Simulação Distribuída Utilizando Protocolos Independentes e Troca Dinâmica nos Processos Lógicos

Célia Leiko Ogawa Kawabata

Orientadora: ***Profa. Dra. Regina Helena Carlucci Santana***

Tese apresentada ao Instituto de Ciências Matemáticas e de Computação - ICMC-USP, como parte dos requisitos para obtenção do título de Doutor em Ciências – Ciências de Computação e Matemática Computacional.

"VERSÃO REVISADA APÓS A DEFESA"

Data da Defesa:	26/09/2005
Visto do Orientador:	

Ao meu marido, Masaki,
e ao meu filho, Daniel,
inspirações que me dão motivo para
sempre melhorar.

Agradecimentos

À Deus, por todos os presentes que tenho recebido, e por todos os momentos bons que eu tenho passado.

À Profa. Dra. Regina Helena Carlucci Santana, meus sinceros agradecimentos pela orientação, amizade e paciência, que já duram 12 anos.

Ao Prof. Dr. Marcos José Santana, pela amizade e pela ajuda no desenvolvimento desta tese.

Ao meu marido Masaki, pela paciência, pela ajuda na montagem e na configuração do cluster e por tudo o que ele significa na minha vida. Ao meu filho Daniel, por ser a melhor coisa que aconteceu na minha vida. Eu amo vocês dois.

Aos meus pais, pelas lições de luta e humildade. Às minhas irmãs Rosa, Dirce e Leila pela amizade e confiança.

Aos integrantes do grupo de Sistemas Distribuídos e Programação Concorrente. A todos os que lá passaram desde 1993 até hoje.

À Sarita, grande amiga, sem a sua ajuda esse trabalho provavelmente não teria saído....

À Naninha, pela grande ajuda (principalmente no ParSMPL!!), pelas caminhadas, pela paciência.... Muito obrigada!!!!

À Kali, grande amiga, que eu não poderia esquecer jamais.

Ao Edmilson, cujo apoio foi decisivo para a conclusão desse trabalho.

Ao Daniel e à Renata, amigos muito especiais!

Aos amigos, professores e alunos, da UNICEP/São Carlos.

Ao Luís, à Debbie, à Dani e à Cris, amigos que mesmo distantes são muito importantes na minha vida.

Aos meus queridos alunos Tiago, Érika, Paulo e Léo.

À Bete, Laura e Ana Paula, pela ajuda e pela paciência!!!

À Ângela, Carmen e Sueli, sempre amigas!

E aos demais funcionários do ICMC/USP que sempre foram extremamente prestativos.

E, a todos aqueles que torceram para que esse dia chegasse. A todos vocês, muito obrigada!!!

Para ser grande, sê inteiro: nada
teu exagera ou exclui.
Sê todo em cada coisa. Põe quanto és
no mínimo que fazes.
Assim em cada lago a lua toda
brilha, porque alta vive.
(Fernando Pessoa)

Resumo

KAWABATA, C. L. O. (2005). *Simulação Distribuída utilizando Protocolos Independentes e Troca Dinâmica nos Processos Lógicos*. Tese (Doutorado) – Instituto de Ciências Matemáticas e de Computação, Universidade de São Paulo, São Carlos, 2005.

Esta tese apresenta uma avaliação do desempenho de simulações distribuídas em tempo de execução. Baseando-se nos resultados obtidos nessa avaliação é proposto um mecanismo em que diferentes protocolos de sincronização coexistem em uma mesma simulação. Esse mecanismo tem por objetivo adequar a simulação em execução ao melhor protocolo de sincronização, para garantir melhor desempenho e, conseqüentemente, resultados mais rápidos. Todas as modificações que são necessárias nos protocolos e a definição da troca de mensagens entre os processos são detalhadas neste trabalho. Esta tese apresenta também os resultados dos testes realizados para identificar os casos onde é melhor manter o protocolo conservativo ou onde uma troca de protocolo deve ser considerada. Os resultados obtidos são apresentados e mostram em que momento a troca deve ser considerada. Diferentes abordagens podem ser utilizadas para avaliar o desempenho da simulação, considerando cada processo individualmente ou todos os processos globalmente. De maneira análoga, a troca de protocolos pode ser realizada de forma local ou global. Essas considerações permitem a criação de uma taxonomia para a troca de protocolo que também é apresentada nesta tese.

Palavras-chave: simulação, simulação distribuída, avaliação de desempenho, protocolos de sincronização.

Abstract

KAWABATA, C. L. O. (2005). *Distributed Simulation using independent protocols and dynamical change in logical processes*. Ph.D. Thesis – Instituto de Ciências Matemáticas e de Computação, Universidade de São Paulo, São Carlos, 2005.

This thesis presents a performance evaluation of distributed simulations during execution time. According to the results obtained in this evaluation, it is proposed a mechanism where different synchronization protocols can be used in the same simulation. This mechanism aims at tunning the simulation in execution to the best protocol in order to reach better performance. All modifications needed in the protocols and the definition of the exchanged messages between logical processes is presented in this work. This thesis also presents the results of the tests realized to identify the cases where it is better to keep the conservative protocol or it is better to swap the protocol. The results obtained are presented and shown when it is necessary to swap the protocol. Different approaches can be used to evaluate the simulation performance considering that it is possible to evaluate each logical process locally or all of them globally. The change of the protocol can also be applied in just one logical process or in all of them. These considerations allowed the definition of a taxonomy that is also presented in this thesis.

Keywords: simulation, distributed simulation, performance evaluation, sincronization protocols.

Lista de Figuras

Lista de Tabelas

Lista de Abreviaturas

CC	*Channel Clock*
CMB	Chandy – Misra – Bryant
EVL	*Event List*
FIFO	*Fist-in-first-out*
GVT	*Global Virtual Time*
HLA	*High Level Architecture*
IB	*Input Buffer*
IQ	*Input Queue*
LMTS	Limite mínimo do Timestamp
LVT	*Local Virtual Time*
LVTH	*Local Virtual Time Horizon*
MRIP	*Multiple Replication in Parallel*
Msg	Mensagem
OB	*Output Buffer*
OQ	*Output Queue*
pf	Processo físico
PG	Processo Gerenciador
pl	Processo lógico
RPC	*Remote Procedure Call*
S	*Status*
SRIP	*Single Replication in Parallel*
SS	*Simulation Status*
TW	*Time Warp*
TGAG	Análise Global Troca Global
TGAL	Análise Local Troca Global
TLAG	Análise Global Troca Local
TLAL	Análise Local Troca Local
TWLP	*Time Warp Logical Process*
Vcc	Valor de clock do canal

Sumário

1 Introdução

No último século foi verificado um crescimento muito grande na difusão do conhecimento e no compartilhamento da informação. A Internet representa um papel fundamental nesse cenário, possibilitando a comunicação de pessoas entre partes diferentes do mundo a um custo bastante acessível. Assim, os computadores se tornaram ferramentas comuns e essenciais para o dia-a-dia das pessoas.

O desenvolvimento de novos projetos leva à necessidade de ferramentas para apoio ao desenvolvimento e avaliação dos sistemas. É preciso encontrar formas de verificar qual a abordagem mais apropriada a ser seguida para a implementação de um sistema sem ter a necessidade da existência desse sistema, de forma a prever como será o seu desempenho.

A validação de novas propostas (como novas arquiteturas) tem exigido a utilização de ferramentas de apoio, uma vez que a construção de protótipos torna-se cada vez mais difícil, ou não apropriada, tanto em termos de dificuldade de implementação quanto em termos de custo.

Conforme os sistemas vão se tornando mais complexos, as análises, validações e avaliações vão se apoiando cada vez mais em técnicas de modelagem (Fujimoto, 2003). Modelar um sistema significa abstrair suas características essenciais (que dependem do enfoque considerado) e reuni-las de modo coeso em um modelo, que represente o sistema com certo grau de identidade.

A simulação é uma solução que vem cada vez mais sendo utilizada (Agarwal et al 2005, Bauer 2005, Kalanterry 2004) devido à sua flexibilidade e ao seu baixo custo. Simular um modelo implica em construir um programa, por meio de uma linguagem de programação, e executá-lo em um sistema computacional para a geração dos resultados

necessários para a avaliação. Um programa computacional deve, em geral, ser escrito e validado por meio de testes, para só então poder gerar resultados confiáveis.

A simulação de modelos de maneira seqüencial, para modelos complexos em que muitos detalhes são considerados, pode requerer horas ou até mesmo dias de processamento computacional. Essa situação vem mudando com o uso de sistemas distribuídos formados por múltiplos processadores conectados por meio de infra-estruturas com alta velocidade de comunicação, onde o paralelismo pode ser explorado (Kumar et al, 2003, Foster et al, 2002).

As técnicas de simulação evoluíram de forma a possibilitar a exploração do paralelismo dessas máquinas. A idéia de simulação distribuída foi proposta por Chandy e Misra (Chandy & Misra, 1979) em uma série de conferências na Universidade de Waterloo e por Bryant (Misra, 1986, Fujimoto, 1995). A partir dessa época foram propostos esquemas básicos para a evolução de uma simulação distribuída. O estudo de simulação distribuída vem se tornando cada vez mais importante devido ao fator limitante do tempo computacional (Bononi et al, 2005, Curry et al, 2005, Lee, 2005, Lee et al, 2005). A solução seqüencial de modelos de sistemas mais complexos é mais difícil devido a grande quantidade de variáveis e à grande quantidade de dados produzidos e armazenados.

Uma simulação distribuída pode ser classificada em conservadora, otimista ou mista (Spolon et al, 1999). Essa classificação se baseia na forma como é realizada a sincronização dos processos lógicos por meio da execução dos eventos que compõem o modelo simulado. Na simulação conservadora somente eventos que são considerados seguros são executados (Chen & Szymanski, 2002b, Curry et al, 2005, Park et al, 2004). Na abordagem otimista, os eventos são executados sem verificar se são seguros, e se for verificado que a simulação foi executada fora de ordem, mecanismos que recuperam a simulação até um ponto seguro são acionados (Li & Tropper 2004, Santoro & Quaglia 2005, Steinman 2005). Já a abordagem mista possui características das duas abordagens.

Em uma quarta abordagem, proposta na década de 90, diversas replicações de uma simulação são executadas em paralelo e cada replicação é executada seqüencialmente. A diferença básica desta abordagem para as otimistas e conservadoras é que nestas últimas cada replicação é paralelizada. Nesta abordagem, denominada

MRIP (múltiplas replicações em paralelo) (Mota et al, 1999, Pawlikowski, 2003, Bause & Eickhoff, 2003, Jeong et al, 2005), a minimização da sobrecarga para sincronização dos processos paralelos é a principal vantagem.

Desde a proposta dos protocolos otimistas e conservadores para simulação distribuída diversos trabalhos vem sendo desenvolvidos visando melhorar o desempenho desses protocolos. Em relação ao protocolo conservador destacam-se os avanços em relação ao cálculo de *lookahead* (Deelman et al, 2001 e Meywe & Bagrodia, 1998), a novas propostas de sincronização (Chen & Szymanski, 2002 e Chen & Szymanski, 2002b), dentre outros. Em relação ao protocolo otimista têm-se pesquisado formas de diminuir a sobrecarga de salvamento de estados (Li & Trooper, 2004), abordagens de cancelamentos errôneos que sobrecarregam menos a simulação (Zeng et al, 2004), formas de melhorar o desempenho da simulação por meio do uso de incerteza temporal (Quaglia & Beraldi, 2004), dentre outras (Zeng et al, 2004, Xu & Tropper, 2005, Santoro & Quaglia, 2005).

1.1 Motivação

Uma questão bastante importante a respeito dos protocolos de sincronização conservador ou otimista é qual protocolo pode se adequar a uma determinada aplicação. Existe um consenso que a formulação de regras para determinação da superioridade de um protocolo em relação a outro é tarefa bastante difícil e que envolve uma série de características, tanto do modelo quanto da plataforma utilizada (Ulson, 1999, Bagrodia et al, 1999, Teo et al, 2002, Fujimoto, 2003, Wilmarth et al, 2005, Teo et al, 2005, Lee et al, 2005).

Diversos trabalhos têm sido desenvolvidos no grupo de Sistemas Distribuídos e Programação Concorrente do ICMC-USP com a proposta de facilitar a utilização da simulação distribuída (Ulson, 1999, Bruschi, 2003, Bruschi et al, 2004, Morselli, 2000, Spolon, 2001). Porém, mesmo atento a essas propostas, o usuário deve decidir qual protocolo utilizar. A utilização de um protocolo que não é o mais adequado para um modelo / plataforma pode tornar a simulação distribuída mais lenta que a seqüencial. Dessa forma, equacionar esses parâmetros em busca do protocolo mais adequado torna-se essencial.

Para facilitar essa tarefa foi proposto, em um trabalho de doutorado (Morselli, 2000), um mecanismo de troca entre os protocolos *Time Warp* (protocolo otimista) e CMB (protocolo conservador), automático, durante a execução da simulação.

Esse mecanismo permite que um programa de simulação seja iniciado com um dos protocolos e durante a execução da simulação é feita uma análise do desempenho. De acordo com os resultados obtidos por essa análise é possível verificar se o protocolo está tendo um bom desempenho, caso contrário é efetuada a troca do protocolo.

A existência de um mecanismo que realize a troca de protocolos em tempo de execução minimiza a responsabilidade do usuário de simulação em determinar qual o melhor protocolo. No entanto, torna-se necessário o desenvolvimento de um mecanismo que avalie o andamento da simulação e determine se a troca de protocolo deve ser considerada.

Um outro ponto a ser analisado na proposta de Morselli (2000) é a sobrecarga imposta pelo mecanismo de troca de protocolo. Se um dos processos lógicos apresenta um baixo desempenho, todos os processos lógicos são obrigados a trocar de protocolo, causando uma sobrecarga considerável.

1.2 Objetivos

O objetivo desta tese de doutorado é avaliar simulações distribuídas em tempo de execução de forma a viabilizar a troca de protocolo CMB para o *Time Warp* durante o tempo de execução considerando dois pontos: quando a troca deve ser considerada e como minimizar a sobrecarga imposta pela troca.

Para minimizar a sobrecarga propõe-se uma troca parcial do protocolo de sincronização. Neste caso, a simulação é executada com processos lógicos utilizando o protocolo conservador e processos lógicos utilizando o protocolo otimista. Isto é, os dois protocolos coexistem em uma mesma simulação.

Essa troca parcial permite uma flexibilidade maior ao mecanismo e um ganho de desempenho pela diminuição da sobrecarga que o mecanismo original, proposto por Morselli (2000), impunha a simulação devido à necessidade de interromper a simulação para a troca do protocolo.

A possibilidade de execução simultânea dos dois protocolos na simulação é uma vantagem, pois podem-se acomodar modelos que possuem características em que parte do modelo pode ser simulado de forma mais eficiente com um protocolo e parte com outro protocolo. Possibilita também que a simulação, como um todo, não necessite mais ser interrompida para a troca de protocolo. Isto é, a decisão pela troca do protocolo é realizada localmente em um determinado processo lógico e, apenas esse processo, que apresenta baixo desempenho, tem o seu protocolo alterado, e os demais continuam a simulação.

Para determinar quando se deve considerar a troca de protocolos e também para validar a proposta de troca parcial de protocolo foram feitos testes de simulação no protocolo conservador CMB utilizando o pacote ParSMPL, desenvolvido no Grupo de Sistemas Distribuídos e Programação Concorrente nos trabalhos de doutorado de Ulson (1999) e mestrado de Tatsumi (2003). Esses testes permitiram a definição de métricas que indicam o desempenho do protocolo conservador que poderão ser utilizados para definir o momento da troca de protocolo de conservador para otimista para cada processo lógico.

Os testes realizados permitiram a verificação de que cada processo lógico que compõe uma simulação possui características que definem que um protocolo pode ser mais adequado para esse processo lógico específico e outro protocolo mais adequado para os demais processos lógicos. Essa constatação mostra a importância da proposta desta tese de doutorado, na busca de um melhor desempenho para a simulação.

O mecanismo proposto apresenta um diferencial com relação a proposta definida por Morselli (2000), pois o mesmo apresenta diversas alterações que viabilizam a decisão local de troca de protocolo e execução de processos lógicos executando protocolos diferentes. A forma como os processos lógicos avançam no tempo da simulação também foi modificado de forma a processos conservadores não terem problemas de execução de eventos fora da ordem cronológica. Essas considerações não eram necessárias no mecanismo original, pois a execução da simulação deveria prever apenas que a troca de protocolo fosse efetuada em uma situação consistente. Desta forma, os detalhes de implementação, o funcionamento do mecanismo e diversos outros elementos que compõem a simulação são discutidos nesta tese.

1.3 Organização da tese

Para a definição do mecanismo proposto é necessário o emprego dos conceitos de simulação distribuída e dos protocolos de sincronização. O Capítulo 2 aborda a simulação, a simulação distribuída, a abordagem MRIP e os protocolos de sincronização de simulação distribuída SRIP.

O Capítulo 3 aborda o mecanismo de troca de protocolo proposto por Morselli (Morselli, 2000) mostrando os processos: gerenciador, observador e conversor que compõem o mecanismo. Detalhes do funcionamento do mecanismo, além dos algoritmos do mecanismo são apresentados nesse capítulo.

O capítulo 4 mostra o planejamento efetuado para a realização dos experimentos e o capítulo 5 expõe os testes realizados em simulação distribuída que mostram a importância da ferramenta proposta e mostram o momento em que a troca de protocolo deve ser considerada.

Definido o mecanismo no capítulo 3 e a importância da proposta de troca parcial foi possível criar o mecanismo modificado que é o assunto detalhado no Capítulo 6 que mostra o funcionamento, modificações, vantagens e desvantagens do mecanismo proposto. Nesse capítulo é apresentada também a taxonomia proposta para o mecanismo de troca.

De acordo com o novo mecanismo proposto e os testes realizados no protocolo conservador foi possível detalhar a abordagem de análise de dados local e troca de protocolo local, que é apresentado no Capítulo 7. Por fim, as conclusões, contribuições relevantes deste trabalho, propostas para trabalhos futuros são apresentadas no Capítulo 8. No Apêndice A estão os resultados das simulações efetuadas neste trabalho.

2 Simulação

Este capítulo aborda alguns aspectos sobre simulação, mais especificamente a simulação distribuída, as abordagens MRIP e SRIP e os protocolos de sincronização para simulação distribuída.

2.1 Considerações iniciais

A simulação computacional é um processamento que modela o comportamento dinâmico de algum sistema real ou imaginário. Ela é bastante utilizada atualmente para analisar sistemas como controle de tráfego aéreo ou sistemas de telecomunicações das próximas gerações sem a necessidade da construção desses sistemas. A simulação substitui a construção de protótipos que pode ser muito custosa, não factível e/ou perigosa (Fujimoto, 2000). Outro uso importante da simulação é a criação de mundos virtuais nos quais os seres humanos e os dispositivos físicos estão inseridos de forma natural. Um exemplo disso são os simuladores de vôos que treinam pilotos de avião.

A simulação é uma ferramenta bastante poderosa para o estudo do desempenho de sistemas, sejam eles computacionais ou não, porém ela consome muito tempo e recursos computacionais. Mesmo com os processadores atuais é comum que as simulações demorem horas e até dias para serem executadas. Assim, para minimizar o tempo de processamento de simulações tem-se optado pela utilização de simulação distribuída (Fujimoto, 1990, Nicol & Fujimoto, 1995, Misra, 1986, Fujimoto, 1995, Ferscha, 1995, Fujimoto, 2000, Alonso et al, 1998, Fujimoto, 2003), que explora a utilização de máquinas paralelas e/ou sistemas distribuídos com vários processadores e

alta taxa de comunicação (Foster et al, 2002, Kumar et al, 2003, Almasi & Gottlieb, 1994, Hwang, 1993, Reed & Fujimoto, 1987). A velocidade dos processadores aumenta de acordo com o avanço da tecnologia, porém, existem limites na velocidade que pode ser atingida com um processador.

A alternativa de utilização de máquinas paralelas requer a disponibilidade de máquinas de alto custo, que não estão disponíveis para a grande maioria dos usuários de simulação. Dessa forma, a utilização dessas arquiteturas não é uma solução realista para agilizar uma simulação.

Dessa forma a solução mais viável é a utilização de um sistema distribuído executando uma simulação distribuída. Duas abordagens vêm sendo utilizadas para a execução de simulação distribuída, SRIP e MRIP (Bruschi, 2003, Ewing et al, 1998, Mota et al, 1999, Pawlikowski, 2003, Ewing et al, 2002).

Uma das abordagens da simulação paralela é dividir o modelo de simulação e simular cada uma das partes em um processador diferente, a chamada técnica SRIP (*Single Replication in Parallel*). A SRIP considera a paralelização de uma simulação e, para algumas aplicações (principalmente aplicações altamente paralelizáveis com granulosidade grossa), tem se mostrado bastante eficiente para a redução do tempo de simulação, mas para possibilitar a utilização de diversos processadores na execução de uma simulação é necessário adaptar a simulação seqüencial para esta nova situação.

Nessa adaptação, dois pontos fundamentais podem ser identificados: o sincronismo entre os processos concorrentes e as ferramentas a serem utilizadas para implementação do programa paralelo (Kumar et al, 2003, Foster, 1995, Quinn, 1994).

Deve-se então utilizar um protocolo de sincronização que permita o avanço da simulação de forma correta. Como o processo de simulação é dividido entre os diversos elementos de processamento é necessário prover um mecanismo que permita que os eventos que estão ocorrendo sejam executados de forma que os resultados obtidos sigam a ordem cronológica dos fatos dentro da simulação. Diversos estudos têm sido feitos em simulação distribuída (Fujimoto, 2000, Fujimoto, 2001, Fujimoto, 2003, Park et al, 2004) e os protocolos de sincronização existentes podem ser classificados como: conservadores ou otimistas.

Os protocolos conservadores são os protocolos que executam a simulação apenas quando é seguro, ou seja, a simulação de um evento ocorre apenas se não houver a possibilidade de ocorrer um novo evento, anterior ao evento que está sendo executado.

Nos protocolos otimistas a simulação é executada sem preocupações sobre se é seguro continuar a execução da simulação, pois se supõe que erros não irão ocorrer. Mas, se eles ocorrerem, a simulação deve retornar até o ponto em que o sistema estava consistente e reiniciar a execução a partir desse ponto. Os protocolos de sincronização serão abordados com mais detalhes na seção 2.4.1.

Dependendo da natureza do modelo, a divisão do modelo de simulação em partes adequadas pode ser difícil de ser encontrada. Se o modelo não for dividido de forma a obter um bom balanceamento, o ganho com a paralelização será bem inferior à proporção do número de processadores utilizado.

Neste contexto uma outra abordagem pode ser utilizada na simulação paralela, o MRIP (*Multiple Replication in Parallel*) onde, ao invés de dividir o programa de simulação, são utilizadas múltiplas instâncias de um programa de simulação seqüencial, que ficam executando simultaneamente em processadores diferentes. A abordagem MRIP considera a paralelização das replicações de uma simulação. A seção 2.3 discute esta abordagem com mais detalhes.

2.2 Simulação e tempo de simulação

Em uma simulação computacional quem representa as mudanças de comportamento do sistema (sistema físico) é o programa. Para que o sistema físico possa ser emulado deve haver certas informações no programa que permitam essa representação, assim, o sistema físico contém as mudanças de estado que estão ocorrendo nele, e a simulação deve prover: (1) alguma forma de representação dos estados desse sistema, (2) alguma forma de representação dessas mudanças de estado de forma a mostrar a evolução do sistema físico e (3) algum tipo de representação para o tempo (Fujimoto, 2000).

Existem diferentes noções de tempo que devem ser vistas, pois o tempo de simulação segue certas regras que são diferentes da noção de tempo que as pessoas

possuem, chamado tempo físico.

O tempo de simulação pode ser definido como um conjunto de valores totalmente ordenados onde cada valor representa um instante do tempo do sistema físico sendo modelado. Dessa forma, para quaisquer dois valores de tempo T_1 representando o tempo físico P_1, e T_2 representando P_2, se $T_1 < T_2$ então P_1 ocorre antes de P_2, e $(T_2 – T_1)$ é igual a $(P_2 – P_1) * K$ para alguma constante K para cada processo lógico. Se $T_1 < T_2$ então T_1 ocorreu antes de T_2 e se $T_1 > T_2$ então T_1 ocorreu depois de T_2.

Para se utilizar a simulação distribuída, em geral faz-se a conversão da simulação seqüencial. O grande problema dessa conversão é a natureza seqüencial da execução dos eventos de uma simulação. A abordagem clássica para simulação seqüencial aborda a manipulação de uma estrutura de dados chamada lista de acontecimentos futuros, de acordo com uma variável chamada relógio (*clock*) que representa o tempo de simulação.

2.3 MRIP (Multiple Replications in Parallel)

A técnica *SRIP (Single Replication in Parallel)* é caracterizada por possuir um único programa de simulação, que é dividido em processos lógicos, onde cada processo é simulado em um processador. Já a técnica *Multiple Replications in Parallel* (*MRIP*) é uma alternativa quando se deseja utilizar diversos processadores para realizar uma simulação, onde várias instâncias ou replicações de um mesmo programa de simulação seqüencial são executadas independentemente em diferentes processadores (Glynn & Heildelberger, 1992; Ewing et al, 1998, Ewing et al, 1999, Bause & Eikhoff, 2003, Jeong et al, 2005). Essas instâncias executam de forma independente e enviam continuamente parâmetros de interesse para um processo central controlador, ou seja, os resultados produzidos por cada replicação são enviados para um analisador global. Este processador central calcula uma estimativa média de cada parâmetro das observações de todos os processos que estão sendo executados, calculando o intervalo de confiança das variáveis estimadas. Quando as variáveis atingem a precisão requerida, ou seja, quando é verificado que existem observações o suficiente para se ter a precisão necessária, a simulação é finalizada.

Como as simulações executam independentes, se há “n” cópias da simulação executando em “n” processadores pode-se chegar a conclusão de que serão produzidas observações a uma taxa n vezes mais rápida do que uma cópia da simulação apenas, e, portanto são produzidas em 1/n unidades de tempo, observações o suficiente para se ter a precisão necessária. Dessa forma, por meio da técnica MRIP é possível ter-se um *speedup* da simulação aproximadamente proporcional ao número de processadores utilizados.

A técnica MRIP também provê certo grau de tolerância à falha, pois se um processador falhar, como não importa de qual processador as estimativas são produzidas, o programa que estava sendo executado no processador que falhou pode simplesmente ser reiniciado e continuar a simulação sem problemas. De forma alternativa, a simulação pode continuar nos outros processadores, e o processador falho pode ser ignorado. Neste caso, o tempo de simulação deve ser proporcionalmente aumentado para cobrir a falha desse processador.

As vantagens na utilização da *MRIP* são:

- Pode ser aplicado a qualquer programa de simulação, sem a necessidade de paralelizá-lo ou modificá-lo;
- Desde que as simulações são independentes, se *n* replicações estão sendo executadas em *n* processadores, *n* resultados serão produzidos enquanto na versão seqüencial somente um seria produzido, ou seja, produz um *speedup* aproximadamente igual ao número de processadores;
- Possui tolerância a falha, uma vez que são executadas várias instâncias de um mesmo programa. O único ponto crítico é em relação ao analisador global, pois na ocorrência de problemas, a simulação pára.

2.4 SRIP (Single Replication in Parallel)

A utilização da simulação distribuída não é uma tarefa trivial, uma vez que ela se utiliza das estruturas existentes em uma simulação seqüencial e, adicionalmente, as estruturas necessárias à simulação distribuída (Fujimoto, 2000, Curry et al, 2005, Lee et al, 2005, Robinson, 2005). As estruturas básicas necessárias em uma simulação

seqüencial são: variáveis de estado, lista de eventos futuros e relógio global. Cada evento possui um *timestamp* que tem por objetivo indicar quando uma mudança vai ocorrer no sistema. O programa de simulação consiste basicamente de um laço que continuamente varre a lista de eventos futuros (que é ordenada por tempo de chegada) retirando o próximo evento a ser processado. O processamento do evento pode significar mudanças no estado e/ou no escalonamento de novos eventos. Um ponto importante que deve ser lembrado é que os eventos devem ser executados na sua ordem cronológica de *timestamps* para que não ocorram erros de causa e efeito, isto é, que eventos futuros sejam processados antes de eventos com *timestamp* menor (MacDougall, 1987, Fujimoto, 2000, Reed & Fujimoto, 1987, Fujimoto, 2003, Steinman, 2005, Teo & Onggo, 2004).

O princípio de causa e efeito implica que estados do futuro não devem afetar o passado. O princípio impõe uma ordenação parcial nos estados transitórios de um sistema físico. Se uma transição de estado tem algum efeito em outra transição de estado, de forma que uma deva sempre ocorrer antes da outra, chega-se a conclusão de que a causa sempre deve preceder o efeito. As transições de estado que não tem relacionamento direto nem indireto em outra transição de estado não possuem limitações quanto a sua ordenação no tempo e não precisa ocorrer em nenhuma seqüência de tempo no sistema físico. Entretanto, o princípio de causa e efeito impõe uma ordenação parcial em todas as transições de estado.

Na implementação seqüencial de uma simulação o problema da causa e efeito é facilmente resolvido por meio da ordenação dos eventos de acordo com seu *timestamp*. Porém, é muito mais difícil evitar erros de causa e efeito em simulações onde o programa de simulação é executado em sistemas paralelos ou distribuídos, porque muitos eventos são executados concorrentemente. Dessa forma a simulação SRIP possui, em seus protocolos, regras que garantam a não ocorrência de erros de causa e efeito.

Na simulação *SRIP* o sistema é visto como um conjunto de processos lógicos $pl_0, pl_1, ..., pl_n$ cada um representando um processo físico. Toda a interação entre os processos físicos é modelada por mensagens (eventos com *timestamp*) enviadas entre os processos lógicos.

As próximas seções discutem os protocolos de sincronização para simulação distribuída.

2.4.1 Protocolos de sincronização

A simulação seqüencial baseia-se em uma *lista de eventos futuros* que armazena os eventos a serem executados pela simulação, ordenados pelo tempo em que eles devem ocorrer (MacDougall, 1987). A ativação desses eventos é controlada por um sistema de relógio que monitora o tempo da simulação.

O problema da simulação distribuída está na natureza seqüencial dessa lista de eventos futuros. Um único evento é retirado da lista para ser executado e esse evento poderá gerar novos eventos com diferentes localizações na lista (esses novos eventos gerados podem, por exemplo, ser inseridos no início da lista). Esse mecanismo torna difícil a execução concorrente dos eventos, uma vez que a lista de eventos futuros não pode ser simplesmente dividida.

Para a solução do problema de sincronismo entre os processos de uma simulação distribuída, diferentes protocolos de sincronização, conservadores ou otimistas, têm sido propostos.

No protocolo de sincronização conservador os tempos de ocorrência dos eventos são verificados de forma que nenhum evento seja executado se houver a possibilidade de haver eventos com tempo de ocorrência menor. Desta forma garante-se que os eventos estão sendo executados na ordem cronológica imposta no modelo real simulado. Assim, pode-se considerar como principal característica dos protocolos conservadores a ausência de erros de causa e efeito. Um erro de causa e efeito ocorre quando é realizado um evento fora da sua ordem cronológica. O protocolo CMB (Chandy – Misra – Bryant) (Chandy & Misra, 1979, Chandy & Misra, 1981) é um exemplo dessa categoria de protocolos.

Por outro lado, em um protocolo de sincronização otimista, cujo principal exemplo é o protocolo *Time Warp* (Fujimoto, 1990, Nicol & Fujimoto, 1995, Jefferson 1985), os eventos são executados sem a verificação prévia da ordem cronológica. Dessa

forma podem ocorrer erros de causa e efeito que obrigam a restaurar a simulação até um estado anterior consistente. Feita essa ação, a simulação é reiniciada a partir do ponto em que a simulação é considerada segura.

Os protocolos conservadores são descritos na seção 2.4.2 e os otimistas na seção 2.4.3.

2.4.2 Protocolos conservadores

Os protocolos de sincronização conservadores (Taylor et al, 2005, Teo et al, 2002) previnem que erros de causa e efeito ocorram por meio da execução de eventos que sejam considerados seguros e de forma que não haja possibilidade de que nenhuma mensagem com *timestamp* menor chegue ao processo lógico. Essa situação muitas vezes pode levar a um baixo desempenho e/ou a situações de *deadlock*.

Para que *deadlocks* não ocorram, diversas variantes do protocolo têm sido propostas, dessa forma alguns deles evitam o *deadlock* e outros deixam que o *deadlock* aconteça para depois tomar as providências necessárias.

Um conceito importante nos protocolos conservadores é o *lookahead*, definido como o intervalo de tempo em que se pode prever o que irá acontecer na simulação de forma que é possível avançá-la até esse ponto sem a possibilidade da chegada de mensagens dos outros processos lógicos com *timestamp* menor que o intervalo da duração do *lookahead*. Por exemplo, se um processo lógico está simulando no tempo T e somente pode escalonar novos eventos com um *timestamp* de no mínimo T + L, então o intervalo de tempo L pode ser dito como sendo o *lookahead* do processo lógico (Fujimoto, 2000, Boukerche & Tropper, 2001, Ferscha et al, 2001, Socany & Safarik, 2002, Taylor et al, 2005, Wilmarth et al, 2005).

A noção de *lookahead* é fundamental para os mecanismos de sincronização conservadores. É ele o principal responsável pelo desempenho da simulação. Em geral tenta-se aumentar o *lookahead* o máximo possível de forma a melhorar o desempenho dos protocolos conservadores.

O protocolo conservador que será abordado é o protocolo CMB, que foi proposto por Chandy e Misra (Chandy & Misra, 1979) e independentemente por Byrant (Byrant, 1977) e é um dos mais conhecidos. O princípio de funcionamento do algoritmo é que por meio das mensagens trocadas entre os processos lógicos, cada um dos processos lógicos possa definir quais mensagens pode processar (sem erros de causa e efeito) e avançar na simulação. Porém, se o processo lógico precisa esperar as mensagens dos outros processos lógicos para avançar a simulação, existe um tempo de espera grande, o que causa um problema no desempenho e/ou também um *deadlock*.

Um exemplo de *deadlock* pode ser visto na Figura 2.1 que representa os processos lógicos (*pl*) 1, 2 e 3. Cada *pl* está aguardando uma mensagem do outro processo lógico. Nesse caso, o *pl* 1 está bloqueado (com o *clock* de 9 unidades de tempo) esperando uma mensagem do *pl* 2, que por sua vez está bloqueado (com o menor *clock* de canal de 10 unidades de tempo) esperando uma mensagem do *pl* 3 e que também por sua vez está esperando uma mensagem do *pl*1. Esse ciclo cria uma situação de espera infinita que resulta no *deadlock* da simulação.

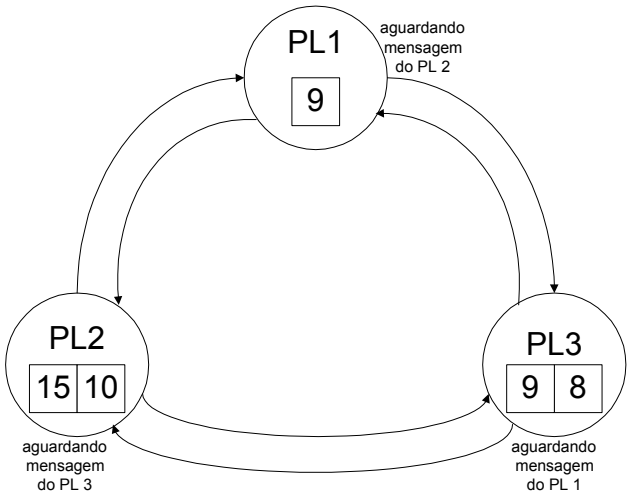


Figura 2.1: Situação de deadlock

Existem diversas estratégias que podem ser utilizadas para evitar situações de *deadlock*. Uma das formas de contornar esse problema é a geração de mensagens nulas (que contém apenas informações de *clock*) para os outros processos para que todos possam avançar seus relógios e continuar os processamentos. A segunda permite que o

deadlock ocorra, porém provê mecanismos de detecção e recuperação do estado de *deadlock*. E por último o *deadlock* é evitado permitindo que os processos executem todos os eventos da fila, podendo ocorrer erros de causa e efeito. Nesse caso, algum mecanismo de retorno ao último estado consistente deve ser previsto. Esse último caso é a estratégia do mecanismo otimista que será abordado na seção 2.4.3.

Existem diversas outras técnicas que podem ser utilizadas para a prevenção de *deadlocks*, porém, a técnica utilizada neste trabalho em questão é a de mensagens nulas. Assim, o uso de mensagens nulas pelo CMB para prevenção de *deadlocks* é discutido em detalhes na próxima seção.

2.4.2.1 Prevenção do *deadlock* por meio da utilização de mensagens nulas

O protocolo de sincronização conservador CMB utiliza canais de comunicação que são estáticos de forma que se sabe desde o início da simulação quais processos podem se comunicar com quais outros processos. As mensagens que chegam de cada canal de comunicação podem ser armazenadas em uma estrutura do tipo FIFO (*First-in-first-out*), pois sempre irão possuir uma ordenação crescente de *timestamp*. Dessa forma, a execução dos eventos do *pl* é intercalada com os processamentos da própria fila de eventos do *pl* e com os eventos da fila dos canais de comunicação. É sempre verificado se existe pelo menos uma mensagem em cada fila e é considerada a mensagem com menor *timestamp* para ser executada. (Fujimoto, 2003, Reed & Fujimoto, 1987, Alonso et al, 1998,).Um exemplo de como funciona esse algoritmo pode ser visto na figura 2.2.

```
Enquanto (simulação ainda não terminou)
        Espere até que cada fila contenha pelo menos uma mensagem
        Remova a mensagem M com menor timestamp
        Clock := timestamp de M
        Processe a mensagem M
```

Figura 2.2: Versão inicial do *loop* de processamento de eventos de um *pl*

Como pode ser visto o *clock* do processo lógico (*LVT – Local Virtual Time)* sempre possuirá o menor valor entre os relógios dos canais de entrada. A cada repetição do *loop*, o processo seleciona o canal com menor *timestamp* e caso haja uma mensagem, processa-a. Caso a fila esteja vazia o processo é bloqueado, pois uma mensagem com um *timestamp* menor do que os dos outros canais pode chegar por este canal (Boukerche & Tropper, 2001, Curry et al, 2005).

Um ponto que deve ser observado nessa estratégia é que algumas vezes os *pl*s podem permanecer em estado de espera como no caso da Figura 2.1 onde cada *pl* estava esperando uma mensagem de outro *pl* formando um ciclo, e com isso, um estado de *deadlock*. Dessa forma, para manter os *pls* informados dos *timestamps* dos *pl*s vizinhos são enviadas mensagens nulas apenas para propósitos de sincronização.

As mensagens nulas servem apenas para atualização do *clock* do processo lógico e não corresponde a nenhuma atividade no processo físico. De maneira geral, uma mensagem nula com *timestamp* T que é enviado de um pl_A para um pl_B é basicamente um meio de informar ao pl_B que o pl_A não irá enviar mensagens posteriores com *timestamp* menor que T. (Fujimoto, 2000, Fujimoto, 2003)

Os processos lógicos determinam o *timestamp* das mensagens nulas por meio dos *timestamps* dos canais de entrada e do conhecimento do tempo de execução da simulação que deve ser feita com a mensagem retirada do canal. Com posse dessas informações é possível para o *pl* prever o limitante inferior da próxima mensagem de saída para cada canal de saída. Se um processo finaliza o processamento de um evento, ele envia uma mensagem nula em cada um dos canais de saída indicando esse limitante inferior. Os processos que recebem essa mensagem podem então recalcular novos limitantes para as suas próprias mensagens nulas a serem enviadas. E assim por diante.

A idéia básica da técnica de mensagens nulas do protocolo CMB é a que foi descrita, porém um outro ponto que deve ser observado é quando devem ser enviadas essas mensagens. Uma abordagem é enviar mensagens nulas para os outros *pl*s após o processamento de cada um dos eventos do *pl*. Isso garante que os processos sempre possuem informações atualizadas dos outros *pl*s. Com essa abordagem o algoritmo descrito na figura 2.2 pode ser revisto para um novo algoritmo descrito na figura 2.3. Nesse algoritmo pode ser observado que as mensagens nulas são processadas de forma a atualizar o *clock* do *pl*.

While (simulação não está terminada)

Espere até que cada FIFO contenha pelo menos uma mensagem

Remova a mensagem M com menor *timestamp*

Clock := *timestamp* de M

Processe M

Envie mensagens nulas para todos os *pl*s vizinhos com *timestamp* igual ao limitante inferior das futuras mensagens. (*clock* + *lookahead*)

Figura 2.3: Algoritmo de mensagens nulas de Chandy/Misra/Byrant

A principal desvantagem dessa abordagem é que, como devem ser enviadas mensagens nulas a cada evento processado, é gerada uma grande quantidade de mensagens nulas, principalmente para valores pequenos de *lookahead*. Dessa forma uma outra abordagem que pode ser utilizada para o envio de mensagens nulas é o envio de mensagens nulas sob demanda. Essa abordagem é descrita na próxima seção.

2.4.2.2 Transmissão de mensagens nulas sob demanda

Neste mecanismo, a transmissão das mensagens nulas é feita sob demanda e não após cada evento (Chandy & Misra, 1981; Fujimoto, 1990, Park et al, 2004). Se um processo verifica que ficará bloqueado porque não há nenhuma mensagem para ser processada na fila com menor valor de *clock*, então esse processo requisita a próxima mensagem, seja ela nula ou não, do processo que tem o menor valor de *clock*. O processo pode então continuar a sua execução após o recebimento da mensagem.

As mensagens também podem ser requisitadas por *timeout*, ou seja, o processo que está em estado de espera há algum tempo faz requisições de mensagens nulas.

Essa abordagem ajuda a reduzir a grande quantidade de mensagens nulas, apesar de aumentar o tempo necessário para recebimento dessas mensagens, uma vez que são necessárias agora duas mensagens.

Para evitar a necessidade de envio de mensagens nulas, uma outra abordagem pode ser utilizada. Essa abordagem difere das anteriores porque ela não tenta evitar o *deadlock*. O mecanismo permite que os *deadlocks* ocorram e provê uma forma de

detectar e recuperar da situação de *deadlock*. Esse mecanismo será visto na próxima seção.

2.4.2.3 Detecção e recuperação do deadlock

Neste mecanismo o *deadlock* não é evitado. Dessa forma, se ele ocorrer, existem procedimentos que irão detectar e recuperar da situação de *deadlock*. Este mecanismo difere dos outros pelo fato de que não existe a transmissão de mensagens nulas (Chandy & Misra 1981; Fujimoto, 1990).

Isso é conseguido por meio da introdução, na simulação, de um processo controlador que tem como função detectar a situação de *deadlock*. Dessa forma o seguinte algoritmo é colocado na simulação:

- É verificado se a simulação está em *deadlock;*
- O controlador envia uma mensagem a um ou mais *pl*s informando-os que certos eventos podem ser processados com segurança, quebrando assim o *deadlock*.
- Os *pl*s processam os eventos que foram declarados seguros. Isso geralmente gera novas mensagens para os outros *pl*s que causam o processamento de mais eventos, que por sua vez geram mais mensagens. A reativação de processos que estavam bloqueados permite a construção de uma árvore, onde cada processo que não está mais bloqueado está na árvore. Se uma mensagem é enviada a um processo que ainda não estava na árvore, então esse processo é adicionado à árvore. Os *pl*s que estão na árvore são ditos comprometidos e os que não estão de não-comprometidos.
- A tendência da árvore é de expandir, porém, em alguns casos, alguns *pl*s podem se tornar bloqueados novamente. Nesse caso, se o *pl* está bloqueado e ele é uma folha da árvore, ele se remove automaticamente da árvore sinalizando a seu pai que não faz mais parte da árvore. Um *pl* se torna uma folha quando todas as folhas que foram porventura adicionadas foram removidas porque se encontravam bloqueadas.
- Se o controlador se tornar um nó folha na árvore, então a simulação está em estado de *deadlock*, completando o ciclo.

Um protocolo de sinalização é utilizado para implementar o paradigma descrito. De forma geral, cada *pl* segue as seguintes regras:

- Quando um processo comprometido recebe uma mensagem, ele imediatamente retorna uma mensagem para quem enviou indicando que a mensagem não causou uma expansão na árvore.
- Quando um processo não-comprometido recebe uma mensagem, ele se torna comprometido; e nenhum sinal é retornado para quem enviou a mensagem a não ser que ele se torne não comprometido novamente.
- Cada *pl* mantém um contador indicando o número de mensagens que foram enviadas sem recebimento de resposta. Quando o contador indica 0, o *pl* é um nó folha na árvore. Se o *pl* está bloqueado e o contador se torna 0, ele se torna não-comprometido, e então é enviado um sinal ao processo que originalmente causou a sua entrada na árvore.

Um *deadlock* pode ser quebrado por meio da observação de que a mensagem que contém o menor valor de *timestamp* em toda a simulação sempre será segura para ser executado. Este seria o próximo evento a ser processado em um programa de simulação seqüencial. Dessa forma, para quebrar o *deadlock*, deve-se apenas identificar o evento que contém o menor *timestamp* e enviar uma mensagem para o(s) *pl*(s) que estão segurando eventos indicando que agora é seguro processá-los.

A localização do evento com menor *timestamp* é tarefa relativamente simples porque a computação está parada por causa do *deadlock*, dessa forma não estão sendo criados novos eventos enquanto o evento com menor *timestamp* está sendo localizado. O controlador manda uma mensagem em *broadcast* para todos os *pl*s requisitando o valor do menor *timestamp* dentro de cada processador. Depois de receber as mensagens de cada um dos processadores, o controlador determina o menor deles e instrui o processador a mantê-lo para o processamento dos eventos. Essa abordagem supõe que não há mensagens em trânsito durante o tempo em que o *deadlock* está sendo desfeito.

Um problema desse algoritmo é a sua característica de especificar que apenas o evento com o menor *timestamp* é seguro para ser processado. Utilizando-se o *lookahead*, um conjunto maior de eventos seguros poderia ser obtido.

2.4.3 Protocolos otimistas

Os algoritmos de sincronização conservadores evitam a violação de erros de causa e efeito, ao contrário dos algoritmos otimistas que permitem que violações ocorram, mas para isso provêem um mecanismo de recuperação. O mecanismo *Time Warp* (Jefferson, 1985) foi o primeiro e continua sendo o mais conhecido algoritmo de sincronização otimista. Apesar de outros algoritmos terem sido propostos, muitos dos conceitos e mecanismos fundamentais utilizados por esses algoritmos como *rollback*, anti-mensagens e tempo virtual global apareceram inicialmente no *Time Warp* (Avril & Tropper, 2001, Balakrishnan et al, 2001, Tang t al, 2005, Li & Tropper, 2005, Quaglia & Beraldi, 2004).

O termo otimista de execução se refere ao fato que os processos lógicos executam os eventos de modo otimista supondo que não irão ocorrer erros de causa e efeito. Da mesma forma como o protocolo conservador, a simulação é composta por uma coleção de processos lógicos que se comunicam exclusivamente por troca de mensagens rotuladas com *timestamp*. Supõe-se que a comunicação entre os processos lógicos é confiável, ou seja, que as mensagens enviadas chegam a seu destinatário. Um processo lógico não precisa necessariamente enviar as mensagens em ordem de *timestamp*, e a rede de comunicação não garante que as mensagens irão chegar na ordem em que foram enviadas. (Quaglia et al, 1999, Bauer et al, 2005)

O algoritmo do *Time Warp* consiste basicamente de duas partes, um mecanismo de controle local e outro de controle global. O mecanismo de controle local é implementado dentro de cada processo lógico, independente dos outros processos. O mecanismo de controle global é utilizado para gerenciar as operações entre os processos lógicos de forma que o resultado da simulação seja o mesmo que seria se a simulação fosse executada seqüencialmente (Choe & Tropper, 2001, Ferscha & Malony, 2001, Ferscha, 1995, Fujimoto, 1999, Glynn & Heildelberger, 1992, Teo et al, 2005, Lee et al, 2005).

2.4.3.1 Mecanismo de controle local

O comportamento dos processos lógicos no sistema *Time Warp* pode ser comparada a uma simulação seqüencial. Na simulação seqüencial tem-se uma estrutura

chamada lista de eventos que incluem todos os eventos que foram escalonados mais ainda não processados. Um processo lógico do *Time Warp* (TWLP) pode ser visto da mesma forma com exceção de:

- Os eventos no TWLP podem resultar de mensagens enviadas deste *pl* para outros *pl*s.
- O TWLP não descarta os eventos depois que eles foram processados, ao invés disso mantêm os eventos processados em uma fila. Isso é necessário porque o TWLP pode precisar de *rollback*, ou seja, os eventos podem ter sido processados erroneamente e devem ser processados novamente na ordem correta.

As seguintes estruturas são utilizadas para definir um processo lógico (Figura 2.4) (Jefferson, 1985):

- **Nome do processo**: deve ser único no sistema;
- **Estado do processo**: são os dados do processo que incluem: a pilha de execução, as variáveis locais e o contador do programa.
- **Relógio virtual local**: determina o *LVT* (Tempo Virtual Local) do processo. O valor do relógio não muda durante a execução do evento, mas somente entre os eventos;
- **Fila de entrada de mensagens**: contêm as mensagens recebidas recentemente ordenadas pelo *timestamp.* Contém mensagens que já foram processadas, mas que são mantidas no caso de necessidade de *rollback*. Somente as mensagens que têm valor do *timestamp* maior ou igual ao *GVT* (Tempo Virtual Global) devem ser mantidas. O GVT será definido na seção 2.4.3.2
- **Fila de saída de mensagens**: contém uma cópia de todas as mensagens recentemente enviadas pelo processo. São necessárias no caso de necessidade de *rollback*, para que possam ser reenviadas indicando erro. Novamente, somente as mensagens que têm valor de *timestamp* maior ou igual ao *GVT* devem ser mantidas.

- **Fila de estados**: contém cópia dos estados recentes do processo. É necessário para possibilitar o *rollback*.

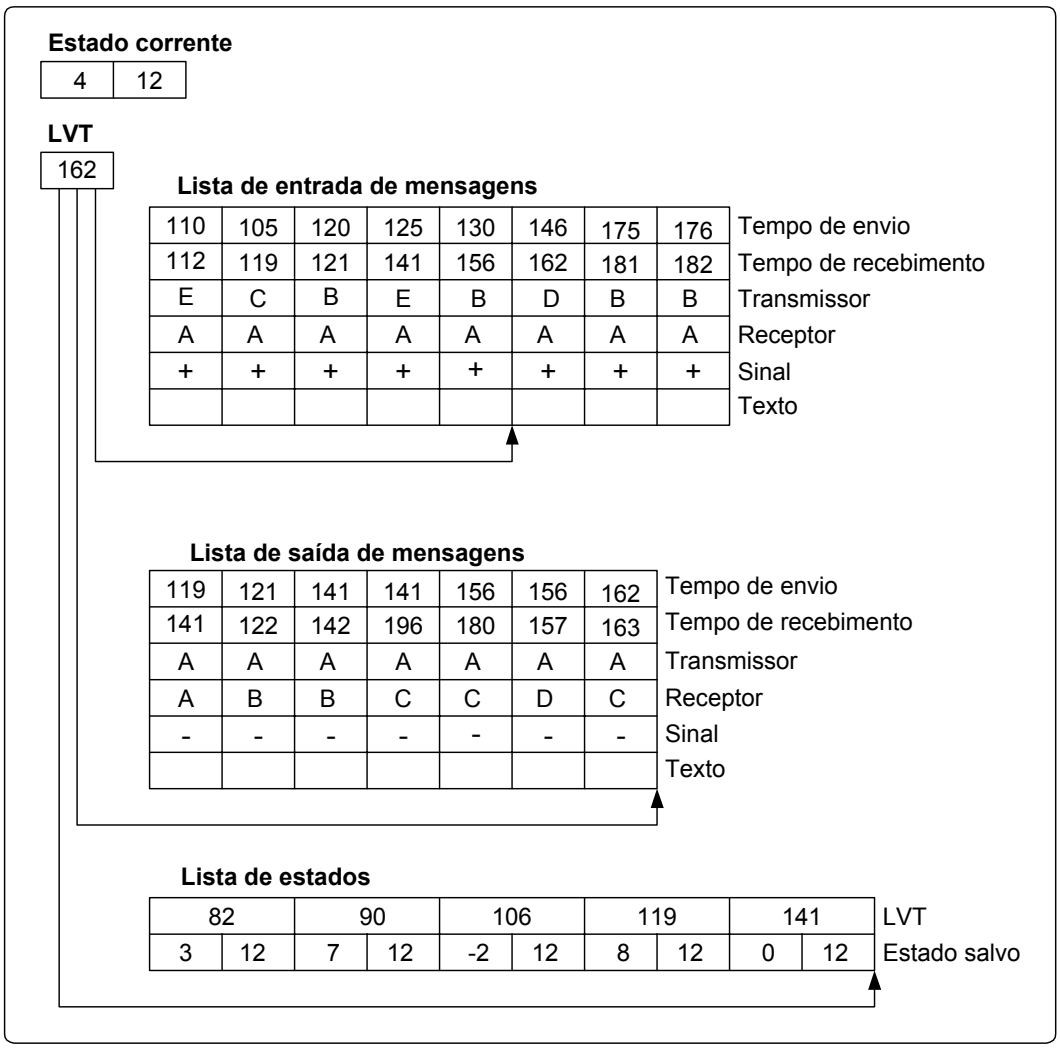


Figura 2.4: Estrutura do processo lógico A no *Time Warp*

Para cada mensagem no *Time Warp* existe uma antimensagem que é exatamente igual em formato e conteúdo exceto em um campo, o sinal. Todas as mensagens enviadas pelo programa têm um sinal positivo (+) e a antimensagem, um sinal negativo (-).Sempre que um processo envia uma mensagem, ela é recebida e guardada na fila de entrada de mensagens do remetente e a antimensagem (com sinal negativo) é guardada na fila de saída de quem está enviando. A antimensagem é armazenada no caso de necessidade de *rollback*.

No mecanismo *Time Warp* a mensagem com menor *timestamp* ainda não processada é sempre segura para ser executada. As mensagens são recebidas e

ordenadas pelo *timestamp*, e vão sendo executadas nessa ordem. Se ocorrer a chegada de uma mensagem com *timestamp* menor do que uma mensagem já processada é acionado o *rollback* (foi detectado um erro de causa e efeito). Essa mensagem que causou o *rollback* é denominada *straggler*.(Quaglia et al, 1999, Xu & Tropper, 2005)

A Figura 2.5 mostra a chegada de uma mensagem *straggler* a um processo. Estava sendo executada a mensagem com *timestamp* 41, com a chegada da mensagem com *timestamp* 18 foi acionado o *rollback* para retornar a simulação até o *timestamp* 12, onde se tem o último evento seguro processado.

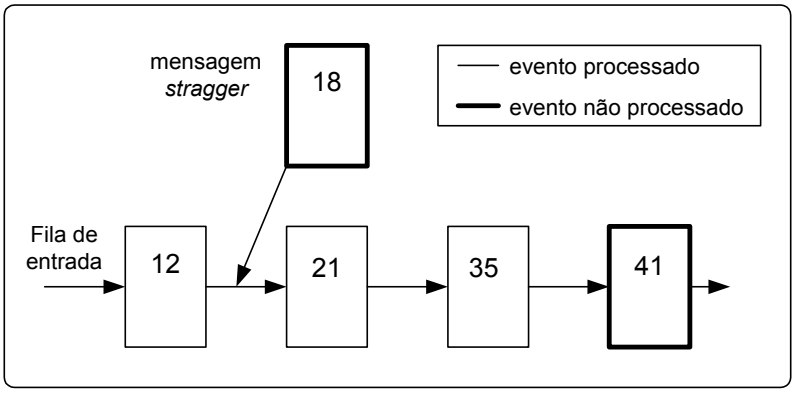


Figura 2.5: Chegada de uma mensagem straggler

O mecanismo de *rollback* serve para desfazer todas as mensagens processadas antes do *timestamp* da mensagem *straggler*. Isso faz com que a simulação retorne ao estado anterior ao processamento errôneo. Para que isso seja possível todos os estados são salvos e são guardadas cópias das mensagens enviadas.

Nesse processo, as mensagens que foram guardadas na lista de saída (antimensagens) são enviadas novamente de forma que os outros processos também possam voltar ao último estado seguro.

Uma antimensagem causa *rollback* no destinatário apenas se o tempo da mensagem recebida for menor que o *timestamp* da mensagem sendo processada. Dependendo-se do tempo, existem diversas possibilidades que podem ocorrer no destinatário (Jefferson, 1985):

1. Se a mensagem positiva chegou, mas ainda não foi processada, então o tempo da mensagem é maior que o tempo virtual local do processo. A mensagem negativa então simplesmente cancela a mensagem que estava na fila, não causando *rollback*. Isso faz com que o processo desconheça que a mensagem existiu, o que é desejável.
2. Outra possibilidade é a mensagem positiva original ter um tempo que está no presente ou no passado em relação ao tempo virtual local do processo e essa mensagem já foi ou está sendo processada, causando modificações no estado do processo e talvez em um outro conjunto de processos. Neste caso, a mensagem negativa irá causar *rollback* no processo de forma a se retornar a um estado seguro. E isso poderá causar impacto em um terceiro conjunto de processos. Caso esse efeito continue ocorrendo pode-se chamar a isso de cascata de *rollbacks*. (Ferscha et al, 2001)
3. A terceira e última possibilidade é a mensagem negativa chegar a seu destino antes da mensagem positiva (Isso é possível considerando-se que o meio de comunicação não oferece mecanismos que garantam a ordem de chegada das mensagens.). Neste caso a mensagem negativa é colocada na fila e nenhuma ação é executada. Somente na chegada da mensagem positiva o mecanismo de *rollback* é acionado.

2.4.3.2 Mecanismo de controle global

O mecanismo de controle local permite garantir que a execução da simulação ocorra da mesma forma que seria em uma simulação seqüencial. Porém, existem problemas que devem ser resolvidos antes do uso desse mecanismo (Fujimoto, 2000, Bauer et al, 2005):

1. A simulação consome memória na execução dos eventos e na criação de novos eventos, mas raramente libera espaço em memória. Deve haver então um mecanismo responsável por verificar espaços de memória que não serão mais necessários no decorrer da simulação. Esse mecanismo é chamado *fossil collection*.

2. Algumas operações não podem ser desfeitas, como por exemplo, operações de I/O.

Ambos os problemas podem ser resolvidos por meio da garantia de que não ocorrerá *rollback* na execução de certos eventos. Por exemplo, se for possível garantir que não ocorrerá *rollback* em um tempo de simulação menor que *T*, então todas as informações guardadas antes do tempo *T* (para que o *rollback* fosse executado, se necessário) podem ser eliminadas, liberando espaço em memória. Da mesma forma, operações de I/O com *timestamp* menor que *T* podem ser executadas sem receio da ocorrência de *rollbacks*. O tempo *T* é um limite inferior que não será ultrapassado por nenhum *rollback*. A esse tempo dá-se o nome de *Global Virtual Time* (GVT).

O GVT *T* é definido como o menor *timestamp* entre todas as mensagens não processadas e parcialmente processadas e anti-mensagens no sistema.

O *Time Warp* calcula o GVT periodicamente ou quando não há mais memória disponível. Dessa forma, existem duas abordagens possíveis para desalocar memória:

- *Batch fossil collection*: quando é feito o *fossil collection*, o *Time Warp* de cada processador desaloca a memória dos eventos que podem ser eliminados.
- *On-the-fly fossil collection*: quando o GVT é calculado, o sistema não precisa desalocar memória imediatamente. Ao invés disso, os eventos processados são colocados em uma lista de eventos (fila FIFO). Quando é necessário memória para mais eventos, o *Time Warp* aloca a memória disponível nessa lista, mas somente dos eventos que possuem *timestamp* com valores menores que o GVT. Essa abordagem evita a procura pela lista de eventos quando da execução do *fossil collection*.

Outro ponto que deve ser abordado no *Time Warp* quando ocorre o *rollback* é a necessidade de fazer o cancelamento das mensagens enviadas de forma a se retornar a um estado seguro para a correta execução da simulação. Existem diversos mecanismos que podem fazer esse cancelamento: cancelamento agressivo, preguiçoso e dinâmico (Fujimoto, 1990, Fujimoto, 2003).

- **Cancelamento agressivo**: as antimensagens são enviadas imediatamente após a detecção do erro de causa e efeito. As antimensagens enviadas podem causar uma cascata de *rollbacks*;

- **Cancelamento preguiçoso**: ao contrário do agressivo, essa abordagem não envia as antimensagens imediatamente após a detecção do erro de causa e efeito. Primeiro é verificado se uma nova execução não irá gerar as mesmas mensagens, evitando um overhead desnecessário. Essa abordagem, em alguns casos, pode melhorar o desempenho do sistema.
- **Cancelamento dinâmico**: cada *pl* decide qual estratégia de cancelamento (agressivo ou preguiçoso) vai utilizar.

De forma geral a estratégia de cancelamento utilizada deve levar em conta o tipo de aplicação sendo simulado de forma a um melhor desempenho do sistema. Outro ponto que também pode causar sobrecarga na simulação é a estratégia de armazenamento das variáveis de estado. Esse armazenamento é necessário para garantir o retorno aos estados anteriores no caso da ocorrência de *rollback*.

O método mais simples é o *copy state saving*, que armazena todos os estados sempre que algum evento é executado, porém, essa abordagem além de tomar muito tempo também requer muito espaço de armazenamento. Uma forma de minimizar isso é a utilização do *sparse state saving*, onde os estados são salvos periodicamente e não mais a cada execução de evento. Outra abordagem que também pode ser considerada é o *incremental state saving*, aonde somente o que foi modificado nos estados são salvos a cada execução de evento (Fujimoto, 2003, Rönngren *et al.*, 1996).

2.4.4 Comparação entre os protocolos conservadores e otimistas

Diversas pesquisas (Ferscha et al, 2001, Suppi et al, 2000, Liljenstam et al, 2001, Ferscha & Malony, 2001, Balakrishnan et al, 2001, Teo et al, 1999, Xu & Chung, 2004, Teo & Onggo, 2004, Teo et al, 2005, Wilmarth et al, 2005) têm sido feitas no sentido de determinar o desempenho dos protocolos de sincronização conservadores e otimistas. Um fato é que esses protocolos são influenciados por uma variedade de fatores como a estruturação dos eventos do modelo de simulação, o particionamento em submodelos, as características de desempenho da plataforma de execução, a implementação da simulação e as otimizações relacionadas aos protocolos.

Do estudo dos dois protocolos é possível construir a seguinte tabela comparativa (Fujimoto, 2000):

Tabela 2.1 :Comparação entre os protocolos conservador e otimista (Fujimoto, 2000)

Protocolo	Conservador	Otimista
Complexidade	Simulação com execução mais simples; Pode precisar de mecanismos especiais no caso de topologia dos *pl*s dinâmica; Possui pouco overhead caso se tenha um bom *lookahead*	Simulação com execução mais complexa, pois requer mecanismos de armazenamento de estados, *fossil collection*, alocação de memória, I./O e erros de execução.
Paralelismo	É limitado pelo pior caso; Necessita de bom *lookahead* para escalabilidade e execução concorrente.	Bom
Desenvolvimento de aplicações	Potencialmente complexo, pois precisa explorar o *lookahead*	Mais robusto a mudanças no modelo; Menos dependência do *lookahead*; Mais transparência no protoloco.
Implementação do protocolo	Fácil	Mais difícil. Deve prover mecanismos que permitam o *rollback*.

A execução de simulação baseada em protocolos conservadores é, em geral, menos complexas do que os protocolos otimistas. E, se a simulação tiver um bom *lookahead*, a execução da simulação será bastante rápida. Isso resultará em um baixo *overhead* e essa abordagem será vantajosa em relação à otimista, que necessita de

armazenamento de informações para a realização do *rollback*. Essa necessidade de armazenamento de estados temporários talvez seja uma das piores desvantagens do protocolo otimista. Porém, o protocolo conservador é extremamente dependente do *lookahead*, pois se esse número for baixo ou nulo, a simulação tende a ser mais lenta que a sequencial.

Os protocolos conservadores não exploram completamente o paralelismo, pois eles se preservam contra o pior caso, já os protocolos otimistas exploram melhor o paralelismo.

Outro ponto bastante importante a ser considerado é que uma boa exploração do *lookahead* torna a tarefa do programador bastante difícil e, modificações na topologia agravam ainda mais esse problema. Se o objetivo do programador é fazer um simulador de propósito geral, talvez o mais sábio seja a utilização dos protocolos otimistas que não necessitam de conhecimento aprofundado sobre o mecanismo de sincronização.

Por outro lado, a reutilização de código seqüencial é mais simples no protocolo conservador, pois o protocolo otimista necessita da implementação de mecanismos que permitam o *rollback*. Mas, deve-se lembrar também, que a abordagem MRIP possui mais vantagens, pois não são necessárias modificações no código seqüencial.

Dessa forma, pode-se concluir que a escolha de um protocolo de sincronização é tarefa difícil, pois se deve considerar diversas características, tanto do modelo quanto do ambiente ou mesmo do programador para que se tenha o desempenho esperado da simulação.

2.5 Considerações finais

A utilização da simulação distribuída tem sido uma opção para sistemas em que o tempo computacional torna-se fator limitante. Diversos estudos têm sido feitos e protocolos de sincronização têm sido propostos. Dentre os mais conhecidos e mais estudados tem-se o CMB e o *Time Warp* e suas variantes.

O CMB é o protocolo de sincronização conservador mais conhecido e possui como característica principal o fato que apenas eventos seguros podem ser processados, evitando que ocorram erros de causa e efeito. O desempenho do protocolo CMB

(Alonso et al, 1998, Ferscha et al, 2001, Boukerche & Tropper, 2001, Teo et al, 2002, Park et al, 2004) está diretamente ligado às características do modelo, na organização dos processos lógicos e nas características do computador sendo usado, além do *lookahead*, onde bons valores conferem um bom desempenho a esse protocolo. Diversos estudos têm sido efetuados visando melhorar o desempenho de simulações conservadoras. Dentre eles cabe-se destacar a comparação entre as abordagens CMB e a baseada em HLA (*High Level Architecture*) de Taylor et al (2005), a simulação conservadora utilizando memória distribuída compartilhada de Teo et al (2002) e a pesquisa sobre *lookahead* de Solkony e Safarik (2002).

Já a abordagem otimista, representada pelo protocolo *Time Warp*, permite que ocorram erros de causa e efeito. Nesse protocolo os eventos são processados sem preocupações quanto a ocorrência de erros, pois se erros ocorrerem basta acionar o mecanismo de *rollback* que fará a simulação voltar ao último estado seguro da simulação.

Muitas pesquisas têm sido feitas para melhorar o desempenho do protocolo *Time Warp*, e variações do protocolo têm sido propostas, como limitação do otimismo (Suppi et al, 2000), balanceamento dinâmico de carga (Choe & Tropper, 2001), utilização de simulação otimista com computação reversa (Tange et al, 2005), criação de um novo algoritmo para cálculo de GVT (Yaun, 2005), sempre com o objetivo de diminuir o tempo de simulação.

As duas abordagens descritas acima são atualmente classificadas como SRIP, onde a simulação é dividida em partes e executadas segundo algum protocolo de sincronização. Outra abordagem possível e que tem sido alvo de muitos estudos é o MRIP (Jeong et al, 2005, Ewing et al, 2002), onde diversas instâncias do mesmo programa são colocadas em diversos processadores de forma paralela. A vantagem dessa abordagem é que ela permite uma análise mais rápida dos dados de saída, uma vez que se tem diversos resultados que podem ser comparados, porém, não resolve o problema de simulação de sistemas muito complexos que requerem muito tempo de simulação.

3 Mecanismo de troca dinâmica de protocolos

Neste capítulo é apresentado o mecanismo proposto por Morselli em (Morselli, 2000), em sua tese de doutorado, que permite a troca entre os protocolos CMB e Time Warp em tempo de execução da simulação e que serviu de base para o desenvolvimento deste doutorado.

3.1 Considerações iniciais

O estudo dos protocolos de simulação distribuída abordado no capítulo 2 leva a uma reflexão sobre o quanto eles são adequados para simular um determinado modelo. Alguns modelos podem ser adequados a um protocolo ou ao outro, mas esse é um aspecto que muitas vezes pode ser detectado apenas durante a execução da simulação, fazendo com que muito trabalho tenha de ser refeito de modo a adequar o modelo a uma nova abordagem de simulação.

Diversas pesquisas têm sido realizadas para avaliar o desempenho dos protocolos conservadores e otimistas (e suas variantes) em diferentes tipos de aplicações. Estas pesquisas têm apontado o *lookahead*, a granulosidade do modelo e a plataforma sendo utilizada como os fatores principais para a determinação do desempenho de uma simulação distribuída (Alonso et al, 1994, Fujimoto, 2000, Teo et al, 2005, Bononi et al, 2005). No entanto, a análise prévia destes fatores para determinar o melhor protocolo a ser utilizado não é uma tarefa trivial, principalmente quando o

usuário não possui grande experiência na utilização de computação paralela. Além disso, em algumas aplicações, alguns fatores podem variar durante a execução da simulação de forma que os dois protocolos acabam sendo eficientes para a execução de partes distintas do sistema. Isto é, as características que determinam qual protocolo utilizar são difíceis de serem definidas, além de se alterarem dinamicamente (Jha & Bagrodia, 1994).

A utilização de protocolos mistos pode ser atrativa para esses casos. No entanto, a quantidade de parâmetros a serem ajustados pelo programador torna a utilização desses protocolos bastante difícil.

Nesse contexto, o mecanismo proposto por Morselli permite que a simulação seja executada sob a abordagem mais próxima daquela que pode ser chamada ideal, pois esse mecanismo verifica o desempenho da simulação e permite uma troca de protocolos em tempo de execução minimizando a atuação do usuário sobre a simulação.

O mecanismo permite, durante a execução de uma simulação distribuída, a troca automática e dinâmica entre os protocolos CMB e *Time Warp* para realizar o sincronismo dos eventos da simulação. Desta forma, utilizando-se de dois protocolos bastante conhecidos, o sistema de simulação proposto poderá decidir, através do comportamento da simulação, qual protocolo de sincronização melhor se adapta ao contexto em que a simulação distribuída está sendo realizada.

De acordo com o estudo desenvolvido por Morselli o mecanismo para a troca de protocolos de sincronização pode ser dividido nas seguintes etapas:

1. Detecção da necessidade de troca de protocolos;
2. Garantia de manutenção da consistência da simulação no momento da troca;
3. Adaptação das estruturas de dados para possibilitar a troca de protocolos de sincronização.

O mecanismo utiliza-se de dois protocolos (CMB e *Time Warp*) e o sistema inicia a simulação com um dos dois protocolos. Durante a execução do sistema, por meio da análise do comportamento da simulação, pode ser efetuada a troca de protocolo.

Na primeira etapa deve-se determinar quando o mecanismo de troca de protocolos de sincronização deve ser considerado. Esta etapa considera duas situações: quando o protocolo CMB está sendo utilizado e deve-se decidir se o *Time Warp* seria mais eficiente; e quando o *Time Warp* está sendo executado e o CMB poderia ser mais eficiente. Para essa análise são coletadas variáveis que indicam o andamento da simulação para que se possa detectar o desempenho do mesmo e a necessidade ou não da troca de protocolos.

Nas etapas 2 e 3 são tomadas providências de forma a manter a consistência da simulação e das estruturas de dados utilizadas pelos protocolos. Dessa forma, essas etapas dependem da identificação das estruturas e processos que caracterizam a implementação dos dois protocolos considerados na construção do mecanismo de troca dinâmica.

Durante a execução da simulação distribuída é necessário obter e analisar as informações que medem o desempenho da simulação para que se possa tomar a decisão da troca de protocolos. E para que isso seja possível foram definidos em (Morselli, 2000, Morselli, 200a) três processos lógicos: observador, gerenciador e conversor. Esses processos serão abordados com detalhes na seção 3.3.

Na próxima seção serão abordados os detalhes de implementação de simulação distribuída de forma a detalhar as modificações que foram introduzidas nos protocolos para a implementação do mecanismo proposto.

3.2 Detalhes de implementação de Simulação distribuída via troca de mensagens

Uma questão importante na implementação de uma simulação em um sistema distribuído é como garantir que os eventos da simulação sejam executados na ordem correta, uma vez que cada processo lógico possui um conjunto de eventos que, como um todo, devem ser executados em ordem cronológica de acontecimentos. Ou seja, deve-se garantir que um evento em um processo lógico com um tempo t qualquer, seja executado antes de um evento (supondo que o evento t modifica o evento u) com um tempo u (onde $u>t$) em um outro processo lógico.

Uma abordagem é utilizar troca de mensagem entre os processos, onde cada processo lógico possui um relógio local, executando a simulação localmente. Quando necessário, ele se comunica com os outros processos lógicos por meio de troca de mensagens.

Porém, um dos pontos principais em uma simulação é que os eventos que são gerados devem ser executados na ordem para que se tenha uma simulação correta. Esse fator torna-se um problema na simulação distribuída uma vez que cada processo lógico receberá parte dessa lista de eventos, o que ocasiona uma fragmentação da lista e a necessidade de sincronização entre os processos lógicos para que os eventos sejam executados na ordem correta.

Dessa forma, é possível observar que um *pl* pode simular as ações de um *pf* até um momento *t* se *pl* tiver conhecimento do estado inicial e de todas as mensagens que o correspondente *pf* recebe até o momento *t*. Nenhuma mensagem futura, isto é, recebida após *t,* poderá afetar o comportamento de *pf* no tempo *t*. Caso isso ocorra, está sendo violada a propriedade de causa e efeito, isto é, o futuro está interferindo no passado.

Esse conceito pode ser exemplificado em um servidor não preempetivo com política de atendimento FCFS (*First Come, First Served* – primeiro a chegar, primeiro a ser servido). Supondo que o servidor está vazio e no instante *t* chega um novo cliente para ser servido. Nesse ponto, o servidor irá simular o evento que acabou de chegar e é possível afirmar que até o tempo *t+Tempo_de_serviço* o servidor não irá produzir nenhum evento. Apenas no tempo *t+Tempo_de_serviço* o servidor irá produzir uma mensagem para o próximo *pl* a que se destina o serviço concluído.

A partir dessas observações, pode-se construir um algoritmo para simulação distribuída. Para reproduzir os acontecimentos presentes nos *pf's*, quando um pf_i envia uma mensagem *m* para um pf_j em um tempo *t*, o pl_i correspondente deve enviar uma mensagem *(m, t)* no tempo *t* para o pl_j.

Outro conceito a ser introduzido é o valor do *clock* do canal (*vcc*), que é o valor *t* da última mensagem recebida pelo canal. Caso nenhuma mensagem tenha sido recebida pelo canal, o valor do *vcc* é zero.

Um pl_i conhece todas as mensagens recebidas pelo seu correspondente pf_i até o tempo:

$$T_i = mín \{t_j\}$$

onde t_j é o valor dos *vcc*'s dos canais que chegam em pl_i. Dessa forma, T_i é o valor mínimo de todos os *vcc*'s, e a esse valor dá-se o nome de *clock* do processo lógico *i*.

Supondo a situação mostrada na Figura 3.1. Nesse exemplo, chegam ao pl_i duas mensagens dos dois canais de entrada. Os valores dos *clocks* dos canais de comunicação são 14 e 12. Dessa forma, o valor mínimo é 12, que é o valor T_i, ou *clock* do pl_i. Consequentemente, isso faz com que o pl_i simule pf_i até o tempo 12 (T_i) de forma segura. Ou seja, todos os eventos com tempo menor que 12 podem ser executados de forma segura.

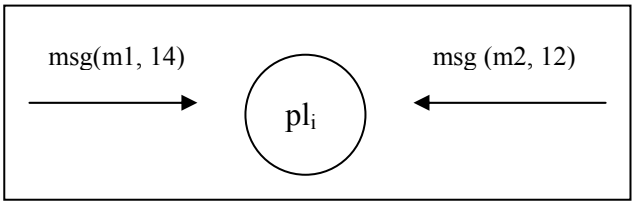


Figura 3.1: *Clock* do processo lógico

O algoritmo descrito faz a sincronização das mensagens de uma simulação distribuída de uma forma simplificada. Em aplicações reais outras questões tornam-se relevantes, onde os protocolos de sincronização precisam possuir mecanismos que permitam a execução da simulação. Porém, de forma geral, os protocolos existentes implementam os princípios básicos do algoritmo descrito. A maneira com que o protocolo reage frente ao problema da propriedade de causa e efeito faz com que sejam classificados em otimistas ou conservadores.

O representante mais conhecido dos protocolos conservadores é o CMB (Chandy & Misra, 1979, Chandy & Misra, 1981) e o representante mais conhecido dos protocolos otimistas é o *Time Warp* (Fujimoto, 1990, Fujimoto, 2000, Jefferson, 1985). A seguir são descritas as implementações desses dois protocolos.

3.2.1 Protocolo CMB

Os protocolos conservadores avançam a simulação apenas quando há garantia de que não irá chegar nenhuma mensagem com tempo inferior aquela que está sendo executada. Essa característica faz com que não ocorram erros de causa e efeito, sendo

essa a principal característica desse protocolo.

Nesses protocolos, a capacidade de *lookahead* do modelo (a capacidade de prever o comportamento futuro e determinar a ocorrência dos eventos) constitui o elemento chave na determinação do desempenho do algoritmo. Em função da importância da determinação do *lookahead* para o desempenho da simulação conservadora, muitos protocolos têm sido desenvolvidos para auxiliar no processo de determinação do *lookahead* (Bagrodia et al., 1998; Fujimoto, 2000; Nicol, 1996).

Com o intuito de facilitar a comparação entre as estruturas utilizadas nos dois protocolos de sincronização é apresentada na Figura 3.2 e na Figura 3.3 a estrutura lógica de cada um dos protocolos.

No protocolo CMB as mensagens são recebidas por meio de canais de comunicação fixos (topologia estática). Para cada canal (ou porta) de chegada de mensagens existe uma estrutura de dados (*IB[i]- Input Buffer*) responsável pelo armazenamento das mensagens que chegam ao processo (Figura 3.2). Da mesma forma as estruturas *OB[i]* (*Output Buffer)* encarregam-se de armazenar as mensagens que deverão ser enviadas a outros processos. As mensagens recebidas são armazenadas por ordem do tempo de ocorrência e o valor do menor tempo de ocorrência é armazenado em uma variável chamada *clock do canal* (*CC[i] – Channel Clock*). A cada ciclo da simulação a mensagem correspondente ao menor valor entre os CC[i] é inserida na lista de eventos futuros (*EVL – Event List*). Para facilitar a identificação do menor valor entre os *CC[i]'s* este valor é calculado e armazenado na variável LVTH (*Local Virtual Time Horizon*).

A função da lista de eventos futuros é armazenar os eventos que serão processados em ordem do seu tempo de ocorrência. Cada evento (ou mensagem) retirado da EVL corresponde a um avanço no relógio do processo (*LVT – Local Virtual Time)*.

A estrutura de dados S (*Status)* armazena as informações estatísticas relacionadas ao contexto do processo lógico. São esses dados que serão utilizados para calcular os resultados que serão apresentados ao usuário quando da simulação do modelo.

Durante a execução da simulação os processos envolvidos no modelo podem, por um período indeterminado de tempo, não possuir mensagens a serem enviadas. Esta condição é suficiente para que ocorra um *deadlock* e assim a simulação fique bloqueada. Para evitar essa situação o protocolo CMB pode se utilizar do mecanismo de *mensagens nulas*. O envio de uma mensagem nula por um processo CMB indica a ausência de processamento por um período de tempo. Desta forma a marca de tempo da mensagem nula irá causar uma atualização no valor do processo lógico que a receberá, permitindo desta forma o avanço do tempo da simulação.

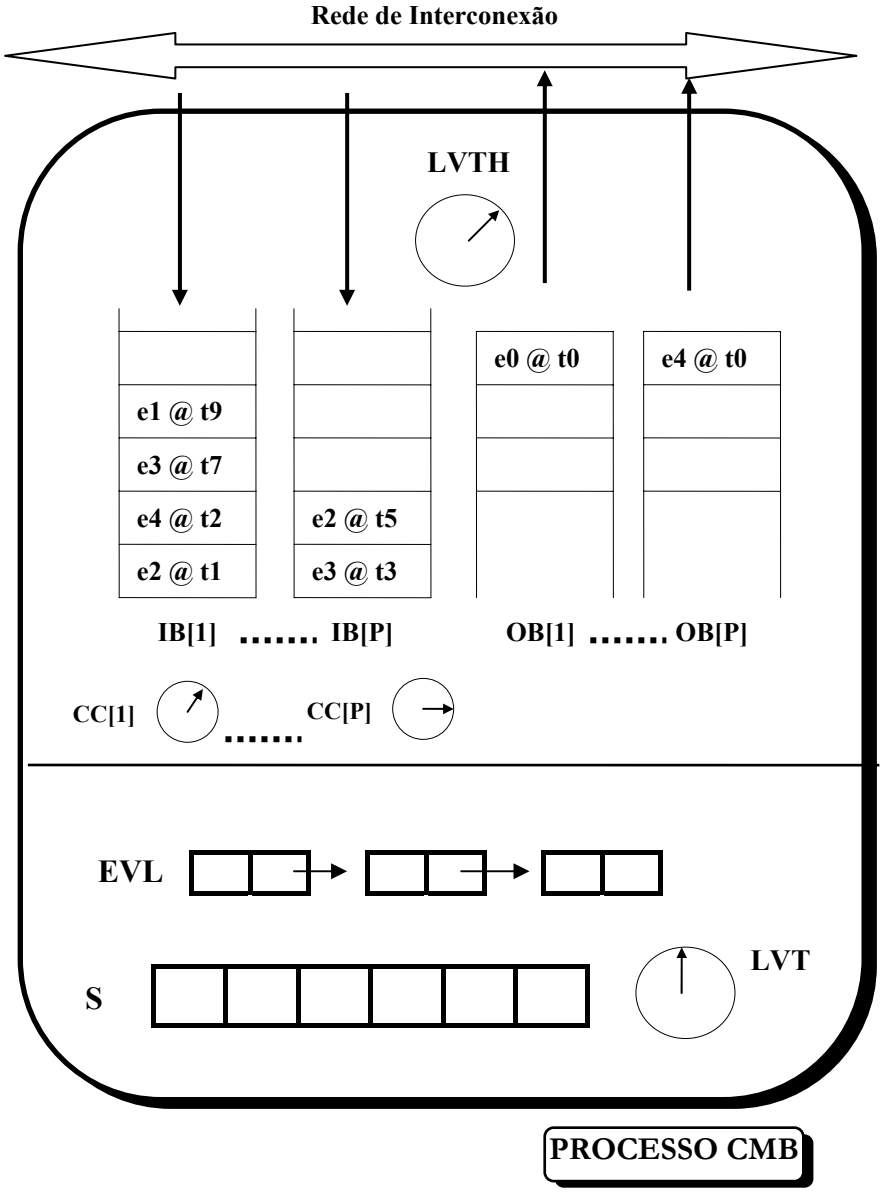


Figura 3.2: Processo Lógico CMB

3.2.2 Protocolo *Time Warp*

No protocolo de sincronização *Time Warp* existe a possibilidade de erros de causa e efeito na execução dos eventos. O conceito otimista supõe que nenhuma mensagem com tempo de ocorrência menor que o LVT (*Local Virtual Time*) será recebida. Entretanto quando essa situação ocorre, um erro de causa e efeito é detectado e o processo lógico restaura um estado anterior (consistente) desfazendo assim as alterações provocadas pela mensagem prematura (mecanismo de *rollback).* Após isso, os eventos são executados na ordem correta de tempo de ocorrência.

No protocolo *Time Warp,* o menor valor entre os tempos de ocorrência das mensagens processadas é chamado de *Global Virtual Time* (*GVT*). Essa marca de tempo garante que nenhum evento com tempo de ocorrência menor que GVT deverá ser executado novamente, garantindo que todo o processamento até esse período está correto.

As estruturas IB (*Input Buffer*) e OB (*Output Buffer*) são responsáveis pelo recebimento e envio das mensagens através da rede de comunicação.

O objetivo do armazenamento temporário das mensagens pelas estruturas *IQ* (*Input Queue*) e *OQ* (*Output Queue*) é corrigir um possível engano causado pelo processamento de uma mensagem fora da ordem cronológica. Essas estruturas armazenam os valores de estados do processo e servem para permitir o processo de *rollback*.

A estrutura *SS* permite que a simulação possa ser restaurada a um estado consistente caso haja um erro de causalidade. O processo de restauração de um estado consistente também envolve a atualização da estrutura S (*Status)* que mantém dados estatísticos referentes a simulação do modelo da aplicação.

No *Time Warp*, para cada mensagem existe uma *antimensagem* cujo conteúdo é idêntico com exceção ao campo de *sinal* (mensagem com sinal + e antimensagem com sinal -) (Jefferson, 1985). Uma antimensagem deve ser enviada em caso de *rollback* e por isso, a mensagem referente a antimensagem deve ser desfeita, pois violou a princípio de causa e efeito.

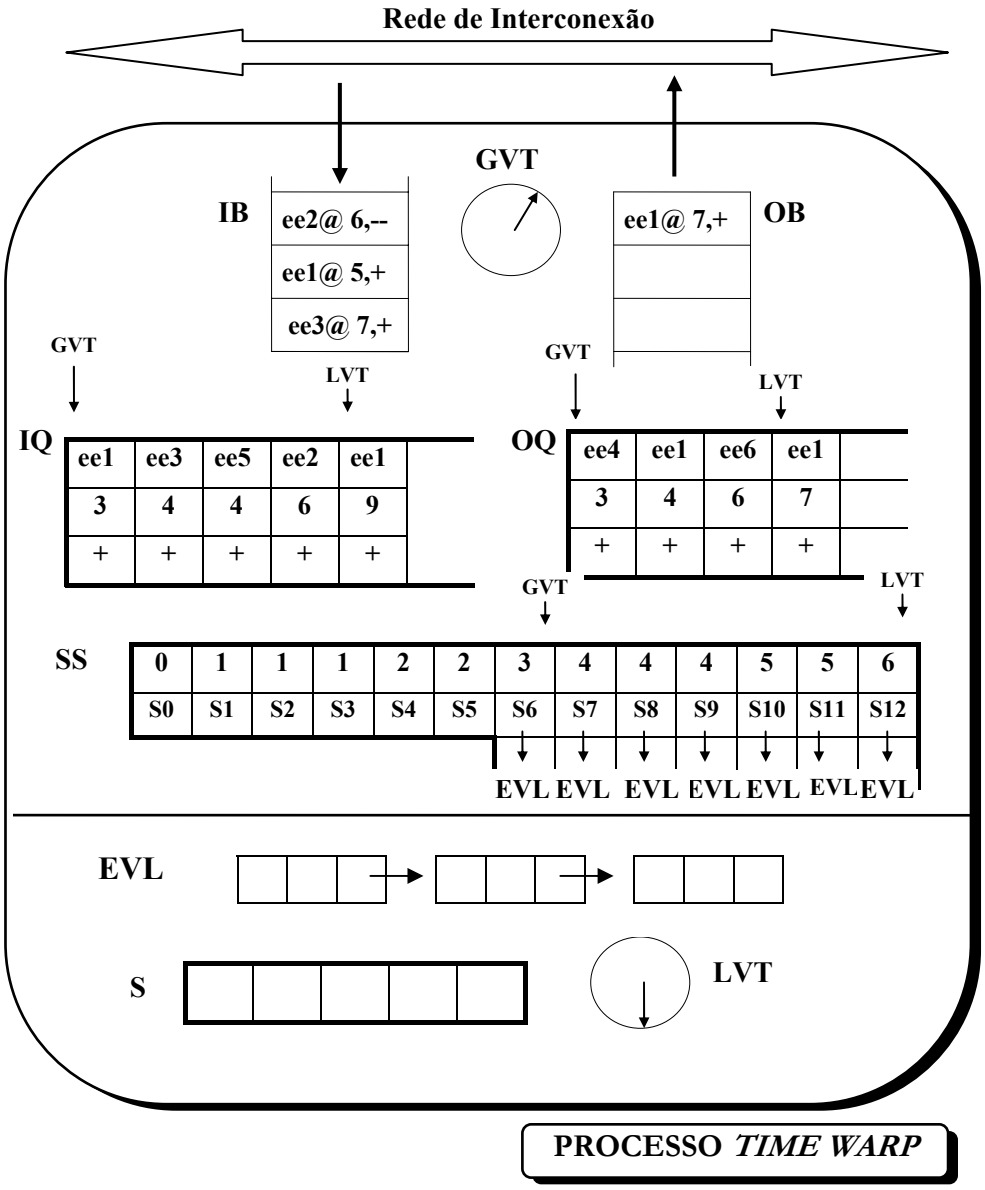


Figura 3.3: Um processo lógico *Time Warp*

A descrição do mecanismo de troca, assim como as alterações necessárias em cada um dos protocolos participantes (CMB e *Time Warp*) são descritas nas próximas seções.

3.3 Mecanismo de troca de protocolos de sincronização para simulação distribuída

Durante a execução da simulação distribuída, independente de qual protocolo esteja sendo utilizado, é necessário que as informações relativas às variáveis que medem

o desempenho da simulação sejam obtidas e analisadas para que se possa decidir se existe a necessidade da troca de protocolos.

Para que isto seja possível, como já dito anteriormente, foram definidos três processos lógicos, cuja função desempenhada por cada um está relacionada apenas com o mecanismo de troca de protocolos de sincronização não tendo nenhuma participação no contexto da simulação do modelo em questão. As funções de cada um destes três processos adicionais são descritas a seguir.

1. *Processo observador.*

Para que o momento da troca de protocolos de sincronização seja identificado é necessário reunir informações a respeito do desempenho da simulação de cada um dos processos lógicos, que estão envolvidos na simulação distribuída. Por exemplo: a quantidade de mensagens nulas e completas transmitidas ou recebidas, a porcentagem de mensagens bloqueadas quando atingem a cabeça da lista de eventos futuros, os valores dos LVT's de cada um dos processos, o número de *rollbacks* e o número de anti-mensagens. É tarefa do processo observador reunir essas informações para que seja possível definir o desempenho da simulação e com isso, a necessidade ou não da troca de protocolos. O processo Observador deve estar sendo executado concorrentemente com cada processo lógico e, em tempos pré-programados, deverá enviar estes valores para o processo Gerenciador.

2. *Processo gerenciador*

O processo Gerenciador tem a função de receber essas informações dos processos Observadores e por meio da análise dos dados, identificar o comportamento do protocolo utilizado, decidindo pela realização ou não da troca de protocolos.

3. *Processo conversor*

O processo Conversor é responsável pela verificação da consistência na simulação e pela manipulação das estruturas de dados e variáveis de tempo dos processos lógicos no momento da troca de protocolos. Esse processo deve ser ativado

concorrentemente com cada processo lógico participante da simulação, no momento da conversão entre protocolos.

A consistência deve ser verificada para que a simulação entre em um estado a partir do qual a conversão de protocolos pode ser considerada. Esta necessidade é uma conseqüência da diferença existente entre os paradigmas (conservador e otimista) adotados pelos protocolos CMB e *Time Warp* respectivamente. Um exemplo de inconsistência é quando o *Time Warp* está sendo utilizado e as anti-mensagens precisam ser eliminadas e os LVT's igualados.

A conversão das estruturas de dados refere-se às modificações dos dados responsáveis pelo armazenamento das mensagens, utilização da variável GVT, etc.

A Figura 3.4 mostra a estrutura dos processos envolvidos no sistema de simulação utilizando a troca dinâmica de protocolos.

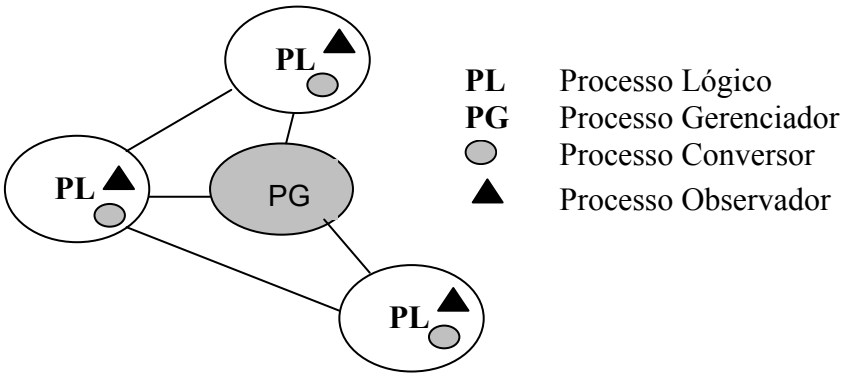
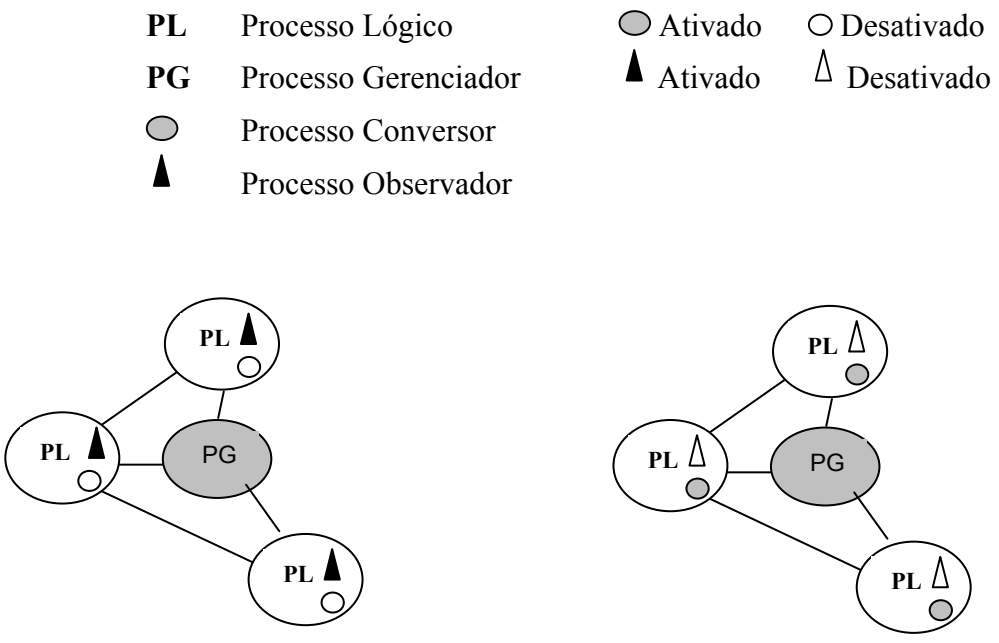


Figura 3.4: Estrutura dos processos envolvidos no sistema de simulação utilizando a troca dinâmica de protocolos.

Dessa forma, deverão estar presentes, em cada processo lógico, uma instância do processo Conversor e uma instância do processo Observador, durante a execução da simulação. O processo Observador está sempre ativo. No momento em que uma conversão é necessária, o processo Conversor é ativado (Figura 3.5).

As próximas seções detalham o processo Conversor, considerando a garantia de consistência (seção 3.4) e o mecanismo para conversão das estruturas de dados (seção 3.5).



a) Durante a simulação

b) Durante a troca

Figura 3.5: Mecanismo de ativação dos processos envolvidos no Mecanismo de Troca.

3.4 Garantia da consistência

O mecanismo de troca de protocolos de sincronização deve prever a mudança de protocolo apenas quando a simulação estiver em um estado consistente. Entende-se por estado consistente a impossibilidade de ocorrência de um processo de recuperação de estados (*rollback- Time Warp*) ou bloqueio (*deadlock* – CMB).

- **Garantia de consistência na conversão CMB para *Time Warp***

Uma simulação distribuída sincronizada por um protocolo conservador estará consistente caso não esteja bloqueada, desta forma a utilização do mecanismo de mensagens nulas garante a consistência de uma simulação distribuída sincronizada pelo protocolo CMB em qualquer momento durante sua execução.

Essa garantia fornecida pelo mecanismo de mensagens nulas possibilita a troca de CMB para *Time Warp* em qualquer instante da simulação.

- **Garantia de consistência na conversão *Time Warp* para CMB**

Ao contrário do que ocorre com uma simulação distribuída sincronizada pelo protocolo CMB, onde as mensagens nulas oferecem a garantia de consistência dos processos durante toda a simulação, o paradigma otimista adotado pelo protocolo *Time Warp* não garante que, em um instante qualquer, a simulação esteja consistente. Um protocolo otimista considera a execução de todas as mensagens independentemente dos seus valores de tempo de ocorrência. Partindo deste princípio, a execução de uma mensagem cronologicamente incorreta deve disparar um processo de reexecução (*rollback*) de todas as mensagens recebidas com tempo de ocorrência menor que o tempo de ocorrência da mensagem defasada. A possibilidade de ocorrer a reexecução das mensagens impede que o processo de troca de protocolos seja realizado.

Dessa forma duas maneiras podem ser consideradas para a garantia de consistência da troca de protocolos quando o *Time Warp* está em execução. A primeira é avançar a simulação até que todos os *pl*s atinjam o maior valor de LVT dentre todos os *pl*s. E a outra forma é retroagir a simulação dos *pl*s para o último valor de GVT.

Para a implementação da primeira abordagem, as diversas instâncias do processo Observador devem incluir o LVT de seu processo lógico no envio das informações para o processo Gerenciador. Quando é detectada a necessidade de conversão do protocolo *Time Warp* para o protocolo CMB, o processo Gerenciador ativa o processo Conversor e informa o maior valor de LVT recebido. Dessa forma, o processo Conversor permite que o processo lógico execute até seu LVT atingir o valor recebido do processo Gerenciador para só então bloquear o processo lógico.

A desvantagem dessa abordagem é, além da subutilização do processo lógico que possui o maior LVT (causado pela espera de que os outros processos lógicos atinjam sua marca), que os outros processos lógicos teriam que aguardar as chegadas das mensagens futuras para que seus LVT's fossem incrementados até o maior LVT. Neste período de espera três situações podem ocorrer:

- A chegada de uma ou mais mensagens fora da ordem cronológica e ocorrer um ou mais *rollbacks*;
- Uma demora muito grande para as chegadas das mensagens que atualizam os LVT's;

- Como o protocolo *Time Warp* é otimista, existe a possibilidade de que um desses processos lógicos, que deveriam avançar seus LVT's, necessite de um determinado evento que o processo lógico que possui o maior LVT deveria causar, mas este estaria parado, aguardando que todos os *pl*'s atingissem as suas marcas de LVT para que ocorresse a troca.

A ocorrência de uma ou mais dessas situações provocaria uma sobrecarga de processamento muito grande prejudicando o desempenho do mecanismo de troca. Por outro lado, a vantagem desta abordagem é que não seria necessário desfazer a execução de eventos já executados, como é o caso da segunda abordagem, que é explicada a seguir.

A segunda maneira de garantir a consistência da simulação parte da própria definição do GVT. O GVT é a variável de tempo cujo valor é o menor entre todos os LVTs da simulação. Sendo assim a realização de um *rollback* intencional de todos os processos *Time Warp* até a marca de tempo fornecida pelo GVT garante que a simulação está em um estado consistente.

Para realizar a implementação desta abordagem, uma vez que cada processo lógico da simulação possui o valor do GVT, seria necessário apenas que os processos Conversores, ao receberem a ordem de conversão do protocolo *Time Warp* para o protocolo CMB, transmitissem uma antimensagem com o valor do GVT. Desta forma, o próprio mecanismo de *rollback* do *Time Warp* se encarregaria de levar todos os processos lógicos até o valor do GVT.

Dependendo do estado geral da simulação, a sobrecarga de processamento exigida pelos *rollbacks* realizados pelos processos otimistas pode, em função de um baixo desempenho da simulação apresentada pelo protocolo *Time Warp* e/ou de uma curta diferença entre o valor do GVT em relação aos LVT's, não interferir no desempenho final da simulação como interferiria a primeira opção discutida anteriormente.

Desta forma a realização intencional dos *rollbacks* para todos os processos lógicos até o valor do GVT para garantir o estado consistente e permitir a realização da troca de *Time Warp* para CMB parece ser mais adequada.

3.5 Mecanismo de conversão

Para cada classe de protocolos existe um conjunto de características que devem ser consideradas no momento da troca de protocolo. E desta forma três pontos devem ser tratados na conversão:

a) As diferenças entre as estruturas de dados necessárias nos dois protocolos considerados (seção 3.5.1);

b) A adaptação da topologia lógica da rede de comunicação utilizada pelos protocolos de sincronização (seção 3.5.2);

c) A necessidade da compatibilização das variáveis de tempo empregadas para manter a ordem cronológica da execução dos eventos (seção 3.5.3);

3.5.1 Estruturas de dados

As estruturas de dados utilizadas nas abordagens otimista e conservadora de simulação são diferentes, devido ao funcionamento típico de cada uma delas. Para cada protocolo de sincronização existe um conjunto de variáveis e estruturas de dados diferentes, ou seja, apesar de algumas variáveis fazerem parte dos dois protocolos, existe um outro conjunto que faz parte de um determinado protocolo apenas.

A troca de protocolos durante a execução da simulação exige que estruturas que não existiam anteriormente em um processo lógico passem a fazer parte deste processo. Para solucionar esse problema foi proposto o emprego de um mecanismo complementar de utilização de estruturas de dados (Morselli 2000). Isto é, a maior parte das estruturas de dados foi reunida em apenas um conjunto. Desta forma, um processo *Time Warp*, por exemplo, irá conter além de suas estruturas de dados originais, as estruturas de dados de um processo lógico CMB. Estas estruturas deverão ser declaradas inicialmente e ativadas (utilizadas) em tempo de execução no momento da troca de protocolos. Este mecanismo não requer uma utilização substancial de memória uma vez que a maioria das estruturas de dados não utilizada durante a execução da simulação estará desativada. Os dados que irão fornecer o valor inicial destas estruturas podem fazer parte de outras

estruturas utilizadas pelo protocolo de origem ou, em caso contrário, são mantidos especificamente para a troca de protocolos.

3.5.2 Canais de comunicação

O protocolo de sincronização CMB clássico, proposto por Chandy e Misra, não prevê um esquema de suporte para alocação dinâmica de canais de comunicação. Estudos posteriores mostram que a conectividade dinâmica também é possível para os protocolos conservadores (embora apresentando restrições) (Jha & Bagrodia, 1994).

O protocolo CMB não prevê a necessidade de incluir nas mensagens os identificadores de processo fonte e processo destino. A identificação dos processos é obtida por meio do canal pelo qual a mensagem está sendo recebida ou enviada. Este tipo de identificação, permitida pela topologia estática, não está incluída no protocolo *Time Warp* (topologia dinâmica). No *Time Warp* a identificação dos processos é necessária para que o mecanismo de antimensagens seja efetuado.

Para solucionar esta incompatibilidade, o mecanismo de troca de protocolos de sincronização utiliza uma estrutura de mensagens, que permite a inclusão do identificador de processos. Dessa forma, os dois protocolos poderão gerenciar o mecanismo de alocação de canais seguindo seus projetos originais.

3.5.3 Variáveis de tempo

Os protocolos de sincronização utilizam variáveis para armazenar os valores de tempo que controlam as execuções dos eventos da simulação em ordem cronológica e para representar o avanço do tempo real do modelo simulado. As variáveis utilizadas pelos protocolos CMB e *Time Warp* estão representadas na Figura 3.6.

Estruturas de Tempo *Clock Channel* (CC[i])

As estruturas de dados IB[i] (*Input Buffer*) do protocolo CMB mantém as mensagens que foram recebidas ordenadas pelo seu tempo de ocorrência na simulação.

Desta forma, a mensagem que ocupa a primeira posição da fila é aquela que possui o menor valor do tempo de ocorrência, recebida por aquele canal de comunicação. Este valor é armazenado em uma das estruturas CC[i].

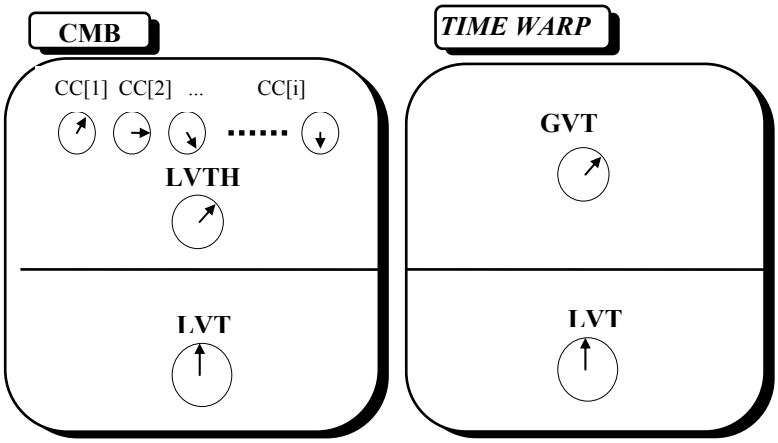


Figura 3.6: Variáveis de Tempo Utilizadas nos Protocolos CMB e *Time Warp*.

Para que o controle local do processo possa decidir qual mensagem deverá ser escalonada (inserida na lista de eventos futuros) é escolhida a mensagem com menor valor entre as estruturas CC[i].

As estruturas CC[i] não são utilizadas pelo protocolo *Time Warp*. Isso ocorre porque no *Time Warp* as mensagens recebidas por um processo lógico são inseridas em apenas uma estrutura (IQ). Assim, basta ao controle local do processo *Time Warp* escalonar a mensagem que ocupa a primeira posição da estrutura IQ.

Desta forma as estruturas CC[i] não são necessárias no modo *Time Warp.*

Variável de Tempo *Local Virtual Time Horizon* (LVTH)

A variável LVTH, utilizada exclusivamente pelo protocolo CMB, armazena o menor valor entre as mensagens que estão nas portas de entrada (CC[i]), para cada canal de comunicação mencionado no item anterior. O evento que possui esta marca de tempo deverá ser escalonado na lista de eventos futuros. Da mesma forma que a estrutura CC, a variável LVTH só tem significado no protocolo CMB.

Variável de Tempo *Local Virtual Time* (LVT)

A variável LVT, tanto para o protocolo CMB como para o protocolo *Time Warp* corresponde a marca de tempo do último evento processado naquele processo lógico. Este valor, denominado *clock do processo*, possui uma grande importância, pois representa a quantidade de tempo virtual que o processo lógico simulou o processo físico correspondente. Além disto, é por meio do valor de LVT que se determina quando uma mensagem está fora da sua ordem cronológica.

Variável de Tempo *Global Virtual Time* (GVT)

Pode-se afirmar que um sistema de simulação simulou o comportamento de um sistema real até o valor de tempo armazenado no GVT. Este valor é obtido por meio do cálculo do menor LVT entre todos os processos do sistema de simulação.

No protocolo *Time Warp* o *fossil collection* (eliminação dos estados da simulação armazenados para a necessidade de efetuar um *rollback*) é realizada por meio do valor do GVT disponível no processo lógico, ou seja, qualquer alteração efetuada no estado da simulação antes do valor de GVT é considerada estável.

O protocolo CMB, ao contrário, não prevê a existência deste valor armazenado em cada processo lógico.

Sendo assim, para que os processos lógicos *Time Warp* tenham acesso ao valor do GVT logo após a troca de CMB para *Time Warp*, o processo Gerenciador deverá calcular, por meio dos valores dos LVT's enviados pelos processos Observadores, o menor LVT e enviá-lo aos processos Conversores. No sentido oposto, ou seja, a troca do *Time Warp* pelo CMB, as variáveis GVT são desativadas.

3.6 Trabalhos relacionados a avaliação de desempenho de simulação distribuída.

Buscou-se na literatura, o estado da arte nos tópicos levantados nesta tese para embasar o encaminhamento científico deste trabalho. Nesse sentido, foram pesquisados os trabalhos relacionados à avaliação de desempenho dos protocolos de simulação distribuída, sistemas de simulação que utilizem as abordagens conservador e otimista

simultaneamente além das pesquisas que tem sido efetuadas na área de simulação distribuída.

Diversos fatores contribuem para que se obtenham bons *speedups* em uma simulação distribuída. Porém, é visível que modelos mais simples com baixa granulosidade poderão, na maior parte dos casos, ter desempenho melhor com a simulação seqüencial. Mesmo a utilização do protocolo otimista traz, em geral, vantagens para modelos mais complexos e com granulosidade mais grossa (Teo & Tay, 1999, Xu & Chung, 2004).

Alguns fatores que afetam diretamente uma simulação distribuída são: o tamanho do problema, balanceamento de carga, granulosidade, sobrecarga da comunicação e particionamento do modelo.

Diversos trabalhos analisam o desempenho dos protocolos de sincronização distribuída. Pode-se citar, por exemplo, o trabalho de Xu e Chung (2004) onde foi desenvolvido um modelo que tem por objetivo prever o desempenho da simulação síncrona por meio da inserção dos fatores mais comuns que afetam a simulação. Nesse modelo são inseridas as características da aplicação e da plataforma onde será feita a simulação e o sistema retorna ao usuário qual será o máximo *speedup* alcançado pela aplicação.

Já o trabalho de Teo e Tay em (Teo & Tay, 1999) mostra um *framework* escalável para simulação paralela que suporta a modelagem de simulação paralela por meio de chamadas a bibliotecas de funções. Esse artigo faz a análise de desempenho desse *framework* de forma a mostrar que se pode obter bons *speedups* com a utilização de modelos de simulação grandes e escaláveis.

Em outra pesquisa, Teo et al (1999) mostra um *framework* que permite a avaliação de desempenho de um sistema por meio da extração de características desse modelo. Ele permite que desenvolvedores verifiquem o potencial desempenho que uma simulação pode ter antes de fazer a sua implementação. A métrica proposta é independente da plataforma e mede apenas o paralelismo de eventos inerente ao modelo de simulação. Ele dá resultados gerais e indicativos se uma determinada aplicação tem tendências de obter bom desempenho com uma simulação distribuída.

O artigo de Teo et al (2002) mostra uma avaliação de desempenho de uma simulação conservadora utilizando memória distribuída compartilhada. Eles concluem

que o desempenho da simulação é altamente dependente da sobrecarga da sincronização e do custo de comunicação entre os processos lógicos.

Bagrodia et al (1999) mostra o uso do COMPASS, que é uma ferramenta que faz a previsão do desempenho de uma simulação. Eles utilizaram diversos modelos de aplicações reais e alguns *benchmarks* para verificar os efeitos da escalabilidade, da sensibilidade à latência na comunicação e da interferência existente entre fatores como padrão da comunicação e sistema de *caching* de arquivo no desempenho da simulação.

Em outro artigo, Bagrodia & Takai (2000) avaliam o desempenho de diversos algoritmos de sincronização conservadores. Entre os algoritmos analisados tem-se o algoritmo de sincronização assíncrono de mensagens nulas, o algoritmo síncrono de eventos condicional, e o algoritmo híbrido chamado aceleração das mensagens nulas que combinam as características dos dois primeiros algoritmos. Esses algoritmos foram analisados sob a ótica de diversas características do modelo como a conectividade do modelo, granulosidade, balanceamento de carga e *lookahead*.

Em relação a pesquisas sobre protocolos, os trabalhos encontrados na literatura relacionam os dois protocolos em sistemas híbridos, onde os protocolos conservadores e otimistas estão combinados dentro do mesmo protocolo, ou seja, são extraídas algumas características dos protocolos conservadores e algumas dos protocolos otimistas. Boukerche et al (2001) combina os mecanismos conservadores e otimistas em um *framework*, o SWiMNet. Esse *framework* permite a modelagem de redes móveis e sem fio de forma detalhada e realista. Parte deste sistema possui características dos protocolos conservadores e parte dos protocolos otimistas.

Das (1996) propõe a utilização de protocolos adaptativos onde os processos lógicos devem ter a habilidade de modificar dinamicamente um ou mais parâmetros de controle de acordo com o conhecimento de aspectos do estado da simulação.

O trabalho de Jha (1996) mostra um framework que possui um mecanismo de controle global que avança a simulação dos processos, que pode ser conservador ou otimista, de acordo com o desempenho apresentado pela simulação.

O artigo de Rajaei et al (1993) propõe um método de sincronização que combina o *Time Warp* com a janela de tempo do protocolo conservador conseguindo com isso uma diminuição na cascata de *rollbacks*.

Kalanterry (2004) propõe o uso de comunicação orientada a conexão (característica de protocolos conservadores) para o protocolo otimista. Os testes realizados pelo autor mostram melhoras no desempenho da simulação, reduzindo a sobrecarga de *rollbacks*.

Perrumalla (2005) propõe a definição de um micro-kernel que serve de base para a simulação distribuída, onde os processos lógicos podem ser conservadores, otimistas, otimista baseado em computação reversa dentre outros. Por exemplo, quando os processos lógicos que compõe a simulação, e que estão executando com o protocolo conservador, ficam bloqueados, a abordagem faz com que a simulação seja executada com o protocolo otimista. Nesse caso, o esquema de processamento de eventos pode ser trocado durante a execução da simulação.

A proposta do mecanismo apresentado nesta tese propõe a execução simultânea dos dois protocolos de sincronização, como a proposta de Jha e de Perrumalla, porém, explorando o controle local que deve estar preparado para avançar com a simulação. A proposta de Jha avança a simulação de acordo com os comandos de um controle global que diz para os processos lógicos quando eles podem avançar na simulação no caso conservador, e quando é necessário desfazer uma execução errônea no caso otimista. A proposta de Perrumalla leva em conta, por exemplo, o momento em que a simulação está bloqueada (no caso conservador) para continuar a simulação com o protocolo otimista, não há uma coleta de dados para avaliar o desempenho da simulação. A proposta apresentada aqui mostra uma avaliação de desempenho local a cada processo lógico e simulação simultânea com os processos lógicos avançando na simulação de acordo com o tipo de mensagem que recebe dos demais processos lógicos. Essa proposta modifica a estrutura dos protocolos originais de forma a adequá-los para o mecanismo de troca de protocolo.

As pesquisas na área de simulação distribuída se diversificam, porém todas têm o propósito de obter melhor desempenho por meio da modificação dos protocolos ou proposta de novos protocolos. Nesse sentido, a pesquisa de Lee et al (2005) mostra o projeto de um sistema em cluster por satélite. É mostrada a viabilidade desse sistema e algumas análises de desempenho realizadas.

Robinson (2005) discute o uso adequado da simulação distribuída. Nesse contexto são analisadas quais são as potencias aplicações para a simulação distribuída, além da existência de uma possível contradição entre a simulação distribuída e a prática

da boa modelagem, ou seja, a habilidade em desenvolver modelos grandes e complexos (mais adequados a simulação distribuída) e a recomendação do desenvolvimento de modelos simples. Essas discussões levam o autor a três conclusões: não são todos que necessitam da simulação distribuída (a simulação seqüencial resolve a maior parte dos problemas), a simulação distribuída ainda não está acessível a maior parte da comunidade que possa estar interessada no seu uso devido a grande quantidade de conhecimento necessário, e as possibilidades da simulação distribuída podem fazer com que modelos sejam mais complexos sem necessidade (apenas porque é possível a sua utilização) o que mostra uma necessidade de treinamento dos usuários de simulação.

Jeong et al (2005) mostra uma simulação de redes de telecomunicações no ambiente MRIP. Esse artigo mostra as vantagens do uso dessa abordagem para a redução do tempo de simulação.

Chen & Szymanski (2002) propõe uma nova forma de explorar o paralelismo em uma simulação por meio de um conceito que eles chamaram de *lookback*. *Lookback* é a habilidade que um processo lógico tem em mudar o seu passado local sem envolver outros processos lógicos. Essa habilidade não é encontrada em todos os processos lógicos, porém, naqueles em que existe, eventos podem ser processados fora da ordem de *timestamp*, permitindo novos tipos de protocolos de sincronização.

O artigo de Li & Tropper (2004) propõe o uso de reconstrução de eventos ao invés de *checkpoint* no *Time Warp*. A reconstrução de eventos significa o uso da diferença entre estados adjacentes ao invés do salvamento de eventos em uma fila (*checkpoint*). Os experimentos realizados indicam que para simulações com granulosidade fina e tamanho de estados pequeno, a reconstrução de eventos consegue uma melhora no tempo de execução e promove uma diminuição na utilização de memória.

A pesquisa de Hybinette (2004) propõe o uso de uma nova abordagem para a técnica de clonagem em simulação distribuída. A clonagem é uma técnica onde são executadas diversas simulações paralelas similares onde muitas computações são compartilhadas ao invés de duplicadas. A simulação é clonada quando o evento que chega está relacionado a apenas um conjunto de processos lógicos em particular, e não a todos eles. A proposta apresentada nesse artigo é a clonagem *just-in-time* onde se

procura melhorar o desempenho da simulação permitindo que a clonagem seja adiada por meio do relaxamento das regras para a geração de clones.

Em outra pesquisa, Agarwal et al (2005) propõe o reverso da clonagem, ou seja, o casamento dos processos que foram clonados, quando se verifica que os resultados estão convergindo e assim, os clones dariam resultados iguais.

O trabalho de Zeng et al (2004) propõe um novo esquema de cancelamento de eventos no *Time Warp*. Esse esquema, chamado "*batch based*" sugere o uso de um vetor de estados que contém a dependência dos eventos. Esse esquema, da forma como foi proposto, permite que a ocorrência de mensagens *straggler* conduza a apenas um *rollback*, ou seja, um *rollback* ótimo. Os testes efetuados mostraram que o esquema proposto teve sucesso na redução do número de antimensagens e na melhora no desempenho da simulação.

O artigo de Verbraeck (2004) propõe o uso da teoria de desenvolvimento baseado em componentes da engenharia de software para preparar um modelo de simulação para sua simulação distribuída, permitindo também que partes do modelo possam ser reutilizados. Os resultados dos experimentos conduzidos pelo autor mostram que o uso de componentes pode trazer vantagens, porém, a criação de modelos em uma forma baseada em componentes não é tarefa fácil e os métodos de modelagem de simulação corrente precisariam ser adaptados.

O trabalho de Quaglia & Beraldi (2004) propõe o uso dos conceitos de incerteza temporal no protocolo de sincronização otimista. Esse conceito é a possibilidade de um evento ocorrer em um intervalo do tempo da simulação ao invés de ser em um instante específico. No artigo, os autores propõem o uso desse conceito como a possibilidade de um evento da simulação ocorrer em um dos pontos do conjunto do espaço do sistema de simulação. Os testes realizados mostraram que o uso desse conceito permite uma maior flexibilidade na sincronização com um conseqüente melhoramento no comportamento do tempo de execução do sistema de simulação.

O artigo de Park et al (2004) descreve uma simulação de redes de computadores de grande escala utilizando sincronização conservadora. No artigo, os autores comparam o desempenho e a escalabilidade de algoritmos síncronos e assíncronos, para isso, desenvolveram um modelo que contempla os algoritmos mais comuns da simulação conservadora. Os resultados mostraram que máquinas paralelas contendo centenas de processadores simulando modelos de rede de larga escala possuem

desempenho melhor quando se utiliza o algoritmo de mensagens nulas do que quando se utiliza protocolos síncronos.

Como é possível observar, as pesquisas em simulação distribuída são diversificadas e as propostas são variadas, porém, os conceitos básicos das simulações conservadoras e otimistas ainda perduram na maioria delas.

3.7 Considerações finais

Neste capítulo foi apresentado o mecanismo de troca dinâmica de protocolos de sincronização proposto por Morselli (2000). Tem-se feito muita pesquisa para avaliar o desempenho dos protocolos conservadores, otimistas e suas variantes em diferentes tipos de aplicações. E existe um consenso mostrando que a adequação de um protocolo a um determinado problema pode ser muito dinâmico, o que pode levar a um baixo desempenho da simulação.

Esse mecanismo age sobre a simulação coletando parâmetros que são analisados, e caso seja detectado um baixo desempenho, é efetuada a troca de protocolos de sincronização. A troca dinâmica de protocolos de sincronização envolve duas questões principais: como realizar a troca considerando-se as características distintas de cada protocolo e quando realizar a troca, isto é, determinar o melhor momento para a troca.

Antes que a troca de protocolos de sincronização seja iniciada é necessário que a simulação esteja em um estado consistente. Precisa-se dessa garantia porque a realização da troca de protocolos em um estado transitório poderia causar um erro de causa e efeito na execução das mensagens.

O mecanismo é composto por três processos lógicos: o gerenciador, o observador e o conversor. O gerenciador é responsável pela manipulação dos dados e pela definição de necessidade de troca de protocolos. Os dados que alimentam o processo gerenciador são coletados pelos processos observadores. E por último, os processos conversores são responsáveis por realizar a troca de protocolos, garantindo um estado consistente da simulação.

4 Planejamento do Experimento para Avaliação do Desempenho de Simulação Distribuída

Este capítulo apresenta o planejamento dos experimentos realizados em uma implementação do protocolo CMB (ParSMPL) de forma a obter os parâmetros que são necessários para a definição do momento da troca de protocolo.

4.1 Considerações iniciais

Diversas pesquisas mostram que a escolha apropriada do protocolo de sincronização para uma simulação distribuída depende de fatores relacionados ao modelo que se deseja simular, a plataforma computacional utilizada e a divisão proposta da simulação em processos lógicos (Xu & Chung, 2004, Choi & Chung, 1995, Alonso et al, 1994, Ulson, 1999). A granulosidade, o particionamento do modelo, o balanceamento de carga, o *lookahead* são alguns fatores que também podem influenciar o desempenho da simulação.

A extração de características que indiquem o comportamento de uma simulação não é uma tarefa fácil. As simulações de diferentes modelos possuem características distintas que torna difícil a postulação de regras que sejam aplicáveis a todos os tipos de modelos. Diversos estudos mostram a influência de diversos parâmetros no desempenho de uma simulação distribuída (Bagrodia & Takai, 2000, Teo et al, 2002, Teo & Onggo, 2004, Park et al, 2004, Xu & Chung, 2004). Todos esses estudos analisam os

parâmetros visando avaliar a simulação como um todo e considerando a obtenção de subsídios para a escolha do melhor protocolo para iniciar a simulação.

Considerando a proposta desta tese de doutorado, a avaliação de outros parâmetros é necessária, uma vez que a simulação deve ser analisada durante a sua execução, quando se deve concluir sobre a troca ou não do protocolo utilizado.

Desta forma, tornou-se necessária a instrumentação de uma plataforma utilizada para simulação distribuída visando a obtenção de dados em tempo de execução que permitissem antever o desempenho da simulação.

Os dados obtidos durante a simulação serão utilizados com dois objetivos distintos:

1. Obtenção de parâmetros que possibilitam antever o desempenho de uma simulação. Para atingir esse objetivo foram analisados diversos resultados parciais de diversas simulações visando compará-los com o *speedup* final alcançado com a simulação distribuída. Com esse estudo pode-se determinar quais resultados parciais devem ser utilizados para decidir a troca ou não do protocolo.

2. Verificação da uniformidade do desempenho de processos lógicos que compõem uma simulação. Neste caso, o objetivo é verificar se quando um processo lógico apresenta um desempenho ruim, necessariamente todos precisam ser avaliados e terem o protocolo de sincronização trocado. Pretende-se, com esse estudo, mostrar que em uma simulação distribuída pode-se ter processos que estão caminhando de forma adequada no protocolo considerado e outros estão prejudicando o desempenho da simulação. Pretende-se observar ainda se são possíveis situações onde processos lógicos podem demorar mais que outros, porém, mesmo assim, o andamento geral da simulação pode não estar sendo afetado por um processo lógico mais lento.

Nesse contexto, verificou-se a necessidade de execução de modelos que esclarecessem certas características da simulação de forma a postular algumas regras em que seja possível descrever o andamento da simulação. Esses testes foram efetuados em simulações seqüenciais e paralelas onde, nos dois casos, utilizou-se a extensão funcional

SMPL (McDougall, 1987) em implementações no linux.

Desta forma, com a execução de diversos modelos pretende-se observar algumas características que possam permear a decisão de troca ou mesmo delinear o comportamento de um determinado modelo em uma simulação distribuída. Algumas características foram consideradas:

- Existência de processos lógicos com características de desempenho diferente.
- Avaliação de como o desempenho de cada processo lógico influencia no resultado final da simulação. Foi verificado que cada processo lógico pode ser analisado de forma isolada em relação ao restante da simulação.
- Avaliação de como a granulosidade influencia no desempenho da simulação. Diferentes granulosidades foram utilizadas de forma a ter-se granulosidades fina, média e grossa.
- Como a parametrização do modelo influencia na troca de mensagens entre os processos lógicos e com isso, verificar como se comporta o desempenho da simulação.
- O modelo pode ser aberto (com intervalo de tempo de chegada ao sistema) ou fechado (com um número pré-definido de tarefas circulando no sistema), dessa forma, deve ser analisado como essa característica influencia no desempenho da simulação.

4.2 Metodologia adotada

A obtenção dos dados necessários para a avaliação de uma simulação distribuída em tempo de execução torna-se possível por meio da implementação de um monitor de software que coleta diversos resultados parciais da simulação e os compara com o desempenho global alcançado no final da execução.

O desempenho global considerado é o *speedup* alcançado com a execução da simulação distribuída. Os resultados parciais analisados avaliam como a simulação está

se desenvolvendo. Algumas características que foram coletadas são:

- Quantidade de mensagens completas que chegam a um processo lógico e são executadas, ao invés de desbloquear uma mensagem que estava na fila de eventos.
- Quantidade de mensagens completas que chegam a um processo lógico e desbloqueiam um evento da lista de eventos.
- Quantidade de mensagens nulas que chegam e desbloqueiam um evento da lista de eventos futuros.
- Quantidade total de mensagens completas enviadas.
- Quantidade total de mensagens nulas enviadas
- Quantidade total de mensagens completas recebidas.
- Quantidade total de mensagens nulas recebidas.
- Tempo que o processo lógico ficou bloqueado.
- Tempo que o processo lógico ficou bloqueado esperando uma mensagem completa.
- Tempo que o processo lógico ficou bloqueado esperando uma mensagem completa que desbloqueou uma mensagem que estava na fila de eventos.
- Tempo que o processo lógico ficou bloqueado esperando uma mensagem nula que desbloqueou uma mensagem da fila de eventos futuros.
- Tempo de execução da simulação.

Os resultados e as relações que podem ser extraídas dos valores coletados acima serão detalhados na seção 4.3.

Para a obtenção desses dados foi utilizado um pacote de simulação distribuída – ParSMPL (apresentado na seção 4.2.1) e diversos modelos (seção 4.2.2) foram executados em uma plataforma distribuída (descrita na seção 4.2.3).

As seções 4.3.e 4.4 apresentam as alterações realizadas no ParSMPL para possibilitar a coleta dos resultados e como foram simuladas diferentes granulosidades dos processos de simulação.

4.2.1 ParSMPL

O ParSMPL é uma implementação do protocolo CMB desenvolvida por Ulson (Ulson, 1999). Inicialmente essa implementação foi desenvolvida para a plataforma IBM/RS6000 e IBM SP2 com sistema operacional AIX. Posteriormente foi adaptada para a plataforma Linux (Tatsumi, 2003) e esta versão passou então a ser chamada ParSMPLX.

Essa implementação permitiu a instrumentação do código para a realização de testes que pudessem detectar como avaliar o desempenho da simulação em tempo de execução. A técnica utilizada foi a coleta de dados (técnica de aferição) por meio de monitores de software. Como o ParSMPL é um ambiente computacional já estabelecido foi possível a utilização de experimentação.

O SMPL é uma extensão funcional da linguagem C para simulação proposta por MacDougall (1987), para plataformas compatíveis com o IBM-PC. Essa extensão apresenta uma biblioteca (subsistema SMPL) na qual, em conjunto com a linguagem C, compõem uma linguagem de simulação orientada a eventos. As operações de simulação são realizadas através de chamadas às funções do subsistema de simulação.

O ParSMPL (Ulson, 1999) é uma ferramenta de simulação distribuída que utiliza o protocolo conservador CMB para a sincronização entre os processos lógicos. Esta ferramenta baseia-se no SMPL (MacDougall, 1987), o que faz do ParSMPL uma extensão funcional da linguagem de programação C para simulação distribuída. O ParSMPL, assim como o SMPL é orientado a eventos, foi desenvolvido para plataformas IBM/RS6000 e IBM SP2 utilizando sistema operacional AIX, realizando comunicação e sincronização por meio dos ambientes de passagem de mensagem PVM e PVMe.

Esta extensão oferece os recursos necessários para a elaboração de programas de simulação distribuída, facilitando o trabalho tanto de usuários com pouca experiência em programação paralela, quanto daqueles mais experientes. Utilizando esta extensão, o usuário não precisa se preocupar com os aspectos de sincronização e comunicação entre processos, nem com características do protocolo de sincronização utilizado para

simulação distribuída. Esses mecanismos estão inseridos no ParSMPL e são transparentes ao usuário.

O SMPL foi implementado em plataformas compatíveis com o IBM-PC (MacDougall, 1987). A partir do SMPL foi desenvolvida uma nova versão para plataformas Unix, o SMPLX (Ulson, 1999), com o intuito de adequá-lo às bibliotecas de passagem de mensagem PVM e PVMe, utilizadas para a comunicação entre os processos. A principal diferença entre o SMPL e o SMPLX é que o SMPL utiliza estruturas de dados estáticas, enquanto o SMPLX, assim como o ParSMPL, utilizam estruturas dinâmicas. A razão desta alteração foi a limitação imposta à complexidade dos modelos implementados, já que as dimensões das estruturas deveriam ser definidas em tempo de compilação. A partir de modificações no SMPLX foi então desenvolvido o ParSMPL.

4.2.2 Modelos utilizados

Os modelos utilizados para os testes de desempenho com o ParSMPLX foram modelos que representam sistemas computacionais fechado (Modelo 1) e aberto (Modelo 3 e 4) e dois exemplos hipotéticos, onde tem-se um modelo fechado (Modelo 5) e um modelo aberto (Modelo 6). Foram utilizados modelos que representam sistemas computacionais, pois são modelos que simulam sistemas reais permitindo a avaliação de características como o impacto de ser um modelo aberto ou fechado, a sobrecarga da CPU em relação ao seu tempo de serviço e quantidade de tarefas, dentre outros fatores que podem ser extraídos desses modelos. Os modelos hipotéticos são modelos maiores em que podem ser verificados: o impacto do particionamento do modelo, a interferência da granulosidade dentre outros fatores.

As diferentes configurações de um mesmo modelo dizem respeito à forma como estes foram particionados em processos lógicos.

Os seguintes modelos foram utilizados nos testes:

Modelo 1 – Servidor Central (MacDougall, 1987)

Representa um sistema computacional constituído por um processador e quatro unidades de disco. Nesse sistema existe um número fixo de tarefas, que requisitam a utilização da CPU e, após o término do processamento, escolhem uma entre as quatro unidades de disco. Após a utilização do disco, a tarefa volta a requisitar a CPU e o ciclo se reinicia.

Das opções de particionamento propostas em (Ulson, 1999), os testes foram efetuados considerando-se a configuração 1 (Figura 4.1) e a configuração 2 (Figura 4.2).

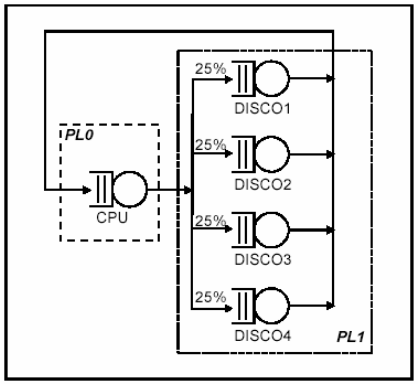


Figura 4.1: Modelo 1 - servidor central, particionado em dois processos lógicos.

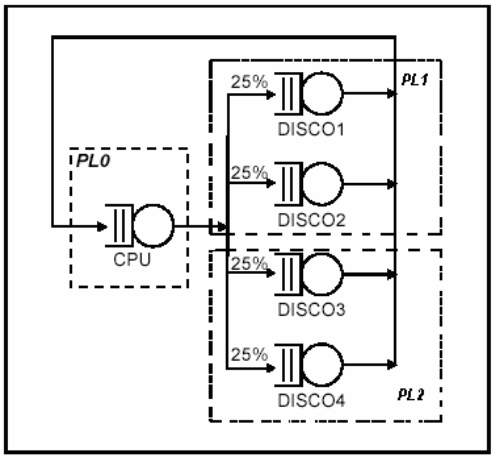


Figura 4.2: Modelo 1 - servidor central, particionado em três processos lógicos.

A parametrização utilizada para o modelo 1 pode ser vista na **Tabela 4.1**.

Tabela 4.1: Parametrização do modelo 1.

Parâmetro	Valor
Recurso: CPU	Tempo de serviço = expntl (10,0)
Recurso: DISCO 1	Tempo de serviço = erlang (30; 7,5)
Recurso: DISCO 2	Tempo de serviço = erlang (30; 7,5)
Recurso: DISCO 3	Tempo de serviço = erlang (30; 7,5)
Recurso: DISCO 4	Tempo de serviço = erlang (30; 7,5)
Número de tarefas no sistema	50
Tempo de simulação	1.000.000

Modelo 3 – Sistema computacional simplificado

Este modelo representa um sistema computacional composto por uma unidade de processamento (CPU) e uma unidade de disco, sendo uma simplificação do modelo 1 (servidor central). Neste modelo as tarefas são escalonadas de acordo com a chegada à CPU. Se a CPU estiver livre, o evento é executado, senão é colocado na fila. Após o término do serviço a tarefa pode deixar o sistema (com 60% de probabilidade) ou solicitar os serviços da unidade de disco. Terminado o serviço no disco, a tarefa volta para a utilização da CPU.

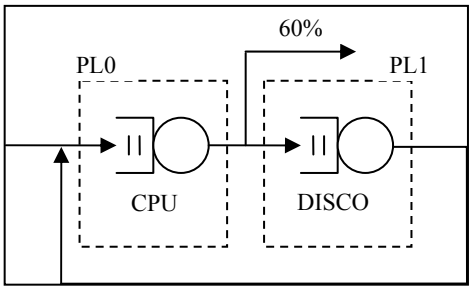


Figura 4.3: Modelo computacional simplificado.

A parametrização desse modelo pode ser vista na Tabela 4.2.

Tabela 4.2: Parametrização do modelo 3, sistema computacional simplificado.

Parâmetro	Valor
Recurso: CPU	Tempo de serviço = expntl (10,0)
Recurso: DISCO	Tempo de serviço = erlang (30; 7,5)
Tempo entre chegadas	50
Tempo de simulação	1.000.000

Modelo 4 – Sistema computacional com CPU e dois discos

Este modelo representa um sistema computacional constituído por uma unidade de processamento e duas unidades de disco. As tarefas chegam ao sistema obedecendo a uma taxa de chegada e requisitam a utilização da CPU. Essas requisições são imediatamente atendidas, caso a CPU esteja livre. Caso contrário, elas são colocadas em fila. Após o processamento, a tarefa pode deixar o sistema ou escolher uma das duas unidades de disco. Sempre que uma tarefa escolher utilizar um dos discos, a tarefa deve voltar para a CPU depois da utilização do disco escolhido.

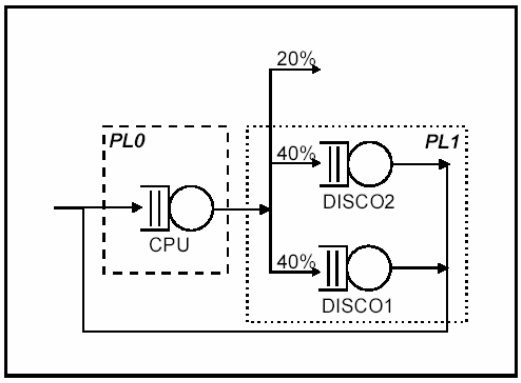


Figura 4.4: Sistema computacional com CPU e dois discos.

Tabela 4.3: Parametrização do modelo com CPU e dois discos

Parâmetro	Valor
Recurso: CPU	tempo de serviço = expntl (10.0)
Recurso: DISCO 1	tempo de serviço = erlang (30, 7.5)
Recurso: DISCO 2	tempo de serviço = erlang (30, 7.5)
Número de tarefas iniciais no sistema	1
Chegada de novas tarefas	tempo entre chegadas = expntl (50)
tempo simulado	1 000 000

Modelo 5 – Sistema de redes de fila 1 (modelo fechado)

Esse modelo descreve um sistema de redes de fila composto por sete recursos, sendo que três deles estão organizados em série e os demais estão relacionados a pontos de decisão.

O número de tarefas circulando no sistema é fixo e sempre que uma tarefa chega ao sistema, deve utilizar os serviços dos recursos identificados por REC1, REC2 e

REC3. Ao término do processamento no recurso REC3 é feita uma escolha probabilística, e assim o próximo recurso a ser utilizado é selecionado. Em função desse recurso, a tarefa pode ou retornar para o recurso REC3 ou retornar ao início do sistema para o recurso REC1.

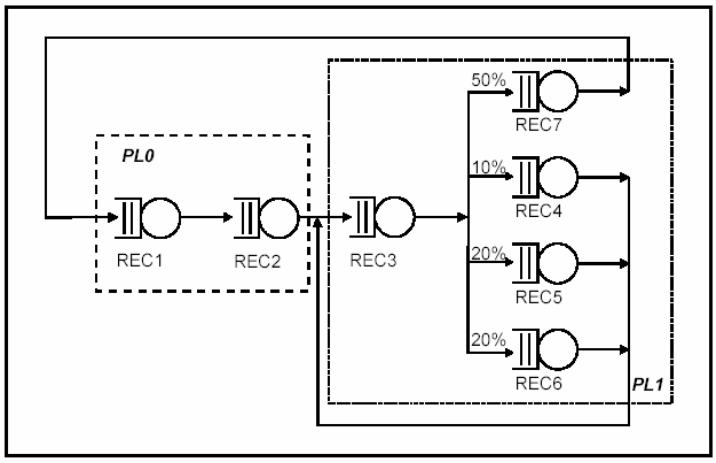


Figura 4.5: Sistema de redes de fila fechado com 7 recursos

Tabela 4.4: Parametrização do modelo de redes de fila fechado com 7 recursos

Parâmetro	Valor
Recurso: REC1	tempo de serviço = expntl (10.0)
Recurso: REC2	tempo de serviço = expntl (10.0)
Recurso: REC3	tempo de serviço = erlang (30.0, 7.5)
Recurso: REC4	tempo de serviço = erlang (30.0, 7.5)
Recurso: REC5	tempo de serviço = erlang (30.0, 7.5)
Recurso: REC6	tempo de serviço = erlang (30.0, 7.5)
Recurso: REC7	tempo de serviço = erlang (30.0, 7.5)
Número de tarefas iniciais no sistema	10
Tempo simulado	1 000 000

Modelo 6 – Sistema de redes de fila 2 (modelo aberto)

O modelo 6 é semelhante ao modelo 5, sendo diferenciado deste pelo número de tarefas que circulam pelo sistema. Enquanto no modelo 5 esse número era fixo, o modelo 6 é um sistema aberto, no qual novas tarefas chegam ao sistema obedecendo uma certa taxa de chegada e existe uma probabilidade para que as tarefas deixem o sistema.

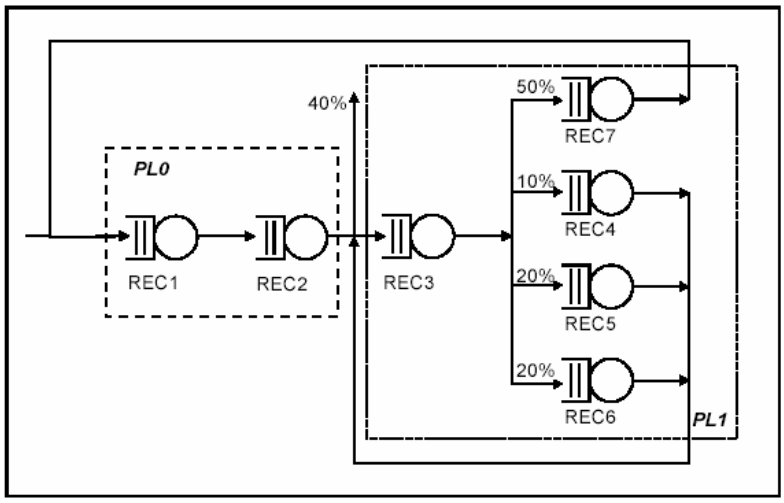


Figura 4.6: Sistema de redes de fila aberto com 7 recursos - configuração 1

Tabela 4.5: Parametrização do modelo de redes de fila aberto com 7 recursos

Parâmetro	Valor
Recurso: REC1	tempo de serviço = expntl (10.0)
Recurso: REC2	tempo de serviço = expntl (15.0)
Recurso: REC3	tempo de serviço = erlang (30.0, 7.5)
Recurso: REC4	tempo de serviço = erlang (30.0, 7.5)
Recurso: REC5	tempo de serviço = erlang (30.0, 7.5)
Recurso: REC6	tempo de serviço = erlang (30.0, 7.5)
Recurso: REC7	tempo de serviço = erlang (30.0, 7.5)
Número de tarefas iniciais no sistema	1
Tempo simulado	100 000
Probabilidade de saída	40%

O modelo aberto foi simulado com 3 configurações diferentes para que fosse possível definir as diferenças de desempenho frente a diferentes particionamentos do modelo. A configuração 2 procura minimizar a comunicação apenas e a configuração 3 é a configuração que apresenta melhor balanceamento e menor sobrecarga na comunicação de acordo com o trabalho de mestrado de Silva (2004).

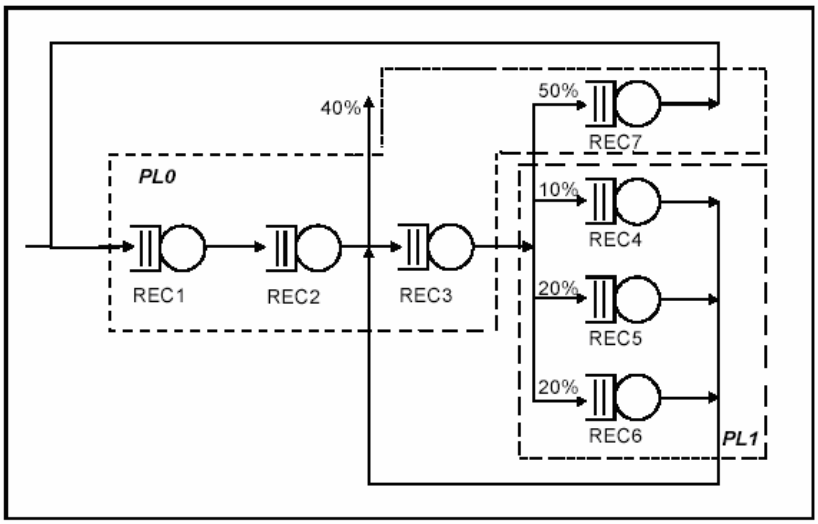


Figura 4.7: Modelo de redes de fila aberto com 7 recursos - configuração 2

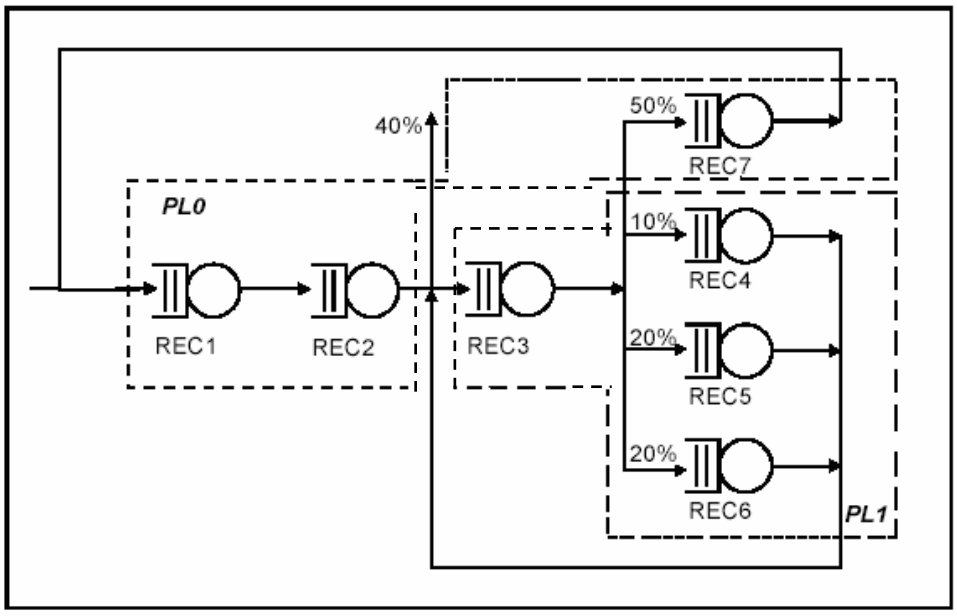


Figura 4.8: Modelo de redes de fila aberto com 7 recursos - configuração 3

4.2.3 Características da arquitetura

Para a realização dos experimentos montou-se uma rede com 5 máquinas. Essa rede de 100 Mbits é interligada por meio de um Hub-switch de 8 portas. As 5 máquinas são computadores pessoais executando o sistema operacional Linux com a distribuição

Red Hat, versão 7.3, kernel 2.4.20-28.7. A versão do compilador C utilizado é 2.96 e a versão do PVM é 3.4.4.

A Tabela 4.6 mostra a configuração (IP, processador, quantidade de memória, disco e placa de rede) das máquinas utilizadas.

Tabela 4.6: Configuração das máquinas utilizadas nos experimentos.

Máquina	IP	Processador	Memória	Disco	Rede
C1	143.107.233.191 / 10.0.0.1	Pentium 2 – 266 MHz	96 Mb	40 Gb	10/100
C2	10.0.0.2	Pentium MMX 200 MHz	64 Mb	1,6 Gb	10/100
C3	10.0.0.3	Pentium MMX 200 MHz	64 Mb	6,4 Gb	10/100
C4	10.0.0.4	Pentium 2 – 333 MHz	128 Mb	6,4 Gb	10/100
C5	10.0.0.5	AMD K6-2 550 MHz	128 Mb	2,5 Gb	10/100

Para a execução dos testes a rede local estava isolada de qualquer tráfego externo e também não havia nenhuma outra aplicação sendo executada nas máquinas.

4.3 Métricas extraídas da simulação

Diversas métricas foram extraídas da simulação de forma que fosse possível fazer uma análise do desempenho. Essas métricas permitem analisar as razões e a influência dos bloqueios aos quais a simulação distribuída é submetida quando os protocolos conservadores são utilizados. Durante o tempo em que está bloqueada, a simulação aguarda por um evento. O tempo que a simulação está bloqueada, por que a simulação está bloqueada e a influência da granulosidade na simulação são algumas das métricas obtidas.

Cada uma dessas métricas e a forma como foram extraídas da simulação estão explicadas nesta seção.

4.3.1 Quantidade de bloqueios que ocorreram com o processo lógico e tipo do bloqueio

Um dos testes realizados verifica nos processos lógicos qual o tipo de bloqueio que está ocorrendo. Como o protocolo CMB somente executa eventos com *timestamp*

menor ou igual ao menor tempo do *clock* dos canais, foi extraído da simulação qual o tipo de bloqueio que estava ocorrendo, para saber como o processo lógico foi desbloqueado, ou seja, se o próximo evento a ser executado já estava na lista de eventos a serem executados ou se o evento que chegou é o evento que irá ser executado.

Se a mensagem que chega é nula, as possibilidades são: executar um evento da lista de eventos futuros, ou seja, o *timestamp* da mensagem nula atualizou o *clock* do canal e desbloqueou o evento da cabeça da lista de eventos, ou então não fazer nada, uma vez que pode ter chegado uma mensagem nula com tempo menor que o *timestamp* da cabeça da lista de eventos futuros.

Por outro lado, se a mensagem que chega é uma mensagem completa, existem as seguintes possibilidades: a mensagem deve ser executada, pois possui o menor *timestamp* da lista de eventos ou a mensagem é colocada na lista de eventos futuros, pois tem *timestamp* menor que a cabeça da lista de eventos futuros.

Esses eventos podem tanto manter o processo lógico bloqueado ou desbloquear a execução da simulação. A computação desses valores é importante para a avaliação de desempenho da simulação. A forma como esses valores são obtidos e como é feita a avaliação de desempenho são descritas a seguir.

A lista de eventos futuros armazena as mensagens completas que serão executadas e a lista de *clocks* dos canais guarda o último *timestamp* de cada um dos canais de entrada do processo lógico. Um evento da lista de eventos futuros só será executado se o tempo do evento da cabeça da lista tiver *timestamp* com tempo menor ou igual ao menor *clock* dos canais.

Existem seis situações distintas que podem ocorrer. Os valores em preto significam a situação anterior à chegada da mensagem e os tempos em vermelho significam os eventos que chegaram e as mudanças que ocorreram na simulação.

Situação 1: a lista de eventos futuros estava vazia e chegou uma mensagem completa com *timestamp* maior que o *timestamp* do *clock* do canal.

Lista de eventos futuros	10				

Clock dos canais	1	2	3	4
	9 → 10	12	15	11

- Chegou uma mensagem completa com *timestamp* 10 do canal 1.
- Atualizou *clock* do canal.
- A lista de eventos futuros estava vazia.
- O *timestamp* do evento é menor ou igual ao menor dos tempos da lista de eventos futuros e o evento é executado.
- Não faz atualização nas marcas de bloqueio, pois a mensagem completa não desbloqueou uma mensagem já existente na lista e nem foi um bloqueio necessário, pois a lista estava vazia (o processo lógico não estava bloqueado, estava sem eventos para serem processados).

Situação 2: a lista de eventos futuros não estava vazia, ela continha um evento para ser executado e chegou uma mensagem completa com *timestamp* menor do que o *timestamp* da mensagem que estava na fila.

Lista de eventos futuros

10	11			

Clock dos canais

1	2	3	4
9 → 10	12	15	11

- Chegou uma mensagem completa com *timestamp* 10 do canal 1.
- Atualizou clock do canal.
- A lista de eventos futuros não estava vazia. O valor 10 é colocado na lista.
- O *timestamp* do evento que chegou é menor ou igual ao menor dos tempos da lista de eventos futuros e o evento é executado.
- O processo lógico estava bloqueado e tratava-se de um bloqueio necessário, pois a mensagem completa que chegou trouxe o evento a ser executado e a lista não estava vazia, ou seja, a mensagem 11 estava corretamente bloqueada, pois a execução da mensagem 11 antes da mensagem 10 causaria um erro de causa e efeito. Se um protocolo otimista estivesse sendo utilizado, ocorreria um *rollback*.

Situação 3: a lista de eventos futuros não estava vazia, mas a mensagem que chega tem *timestamp* maior que a cabeça da lista de eventos.

Lista de eventos futuros

11	12			

Clock dos canais

1	2	3	4
10	12	15	11

- Chegou uma mensagem completa com *timestamp* 12 do canal 2.
- Atualizou clock do canal.
- A lista de eventos futuros não estava vazia.
- O *timestamp* do evento é maior que o evento que estava na fila de eventos futuros. O evento é colocado na fila. Nenhum evento é executado, pois o menor *clock* dos canais continua tendo valor menor que o *timestamp* do evento da cabeça de eventos futuros.
- Não faz atualização nas marcas de bloqueio, pois não houve nem bloqueio nem desbloqueio de evento.

Situação 4: a lista de eventos futuros não estava vazia e chegou uma mensagem com *timestamp* maior que a cabeça da lista de eventos. A atualização de *clock* do canal possibilitou o desbloqueio da mensagem que estava na lista.

Lista de eventos futuros

10	12			

Clock dos canais

1	2	3	4
10	8 → 12	15	11

- Chegou uma mensagem completa com *timestamp* 12 do canal 2.
- Atualizou *clock* do canal.
- A lista de eventos futuros não estava vazia. O evento com *timestamp* 10 que estava na lista de eventos futuros é executado.

- O bloqueio que estava sendo imposto ao evento 10 era desnecessário, pois ele poderia ter sido executado antes, sem causar problemas. Dessa forma, foi um bloqueio desnecessário por envio de mensagem completa.

Situação 5: a lista de eventos futuros estava vazia e chegou uma mensagem completa. Porém, essa mensagem não será executada pois o menor *clock* dos canais tem tempo menor que a mensagem que chegou.

Lista de eventos futuros

12				

Clock dos canais

1	2	3	4
10	8 → 12	15	11

- Chegou uma mensagem completa com *timestamp* 12 do canal 2.
- Atualizou *clock* do canal.
- A lista de eventos futuros estava vazia. Mas o *timestamp* dessa mensagem é maior que o *clock* do canal 1 e o evento não pode ser executado.
- Não faz atualização nas marcas de bloqueio.

Situação 6: a lista de eventos futuros não estava vazia e chegou uma mensagem nula que atualizou o *clock* do canal de forma a desbloquear a mensagem que estava na lista de eventos futuros.

Lista de eventos futuros

9	12			

Clock dos canais

1	2	3	4
10	8 → 12	15	11

- Chegou uma mensagem nula com *timestamp* 12 do canal 2.
- Atualizou *clock* do canal.
- A lista de eventos futuros não estava vazia. A mensagem nula que chegou desbloqueou o evento de *timestamp* 9 que estava na lista de eventos futuros e esse evento é executado.
- O bloqueio que estava sendo imposto ao evento 9 era desnecessário, pois ele

poderia ter sido executado antes, sem causar problemas. Dessa forma, foi um bloqueio desnecessário por envio de mensagem nula.

A chegada de uma mensagem nula com timestamp maior que a cabeça da lista de eventos poderia ser uma 7ª situação, porém, como ela não modifica as métricas de bloqueios ela não precisa se analisada.

Por meio da análise das seis situações descritas, os bloqueios (necessários e desnecessários) são computados.

4.3.2 Tempo que o evento ficou bloqueado e o tempo de cada um dos tipos de bloqueio avaliados

Uma das características do protocolo conservador é executar eventos que ele sabe que são seguros. Dessa forma, grande parte do tempo que a simulação fica parada é por falta de mensagens (sejam elas nulas ou completas) que permitam a execução da simulação. Esse tempo que a simulação fica "parada", aguardando mensagens que possam ser executadas ou mensagens que possam desbloquear eventos da lista será chamado a partir desse ponto de tempo bloqueado.

Dessa forma, uma métrica que pode ser utilizada para avaliar a simulação em tempo de execução é medir esse tempo bloqueado, em que o processo lógico fica aguardando o recebimento de novas mensagens.

Esse tempo é medido desde a última execução até o tempo que leva para a execução do próximo evento. E, esse tempo "bloqueado" pode ser decomposto para os diferentes tipos de bloqueios computados, ou seja, o tempo bloqueado total pode ser decomposto em tempo bloqueado necessário, tempo bloqueado desnecessário e finalizado por mensagem nula, e tempo bloqueado desnecessário e finalizado por mensagem completa. Esses números permitem a análise de onde o processo lógico tem esperado mais tempo para a execução dos eventos.

Caso o processo lógico esteja muito tempo bloqueado necessariamente e pouco tempo bloqueado desnecessariamente, provavelmente, o desempenho do protocolo conservador está satisfatório. Por outro lado, caso ocorra o contrário da situação descrita, uma troca de protocolo deve ser considerada.

4.3.3 Relação tempo executando e tempo bloqueado para cada processo lógico

Uma outra métrica que pode ser extraída dos modelos diz respeito à relação entre o tempo que o processo lógico fica bloqueado e o tempo que ele está em execução. O tempo que o processo lógico está executando é resultado da diferença entre o tempo que o processo lógico ficou bloqueado e o tempo da simulação. Com esse número, faz-se a divisão do tempo executando sobre o tempo bloqueado. Essa relação indica quanto a simulação ficou executando mais do que ficou bloqueada. Ou seja, um valor igual a um indica que a simulação ficou executando o mesmo tempo que ficou bloqueado. Quanto maior esse valor, maior o tempo que a simulação ficou em execução do que bloqueado.

Caso essa relação mostre que o processo lógico se encontra mais tempo em estado bloqueado do que executando, deve-se verificar se o processo lógico possuía eventos que poderiam ter sido executados durante a fase de bloqueio. O tipo de evento que poderia ter sido executado são aqueles que já estavam na fila, e são caracterizados pelos bloqueios desnecessários.

4.4 Parâmetro inserido na simulação para avaliação de desempenho: Granulosidade

Os modelos propostos na seção 4.2.2 mostram o fluxo das tarefas que são executadas e os tempos de serviço em cada centro de serviço. No entanto, são modelos hipotéticos, onde nenhum processamento é necessário para simular cada centro de serviço. Em modelos reais, tem-se normalmente, a execução de diversas características e diversas decisões a serem consideradas. Desta forma, para tornar os modelos mais realistas, é necessário acrescentar a simulação esse tempo gasto nos modelos reais, e assim, incluiu-se no código da simulação uma carga de processamento simulando granulosidade fina, média e grossa. Esse teste verifica o impacto dessa granulosidade no *speedup* da simulação.

Esta seção descreve como foi efetuada a inclusão da granulosidade nos testes realizados nesta tese de doutorado.

Inicialmente foi utilizada uma estrutura de repetição sem conteúdo para a

simulação da granulosidade. Porém, testes preliminares mostraram que essa estrutura pode trazer alguns resultados diferentes daqueles que se deseja, pois o tempo que o programa leva para executar o laço varia. Para demonstrar esse problema foi elaborado um programa simples (Figura 4.9) cujos resultados podem ser visualizados na Tabela 4.7. Os resultados são para o valor de GRAN igual a 500.000 e o programa foi executado na máquina C1.

```
#include <stdio.h>
#include <ctype.h>
#include <sys/time.h>

#define TEMPO  "/home/celia/smplx/mod3/tempo"

int main(int argc, char *argv[])
{
  int i,gran,GRAN =0;
  FILE *tempo;         /* arquivo para armazenar resultado da simulacao */
  struct timeval t7,t8;
  float dif0,dif1,dif2,dif3;

  tempo = fopen (TEMPO,"a");

  if (argc != 0)
    GRAN = atoi(argv[1]);

for (i=0; i<5000;i++)
{
        gettimeofday (&t7, (struct timezone*)0);
          for (gran=0;gran<=GRAN;gran++);
        gettimeofday (&t8, (struct timezone*)0);
        dif0 = (float) (t8.tv_sec - t7.tv_sec) + (float) (((float) (t8.tv_usec - t7.tv_usec)) / 1000000);

        gettimeofday (&t7, (struct timezone*)0);
          for (gran=0;gran<=GRAN;gran++);
        gettimeofday (&t8, (struct timezone*)0);
        dif1 = (float) (t8.tv_sec - t7.tv_sec) + (float) (((float) (t8.tv_usec - t7.tv_usec)) / 1000000);

        gettimeofday (&t7, (struct timezone*)0);
            for (gran=0;gran<=GRAN;gran++);
        gettimeofday (&t8, (struct timezone*)0);
        dif2 = (float) (t8.tv_sec - t7.tv_sec) + (float) (((float) (t8.tv_usec - t7.tv_usec)) / 1000000);

        gettimeofday (&t7, (struct timezone*)0);
            for (gran=0;gran<=GRAN;gran++);
        gettimeofday (&t8, (struct timezone*)0);
        dif3 = (float) (t8.tv_sec - t7.tv_sec) + (float) (((float) (t8.tv_usec - t7.tv_usec)) / 1000000);
        fprintf (tempo, "%f \t %f \t %f \t %f\n",dif0,dif1,dif2,dif3);
}
  fclose (tempo);
  exit (0);
}
```

Figura 4.9: Programa fonte utilizado para avaliar o tempo de execução de estrutura de repetição.

Tabela 4.7: Resultados da execução do programa de teste de laço.

	Tempo total do *loop* (seg)	Média por loop (seg)	Desvio padrão
1º *loop*	37,91	0,007583	0,000251
2º *loop*	65,94	0,013188	0,000140
3º *loop*	65,97	0,013195	0,000301
4º *loop*	46,83	0,009366	0,001488

Como pode ser visto na Tabela 4.7, a utilização de um laço sem processamento para simular a granulosidade resulta em tempos muito diferentes. A quantidade de vezes que foi executado o primeiro *loop* é a mesma que o segundo, o terceiro e o quarto, porém, o tempo do primeiro é de aproximadamente 37 segundos, enquanto o segundo *loop* foi de aproximadamente 65 segundos, o que mostra uma discrepância muito grande. Essa diferença de tempo pode ser explicada pela existência de cache de instruções no processador que faz com que o primeiro laço tenha tempos muito menores que os demais laços. Essa diferença faz com que os resultados que se obtém com essas simulações apresentem *speedups* incompatíveis com a realidade da simulação. Um exemplo dessas incompatibilidades é uma simulação com dois processos lógicos, onde cada processo lógico é executado em um processador e como resultados ter-se valores de *speedups* muito maiores que 2.

Por essa razão, escolheu-se utilizar, para simulação de granulosidade, multiplicação de matrizes. As matrizes escolhidas foram: 25X25, 50X50, 75X75 e 100X100.

Uma multiplicação de matriz apresenta duas fases, onde a primeira é a atribuição de valores para as matrizes a serem multiplicadas e a segunda é a multiplicação propriamente dita, dessa forma, a execução de uma multiplicação de matriz 10X10, por exemplo, indica a execução de 100 atribuições mais 1000 operações em ponto flutuante. A execução prévia de um teste semelhante ao programa da Figura 4.9 com multiplicações de matrizes substituindo os laços mostrou que os tempos obtidos não apresentam os problemas do laço sem processamento.

Na Tabela 4.8 tem-se os resultados obtidos com a multiplicação de matrizes para matrizes de 50X50. Foram efetuadas mil multiplicações de matrizes. Como é possível

observar, os tempos obtidos são bastante próximos e não mostram a discrepância observada nos laços do exemplo já avaliado.

Dessa forma, optou-se por utilizar a multiplicação de matrizes para a simulação da granulosidade.

Tabela 4.8: Resultados da execução do teste de multiplicação de matrizes.

	Tempo total das multiplicações de matrizes (seg)	Média por multiplicação de matriz (seg)	Desvio padrão
1º matriz	4,00	0,0040	0,00003
2º matriz	3,94	0,0039	0,00023
3º matriz	3,94	0,0039	0,00003
4º matriz	3,93	0,0039	0,00002

A granulosidade fina pode ser representada pela multiplicação de matriz de tamanho 25X25 onde são efetuadas 16.625 operações em ponto flutuante. A granulosidade média são as multiplicações de matriz 50X50 e 75X75 onde se tem respectivamente, 125.000 e 421.875 operações em ponto flutuante. E a granulosidade grossa é representada pela matriz 100X100 onde se tem 1.000.000 de operações em ponto flutuante.

4.5 Considerações finais

Neste capítulo foram definidos os modelos e suas parametrizações, as variáveis que foram extraídas desses modelos e o ambiente computacional em que foram realizadas as execuções da simulação. Esse planejamento serviu de base para a coleta dos resultados que serão apresentados no capítulo 5.

Os modelos que foram apresentados têm sido utilizados em diversos trabalhos desenvolvidos no grupo de Sistemas Distribuídos e Programação Concorrente do ICMC.

5 Análise dos Resultados e Conclusões sobre o Desempenho de Simulação Distribuída

Este capítulo apresenta os resultados obtidos com a execução da simulação dos modelos apresentados no Capítulo 4 e uma avaliação do desempenho de simulações distribuídas.

5.1 Considerações iniciais

De acordo com os dados que foram extraídos das simulações executadas foi possível analisar o *speedup* alcançado com a execução paralela da simulação dos modelos apresentados no capítulo 4, considerando-se tempo bloqueado, quantidade de bloqueios e granulosidade.

Cada modelo foi executado para diversas sementes diferentes para cada uma das granulosidades. Os resultados apresentados são a média das diversas execuções com intervalo de confiança de 95%. O desvio padrão obtido com as execuções das sementes foi um valor pequeno, ou seja, a dispersão dos dados foi pequena. Para os modelos com dois processos lógicos utilizou-se a máquina C1 como mestre e a C4 como escrava. Para os modelos com três processos lógicos utilizou-se a máquina C1 como mestre e C4 e C3 como escravas.

A execução da simulação paralela utilizando o protocolo CMB e a monitoração, tanto dos resultados quanto do *speedup* final, tona possível determinar, a partir do

comportamento da simulação em execução, o *speedup* que seria alcançado. Assim, a seção 5.2 apresenta os resultados obtidos com a simulação, tanto durante a execução quanto após sua finalização, e na seção 5.3 é apresentada uma análise geral e as conclusões que podem ser obtidas por meio da análise dos resultados.

5.2 Resultados obtidos

Esta seção apresenta, para cada modelo considerado, um resumo dos resultados obtidos com as simulações. As tabelas contendo todos os resultados das simulações encontram-se no Apêndice A.

5.2.1 Servidor Central com dois processos lógicos

O modelo do servidor central com a configuração 2 *pls* (2 processos lógicos) foi executado com diferentes granulosidades e obteve-se os resultados apresentados na Tabela 5.1 e Figura 5.1.

Tabela 5.1: *Speedup* resultante da simulação do modelo 1

GRAN	Tempo de execução (Sequencial)	Tempo de execução (Paralelo)	Speedup
0	6,59	176,39	0,037
25	174,84	279,70	0,625
50	1060,14	1022,36	1,037
75	3252,36	2873,73	1,132
100	7320,47	6303,40	1,161

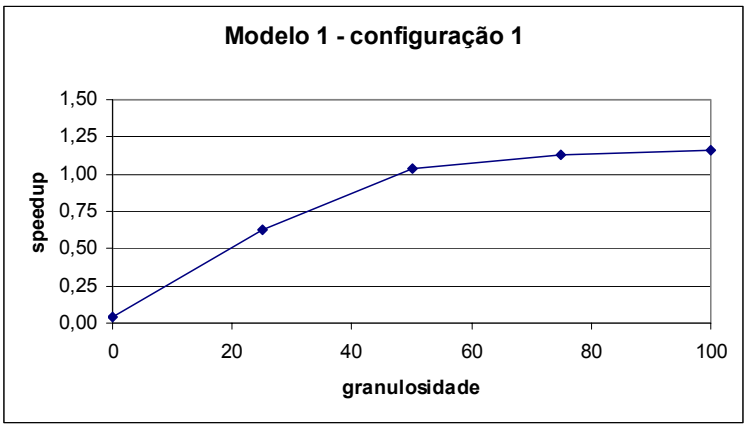


Figura 5.1: *Speedup* X granulosidade para o modelo 1

Como pode ser observado na Tabela 5.1, com o aumento da granulosidade aumenta-se o valor do *speedup*, onde a partir do valor 50 tem-se *speedups* com valores acima de 1. O gráfico da Figura 5.1 mostra o avanço do *speedup* em relação à granulosidade utilizada. Verifica-se que o aumento do *speedup* não é linear de acordo com as granulosidades, mesmo que o aumento da carga com as granulosidades seja quadrático. Esse tipo de curva apresentada no gráfico mostra que um aumento na granulosidade, além das já utilizadas poderão não resultar em *speedups* muito maiores.

Verificou-se também o tempo bloqueado de cada um dos processos lógicos. Com granulosidade 0 o modelo considerado apresenta uma porcentagem de tempo bloqueado muito grande, que diminui com o aumento da granulosidade, conforme pode ser observado na figuras 5.2 (o eixo X dessa figura representa o *speedup* alcançado pelo modelo, porém, cada ponto da curva representa as diferentes granulosidades consideradas).

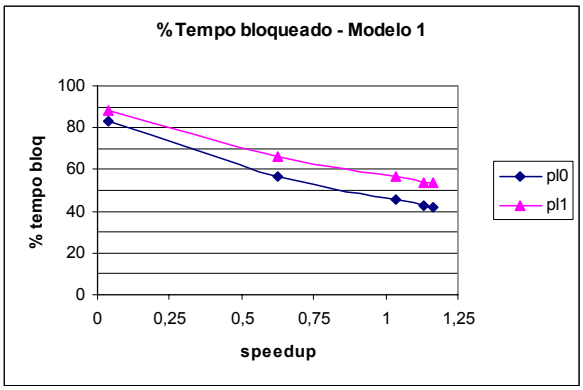


Figura 5.2: Gráfico que mostra a porcentagem de tempo bloqueado em função do *speedup* para o modelo 1

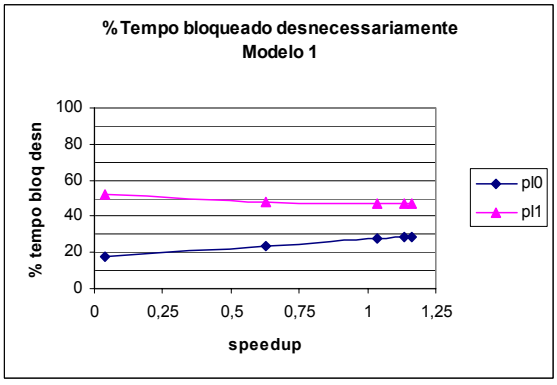


Figura 5.3: Gráfico que mostra o avanço da porcentagem de tempo bloqueado desnecessariamente em função do *speedup* para o Modelo 1

Já a porcentagem de tempo bloqueado desnecessariamente nos dois processos lógicos não diminuiu. No processo lógico 0 ocorreu um aumento, porém, em valor bem menor que o processo lógico 1, que apesar de ter diminuído, continuou com um percentual alto de tempo bloqueado desnecessariamente. Isso é um indicativo de que muitos eventos que estavam na fila de eventos poderiam ter sido executados sem causar problemas de causa e efeito.

A Tabela 5.2 mostra o comportamento do processo lógico 0 em relação ao recebimento de mensagens e qual o comportamento do processo em relação a essas mensagens recebidas.

Tabela 5.2: Comportamento das mensagens que chegam ao processo lógico 0 do Modelo do servidor central com dois processos lógicos.

GRAN	pl0 (tempo executando / tempo bloq)	% msg completas bloq nec (pl0)	%msg completas bloq desn (pl0)	%msg nulas bloq desn (pl0)
0	0,20	73,79	16,81	80,53
25	0,75	75,66	6,77	77,05
50	1,20	75,67	6,77	77,03
75	1,35	75,66	6,78	77,03
100	1,38	75,67	6,77	77,03

A primeira coluna mostra as granulosidades, a segunda coluna, mostra a relação tempo executando / tempo bloqueado do processo lógico (Tabela 5.2). A coluna 3 mostra a porcentagem das mensagens completas que chegaram e desbloquearam eventos cujos bloqueios foram necessários. A coluna 4 mostra a porcentagem de mensagens completas que desbloquearam eventos cujos bloqueios foram desnecessários e a última coluna a porcentagem de mensagens nulas que desbloquearam eventos cujos bloqueios foram desnecessários.

A porcentagem de mensagens completas que desbloquearam eventos cujos bloqueios foram necessários ou desnecessários e a porcentagem de mensagens nulas que desbloquearam eventos cujos bloqueios foram desnecessários em geral não se modifica com o aumento da granulosidade. Isso pode ser explicado, pois a simulação em si não se modifica com a adição da granulosidade. A quantidade de mensagens que a simulação troca com ou sem granulosidade se mantém a mesma. O que muda é o tempo que essas mensagens demoram a serem enviadas e recebidas. Apesar disso, esse dado permite analisar a quantidade de mensagens que trafegam e como elas se comportam na simulação, podendo ser utilizado em conjunto com os outros de forma a se definir

regras que possam ser utilizadas para definir a troca ou não do protocolo de sincronização.

A Figura 5.4 mostra o gráfico da relação tempo executando / tempo bloqueado dos dois processos lógicos. É possível verificar que o tempo bloqueado diminui e o tempo executando aumenta com o aumento do *speedup* (granulosidade). Isso é mais acentuado no processo lógico 0, que tem relações com valores acima de 1, que indica que ele passa mais tempo executando que bloqueado, diferente do processo lógico 1 que passa mais tempo bloqueado que executando, visto que em nenhuma granulosidade a relação tem valor maior que 1.

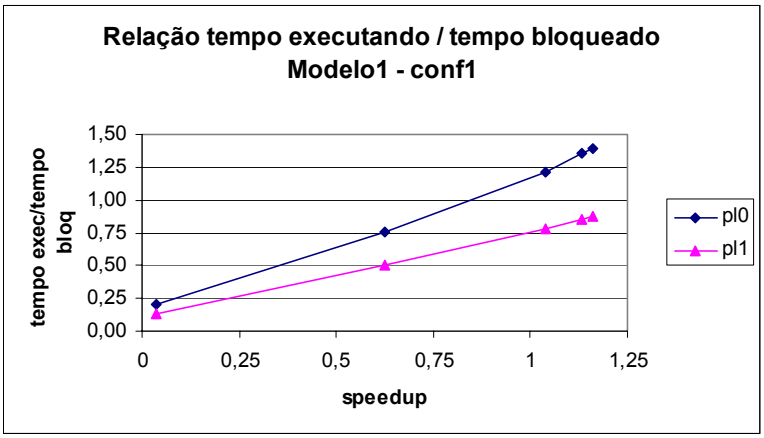


Figura 5.4: Gráfico que mostra a relação tempo executando / tempo bloqueado para os dois processos lógicos do Modelo 1

Analisando-se o comportamento do processo lógico 0, é possível verificar que em torno de 75 % das mensagens completas que chegaram foram as mensagens que foram executadas (bloqueios necessários), dessa forma, esse valor alto indica que a substituição do protocolo conservador pelo protocolo otimista pode não ser interessante, pois o avanço da simulação poderia resultar em muitos *rollbacks* e não haveria ganho no *speedup*.

Tabela 5.3: Comportamento das mensagens que chegam ao processo lógico 1 do Modelo do servidor central com dois processos lógicos.

GRAN	pl1 (tempo executando / tempo bloq.)	% msg compl bloq nec (pl1)	%msg compl bloq desn (pl1)	%msg nulas bloq desn (pl1)
0	0,14	51,36	0	98,50
25	0,51	51,03	0	95,28
50	0,78	51,05	0	95,36
75	0,85	51,05	0	95,36
100	0,87	51,05	0	95,36

Por outro lado, no processo lógico 1 a porcentagem de mensagens completas que causaram bloqueios necessários é de 51% e a porcentagem de mensagens nulas que causaram de bloqueios desnecessários é de 95%. E, a porcentagem de tempo que o processo lógico ficou bloqueado desnecessariamente foi de 47%. Essa relação mostra que uma mudança desse processo lógico para o protocolo otimista traria vantagens visto que quase metade do tempo da simulação (47%) o processo se encontra bloqueado desnecessariamente. Outro fator que deve ser verificado é que o aumento da granulosidade no processo lógico 1 não promove uma relação executando/bloqueado muito boa, o que justificaria a troca do protocolo para que o processo lógico 1 possa avançar na simulação, e com isso aumentar o *speedup*.

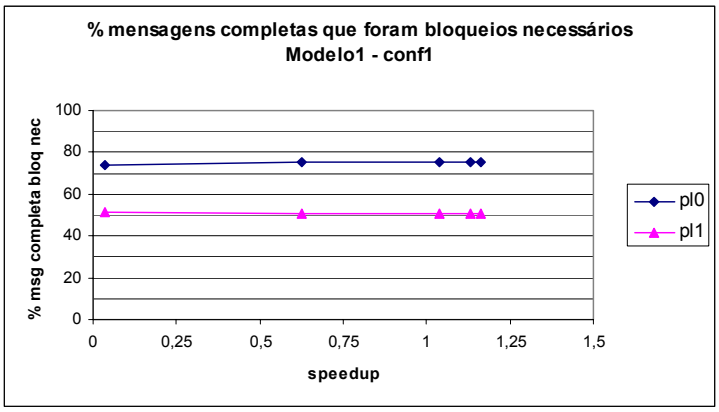


Figura 5.5: Gráfico que mostra as porcentagens das mensagens completas que foram bloqueios necessários.

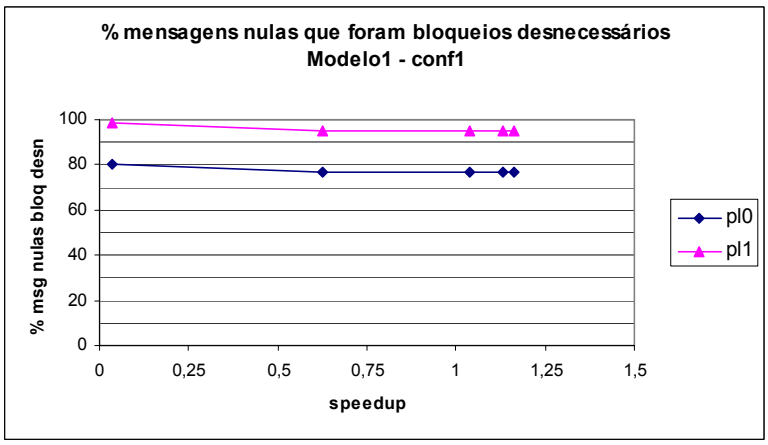


Figura 5.6: Gráfico que mostra as porcentagens das mensagens nulas que foram bloqueios desnecessários.

A Figura 5.5 mostra a porcentagem de mensagens completas que causaram bloqueios necessários para os dois processos lógicos para os *speedups* alcançados e a

Figura 5.6 mostra a porcentagem de mensagens nulas que causaram bloqueios desnecessários para os dois processos lógicos.

Com a análise dos resultados obtidos com o modelo servidor central – 2 *pls*, pode-se observar que:

I. Analisando a simulação em geral

- Quando a porcentagem de tempo bloqueado é alta (granulosidade 0, % tempo bloqueado = 82,82%), a porcentagem de tempo bloqueado desnecessariamente é baixa (tempo bloqueado desnecessariamente = 17,72%), a relação executando / bloqueado é menor que 1 ou muito menor que 1 e existe uma porcentagem de mensagens completas que são bloqueios necessários alto (73,79%) indica que a simulação seqüencial ou a simulação MRIP pode ser a mais adequada. Isso pode ser explicado pelos seguintes fatores: existem muitas mensagens completas que são as que devem ser executadas, e os bloqueios estão ocorrendo, na sua maioria, porque são necessários. Isso indica que a mudança para o protocolo otimista não trará vantagens. Nesse caso, a simulação seqüencial pode ser a melhor opção.

- Quando se tem uma porcentagem de tempo bloqueado alta (acima de 50%), uma porcentagem de tempo bloqueado desnecessário alta (acima de 40%), uma relação executando/bloqueado com valores abaixo de um, uma porcentagem de mensagens completas que foram bloqueios necessários baixo (abaixo de 50%) e uma porcentagem alta de mensagens nulas que bloquearam a simulação desnecessariamente (acima de 50%), então, uma troca de protocolo deve ser considerada. Neste caso, o que está diminuindo o desempenho da simulação é a não execução de eventos que estão na lista e que são desbloqueados com a chegada de mensagens nulas que apenas atualizam os *clocks* dos canais.

II. Analisando cada processo lógico.

- O *pl* 0 apresenta uma relação executando / bloqueado satisfatório a partir da granulosidade 50 (valores maiores que 1,2) o que indica que a partir desse ponto o tempo executando supera o tempo bloqueado. Nessa

granulosidade o *pl* 0 possui uma porcentagem de tempo bloqueado de aproximandamente 45% variando até 41 % para granulosidade 100, onde de 17% a 12% é para tempo bloqueado necessário e de 27% a 28% para tempo bloqueado desnecessário. A porcentagem de mensagens que causaram bloqueios necessários é de 75% e de bloqueios desnecessários de 84% (somando as mensagens completas e nulas). Nesse caso, pode-se verificar que apesar da porcentagem de bloqueios desnecessários ser alta (84%), a porcentagem de bloqueios necessários também é alta (75%), a relação executando / bloqueado é satisfatória (maior que 1) e a porcentagem de tempo bloqueado tem valores menores que 50%. De acordo com esses valores é possível concluir que o protocolo conservador está tendo bom desempenho e deve ser mantido.

- No *pl* 0, em relação ao desempenho com granulosidade baixa (0 e 25) é possível observar que a porcentagem de tempo bloqueado é alta, 82% e 56% para granulosidade 0 e 25 respectivamente. A relação executando / bloqueado é menor que 0,75. A porcentagem de tempo bloqueado necessário é maior que o tempo bloqueado desnecessário (60% contra 17% para granulosidade zero, e 31% contra 23% para granulosidade 25). A porcentagem de mensagens que causaram bloqueios necessários é alta (73% e 75%). Analisando os dados é possível verificar que uma troca de protocolo não irá trazer benefícios para esse processo lógico. Uma verificação de desempenho dos demais processos lógicos é necessária para que seja possível analisar se uma troca para simulação seqüencial ou MRIP deva ser efetuada.

- O *pl* 1 não apresentou uma relação executando / bloqueado boa em nenhuma das granulosidades (sempre valores menores que 1) o que indica que o seu desempenho não está satisfatório. Apesar dessa relação melhorar com o aumento da granulosade (de 0,13 até 0,87 – granulosidade 100) o valor 0,87 é bastante ruim uma vez que indica que o processo lógico passou mais tempo bloqueado que executando. Analisando os outros dados é possível observar que a porcentagem de tempo bloqueado manteve-se alta (sempre acima de 50% para todas as

granulosidades) e a porcentagem de tempo bloqueado desnecessário manteve-se alta também (52% para granulosidade 0 e 47% para as demais granulosidades). Apesar da porcentagem de mensagens completas que causaram bloqueios necessários ser razoavelmente alta (51%), a porcentagem de tempo bloqueado necessário foi baixa para granulosidade acima de 50 (abaixo de 10% - 8,8%, 6,8% e 6,2 para granulosidades 50,75 e 100, respectivamente) e a porcentagem de mensagens nulas que causaram bloqueios desnecessários foi bastante alta (95%). Verificando-se então que o processo lógico teve relação executando / bloqueado bastante ruim e analisando os dados apresentados, conclui-se que uma troca de protocolo para otimista é indicada. Mesmo para granulosidade baixa, a troca de protocolo pode resultar em melhor desempenho, pois os bloqueios desnecessários podem ser eliminados e com isso pode resultar em melhor desempenho inclusive para o *pl* 0, que poderá ter diminuída a quantidade de bloqueios necessários.

Observa-se nestes resultados que para granulosidade alta, o *pl* 0 apresenta bom desempenho enquanto o *pl* 1 sempre apresenta um desempenho não aceitável para qualquer granulosidade.

5.2.2 Servidor Central com três processos lógicos

O modelo do servidor central com 3 *pls* (particionamento em 3 processos lógicos) foi executado e obtiveram-se os resultados da Tabela 5.4. Como pode ser observado não foram obtidos *speedups* acima de 1 em nenhuma granulosidade. Dessa forma, é possível notar que o particionamento do modelo em mais processos lógicos (de 2 para 3) não traz melhora ao desempenho da simulação. Pelo contrário, existe uma diminuição no speedup da simulação. Isso pode ser explicado pelo aumento da comunicação na rede e o aumento na necessidade de sincronização dos processos lógicos para o avanço da simulação.

Tabela 5.4: Resultado apresentado pelo modelo 1, configuração 2

GRAN	Tempo de execução (Sequencial)	Tempo de execução (Paralelo)	Speedup
0	6,58	262,06	0,025
25	174,81	367,01	0,476
50	1060,02	1279,51	0,828
75	3251,92	3583,20	0,908
100	7319,93	7831,48	0,935

Os resultados apresentados na Tabela 5.4 mostram que a utilização do protocolo conservador para essa simulação não é adequado. No entanto, a avaliação dos resultados é importante para complementar o estudo em questão e verificar qual o andamento da simulação que leva a resultados tão desfavoráveis (Figura 5.7).

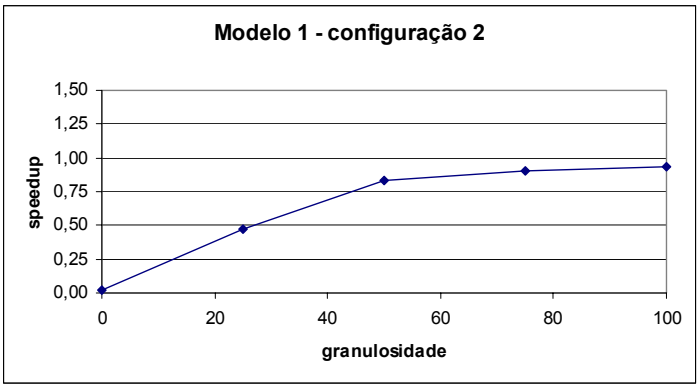


Figura 5.7: *Speedup* X granulosidade para o modelo 1 configuração 2

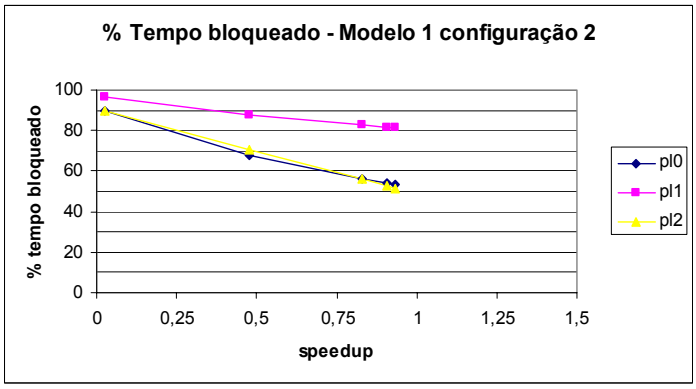


Figura 5.8: Porcentagem de tempo bloqueado de cada um dos processos lógicos do Modelo 1 - configuração 2

A porcentagem de tempo que cada um dos processos lógicos ficou bloqueado foi alto e apesar de ter diminuído com o aumento da granulosidade, continuou com valores acima de 50%, mesmo para a granulosidade mais alta (Figura 5.8). O processo lógico 0

e 2 tiveram diminuição do tempo bloqueado mais acentuado que o processo lógico 1. Isso pode ser explicado pela diferença de desempenho das máquinas, pois a carga de trabalho imposta para o pl1 e para o pl2 é a mesma, mas o poder computacional da máquina que executou o pl1 é maior que a que executou o pl2. Dessa forma, apesar do pl1 executar mais rápido, ele fica mais tempo bloqueado esperando mensagens que possam desbloquear a execução da simulação.

A heterogeneidade das máquinas mostra a diferença de desempenho que processos lógicos podem atingir. Essa diferença pode ser minimizada com a utilização simultânea de protocolos diferentes para cada processo lógico. Se a máquina mais fraca puder permanecer a maior parte do tempo processando com o protocolo otimista (tentando minimizar a ocorrência de *rollbacks*), ela irá gerar mais mensagens que poderão desbloquear os demais processos lógicos e com isso melhorar o desempenho da simulação de forma geral. A Figura 5.9 mostra a porcentagem de tempo bloqueado desnecessariamente dos processos lógicos. A porcentagem aumenta para o processo lógico 0 e diminui para os processos 1 e 2, porém, mais acentuadamente para o pl2.

A relação tempo executando / tempo bloqueado dos processos lógicos (Figura 5.10) mostra que nenhum dos processos lógicos passou mais tempo executando do que bloqueado, uma vez que nenhum deles atingiu o valor 1. Essa relação é pior para o pl1, que foi executado com a máquina mais rápida. Dessa forma, é possível observar que a dependência que os pls 1 e 2 possuem do pl0 (de onde recebem as mensagens para serem processadas) torna-se pior para máquinas mais rápidas, que processam granulosidades mais grossas porém, como o pl0 depende das mensagens que chegam dos pl1 e pl2, a lentidão da execução faz com que o pl0 também seja bloqueado e com isso diminua o desempenho da simulação.

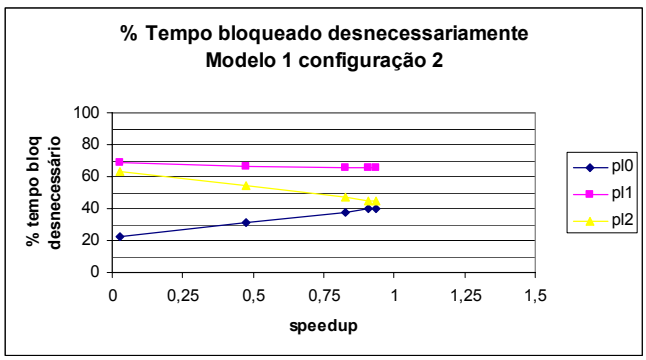


Figura 5.9: Porcentagem de tempo bloqueado desnecessariamente de cada um dos processos lógicos do Modelo 1 - configuração 2

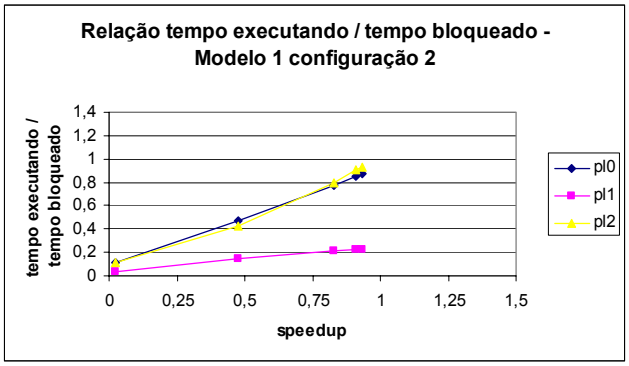


Figura 5.10: Relação tempo executando / tempo bloqueado para o Modelo 1 - configuração 2

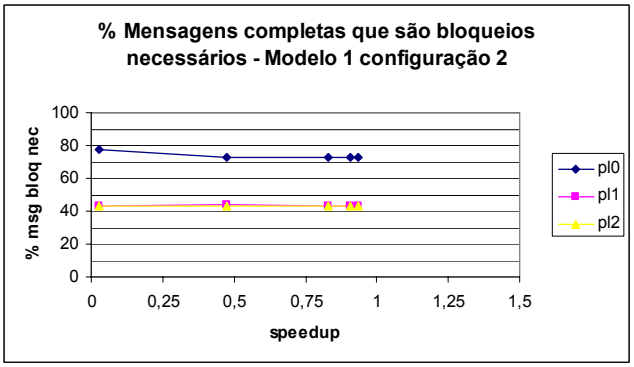


Figura 5.11: Porcentagem de mensagens completas que resultaram em bloqueios necessários no Modelo 1 - configuração 2

Como se pode ver nas figuras 5.11 e 5.12, a porcentagem de mensagens completas que resultaram em bloqueios necessários para o pl0 é grande, o que mostra que uma mudança no protocolo pode resultar em muitos *rollbacks*, apesar da quantidade de mensagens nulas que foram bloqueios desnecessários também ser razoavelmente alto (em torno de 50%).

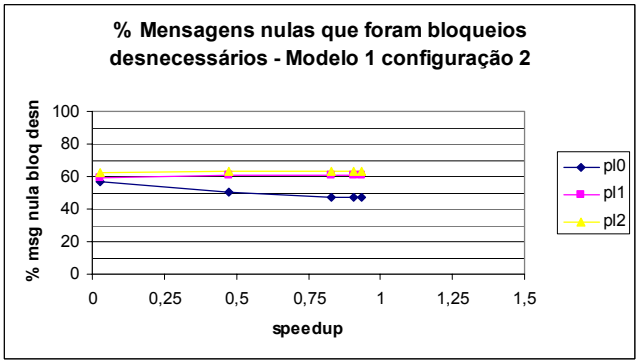


Figura 5.12: Porcentagem de mensagens nulas que resultaram em bloqueios desnecessários no Modelo 1 - configuração 2

Analisando-se os resultados obtidos com o modelo do servidor central particionado em três processos lógicos é possível chegar a seguintes conclusões:

I. Analisando a simulação em geral.

- O particionamento utilizado para o modelo 1 com dois processos lógicos (exemplo anterior descrito) foi melhor do que com 3 processos lógicos. Isso mostra que o particionamento adequado do modelo é importante para a simulação distribuída. Apesar de poder existir melhora de desempenho com a troca de protocolo dos processos lógicos, modelos mal particionados podem resultar em desempenhos piores que a simulação seqüencial. Dessa forma, a migração do modelo para a forma seqüencial ou utilização de MRIP também devem ser consideradas quando a relação tempo executando / tempo bloqueado de todos os processos lógicos for abaixo do esperado.

- A porcentagem de tempo bloqueado foi alta (acima de 50%) para todos os processos lógicos em todas as granulosidades, o que mostra que o desempenho da simulação está muito aquém do esperado e que algum problema está ocorrendo. Essa porcentagem alta faz com que a relação tempo executando / tempo bloqueado não atinja o valor 1 para qualquer processo lógico em qualquer granulosidade e assim é possível concluir que a interrupção da simulação é necessária e uma atitude seja tomada. Nesse caso, ou o reparticionamento ou a execução da simulação seqüencial ou MRIP.

- O reparticionamento de uma simulação em tempo de execução torna-se custoso por se tratar de uma tarefa muito complexa, e que imporia uma sobrecarga muito grande à simulação. Para evitar particionamentos inadequados Silva (2004) propõe a utilização de algoritmos genéticos para definir um particionamento eficiente baseado no poder computacional dos processadores onde a simulação será executada, no fluxo das mensagens trocadas entre os processos lógicos e na quantidade de processamento requerida pelos processos lógicos.

II. Analisando cada processo lógico.

- O pl 0 possui uma porcentagem de mensagens completas que causaram bloqueios necessários alta (aproximadamente 73%), o que indica que mesmo com a relação executando / bloqueado ruim para todas as granulosidades (valores abaixo de 1), uma mudança de protocolo não irá trazer vantagens a simulação. Nesse caso, mesmo com desempenho ruim, a simulação conservadora é a mais indicada para esse processo lógico.

- O pl 1 tem a porcentagem de tempo bloqueado extremamente alta (acima de 81% para todas as granulosidades) o que mostra uma subutilização muito grande da capacidade de processamento da máquina. A relação executando / bloqueado é ruim (abaixo de 0,3 para todas as granulosidades), a porcentagem de tempo bloqueado desnecessário é alta (maior que 65%), a porcentagem de mensagens completas que causaram bloqueios necessários é razoavelmente baixa (43%) e a porcentagem de mensagens nulas que causaram bloqueios desnecessários está em torno de 61%. Considerando-se esses dados, uma troca de protocolo pode ser considerada para a melhora no desempenho da simulação, apesar do desempenho ruim que a simulação apresentou como um todo. O protocolo otimista poderia nesse caso, desbloquear as mensagens que foram bloqueadas desnecessariamente e com isso enviar mais mensagens para o processo lógico 0 melhorando inclusive o desempenho do pl 0.

- O pl 2, apesar de conter os mesmos tipos de recursos do pl 1 apresenta um desempenho diferente deste por ter sido executado em uma máquina com processamento mais lento. Nesse caso, a porcentagem de tempo bloqueado nesse processo lógico ficou em valores acima de 51% (o pl 1 ficou acima de 81%), porém, esse valor ainda pode ser considerado bastante alto pois a relação executando / bloqueado não atingiu o valor 1 em nenhum das granulosidades. A porcentagem de tempo bloqueado desnecessariamente foi alto (acima de 44% em todas as granulosidades), a porcentagem de mensagens completas que causaram bloqueios necessários foi a mesma verificada no pl 1 (em torno de 43%) e a porcentagem de mensagens nulas que causaram bloqueios desnecessários também foi alta (em torno de 63%). Analisando-se esses valores, a troca de protocolo para otimista pode ser considerada e uma análise análoga ao pl 1 pode ser feita, pois o desbloqueio das mensagens que

estavam na fila de eventos nesses dois processos lógicos pode resultar em um desempenho melhor para o pl 0 pois esse processo lógico depende das mensagens enviadas pelos processos lógicos 1 e 2. Nesse caso, o desempenho da simulação pode melhorar não apenas para os processos lógicos 1 e 2 como também para o *pl* 0.

5.2.3 Modelo 3

O modelo 3 é um sistema computacional simplificado (uma CPU e uma unidade de disco) com tempo de serviço de CPU de 50 unidades de tempo foi executado e os resultados obtidos estão apresentados na Tabela 5.5 e na Figura 5.13:

Tabela 5.5: Resultados da execução do modelo 3 (paralelo e seqüencial) e o *speedup* resultante.

GRAN	Tempo de execução (Sequencial)	Tempo de execução (Paralelo)	Speedup
0	19,83	35,92	0,552
25	43,44	53,06	0,819
50	166,35	161,29	1,031
75	472,19	425,48	1,110
100	1036,41	917,64	1,129

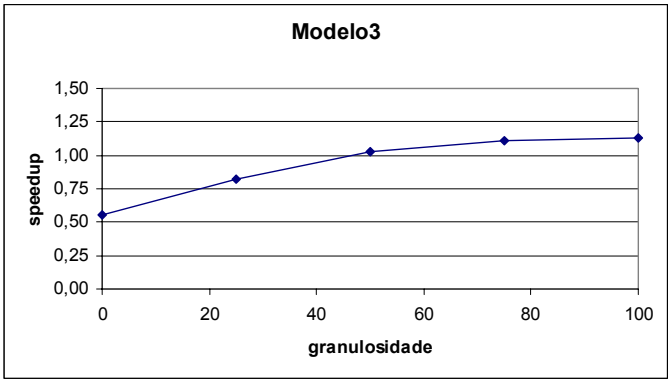


Figura 5.13: Speedup X granulosidade para o Modelo 3

Como pode ser observado na Figura 5.14, a porcentagem de tempo bloqueado dos dois processos lógicos diminui com o aumento do *speedup*. No caso do pl0 isso é mais acentuado.

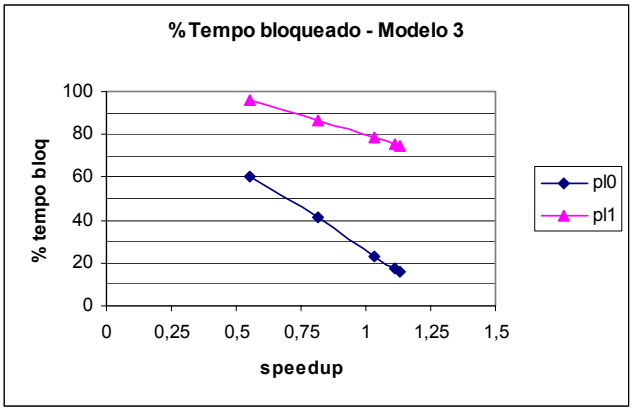


Figura 5.14: Porcentagem de tempo bloqueado - Modelo 3

O tempo de bloqueio desnecessário no modelo3 diminui para o pl0, mas mantém aproximadamente o mesmo percentual no pl1. (Figura 5.15).

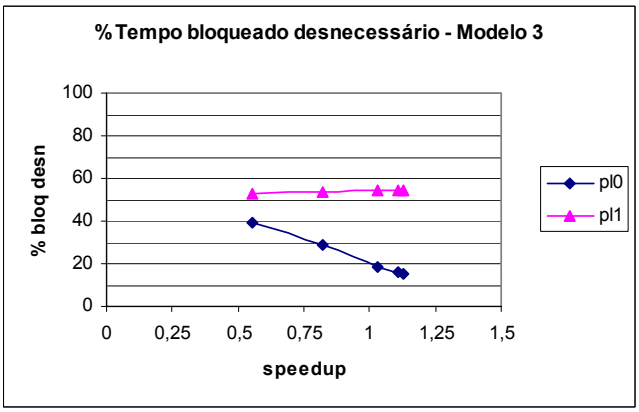


Figura 5.15: Porcentagem de tempo bloqueado desnecessariamente - Modelo 3

Já a relação tempo executando / tempo bloqueado dos dois processos lógicos aumenta com o aumento da granulosidade (*speedup*). Porém, nota-se que o aproveitamento do tempo no pl0 é muito melhor que o pl1 que passa grande parte do tempo bloqueado (Figura 5.16). O pl0 atinge com a granulosidade 100 o valor de 5,35, ou seja, ele fica executando 5 vezes mais do que fica bloqueado. O pl1, por outro lado, tem valores abaixo de 0,34 em todas as granulosidades. O valor alto do pl0 e o valor baixo do pl1 em relação ao número de processos lógicos (2 pls) pode indicar um desbalanceamento no particionamento do modelo. Porém, o modelo 3 é um modelo simples com apenas 2 recursos e o particionamento utilizado é o particionamento possível que faz com que um processsador simule um recurso e o outro processador simule o outro recurso. Nesse caso, o desbalanceamento encontrado no particionamento do modelo não está na forma como o modelo foi particionado, e sim na carga de

trabalho que um recurso tem em relação ao outro. No caso específico desse modelo, atingiu-se algum *speedup* com granulosidade acima de 50, e nesse caso, a simulação conservadora deve ser mantida. Porém, para granulosidades mais baixas (0 e 25) a solução é a adoção da simulação seqüencial ou MRIP.

Desbalanceamentos no particionamento podem ser resolvidos por meio de ferramentas que particionam o modelo automaticamente de acordo com a arquitetura e características do modelo, porém, desbalanceamentos intrínsecos do modelo podem resultar em desempenho ruim para a simulação distribuída.

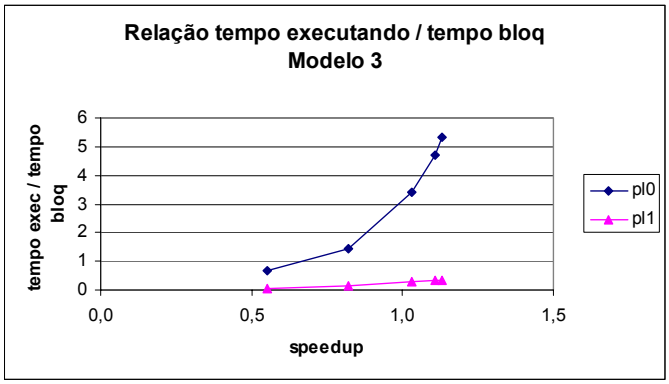


Figura 5.16: Relação tempo executando / tempo bloqueado para o Modelo 3

O desbalanceamento existente no modelo 3, que pode ser visualizado na Figura 5.16, é intrínseco ao modelo devido ao fato da CPU (pl0) ser mais utilizada que o DISCO (pl1). Nesse caso, uma análise do desempenho dos dois processos lógicos é necessária para que seja possível verificar se esse desempenho pode ser melhorado com a troca de protocolo de um dos processos lógicos.

- O pl0 possui uma relação executando / bloqueado melhor que o pl1, e possui uma porcentagem alta de mensagens completas que foram bloqueadas necessariamente (71%). Isso indica que o protocolo conservador está tendo um desempenho bom. Apesar da porcentagem de mensagens nulas que foram bloqueios desnecessários ser alta (98% - Figura 5.18), a porcentagem de tempo que o pl0 ficou bloqueado a partir da granulosidade 50 foi pequeno em relação a execução da simulação (variação de 22% para granulosidade 50 até 15 para granulosidade 100). No caso da granulosidade baixa (0 e 25) é possível observar que uma mudança para o protocolo otimista não deve trazer muitas melhoras devido à porcentagem de mensagens completas que causaram

bloqueios necessários ser alta (71% - Figura 5.17). Nesse caso, a simulação seqüencial ou MRIP é a mais indicada.

- No caso do pl1, a porcentagem de mensagens completas que foram bloqueios necessários foi baixa (aproximadamente 9% - Figura 5.17) e a porcentagem de mensagens nulas que foram bloqueios desnecessários foi relativamente baixo (aproximadamente 44% - Figura 5.18). Porém, como o desempenho do pl1 (relação executando / bloqueado sempre menor que 1) foi baixo e a porcentagem de tempo bloqueado desnecessariamente foi alto (acima de 53%), a troca de protocolo pode ser considerada, pois a execução das mensagens que foram bloqueadas desnecessariamente pode acelerar a simulação. E a possibilidade de execução de mensagens erradas fica diminuída pelo baixo valor do número de mensagens completas que foram bloqueios necessários.

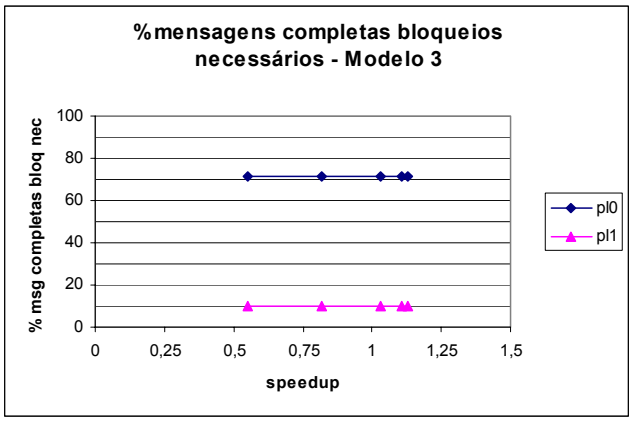


Figura 5.17: Porcentagem de mensagens completas que foram bloqueios necessários para o Modelo 3

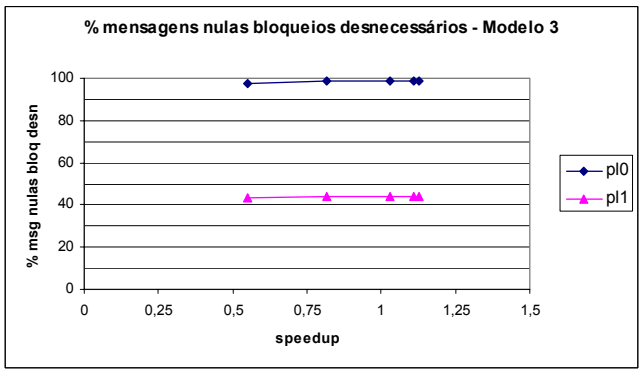


Figura 5.18: Porcentagem de mensagens nulas que foram bloqueios desnecessários para o Modelo 3

5.2.4 Modelo 4 – Sistema computacional com 1 CPU e 2 discos.

Os resultados da execução do modelo 4 estão na Tabela 5.6 e na Figura 5.19:

Tabela 5.6: *Speedup* resultante da simulação do modelo 4.

GRAN	Tempo de execução (Sequencial)	Tempo de execução (Paralelo)	Speedup
0	6,77	121,10	0,056
25	138,42	202,27	0,684
50	839,06	761,96	1,101
75	2572,77	2153,16	1,195
100	5805,85	4727,13	1,228

Esse modelo, assim como o modelo 1 (configuração com 2 processos lógicos) e o modelo 3, apresentou *speedup* a partir da granulosidade 50 (Figura 5.19).

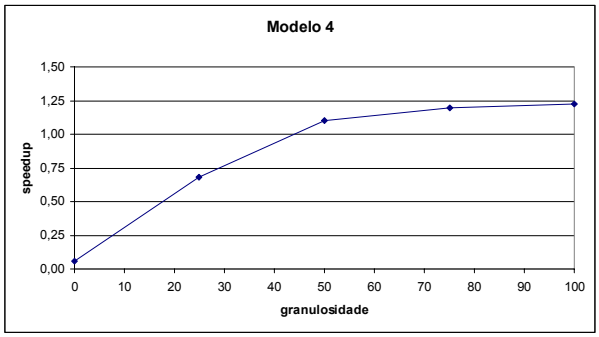


Figura 5.19: Speedup X granulosidade do Modelo 4

O modelo 4 repete o mesmo comportamento do modelo 1 (configuração 1) e do modelo 3. A porcentagem de tempo bloqueado diminui com o aumento do *speedup* (granulosidade). Com granulosidade 0 o tempo bloqueado é o mesmo para os dois processos lógicos, porém, com o aumento da granulosidade o pl0 tem uma diminuição mais acentuada do que o pl1.

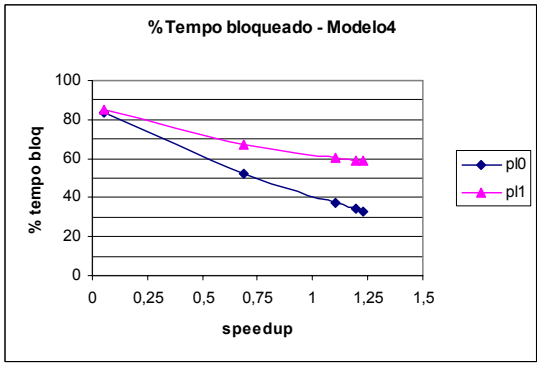


Figura 5.20: Porcentagem de tempo bloqueado do Modelo 4

A porcentagem de tempo bloqueado desnecessário (Figura 5.21) aumentou para o pl0 (entre 16% para granulosidade 0 e 28% para granulosidade 100) e manteve-se para o pl1 (aproximadamente 50%).

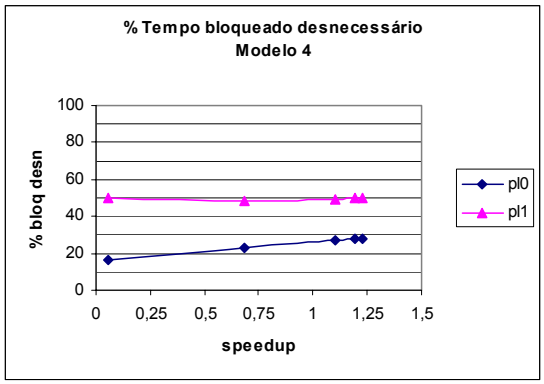


Figura 5.21: Porcentagem de tempo bloqueado desnecessário do Modelo 4

A relação tempo executando / tempo bloqueado é melhor para o pl0 (Figura 5.22). O pl1 não alcança o valor 1 com nenhuma das granulosidades.

No pl0, a porcentagem de mensagens completas que são bloqueios necessários (Figura 5.23) é alta (em torno de 84%) e dessa forma, é possível verificar que o protocolo conservador está tendo o desempenho dentro do esperado.

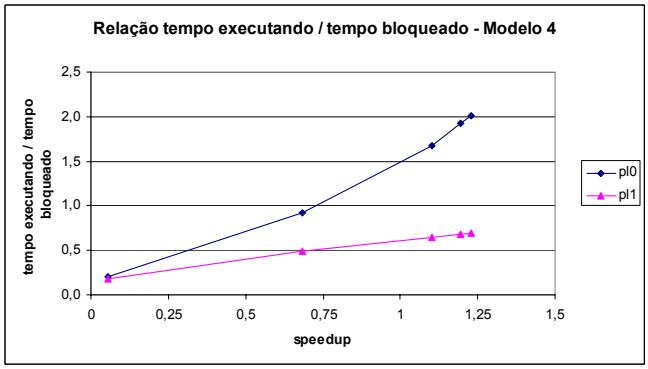


Figura 5.22: Relação tempo executando / tempo bloqueado para o Modelo 4

Já no pl1, a porcentagem de mensagens completas que foram bloqueios necessários (Figura 5.23) foi de aproximadamente 51% e a porcentagem de mensagens nulas que foram bloqueios desnecessários foi de 95%. Porém, os bloqueios desnecessários (Figura 5.24) representam metade do tempo da simulação bloqueado, o que pode representar um indicativo significante para a troca de protocolo para otimista, apesar da alta porcentagem de mensagens completas que são bloqueios necessários.

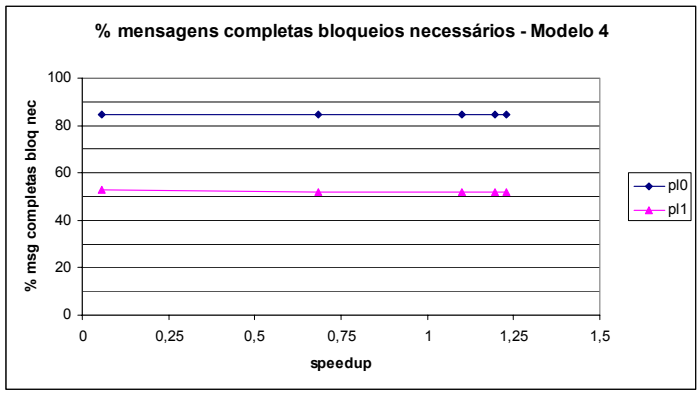


Figura 5.23: Porcentagem de mensagens completas que foram bloqueios necessários para o Modelo 4

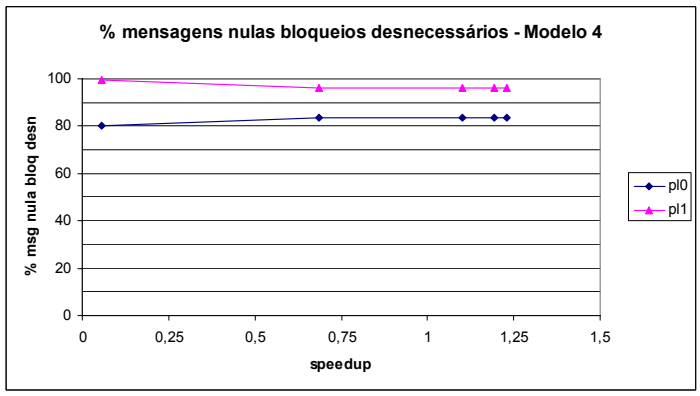


Figura 5.24: Porcentagem de mensagens nulas que foram bloqueios desnecessários para o Modelo 4.

Uma análise detalhada do modelo 4 permite as seguintes conclusões:

- O pl0 para granulosidade fina (0 e 25) tem relação executando / bloqueado com valores abaixo de 1, o que indica que esse processo lógico fica mais tempo bloqueado que executando. A porcentagem de tempo bloqueado é alta (83% e 51% para granulosidade 0 e 25 respectivamente), a porcentagem de tempo bloqueado desnecessariamente é baixa (16% e 23% para granulosidade 0 e 25 respectivamente) e a porcentagem de mensagens que causaram bloqueios necessários é alta (84%). Esses valores mostram que o desempenho do processo lógico está ruim, mas uma mudança de protocolo não irá trazer benefícios para a simulação devido à grande porcentagem de mensagens completas que causaram bloqueios necessários. Nesse caso, a melhor opção é a execução da simulação seqüencial ou MRIP.

- Para o pl0 com granulosidades média e alta (50, 75 e 100) a relação executando / bloqueada é satisfatória (valores acima de 1), o que indica um bom desempenho. A porcentagem de tempo bloqueado é baixa (em média 35%), a porcentagem de tempo bloqueado desnecessário é baixa (27%), a porcentagem de mensagens completas que causaram bloqueios necessários é alta (84%) e a porcentagem de mensagens nulas que foram bloqueios desnecessários á alta (83%). Esses valores indicam que o protocolo conservador deve ser mantido para esse processo lógico.
- O pl1 possui um desempenho ruim para todas as granulosidades, pois a relação executando / bloqueado é sempre menor que 1 nos testes realizados. A porcentagem de tempo bloqueado é sempre alta (varia de 85% a 58% para granulosidade de 0 a 100 respectivamente), a porcentagem de tempo bloqueado desnecessariamente é alta também (aproximadamente 50% para todas as granulosidades), a porcentagem de mensagens completas que causaram bloqueios necessários foi 51% e a porcentagem de mensagens nulas que causaram bloqueios desnecessários foi de 95%. Considerando que o processo lógico possui uma porcentagem bastante alta de bloqueios desnecessários que consomem 50% do tempo da simulação, a mudança de protocolo é indicada nesse caso.

5.2.5 Modelo de redes de fila 1 (fechado)

O modelo de redes de fila 1 foi executado e obteve-se os resultados apresentados na Tabela 5.7e na Figura 5.25:

Tabela 5.7: Resultados obtidos com a simulação do Modelo de redes de fila 1

GRAN	Tempo de execução (Sequencial)	Tempo de execução (Paralelo)	Speedup
0	1,85	49,35	0,04
25	85,51	105,77	0,81
50	524,83	453,57	1,16
75	1612,16	1315,69	1,23
100	3628,10	2909,86	1,25

O modelo de redes de fila 1 mostra os saltos de desempenho que a simulação tem com a adição de granulosidade. Com granulosidade baixa, o modelo apresenta um desempenho extremamente ruim para a simulação distribuída, porém, com a granulosidade 100 ele mostra um bom *speedup* (1,25).

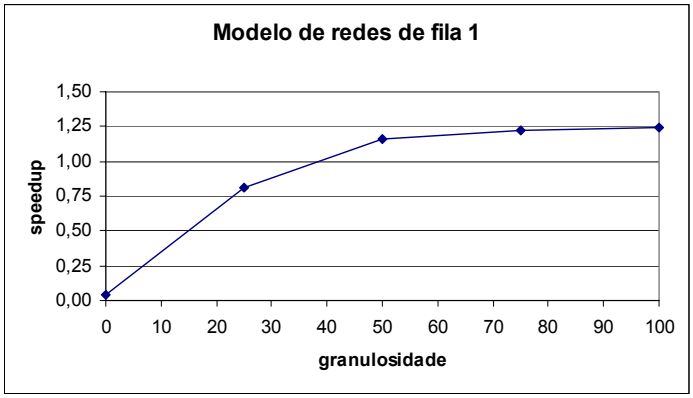


Figura 5.25: Speedup X granulosidade para o modelo de redes de fila 1

A porcentagem de tempo bloqueado do pl0 tem uma queda menor que o pl1, diferente dos modelos anteriormente analisados (modelo 1, modelo 3 e modelo 4, onde a porcentagem de tempo bloqueado era menor no pl0) (Figura 5.26).

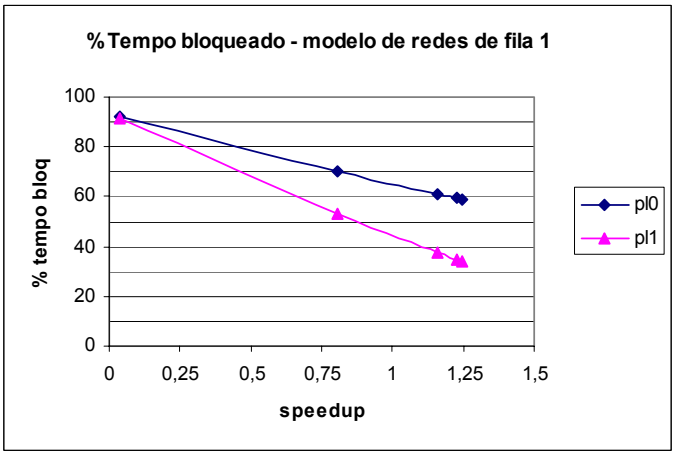


Figura 5.26: Porcentagem de tempo bloqueado no modelo de redes de fila 1

A porcentagem de tempo bloqueado desnecessariamente também é menor no pl1. E no pl0 a partir da granulosidade 50 a porcentagem de tempo bloqueado desnecessário se mantém (Figura 5.27).

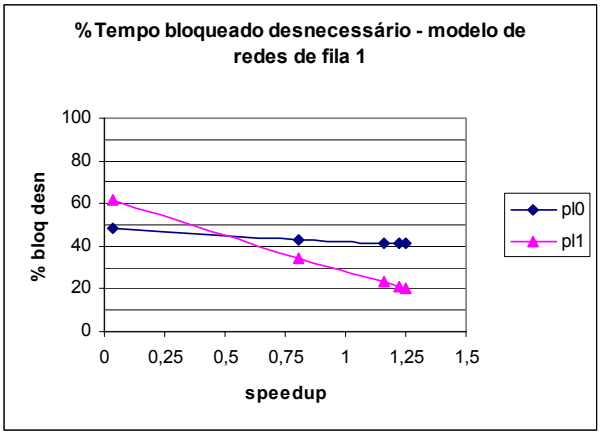


Figura 5.27: Porcentagem de tempo bloqueado desnecessariamente no modelo de redes de fila 1

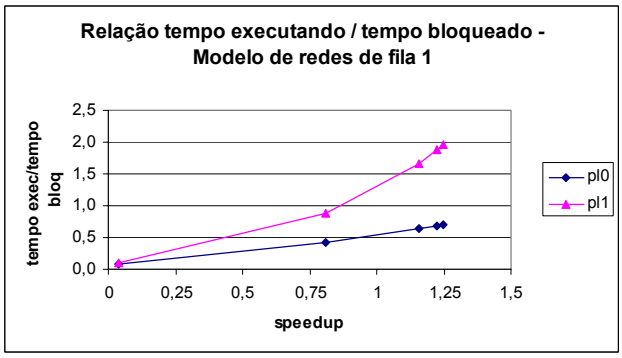


Figura 5.28: Relação tempo executando / tempo bloqueado para o modelo de redes de fila 1

A relação tempo executando / tempo bloqueado é melhor no pl1, onde o pl0 não alcançou o valor 1 para nenhuma das granulosidades analisadas (Figura 5.28). Nesse caso observa-se que o processo lógico 1 está tendo desempenho melhor que o pl 0.

Como é possível ver na Figura 5.29 e na Figura 5.30, a porcentagem de mensagens completas que causaram bloqueios necessários no pl0 é bastante baixo (em torno de 9%) e a porcentagem de mensagens nulas que causaram bloqueios desnecessários é relativamente alto (55%) em relação aos bloqueios necessários. E, no pl1, a porcentagem de mensagens completas que causaram bloqueios necessários foi 65% e a porcentagem de mensagens nulas que causaram bloqueios desnecessários foi de 99%.

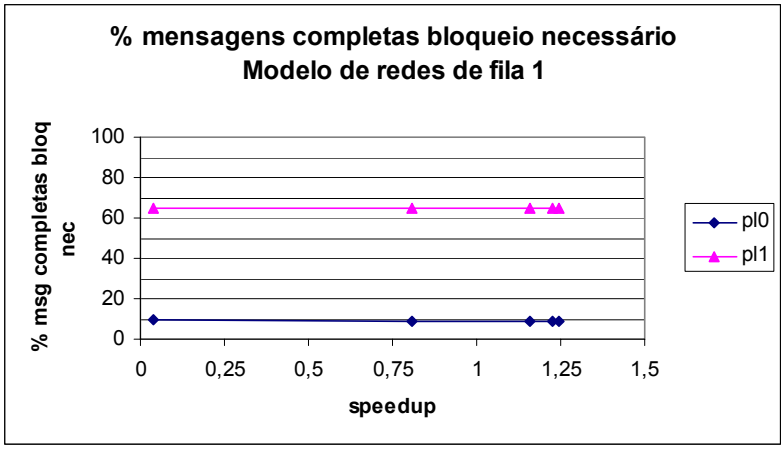


Figura 5.29: Porcentagem de mensagens completas que foram bloqueios necessários do Modelo de redes de fila 1

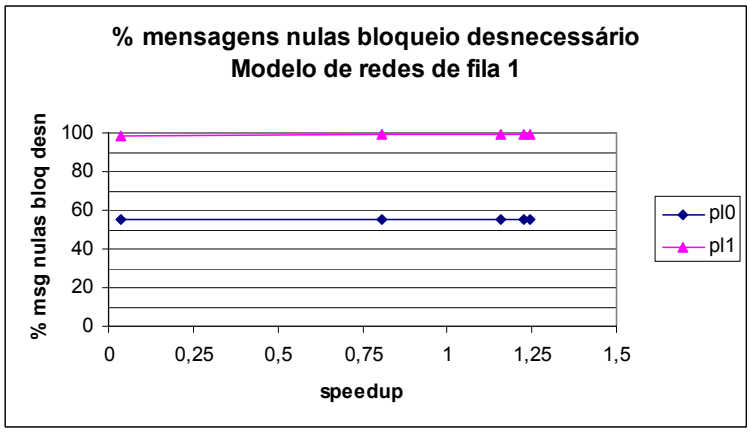


Figura 5.30: Porcentagem de mensagens nulas que foram bloqueios desnecessários do modelo de redes de fila 1

Analisando os resultados observa-se que:

- O pl0 não apresentou uma relação executando / bloqueado maior que 1 para nenhuma das granulosidades, o que indica que teve um baixo desempenho. Analisando-se o tempo bloqueado (variou de 92% até 58% para granulosidades 0 e 100 respectivamente), é possível observar que esse número é bastante alto, e grande parte desse tempo foi de tempo bloqueado desnecessariamente (aproximadamente 42% para todas as granulosidades). Aliando-se isso ao fato desse processo lógico ter uma porcentagem de mensagens completas que causaram bloqueios necessários baixa (9%) e a porcentagem de mensagens nulas que causaram bloqueios desnecessários alta (55%), a troca de protocolo para otimista é indicada.

- No caso do pl1, para granulosidade fina (0 e 25) a relação executando / bloqueado foi menor que 1, a porcentagem de tempo bloqueado foi alta (91% e 52% para granulosidade 0 e 25 respectivamente) e a porcentagem de tempo bloqueado desnecessariamente foi alta também (61% e 34% para granulosidade 0 e 25 respectivamente). A porcentagem de mensagens completas que causaram bloqueios necessários é alta (65%) e inviabiliza uma troca de protocolo, mesmo com o desempenho ruim apresentado pela relação executando / bloqueado. Nesse caso, é indicado o uso da simulação seqüencial ou MRIP.
- O processo lógico 1 possui uma relação executando / bloqueado boa para granulosidades acima de 50, e a porcentagem de mensagens completas que são bloqueios necessários é alta (em torno de 65 %) o que mostra que o desempenho do protocolo conservador está bom.

5.2.6 Modelo de redes de fila 2 – configuração 1, 2 e 3

O modelo de redes de fila 2 foi executado com 3 tipos de particionamentos diferentes. Os resultados podem ser vistos na Tabela 5.8 e Figura 5.31.

Tabela 5.8: Resultados obtidos com a simulação do modelo de redes de fila 2, configurações 1, 2 e 3

GRAN	Speedup (configuração 1)	Speedup (configuração 2)	Speedup (configuração 3)
0	0,306	0,363	0,335
25	0,809	0,823	0,854
50	1,000	0,967	1,041
75	1,036	0,994	1,076
100	1,046	0,999	1,087

Como é possível observar, os *speedups* das três configurações estão aproximados. Apesar da configuração 2 mostrar o melhor speedup para granulosidade 0, com o aumento da granulosidade o *speedup* alcançado foi o pior. Isso pode ser explicado pelo estado de desbalanceamento do modelo, que sobrecarrega o processo lógico 0. A configuração que possui o melhor parcionamento (levando-se em conta comunicação e balanceamento de carga) é a configuração 3, que apresentou o melhor *speedup* com a granulosidade mais grossa.

A execução desse mesmo modelo com três particionamentos diferentes mostra a importância que diferentes características podem assumir na simulação. Granulosidades mais altas mostram que o balanceamento de carga torna-se um fator mais importante para um bom desempenho do que a redução do custo da comunicação em rede. Porém, em simulações com pouca granulosidade, o speedup alcançado é ruim, mesmo com um bom balanceamento de carga. Isso confirma os resultados obtidos em (Xu & Chung, 2004) onde os autores propõem um modelo que faz previsões sobre o desempenho de uma simulação. Este modelo considera principalmente o balanceamento de carga e a comunicação.

O que é possível observar também é que este modelo não apresenta bom desempenho com o protocolo conservador, pois apresentou um *speedup* bastante baixo, mesmo com granulosidade grossa, em todas as configurações testadas. (Figura 5.31)

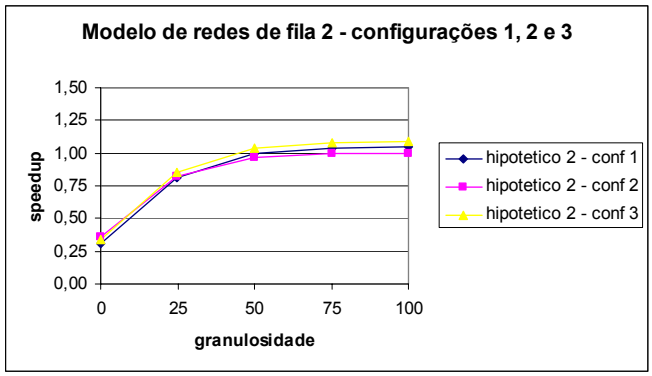


Figura 5.31: Speedup X granulosidade para o modelo de redes de fila 2, configurações 1, 2 e 3

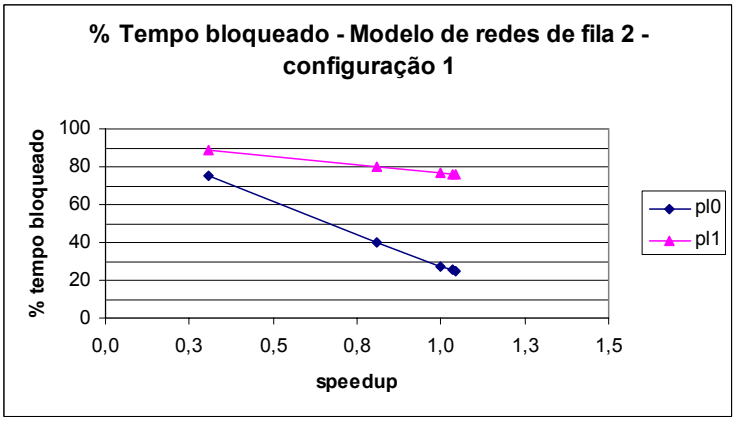


Figura 5.32: Porcentagem de tempo bloqueado para o modelo de redes de fila 2, configuração 1

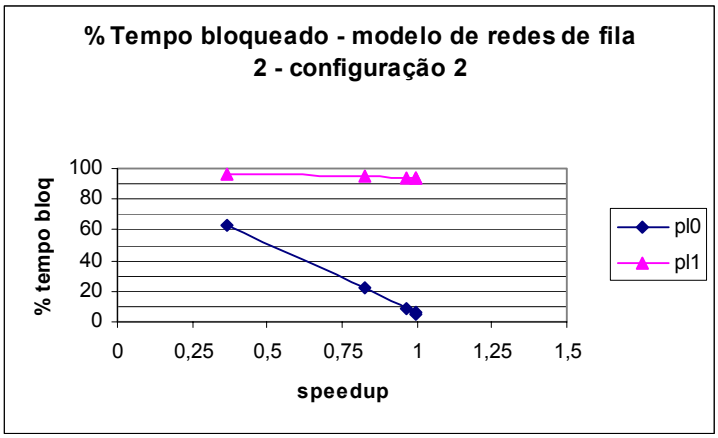


Figura 5.33: Porcentagem de tempo bloqueado para o modelo de redes de fila 2, configuração 2

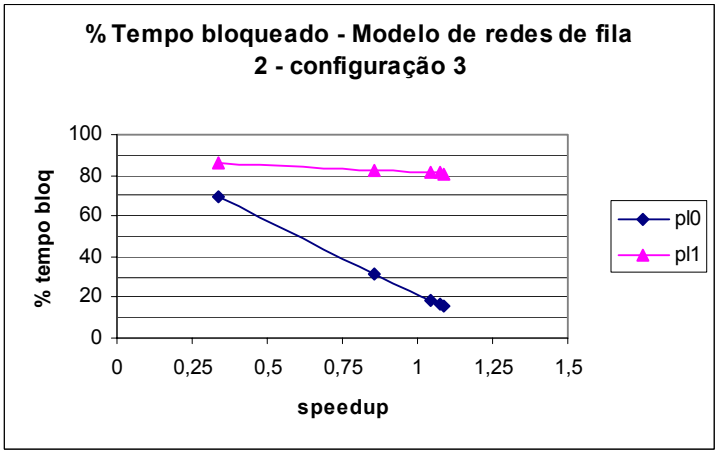


Figura 5.34: Porcentagem de tempo bloqueado para o modelo de redes de fila 2, configuração 3

Em relação ao comportamento do tempo bloqueado no modelo, nas três configurações (Figura 5.32, Figura 5.33 e Figura 5.34), observa-se que o processo lógico 0 apresenta uma queda bastante acentuada na porcentagem de tempo bloqueado com o aumento da granulosidade. A configuração 2 apresenta as menores porcentagens e atinge o menor valor com a maior granulosidade (4,94%). O mesmo não acontece com o processo lógico 1, que permanece com porcentagens altas de bloqueio para as três configurações. O pior caso, ou seja, a maior porcentagem de bloqueio pode ser verificada na configuração 2. Isso pode ser explicado devido ao desbalanceamento de carga que existe nessa configuração, ou seja, o processo lógico 0 está muito carregado e concentra a maior parte do processamento e dessa forma fica pouco bloqueado na maior granulosidade (4,94%) e o processo lógico 1 que está com pouca carga passa a maior

parte do tempo bloqueado, mesmo com a maior granulosidade (94,13%).

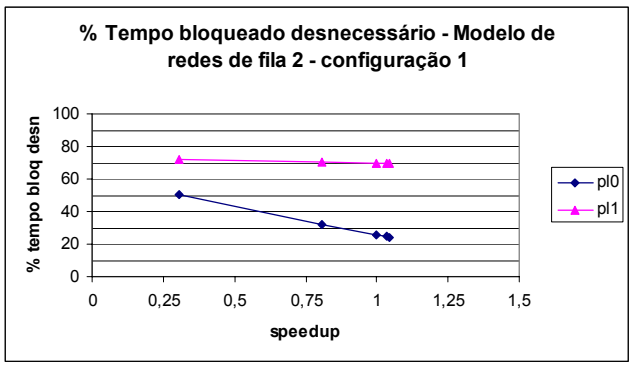


Figura 5.35: Porcentagem de tempo bloqueado desnecessariamente para o modelo de redes de fila 2, configuração 1

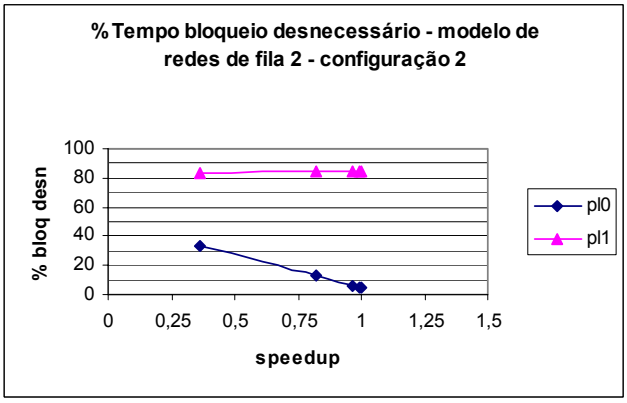


Figura 5.36: Porcentagem de tempo bloqueado desnecessariamente para o modelo de redes de fila 2, configuração 2.

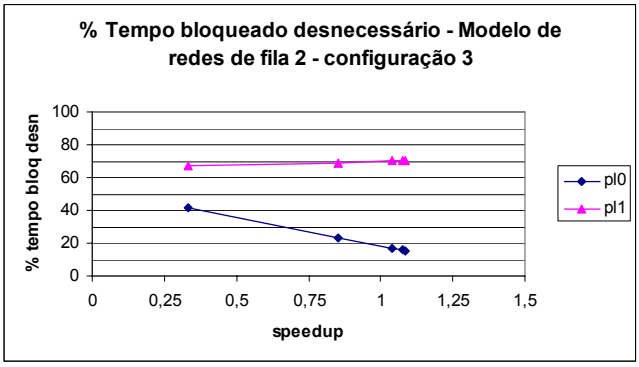


Figura 5.37: Porcentagem de tempo bloqueado desnecessariamente, modelo de redes de fila 2, configuração 3.

Em relação ao tempo bloqueado desnecessariamente (Figura 5.35, Figura 5.36 e Figura 5.37) pode-se notar que a porcentagem diminui para o processo lógico 0 nas três configurações, mas para o processo lógico 1 há até aumento na porcentagem na configuração 2 e 3 e uma leve queda na configuração 1. Isso mostra que o desempenho

do processo lógico 0 pode ser considerado bom para o protocolo conservador para granulosidades mais grossas, porém, o processo lógico 1 não está tendo bom desempenho, uma vez que permanece grande parte do tempo bloqueado. Mesmo com o aumento da granulosidade, a porcentagem de tempo bloqueado não se altera de forma significativa.

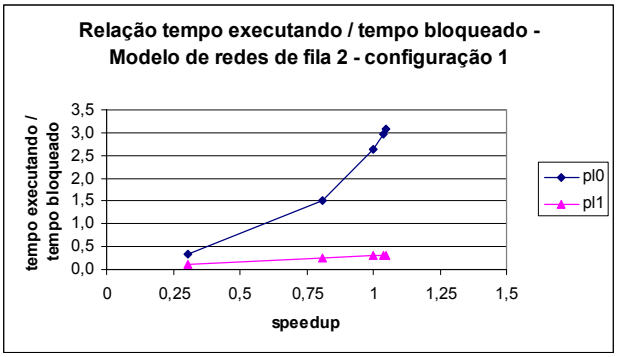


Figura 5.38: Relação tempo executando / tempo bloqueado para o modelo de redes de fila 2, configuração 1

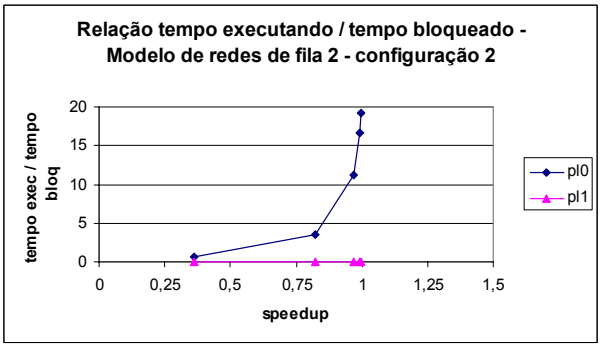


Figura 5.39: Relação tempo executando / tempo bloqueado para o modelo de redes de fila 2, configuração 2

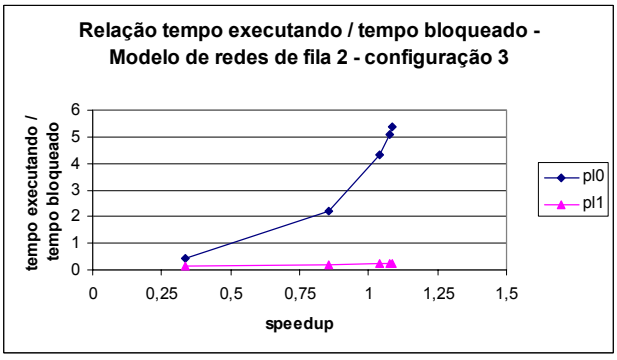


Figura 5.40: Relação tempo executando / tempo bloqueado para o modelo de redes de fila 2, configuração 3

Na questão da relação tempo executando / tempo bloqueado (Figura 5.38, Figura 5.39 e Figura 5.40) pode-se novamente verificar que o desempenho do processo lógico 1 está bem aquém do processo lógico 0. No gráfico da Figura 5.39, que retrata a configuração 2, nota-se um aumento muito grande nessa relação para o processo lógico 0 e uma estagnação para o processo lógico 1.

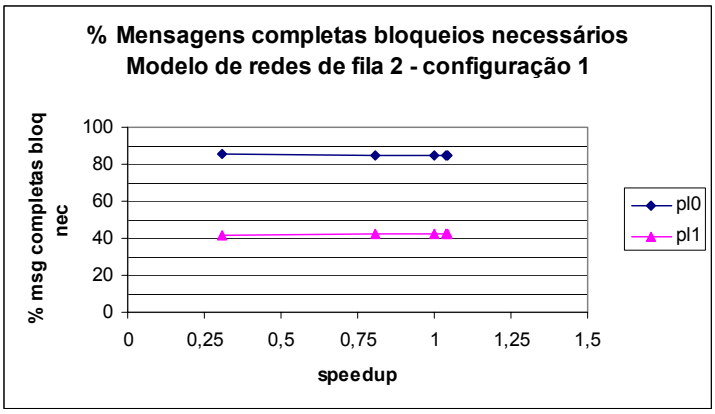


Figura 5.41: Porcentagem de mensagens completas que foram bloqueios necessários no modelo de redes de fila 2, configuração 1.

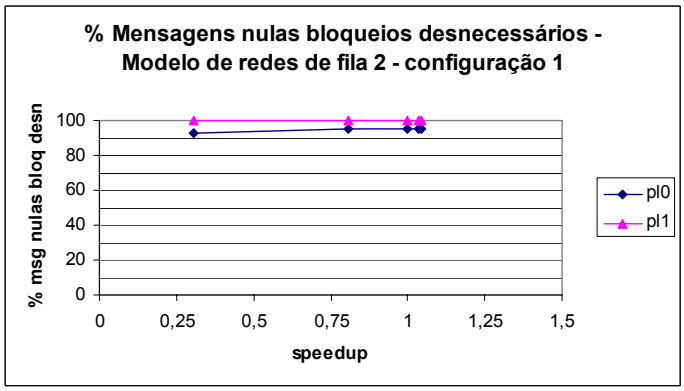


Figura 5.42: Porcentagem de mensagens nulas que foram bloqueios desnecessários no modelo de redes de fila 2, configuração 1.

O comportamento das mensagens para a configuração 1 mostra que a maior parte das mensagens completas recebidas pelo processo lógico 0 (mais que 80% - Figura 5.41) foram bloqueios necessários, o que indica que esse processo lógico está com bom desempenho para granulosidades mais grossas. Apesar da porcentagem de mensagens nulas que foram bloqueios desnecessários ser alto, a porcentagem de tempo bloqueado desnecessariamente (Figura 5.35) nesse processo lógico é baixo o que mostra um bom desempenho.

Verificando-se, por outro lado, o processo lógico 1 é possível verificar que a porcentagem de mensagens completas que foram bloqueios necessários fica em torno de 40% para todas as granulosidades e a porcentagem de mensagens nulas que foram bloqueios desnecessários ficou em torno de 99%. Esses números, somada a porcentagem alta de tempo bloqueado desnecessário (em torno de 70% - Figura 5.35) mostra que uma mudança de protocolo para otimista pode trazer um desempenho melhor para a simulação uma vez que esse processo lógico poderia ter executado diversos eventos sem causar problemas de causa e efeito.

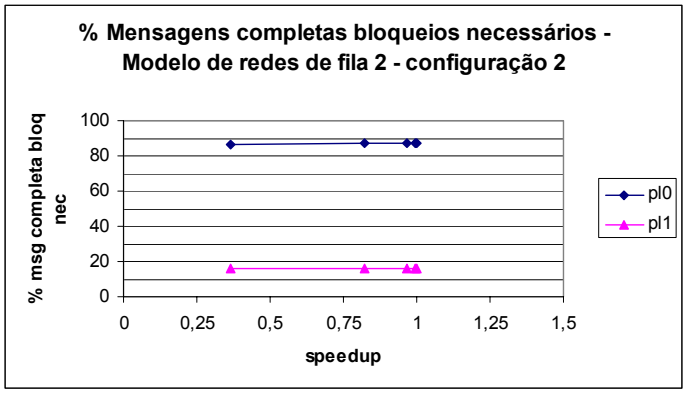


Figura 5.43: Porcentagem de mensagens completas que foram bloqueios necessários no modelo de redes de fial 2, configuração 2.

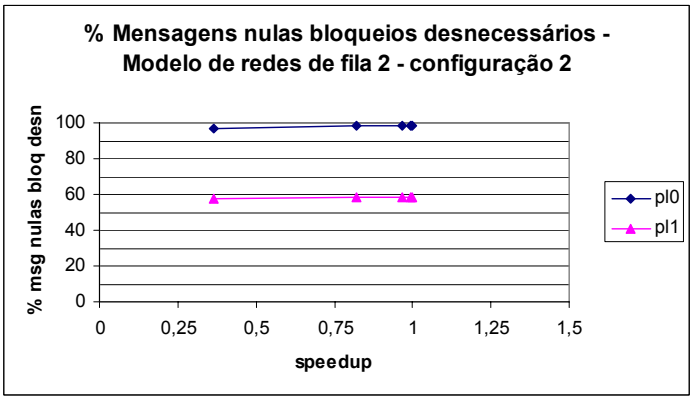


Figura 5.44: Porcentagem de mensagens nulas que foram bloqueios desnecessários no modelo de redes de fila 2, configuração 2.

Este modelo, particionado de três formas diferentes, e onde uma das partições procura otimizar comunicação e carga, mostra que alguns modelos não se adaptam ao protocolo conservador. Apesar de apresentar algum *speedup* (melhor para a configuração 3) esse *speedup* pode ser considerado quase desprezível (1,087) para a granulosidade mais grossa. Nesse sentido, a simulação seqüencial ou a simulação com o

protocolo otimista podem ser consideradas.

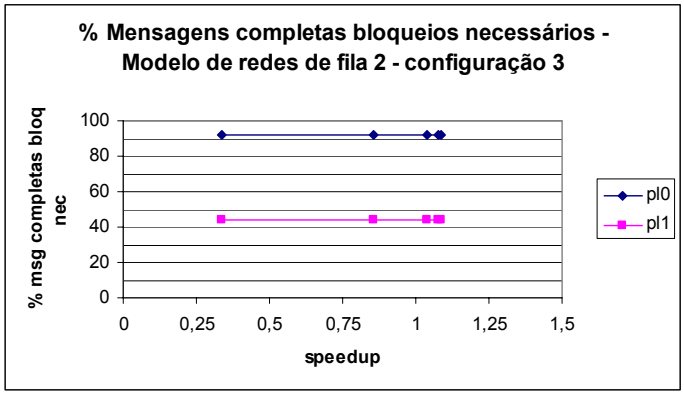


Figura 5.45: Porcentagem de mensagens completas que foram bloqueios necessários no modelo de redes de fila 2, configuração 3.

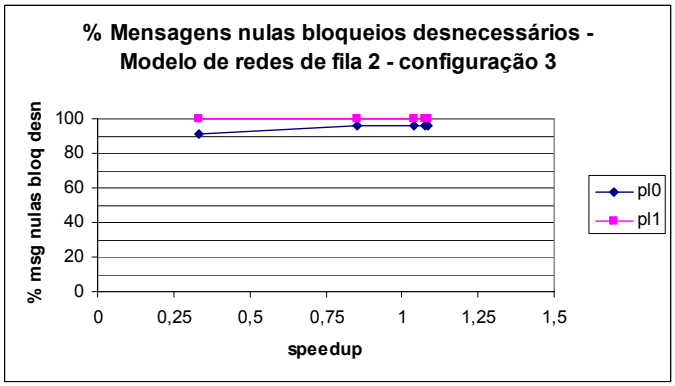


Figura 5.46: Porcentagem de mensagens nulas que foram bloqueios desnecessários no modelo de redes de fila 2, configuração 3.

Analisando isoladamente cada uma das configurações do modelo de redes de fila 2 pode-se chegar a algumas conclusões:

Modelo de redes de fila 2 – configuração 1

- O *pl0* apresentou uma relação executando / bloqueado com valores acima de um para granulosidades maiores que 25, o que mostra que esse processo lógico teve um bom desempenho. E, como a porcentagem de mensagens completas que causaram bloqueios necessários foi alta (85%), o protocolo conservador deve ser mantido.
- Para granulosidade 0, o *pl0* apresentou uma relação executando / bloqueado com valor abaixo de 1, o que mostra que esse processo lógico passou mais tempo

bloqueado que executando. Considerando que a porcentagem de mensagens completas que causaram bloqueios necessários foi de 85%, a melhor opção é a simulação seqüencial ou MRIP.

- No caso do pl1, a relação executando / bloqueado foi abaixo de um para todas as granulosidades, nesse caso, deve-se verificar os demais valores para decidir se uma troca de protocolo deve ser considerada. A porcentagem de mensagens completas que causaram bloqueios necessários foi razoavelmente baixa (42%), porém, a porcentagem de mensagens que foram bloqueios necessários foi bastante alta (99%). E, a porcentagem de tempo bloqueado desnecessariamente foi bastante alta também (acima de 69% para todas as granulosidades). Nesse caso, o protocolo otimista é a melhor opção.

Modelo de redes de fila 2 – configuração 2

- O *pl0* apresentou valores bastante altos para a relação executando / bloqueado para granulosidades maiores que 25 (3,56 para granulosidade 25 crescendo gradativamente até 19,24 para granulosidade 100). Esses valores extremamente altos (considerando que existem apenas 2 processos lógicos) evidencia um desbalanceamento muito grande entre os processos lógicos, e, nesse caso particular, um desbalancemanto no particionamento do modelo. Esses valores mostram que o processamento está concentrado nesse processo lógico e provavelmente o outro processo lógico deve estar bastante ocioso. Nesse caso, o processamento desse processo lógico está bom e isso pode ser confirmado pela porcentagem alta de mensagens completas que causaram bloqueios necessários (87%).
- Em relação ao *pl0* em granulosidade 0, verifica-se que a relação executando / bloqueado é ruim (0,59) e, devido ao valor alto de mensagens completas que causaram bloqueios necessários (86%) esse processo lógico não deve ser mudado para otimista. Nesse caso, o mais indicado é a simulação seqüencial ou MRIP.
- Para o *pl1*, os valores da relação executando / bloqueado são bastante baixos (valores menores que 0,063) o que confirma o desbalanceamento do modelo. A

porcentagem de tempo bloqueado é bastante alto (94%), a porcentagem de mensagens completas que causaram bloqueios necessários é baixa (16%) e a porcentagem de mensagens nulas que causaram bloqueios desnecessários foi razoavelmente alto (58%). Nesse caso, uma mudança de protocolo para otimista é o indicado.

Modelo de redes de fila 2 – configuração 3

- O *pl0* apresenta uma relação executando / bloqueado com valores acima de um a partir da granulosidade 25. Para essas granulosidades, verifica-se que o processo lógico tem bom desempenho com o protocolo conservador, e a porcentagem de mensagens que causaram bloqueios necessários é alta (92%).
- Para granulosidade 0, o *pl0* apresenta um valor baixo (0,43), e como a porcentagem de mensagens completas que causaram bloqueios necessários é alta, a mudança para protocolo otimista não irá trazer vantagem. Assim, o uso de simulação seqüencial ou MRIP é o mais indicado.
- O *pl1* apresentou valores baixos para a relação executando / bloqueado para todas as granulosidades (valores menores que 0,24). Como a porcentagem de mensagens que causaram bloqueios necessários é baixa (44%) e a porcentagem de mensagens nulas que causaram bloqueios desnecessários é alta (99%) a mudança de protocolo para otimista deve ser considerada.

5.3 Análise geral dos resultados obtidos com a simulação dos modelos

Esta seção apresenta uma análise geral dos resultados obtidos com os experimentos realizados. As diversas características avaliadas são consideradas, possibilitando diversas conclusões em relação a predição do desempenho de uma simulação distribuída durante sua execução. Os resultados obtidos permitem não apenas identificar quais características da simulação influenciam o seu desempenho, mas também quantificar que valores irão comprometer esse desempenho.

Visando a realização de uma análise geral dos resultados obtidos, a seção 5.3.1 apresenta um resumo dos valores mais importantes e uma análise dos resultados considerando o modelo como um todo. A seção 5.3.2 apresenta uma análise quantitativa dos resultados, apresentando para que valores de cada característica, mudanças no protocolo utilizado devem ser consideradas. Para essa análise é considerado o modelo completo. A seção 5.3.3 apresenta os resultados e as conclusões sobre o comportamento de cada processo lógico das simulações.

5.3.1 Análise qualitativa dos resultados

O gráfico da Figura 5.47 mostra os *speedups* obtidos na execução dos modelos descritos nesse capítulo. Como é possível observar, o melhor *speedup* é atingido com o modelo de redes de fila 1, e o pior desempenho é apresentado pelo modelo 1, configuração 2 com três processos lógicos.

Em linhas gerais, foram obtidos os seguintes resultados:

- A porcentagem de tempo bloqueado diminui com o aumento da granulosidade.
- A porcentagem de tempo bloqueado desnecessário diminui com o aumento da granulosidade.
- A relação tempo executando / tempo bloqueado aumenta com o aumento da granulosidade.
- A porcentagem do número de mensagens completas que causaram bloqueios necessários em relação ao total de mensagens completas que chegou ao processo lógico pode ser correlacionada com o tempo bloqueado de forma a delinear o desempenho do processo lógico e assim definir se o desempenho está bom ou ruim e se uma troca de protocolo deve ser considerada.
- Da mesma forma é possível verificar se a maior parte das mensagens nulas bloqueou o processo lógico desnecessariamente e qual foi a porcentagem de tempo perdido com esse bloqueio para verificar se a troca deve ser considerada.

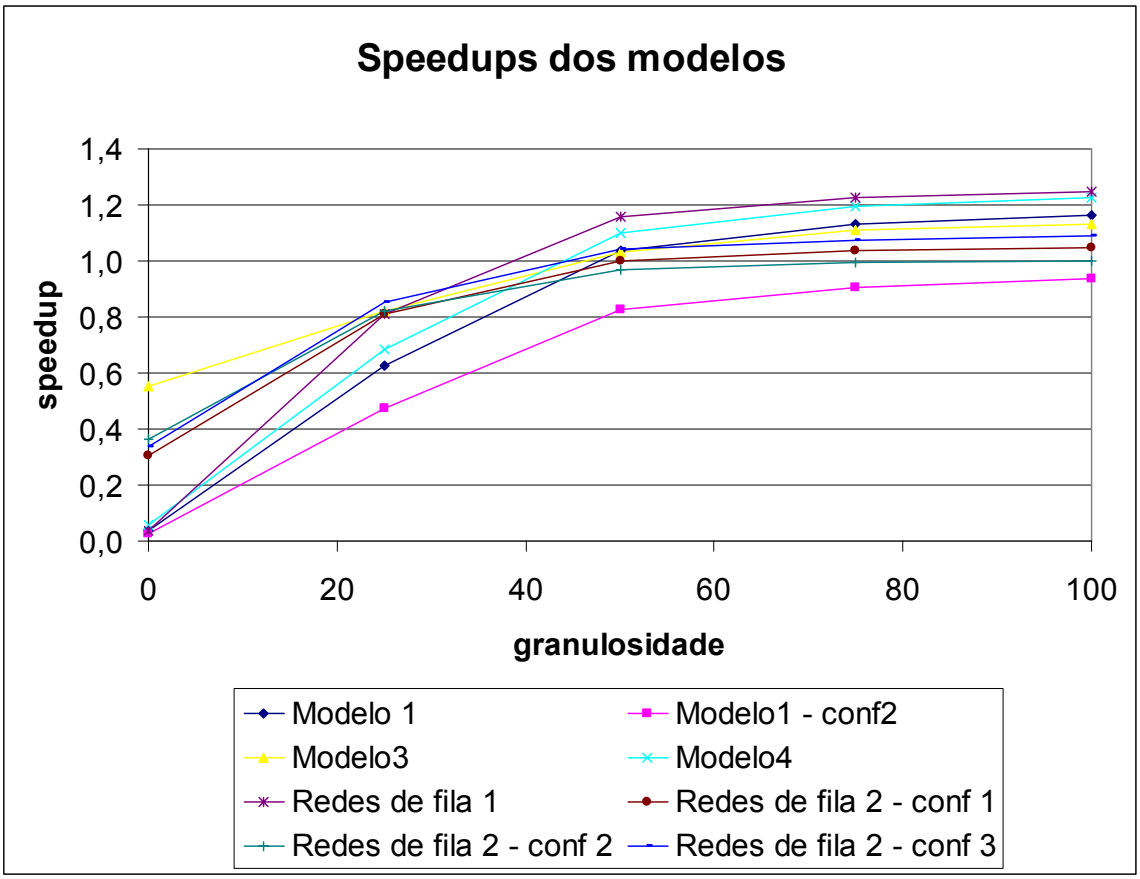


Figura 5.47: Gráfico com os *speedups* atingidos por todos os modelos executados.

Uma outra análise que pode ser feita com as simulações dos modelos deste capítulo pode ser vista nas tabelas 5.9 até 5.16. Nessas tabelas, tem-se: na primeira coluna a granulosidade, na segunda e terceira colunas a relação executando / bloqueado dos *pls* (na tabela 5.10 tem-se também o pl2), na quarta coluna a média das relações executando / bloqueado dos *pls*, na quinta coluna tem-se o desvio padrão e na última coluna o *speedup* atingido pelo modelo.

Tabela 5.9: Modelo 1

GRAN	pl0 (E/B)	pl1 (E/B)	Média (E/B)	desvio padrão	Speedup
0	0,207	0,137	0,172	0,035	0,037
25	0,759	0,507	0,633	0,126	0,625
50	1,208	0,777	0,992	0,216	1,037
75	1,350	0,850	1,100	0,250	1,132
100	1,389	0,875	1,132	0,257	1,161

Tabela 5.10: Modelo 1 - configuração 2

GRAN	pl0 (E/B)	pl1 (E/B)	pl2 (E/B)	Média (E/B)	desvio padrão	Speedup
0	0,114	0,038	0,112	0,088	0,036	0,025
25	0,474	0,143	0,424	0,347	0,146	0,476
50	0,771	0,209	0,790	0,590	0,270	0,828
75	0,846	0,223	0,905	0,658	0,308	0,908
100	0,877	0,228	0,934	0,680	0,320	0,935

Tabela 5.11: Modelo 3

GRAN	pl0 (E/B)	pl1 (E/B)	Média (E/B)	desvio padrão	Speedup
0	0,654	0,045	0,350	0,305	0,552
25	1,444	0,151	0,798	0,647	0,819
50	3,399	0,277	1,838	1,561	1,031
75	4,693	0,323	2,508	2,185	1,110
100	5,346	0,337	2,842	2,505	1,129

Tabela 5.12: Modelo 4

GRAN	pl0 (E/B)	pl1 (E/B)	Média (E/B)	desvio padrão	Speedup
0	0,201	0,174	0,187	0,014	0,056
25	0,927	0,493	0,710	0,217	0,684
50	1,673	0,648	1,161	0,513	1,101
75	1,926	0,686	1,306	0,620	1,195
100	2,012	0,697	1,355	0,657	1,228

Tabela 5.13: Modelo de redes de fila 1

GRAN	pl0 (E/B)	pl1 (E/B)	Média (E/B)	desvio padrão	Speedup
0	0,081	0,095	0,088	0,007	0,038
25	0,428	0,889	0,658	0,230	0,808
50	0,637	1,650	1,143	0,507	1,157
75	0,684	1,885	1,285	0,600	1,225
100	0,699	1,960	1,330	0,630	1,247

Tabela 5.14: Modelo de redes de filas 2 - configuração 1

GRAN	pl0 (E/B)	pl1 (E/B)	Média (E/B)	desvio padrão	Speedup
0	0,325	0,123	0,224	0,101	0,306
25	1,518	0,247	0,883	0,635	0,809
50	2,625	0,301	1,463	1,162	1,000
75	2,957	0,313	1,635	1,322	1,036
100	3,068	0,316	1,692	1,376	1,046

Tabela 5.15: Modelo de redes de fila 2 - configuração 2

GRAN	pl0 (E/B)	pl1 (E/B)	Média (E/B)	desvio padrão	Speedup
0	0,591	0,039	0,315	0,276	0,363
25	3,559	0,057	1,808	1,751	0,823
50	11,255	0,061	5,658	5,597	0,967
75	16,669	0,062	8,365	8,303	0,994
100	19,244	0,062	9,653	9,591	0,999

Tabela 5.16: Modelo de redes de fila 2 - configuração 3

GRAN	pl0 (E/B)	pl1 (E/B)	Média (E/B)	desvio padrão	Speedup
0	0,433	0,157	0,295	0,138	0,335
25	2,186	0,213	1,199	0,986	0,854
50	4,327	0,229	2,278	2,049	1,041
75	5,091	0,233	2,662	2,429	1,076
100	5,354	0,235	2,794	2,560	1,087

Analisando-se essas tabelas, verifica-se que para valores acima de um ou muito próximo de um para a média, combinado com valores baixos para o desvio padrão (abaixo de um), obtêm-se *speedup*.

Valores altos de desvio padrão (acima de 2) indicam desbalanceamento no modelo. Os exemplos são: o Modelo 3 (Tabela 5.11) e Modelo de redes de fila 2 – configuração 2 e configuração 3 (Tabelas 5.15 e 5.16). No caso do modelo 3, como já discutido, trata-se de desbalancemento intrínseco do modelo (parametrização). Uma forma de balancear esse modelo seria mudar os seus parâmetros, uma vez que não é possível outra forma de particionamento (o modelo é composto por 2 recursos).

No caso do modelo de redes de fila 2, configurações 2 e 3, o desbalanceamento é mais evidente na configuração 2, onde atinge o valor 9 para granulosidade 100. Essa configuração mostra valores altos para a média e também valores altos para o desvio padrão, o que evidencia ainda mais o desbalanceamento.

A configuração 3 tem um melhor balanceamento, porém como ela também procura minimizar a comunicação o balanceamento não é perfeito. Porém a discrepância da relação executando / bloqueado dos processos lógicos é bem menor se comparada com a configuração 2.

Os resultados obtidos permitem ainda conclusões sobre os problemas relacionados a granulosidade e ao balanceamento dos processos lógicos.

Uma das características que deve ser observada é que a contagem de bloqueios necessários e desnecessários não se modifica com o aumento da granulosidade, o que mostra que a utilização dessa métrica isolada não é viável para a análise de desempenho da simulação. Nesse caso, a informação de bloqueios poderá ser utilizada em conjunto com o tempo bloqueado do processo lógico.

Para granulosidade fina, a simulação paralela não permite um desempenho maior, nesse caso, a utilização de simulação seqüencial pode ser a mais indicada. Isso se deve ao fato de que modelos com granulosidade fina passam mais tempo bloqueados que executando, o que degrada o desempenho. Todos os modelos que não apresentam *speedup* em nenhuma granulosidade mostram que uma melhora no desempenho pode ser a simulação seqüencial.

Modelos que apresentam desbalanceamento em seu particionamento não produzem bom desempenho com a simulação distribuída. Dessa forma, a utilização da simulação seqüencial ou MRIP pode ser mais atrativa, ou então, é indicada a utilização de ferramentas que particionam o modelo de forma a minimizar a comunicação e uniformizar as cargas entre os processos lógicos.

No caso específico de modelos desbalanceados, a mudança de protocolo pode melhorar em parte o desempenho. Isso pode ser explicado devido a sobrecarga que um dos processos lógicos já enfrenta. O envio de mais mensagens a esse processo lógico não consegue acelerar a simulação, pois essas mensagens a mais tendem apenas a serem colocados em fila, não promovendo mais envios de mensagens para o processo lógico ocioso. A ocorrência de *rollbacks* nesse processo lógico pode se tornar mais freqüente, degradando ainda mais o desempenho da simulação.

5.3.2 Análise quantitativa dos resultados

A Figura 5.47 e a Tabela 5.17 resumem o comportamento dos modelos estudados e permitem conclusões sobre o desempenho da simulação distribuída durante sua execução. Com estes resultados pode-se verificar quando é interessante considerar uma troca de protocolo, quando se deve manter o protocolo atual e os casos em que a simulação distribuída SRIP não é apropriada. Neste último caso, deve-se considerar a

execução MRIP ou alterações no número de *pls* considerados ou na divisão e mapeamento dos processos.

Em relação a troca de protocolo ou não, e sobre o comportamento dos modelos, pode-se concluir que:

- Para valores de relação executando / bloqueado maiores que 1 e porcentagem de mensagens completas que causaram bloqueios necessários maior que 50%, o protocolo conservador deve ser mantido.
- Para valores de relação executando / bloqueado menores que 1 e porcentagem de mensagens completas que causaram bloqueios necessários maior que 50%, a simulação seqüencial ou MRIP deve ser considerada.
- Para valores de relação executando / bloqueado menores que 1, porcentagem de mensagens completas que causaram bloqueios necessários menor que 55% e porcentagem de mensagens nulas que causaram bloqueios desnecessários maior que 50% o protocolo otimista deve ser considerado.
- O processo lógico 1 do Modelo 3 pode ser considerado uma exceção à regra pois relação executando / bloqueado é menor que 1, a porcentagem de mensagens completas que causaram bloqueios necessários foi de 9% e a porcentagem de mensagens nulas que causaram bloqueios desnecessários foi de 44% (abaixo dos 50% previstos para a mudança para o protocolo otimista). Porém, considerando-se que a porcentagem de bloqueios necessários é menor que a de bloqueios desnecessários e o processo lógico tem uma porcentagem de tempo bloqueado alta (75%) o protocolo otimista pode ser considerado.

O diagrama da Figura 5.48 resume a análise apresentada. São consideradas, neste caso, três características principais para a verificação do desempenho do processo lógico: a relação executando / bloqueado, a porcentagem de mensagens completas que causaram bloqueios necessários e a porcentagem de mensagens nulas que causaram bloqueios desnecessários.

Esses valores permitem classificar o desempenho da simulação de acordo com os valores que são obtidos com a sua execução. Quando se tem valores executando / bloqueado maiores que um, pode-se concluir que a simulação conservadora está tendo bom desempenho. Os modelos considerados nesta tese não obtiveram valores menores

que 55% para a porcentagem de mensagens completas que causaram bloqueios necessários, porém, nesse caso, pode-se concluir que o protocolo conservador está tendo bom desempenho, pois ele passa mais tempo executando do que bloqueado. Assim, apesar de nenhum modelo ter atingido esses valores, caso algum processo lógico apresente esse comportamento, o protocolo CMB deve ser mantido por estar obtendo um bom desempenho (Figura 5.48).

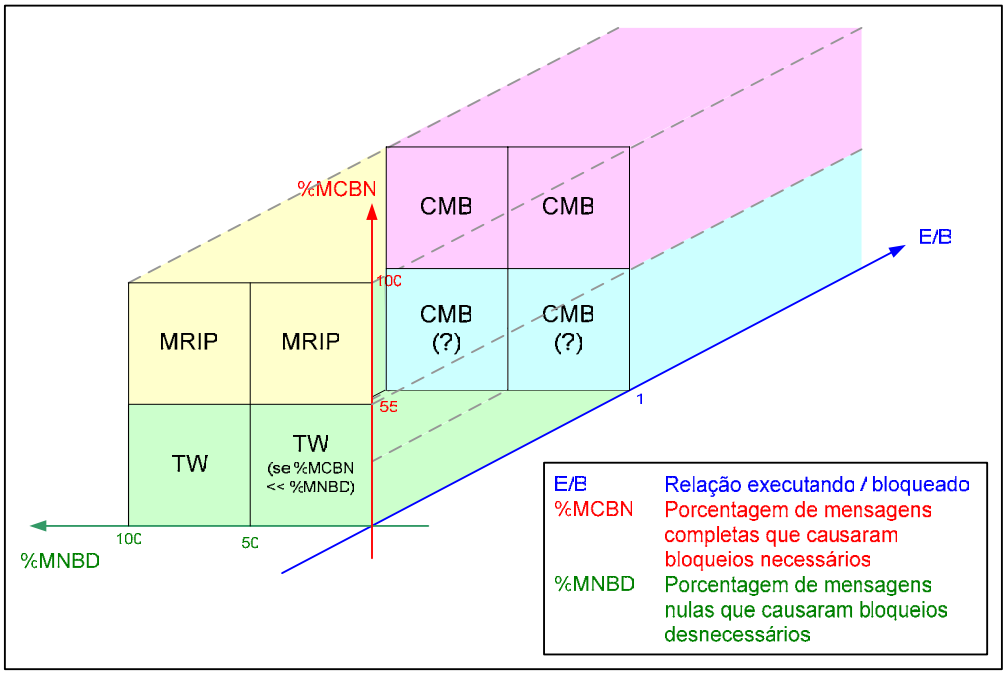


Figura 5.48: Diagrama dos protocolos que devem ser utilizados de acordo com as características apresentadas pelo processo lógico.

A Tabela 5.17 mostra os resultados obtidos com as simulações efetuadas. As colunas possuem os seguintes campos:

- Coluna 1: Modelo
- Coluna 2: processo lógico
- Coluna 3: Granulosidade considerada
- Coluna 4: Relação executando / bloqueado
- Coluna 5: Porcentagem de mensagens completas que causaram bloqueios necessários
- Coluna 6: Porcentagem de mensagens nulas que causaram bloqueios desnecessários.

- Coluna 7: Porcentagem de tempo bloqueado
- Coluna 8: Porcentagem de tempo bloqueado desnecessário
- Coluna 9: Protocolo mais adequado.

Tabela 5.17: Resultados obtidos com todas as simulações efetuadas.

Modelo	*pl*	GRAN	E/B	%MCBN	%MNBD	%TB	%TBD	protocolo
1	0	média e alta	>1	~75	~77	<50	<30	conservador
	0	baixa	<1	~75	~77	>56	>17	seq ou MRIP
	1	todas	<1	~51	~95	87 - 53	~47	otimista
1 (conf 2)	0	todas	<1	~72	~47	~53	22 - 39	seq ou MRIP
	1	todas	<<1	~43	~60	~81	~65	otimista
	2	todas	<1	~42	~63	~50	~44	otimista
3	0	média e alta	>1	~71	~98	22-15	18-14	conservador
	0	baixa	<1	~71	~98	60-40	38-28	seq ou MRIP
	1	todas	<1	~9	~44	75	53	otimista
4	0	média e alta	>1	~84	~83	33-37	27	conservador
	0	baixa	<1	~84	~83	83-51	16-23	seq ou MRIP
	1	todas	<1	~51	~95	85-58	50	otimista
redes de fila 1	0	todas	<1	~9	~55	92-58	41	otimista
	1	média e alta	>1	~65	~99	37-33	23-21	conservador
	1	baixa	<1	~65	~99	91-52	61-35	seq ou MRIP
redes de fila 2 (conf 1)	0	média e alta	>1	~85	~95	28	26	conservador
	0	baixa - 25	>1	~86	~96	39	31	conservador
	0	baixa - 0	<1	~87	~97	75	50	seq ou MRIP
	1	todas	<<1	~42	~99	76	69	otimista
redes de fila 2 (conf 2)	0	média e alta	>1	~87	~98	9	6	conservador
	0	baixa - 25	>1	~87	~98	21	12	conservador
	0	baixa - 0	<1	~86	~97	62	32	seq ou MRIP
	1	todas	<<1	~16	~58	94	84	otimista
redes de fila 2 (conf 3)	0	média e alta	>>1	~92	~95	19	17	conservador
	0	baixa - 25	>1	~92	~95	31	23	conservador
	0	baixa - 0	<1	~92	~91	69	41	seq ou MRIP
	1	todas	<<1	~44	~99	80	67	otimista

5.3.3 Resultados e conclusões sobre o comportamento dos processos lógicos

Os exemplos mostram que os processos lógicos que compõem um modelo possuem características distintas que fazem com que sejam mais adequados a um tipo de protocolo. A utilização de vários protocolos durante a simulação torna-se atrativa para que se alcance um melhor desempenho. A troca de um protocolo em um processo lógico que permita a execução de eventos que estão bloqueados desnecessariamente causa uma aceleração na simulação, pois, ao mesmo tempo que o processo lógico não está mais bloqueado, ele começa a mandar mais mensagens para os outros processos lógicos, o que pode acelerar a simulação.

Dessa forma, propõe-se nesta tese a execução da simulação com diversos protocolos de forma a flexibilizar a simulação e conseguir um desempenho melhor com a simulação distribuída por meio da utilização do protocolo mais adequado para cada parte do modelo.

Nesse sentido, verificou-se que a adoção de um protocolo de sincronização em uma simulação depende de mais fatores do que simplesmente a granulosidade, o particionamento, o *lookahead* dentre outros. Existem fatores que podem ser extraídos em tempo de execução que mostram o desempenho da simulação em cada processo lógico. E esses fatores permitiram analisar a execução de uma simulação, onde se observaram processos lógicos com bom desempenho com um protocolo e outros processos lógicos com um desempenho ruim, com características que permitem concluir que uma mudança de protocolo em alguns processos lógicos poderia melhorar o desempenho da simulação como um todo.

5.4 Considerações finais

De forma geral pode-se observar que na execução de uma simulação distribuída conservadora é possível analisar individualmente cada um dos processos lógicos e dessa forma, obterem-se medidas que indicam o desempenho da simulação.

Essas medidas mostram que cada processo lógico possui características distintas que indicam se o protocolo conservador está tendo bom desempenho ou não. O

comportamento dos modelos analisados mostra que cada processo lógico possui um desempenho diferente se analisado isoladamente. Nos casos analisados, um processo lógico está tendo um bom desempenho enquanto o outro ou os outros poderiam estar sendo executados de forma mais rápida caso estivesse sendo utilizado o protocolo otimista.

Por outro lado, a adoção do protocolo otimista para todos os processos lógicos pode não ser a melhor opção, pois pela análise desenvolvida nesse capítulo, a ocorrência de *rollbacks* em muitos casos poderia degradar a simulação.

Dessa forma, a utilização de um protocolo que melhor se adeque ao processo lógico em questão torna-se atrativo, pois o protocolo é escolhido de acordo com as características desse processo lógico. A criação de uma simulação com processos lógicos executando a simulação com o protocolo mais adequado torna-se atrativa para uma melhora no desempenho da simulação.

A utilização de um tipo de protocolo pode resultar em bons *speedups*, porém, os modelos analisados mostram que os processos lógicos executando o protocolo conservador poderiam ter desempenho melhor caso parte da simulação pudesse ser executada com o protocolo otimista, que executaria, na maioria das vezes, eventos seguros. Considerando-se o estudo efetuado é possível concluir que a utilização de duas abordagens de simulação torna-se bastante atrativa, pois enquanto um processo lógico possui uma grande quantidade de bloqueios necessários (mostrando que o protocolo conservador está sendo a melhor opção) o outro processo lógico fica grande parte do tempo bloqueado desnecessariamente, ou seja, eventos que estavam na sua lista de espera poderiam ter sido executados.

A probabilidade de execução de eventos que resultem em erros de causa e efeito são menores, pois como se pode verificar nos modelos, a porcentagem de bloqueios que foram necessários, ou mesmo o tempo (em porcentagem) de bloqueios necessários é pequeno em relação a porcentagem de tempo em bloqueios desnecessários.

Outro fator que deve ser considerado é que a migração de um processo lógico do protocolo conservador para o protocolo otimista pode melhorar o desempenho, não apenas do processo lógico onde está ocorrendo a migração, mas também do processo lógico que se manteve com o protocolo conservador. Isso ocorre porque a execução de eventos que estavam bloqueados gera mensagens que acabam desbloqueando ou mesmo

sendo executadas no processo lógico destino. Isso gera uma aceleração na simulação, tanto no protocolo otimista quanto no protocolo conservador, que receberá mais mensagens diminuindo o tempo bloqueado.

Essas constatações levam a proposição de um mecanismo de troca de protocolo parcial, onde apenas processos lógicos que mostrem um desempenho aquém do esperado sofrem a mudança de protocolo. O próximo capítulo (capítulo 6) detalha o mecanismo proposto e a forma de execução da simulação com os protocolos conservador e otimista.

6 Proposta de um Mecanismo de Troca Dinâmica de Protocolos Local

Este capítulo propõe um novo mecanismo de troca dinâmica de protocolos onde tanto a decisão quanto a execução da troca é realizada localmente em cada processo lógico.

6.1 Considerações iniciais

Morselli (2000), propõe um mecanismo de troca automática de protocolo de sincronização em tempo de execução com uma abordagem diferente. A proposta, descrita no capítulo 3, mostra um mecanismo que decide, por meio do comportamento da simulação, qual protocolo de sincronização melhor se adapta ao contexto em que a simulação distribuída está sendo realizada. A proposta considera a utilização dos dois mecanismos de sincronização mais populares, o CMB e o *Time Warp*.

O mecanismo proposto por Morselli apresenta algumas desvantagens decorrentes, principalmente, da centralização observada para a decisão e implementação da troca de protocolos. Neste sentido, alguns pontos que provocam a perda de desempenho na troca de protocolos devem ser salientados:

1. Sobrecarga de comunicação

   No protocolo proposto por Morselli, os processos observadores coletam informações e enviam pela rede para o processo gerenciador, que precisa desses

dados para analisar o desempenho da simulação. A decisão pela troca ou não do protocolo é realizada pelo processo gerenciador de forma centralizada, considerando os dados coletados por todos os processos observadores.

2. Sobrecarga na troca de protocolo

   Na proposta de Morselli, todos os processos lógicos devem ser paralisados para que a troca de protocolos seja realizada, gerando uma queda considerável no desempenho da simulação.

3. Todos os processos lógicos devem executar o mesmo protocolo

   Quando o processo gerenciador decide pela troca de protocolo, todos os processos lógicos devem seguir a troca, mesmo os que tiverem apresentando um bom desempenho. No capítulo 5, por meio da análise de cada processo lógico, observa-se que protocolos diferentes podem ser mais adequados para cada processo lógico.

Procurando minimizar essas desvantagens foram estudadas modificações para o mecanismo de troca de protocolos em tempo de execução.

Durante a realização deste estudo, foram inicialmente analisadas as possíveis formas para realizar a troca de protocolos considerando-se principalmente “como decidir a troca” e “como realizar a troca”. Foram avaliadas as diversas possibilidades considerando decisões e trocas, globais e locais.

Este estudo gerou uma proposta de taxonomia para os mecanismos de troca de mensagem, apresentada na seção 5.2. A taxonomia desenvolvida facilita a avaliação e comparação das diversas possibilidades. Essa avaliação está na seção 5.3.

Baseando-se nessa avaliação realizada, optou-se por detalhar a proposta de decisão e troca locais apresentada na seção 5.4 e comparar com a decisão e troca globais na seção 5.5.

Vale ressaltar que a possibilidade de troca de protocolo parcial implica na existência de dois protocolos distintos sendo aplicados nos processos lógicos de uma mesma simulação. A coexistência de mais de um protocolo de sincronização em uma mesma simulação é uma proposta desta tese de doutorado. O detalhamento de como

pode ser tratada essa situação e avaliação da viabilidade da coexistência de protocolos será detalhada no capítulo 7 desta tese.

6.2 Taxonomia proposta para a Troca de Protocolo

O mecanismo de troca de protocolo de sincronização apresenta duas fases principais: uma fase de análise de desempenho e outra de troca do protocolo. Na fase de análise de desempenho é verificado se o desempenho da simulação está bom, ou seja, com base no desempenho apresentado pelo processo lógico é feita a decisão de troca ou não do protocolo. Essa decisão pode partir de um processo lógico apenas (o processo lógico verifica o seu desempenho isoladamente, sem saber sobre o desempenho dos demais processos lógicos e toma a decisão de trocar o protocolo), ou pode existir a idéia do gerenciador global que coleta os dados de todos os processos lógicos, analisando o desempenho e tomando a decisão ou não de troca do protocolo. Na fase de troca do protocolo, a troca pode ser local (a troca afeta apenas o processo lógico em que se detectou o baixo desempenho e os outros processos lógicos continuam a executar sem alteração) ou global, em que todos os processos lógicos terão que fazer a mudança de protocolo de sincronização.

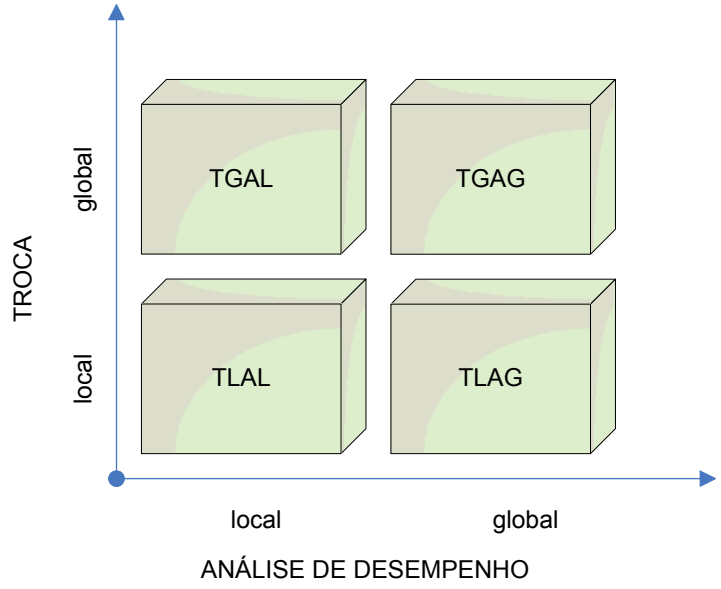


Figura 6.1: Taxonomia do mecanismo de troca de protocolo.

Com base nessas estratégias, é possível classificar as estratégias do mecanismo de troca de protocolos em quatro grupos (Figura 6.1): TLAL (Troca Local Análise Local), TGAL (Troca Global Análise Local), TLAG (Troca Local Análise Global) e TGAG (Troca Global Análise Global).

6.2.1 Estratégia TLAL

Nessa estratégia a análise de desempenho da simulação é local em todos os processadores, ou seja, cada processo lógico faz a análise dos seus dados para avaliar o desempenho da simulação. Durante a execução da simulação, dados sobre o avanço da simulação são computados de forma a medir o desempenho. Caso seja detectado localmente, que o processo lógico em questão está com baixo desempenho, o próprio processo lógico propõe a mudança de protocolo e o algoritmo de troca de protocolo é acionado.

Nessa abordagem tem-se então uma troca de protocolo local, onde podem coexistir diferentes protocolos sendo executados nos diversos processos lógicos.

6.2.2 Estratégia TGAL

Nesta estratégia a análise do desempenho continua sendo local em todos os processadores, ou seja, cada processador faz a análise dos dados coletados durante a sua simulação e consegue identificar se o seu desempenho está bom ou ruim. Se o processo lógico detectar um baixo desempenho, ele envia uma mensagem para o processo gerenciador indicando o seu nível de desempenho. Porém, a troca de protocolo é centralizada, ou seja, um processo gerenciador centraliza as informações de desempenho de todos os processos lógicos e com base nesses dados decreta ou não a troca do protocolo.

É bom ressaltar que nesta abordagem o processo gerenciador não recebe os dados (número de mensagens nulas, tempo bloqueado, por exemplo) dos processos lógicos, mas o resultado de desempenho do processo lógico. Essa diferença é

importante, pois o processamento do processo gerenciador nesse caso é mínimo, pois o cálculo de desempenho já foi realizado nos processos lógicos, cabendo ao processo gerenciador apenas computar quantos e quais são os processos que indicam baixo desempenho para decidir pela troca do protocolo.

Nesta estratégia, todos os processos lógicos recebem o aviso de troca de protocolo e tem a simulação interrompida para a execução do processo de conversão de protocolo. Dessa forma, o processo gerenciador precisa das informações da maioria dos processos lógicos de forma que seja garantida uma melhora de desempenho no caso de troca.

6.2.3 Estratégia TLAG

Na análise de desempenho global todos os processos lógicos enviam, periodicamente, informações sobre o desempenho da simulação para um processo gerenciador. Nesse caso, em cada processo lógico existe um processo observador que coleta informações sobre o desempenho e envia essas informações para o processo gerenciador. O processo gerenciador deve então organizar esses dados, processa-los e conseguir como resultado o estado de desempenho da simulação.

Nessa abordagem o processo gerenciador faz todo o processamento necessário para avaliar o desempenho da simulação, porém, são enviadas mensagens apenas para os processos lógicos que apresentam baixo desempenho para que ocorra a troca do protocolo. Nesse caso, a simulação poderá continuar com a coexistência de diversos protocolos.

6.2.4 Estratégia TGAG

Essa estratégia utiliza a análise de desempenho global e troca de protocolo em todos os processos lógicos. Essa é a abordagem inicial considerada no mecanismo de troca de protocolo proposto por Morselli (2000). Nessa abordagem, o processo gerenciador recebe os dados de interesse para cálculo de desempenho de todos os

processos lógicos e calcula esse desempenho. Caso seja detectado um baixo desempenho, o processo gerenciador manda uma mensagem para todos os processos lógicos avisando a troca e colocando em execução o processo conversor.

6.3 Avaliação das abordagens para Troca de protocolo

As estratégias discutidas possuem diversas vantagens e desvantagens uma em relação à outra que serão detalhadas aqui:

Estratégia TGAL (Análise Local e Troca Global)

As vantagens da estratégia TGAL são:

- A análise local de desempenho diminui o tráfego na rede, pois os dados do processo lógico (número de mensagens nulas, *rollbacks*, tempo bloqueado, dentre outros) não serão mais enviados pela rede, pois a análise desses dados é feita no próprio processo lógico. O único tráfego que ocorre é o envio do *status* de desempenho do processo lógico, ou seja, se o desempenho está satisfatório ou não.
- Como o processamento é feito localmente no processo lógico, o processo gerenciador é um processo que fica apenas coletando os resultados de desempenho dos processos lógicos e por meio da verificação desses dados ele pode optar ou não pela troca de protocolo. Esse processo é menos complexo e mais simples de ser implementado.
- O processo gerenciador não precisa entender detalhes de funcionamento do protocolo que está em execução nos processos lógicos, pois ele apenas precisa da informação do desempenho da simulação. Dessa forma, ele é independente de protocolo e não precisa sofrer modificações com a adição de outros protocolos de sincronização.
- Como a troca de protocolo é global não é necessário fazer modificações na forma de execução da simulação. Caso fosse uma troca local, diversas

modificações seriam necessárias, pois para existirem dois ou mais tipos de protocolos sendo executados para uma mesma simulação, a forma como a simulação avança deve ser modificada.

As desvantagens são:

- O processamento dos dados de desempenho no mesmo processador do processo lógico implica em uma sobrecarga maior, pois além de processar a simulação também deve fazer a coleta e processamento dos dados de desempenho para enviar o resultado para o processo gerenciador.
- A troca global implica em modificar o protocolo de sincronização de todos os processos lógicos, mesmo daqueles processos que estavam tendo bom desempenho.
- A troca global implica também em interromper a simulação para que a troca possa ser feita. Essa interrupção traz uma sobrecarga a mais para a execução da simulação.

Estratégia TGAG (Análise Global e Troca Global)

As vantagens são:

- O processador não terá ciclos de processamento "roubados" pela verificação de desempenho como no caso de análise local, dedicando-se exclusivamente à execução do processo lógico.
- Sendo uma troca global, a forma de funcionamento dos protocolos de sincronização não precisa ser modificada para execução da simulação. Isso torna a implementação mais simples.
- A análise global permite uma visualização mais ampla da execução da simulação, o que pode implicar em menos falhas na ocorrência de troca de protocolo, pois todos os processos lógicos são levados em conta na análise.

As desvantagens são:

- Modelos em que parte dele é mais adequada a um protocolo e parte a outro protocolo, como estudado no capítulo 5, não são contemplados nessa estratégia.
- A simulação precisa ser interrompida para a troca de protocolo. E a simulação de todos os processos lógicos deve passar por uma fase de transição até a efetivação da troca.
- O envio de dados dos processos lógicos para o processo gerenciador sobrecarrega a rede de comunicação, pois são diversos dados (mensagens nulas, *rollbacks*, bloqueios, tempo bloqueado, dentre outros) que precisam ser enviados periodicamente e que competem com a troca de mensagens da simulação, sobrecarregando o meio de comunicação.
- Necessita de um processo gerenciador que colete e processe os dados dos processos lógicos para análise de desempenho. E esse processo deve estar alocado junto a um processo lógico em um processador ou ter um processador dedicado. As duas opções trazem sobrecarga à simulação.

Estratégia TLAG (Análise Global e Troca Local)

As vantagens são:

- A análise global permite que sejam analisados os desempenhos de todos os processos lógicos para que uma troca local seja efetuada. Nesse caso, é possível tentar prever melhor se uma troca de protocolo trará vantagem ao processo lógico com baixo desempenho, sabendo-se o desempenho de todos os demais processos lógicos.
- Modelos que possuem características em que parte do modelo é mais bem simulado com um protocolo e parte com outro são beneficiados com a troca local.
- A simulação é interrompida apenas no processo lógico que precisa realizar a troca.

As desvantagens são:

- A existência de um processo gerenciador que deverá estar alocado junto a outro processo lógico ou em um processador dedicado.
- A sobrecarga na rede devido ao envio dos dados dos processos lógicos para o processo gerenciador para que este analise o desempenho da simulação.
- É uma estratégia que gera tráfego na rede devido a análise global, mas que possui troca local. Nesse caso, é uma estratégia que provavelmente não seria útil na prática.

Estratégia TLAL (Análise Local e Troca Local)

As vantagens são:

- Sobrecarrega menos o meio de comunicação, pois nem dados nem resultados de desempenho são enviados. Como a análise é feita localmente e não considera o desempenho dos demais processos lógicos, não há necessidade de envio de dados para um processo gerenciador como no caso do TGAG. É enviada apenas uma comunicação de um *pl* quando ocorrer mudança de protocolo. Nesse caso, após a troca de protocolo, o *pl* envia mensagens para os demais *pls* informando a mudança de protocolo.
- A simulação não é paralisada em todos os processos lógicos. Apenas o processo lógico que precisa ter seu protocolo modificado é interrompido. Nesse caso, a simulação continua normalmente nos demais processos lógicos, diminuindo a sobrecarga do mecanismo de troca de protocolo.
- Apenas processos lógicos que apresentem baixo desempenho têm seu protocolo trocado. Isso contempla modelos em que em alguns processos lógicos é mais adequada a utilização do protocolo otimista e em outros o conservador.
- Não há a necessidade de implementação de um processo gerenciador que verifica o desempenho.
- Plataformas arquiteturalmente ou configuracionalmente diferentes são beneficiadas com a análise e troca local, pois pode-se utilizar o protocolo mais adequado à realidade de desempenho do processador utilizado.

As desvantagens são:

- São necessárias diversas modificações nos protocolos de forma a compatibilizar a troca de mensagens entre processos lógicos com protocolos diferentes. Por exemplo, no protocolo conservador têm-se mensagens nulas que não existem nos protocolos otimistas. Para o correto funcionamento de ambos, o protocolo otimista também deverá enviar mensagens nulas de forma a possibilitar o avanço da simulação no *pl* executando o protocolo conservador.
- A troca é local e não considera o desempenho dos outros processos lógicos. Isso pode fazer com que o processo lógico que trocou seu protocolo não melhore seu desempenho, pois ele depende diretamente dos outros para avançar na simulação. Nesse caso, duas possibilidades podem ocorrer: o processo lógico volta a trocar de protocolo e retorna ao protocolo original ou os demais processos lógicos também trocam de protocolo. Para exemplificar o primeiro caso, supõe-se um processo lógico que inicia com o protocolo conservador e troca para otimista. A geração de mais mensagens melhora o desempenho desse processo lógico por um intervalo de tempo, mas após esse tempo, a necessidade de *rollback* se intensifica porque os processos lógicos para onde ele está enviando mensagens não estão consumindo essas mensagens na velocidade desejada e o desempenho esperado não é atingido. Nesse caso, a troca de protocolo não trouxe vantagens e uma troca é realizada novamente. No segundo caso, supondo novamente que o processo lógico iniciou com o protocolo conservador e troca para otimista. Supondo que os *pls* para onde ele envia mensagens começam a verificar muitos bloqueios desnecessários, nesse caso, os demais processos lógicos começam a trocar também de protocolo até culminar na troca de todos os processos lógicos. Nesse caso, uma troca global poderia ter sido mais eficiente que a troca parcial, pois os processos lógicos ficaram executando o protocolo conservador por mais tempo do que necessário até a efetivação da troca. Cumpre lembrar que essa possibilidade existe, porém, em todos os modelos que foram testados, verificou-se que na maior parte dos casos, protocolos diferentes acabam sendo mais adequados para a simulação.

Dentre as estratégias analisadas, é possível verificar que a estratégia TLAL resulta em uma menor sobrecarga à simulação, pois diminui o tráfego na rede, não paralisa a simulação de todos os processos lógicos, e é a estratégia que beneficia modelos como os utilizados nos testes descritos no capítulo 5. Sendo assim, um maior detalhamento dessa estratégia será apresentada a partir desse ponto.

6.4 Detalhamento da abordagem TLAL

Conforme apresentado nas seções anteriores, a abordagem TLAL propõe que a troca de protocolos possa ocorrer gradualmente, com a decisão sobre a troca ou não sendo tomada individualmente em cada processo lógico e com a troca de protocolo efetuada apenas em alguns processos lógicos. Nesse contexto, diversas considerações devem ser feitas em relação às estruturas de dados, a garantia de consistência da simulação com a coexistência de dois protocolos de sincronização e a troca de mensagens entre os processos lógicos de forma a garantir a integridade da simulação.

Essa proposta tem por objetivo minimizar a sobrecarga que o mecanismo impõe à simulação diminuindo o tráfego de mensagens dos processos para o processo gerenciador e diminuindo o tempo que o mecanismo leva para efetuar a troca do protocolo.

A diminuição do tráfego de mensagens dos processos para o processo gerenciador pode ser obtida por meio da pulverização da decisão de troca, que deixaria de ser centralizado no processo gerenciador para ser colocado em cada um dos processos lógicos que compõe a simulação. O gerenciador apenas recebe um comunicado do desempenho dos processos lógicos e não mais os dados brutos sobre desempenho (mensagens nulas, tempo bloqueado, etc.).

Na proposta original do mecanismo, é função do processo observador fazer a coleta dos dados e envio dos mesmos para o processo gerenciador, de forma que este possa verificar, por meio de inferências, o desempenho da simulação.

Nessa abordagem, parte do trabalho de verificação do desempenho, é função do processo observador. Ou seja, o processo observador, em intervalos de tempo definidos, faz uma checagem do andamento da simulação no processo e caso seja detectado um

baixo desempenho, faz uma notificação ao processo gerenciador. O processo gerenciador, por sua vez, computa a quantidade de observadores que enviaram a mesma notificação e toma uma decisão de troca ou não de protocolos de acordo com esses dados.

Esse sistema pode ser definido em três fases distintas:

Fase 1 – O processo observador coleta os dados do processo que está observando, como por exemplo, porcentagem de tempo bloqueado, porcentagem de mensagens nulas que foram bloqueios desnecessários, porcentagem de mensagens completas que foram bloqueios necessários, número de *rollbacks*, porcentagem de tempo ocioso, quantidade de mensagens nulas, etc.

Fase 2 – O processo observador faz inferências sobre os dados coletados e verifica o desempenho da simulação. Caso esse processo chegue à conclusão que a simulação está com baixo rendimento, envia uma mensagem ao processo gerenciador notificando o número do processo que está com baixo desempenho. Caso contrário, nenhuma ação é tomada pelo processo observador.

Fase 3 – O processo gerenciador recebe as notificações dos processos observadores e computa a quantidade de mensagens. Caso o número de mensagens exceda um limite, a troca de protocolos é ativada. O limite estabelecido deve ser expressivo o suficiente para validar a necessidade da troca de protocolos.

A vantagem dessa abordagem é que o processo gerenciador se torna independente do protocolo que está sendo empregado nos nós de simulação. Diversos protocolos de sincronização e variantes podem ser utilizados e somente os processos conversores e observadores terão de ser atualizados.

Uma segunda vantagem diz respeito à diminuição da sobrecarga na rede, uma vez que apenas resultados são enviados ao processo gerenciador, e não mais os dados. Dessa forma, a rede de comunicação será menos afetada pelo mecanismo de troca.

Com a decisão de troca de protocolos distribuídos nos processos lógicos, torna-se possível a troca parcial de protocolo. Isto é, em alguns processos um protocolo fica em execução enquanto nos demais tem-se outro protocolo.

Assim, em uma única simulação distribuída, tem-se alguns processos lógicos executando o protocolo conservador e outros o protocolo otimista. Para tornar mais claro o funcionamento deste novo mecanismo, apresenta-se a seguir uma comparação entre os mecanismos TLAL e TGAG.

6.5 Comparação das abordagens TLAL e TGAG

A Figura 6.2 mostra o funcionamento do mecanismo de troca de protocolo seguindo a abordagem TGAG. Nesse exemplo, o processo gerenciador (PG) recebe as mensagens dos processos lógicos com as variáveis (setas verdes) para calcular o desempenho da simulação e verifica, no tempo t, que a simulação está com baixo desempenho. Ele então envia uma mensagem para os processos lógicos para que seja executada a mudança de protocolo. No caso da simulação conservadora, como a simulação sempre está em estado consistente, os processos lógicos interrompem as simulações e executam os processos conversores que fazem a troca do protocolo. No caso do protocolo otimista, a simulação deve entrar em um período de transição para garantir a consistência da simulação para então fazer a conversão do protocolo. Esse período de transição, tanto para o caso conservador como para o caso otimista, está sendo representado na figura pelo tempo t_{troca}.

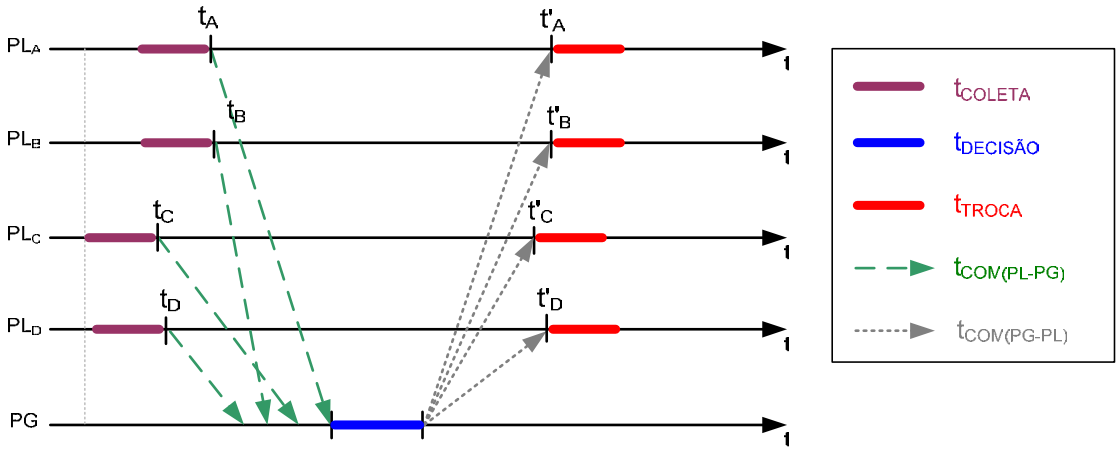


Figura 6.2: Diagrama de tempo mostrando o momento da troca de protocolo da abordagem TGAG.

A proposta desta tese é minimizar o tempo de transição e com isso melhorar o desempenho da simulação. Dessa forma, dois momentos podem ser extraídos do mecanismo de troca de protocolo: o cálculo de desempenho para decisão da troca ($t_{DECISÃO}$) e a troca de protocolo propriamente dita (t_{TROCA}).

A proposta de Morselli (Morselli, 2000) é criar três processos para o funcionamento do mecanismo: observador, gerenciador e conversor. Os observadores enviam os dados dos processos lógicos para o gerenciador que infere sobre os dados e decide a troca. Os conversores realizam a troca propriamente dita.

A abordagem que se propõe é modificar a decisão da troca para cada um dos processos lógicos. Ou seja, é função do processo lógico inferir sobre os dados coletados e calcular o desempenho. Caso seja detectado baixo desempenho, o processo lógico modifica seu protocolo e informa esse estado para os demais processos lógicos.

A Figura 6.3 ilustra como funciona a abordagem proposta. Uma das modificações em relação à Figura 6.2 é a ausência do processo gerenciador na decisão da troca. Isso ocorre porque cada processo lógico verifica seu desempenho, independente do desempenho dos outros processos lógicos. E, cada processo lógico pode modificar o protocolo corrente se detectar baixo desempenho.

Na Figura 6.3 o pl_B verificou, no tempo $t_{B,}$ que estava tendo baixo desempenho. Nesse momento o mecanismo de troca é ativado e o pl_B, em um intervalo de tempo t_{troca}, fica realizando as operações necessárias para a troca do protocolo.

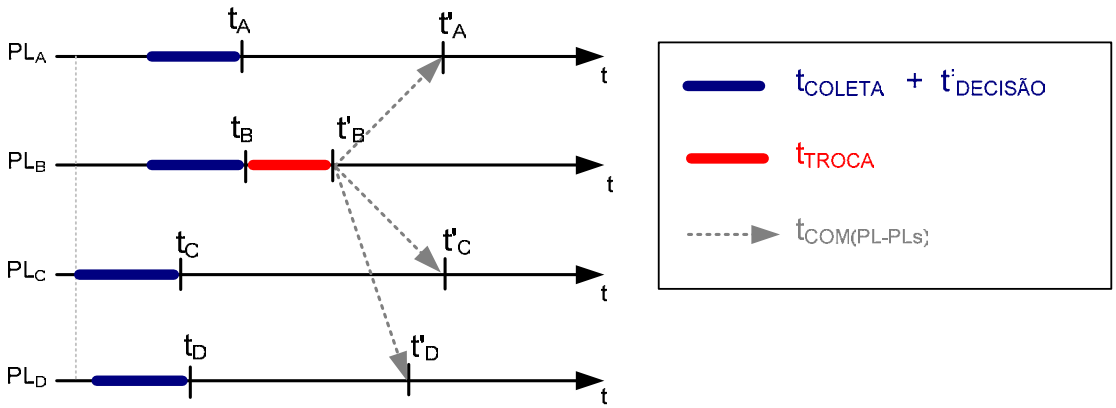


Figura 6.3: Diagrama de tempo mostrando o momento da troca de protocolo de um dos processos lógicos da simulação para a abordagem TLAL.

A maior vantagem da abordagem apresentada na Figura 6.3 é a independência entre os *pl*s e a exclusão do PG (Processo Gerenciador) que deixa de ser uma entidade que recebe todos os dados para análise diminuindo o tráfego na rede de comunicação. Uma análise geral das duas abordagens mostra que:

1. Tempo gasto a cada intervalo de tempo em que o desempenho do *pl* será avaliado:

    - Na abordagem TGAG tem-se uma sobrecarga representada por:

    $$t_{COLETA} + t_{COM(PL\text{-}PG)}$$

    - Na abordagem TLAL, esse mesmo tempo será dado por:

    $$t_{COLETA} + t'_{DECISÃO}$$

    Comparando-se esses dois valores conclui-se que o tempo gasto (a cada intervalo de tempo em que a análise de desempenho deve ser feita) é menor no caso da abordagem TLAL, uma vez que se sabe que o tempo de processamento é menor que o tempo de comunicação (latência e velocidade de transmissão da rede). Assim:

    $$t_{COM(PL\text{-}PG)} > t'_{DECISÃO}$$

    É possível observar que na abordagem TLAL a ação de verificar um desempenho ruim é mais veloz, pois não depende dos outros processos lógicos e assim é possível tomar providências mais rápidas para efetuar a troca do protocolo.

2. Tempo total para decidir a troca de protocolo e efetivar a troca:

    - Na abordagem TGAG, cada processo lógico deve coletar os dados (t_{COLETA}) e enviar para o processo gerenciador ($t_{COM(PL\text{-}PG)}$). Esse, por sua vez, espera os dados de todos os processos lógicos, faz a verificação do desempenho e toma a decisão ou não de troca de protocolo ($t_{DECISÃO}$). Caso a decisão seja pela troca, o processo gerenciador envia uma mensagem para todos os processos lógicos sobre a troca de protocolo ($t_{COM(PG\text{-}PL)}$) e os processos lógicos então executam a troca (t_{TROCA}). Como pode-se observar, o tempo total gasto pode ser representado por:

$$t_{COLETA} + t_{COM(PL\text{-}PG)} + t_{DECISÃO} + t_{COM(PG\text{-}PL)} + t_{TROCA}$$

- Já na abordagem TLAL, cada processo lógico coleta as informações de desempenho (t_{COLETA}) e faz os cálculos necessários para a tomada de decisão de troca ou não de protocolo ($t'_{DECISÃO}$). Caso a decisão seja pela troca de protocolo, o processo lógico entra em status de troca (t_{TROCA}) e comunica aos demais processos lógicos que está executando com um protocolo diferente ($t_{COM(PL\text{-}PLs)}$)

$$t_{COLETA} + t'_{DECISÃO} + t_{TROCA} + t_{COM(PL\text{-}PLs)}$$

Comparando-se esses valores, verifica-se que o tempo de coleta de dados (t_{COLETA}) e o tempo de comunicação ($t_{COM(PG\text{-}PLs)}$ para o TGAG e $t_{COM(PL\text{-}PLs)}$ para o TLAL) é igual para as duas abordagens e para todos os processos lógicos. Porém, o tempo que a abordagem TLAL gasta é menor (considerando-se os processos lógicos como um todo), pois eliminando-se os tempos iguais (t_{COLETA} e $t_{COM(PL\text{-}PLs)}$) é possível observar que:

$$t_{COM(PL\text{-}PG)} + t_{DECISÃO} + t_{TROCA} > t'_{DECISÃO} + t_{TROCA}$$

O tempo de troca t_{TROCA} não foi eliminado porque o no caso da abordagem TGAG o tempo deve ser contabilizado para todos os processos lógicos e na abordagem TLAL apenas no processo lógico que trocou de protocolo. Mas, considerando-se o processo lógico que fez a troca e eliminando-se esse tempo também:

$$t_{COM(PL\text{-}PG)} + t_{DECISÃO} > t'_{DECISÃO}$$

Ou seja, a abordagem TGAG possui uma sobrecarga a mais de comunicação que a abordagem TLAL e é possível observar também que $t_{DECISÃO} > t'_{DECISÃO}$ pois $t'_{DECISÃO}$ tem menos dados a serem analisados (dados de um processo lógico apenas) enquanto $t_{DECISÃO}$ analisa os dados de todos os processos lógicos.

Analisando-se os processos lógicos do TLAL que não estão com desempenho ruim, pode-se verificar que a sobrecarga é composta por $t_{COLETA} + t'_{DECISÃO}$ e o recebimento da mensagem avisando sobre a troca de protocolo no outro processo lógico.

3. Tempo de troca de protocolo

    - Na abordagem TGAG, o tempo de troca de protocolo é representado por:

    $$t_{TROCA}$$

    - E, na abordagem TLAL, o tempo de troca também é representado por:

    $$t_{TROCA}$$

    Apesar do tempo de troca ser o mesmo para as duas abordagens, na abordagem TGAG, esse tempo é necessário para todos os processos lógicos, o que não é verdade na abordagem TLAL onde apenas o processo lógico com desempenho ruim precisa efetuar a troca. Nesse caso, a sobrecarga da troca só é sentida pelo processo lógico com baixo desempenho.

As vantagens da abordagem mostrada na Figura 6.3 (TLAL) em relação a abordagem da Figura 6.2 (TGAG) são:

1. O processo gerenciador deixa de ser uma entidade que recebe todos os dados para análise diminuindo o tráfego na rede de comunicação; como somente um processo lógico está tendo seu protocolo modificado, os demais continuam a sua simulação; essa abordagem contempla modelos que parte dele é mais bem simulada com um protocolo e parte com outro protocolo.

2. A sobrecarga do tempo t_{TROCA} só existe para o processo lógico que precisa ter seu protocolo modificado. O tempo gasto para a troca de um processo lógico acaba sendo menor do que quando a troca é efetuada por todos os processos lógicos, pois não há necessidade de se chegar a um denominador comum (por exemplo, descoberta do Tempo Virtual Global para troca de otimista para conservador).

3. A ação de verificar um desempenho ruim e trocar de protocolo é mais rápida, pois como é tudo feito localmente, não existe atraso de envio / recebimento de mensagens nem atraso do cálculo do desempenho por um processo gerenciador.

4. A sobrecarga de comunicação é menor, uma vez que suprime-se toda a comunicação $t_{COM(PL-PG)}$. Um dos detalhes que se deve levar em conta é que essas mensagens são

maiores que as mensagens que os processos lógicos trocam entre si para a simulação, pois essas mensagens devem informar diversos valores. Por exemplo, no caso do protocolo conservador, é preciso informar: tempo simulado, tempo bloqueado, quantidade de mensagens nulas, quantidade de mensagens completas, quantidade de mensagens completas que causaram bloqueios necessários, quantidade de mensagens completas que causaram bloqueios desnecessários, quantidade de mensagens nulas que causaram bloqueios desnecessários, tempo bloqueado necessário, tempo bloqueado desnecessário e alguns outros dados. Nesse caso, acabam por formar uma mensagem de tamanho razoável, mas são dados necessários para que o processo gerenciador possa calcular o desempenho do processo lógico.

Porém, devem ser observadas também as desvantagens que a abordagem TLAL tem em relação a TGAG:

1. A decisão de troca de protocolo na abordagem TLAL apenas considera os dados de desempenho daquele processo lógico. Não existe uma análise global onde se verifica o desempenho dos demais processos lógicos. Isso faz com que a decisão utilize somente dados parciais e não globais. Nesse caso, na ocorrência de uma mudança de protocolo pode não resultar em bom desempenho, pois estará baseada em dados locais.
2. A implementação de um sistema que permita a utilização de dois protocolos torna-se mais complexa, pois são necessárias diversas modificações no funcionamento dos protocolos e na troca de mensagens entre processos lógicos executando protocolos diferentes.

As vantagens e desvantagens descritas mostram que um detalhamento da abordagem TLAL é necessário e uma visualização dos seus componentes torna-se importante para o seu entendimento. Nesse sentido, a seção 6.6 traz a estrutura gráfica da abordagem TLAL e uma proposta de implementação.

6.6 Estrutura gráfica da abordagem TLAL e definição dos seus componentes

De acordo com a proposta descrita nesse capítulo é possível definir graficamente a abordagem TLAL. Para essa definição gráfica utilizou-se statecharts (Harel, 1988):

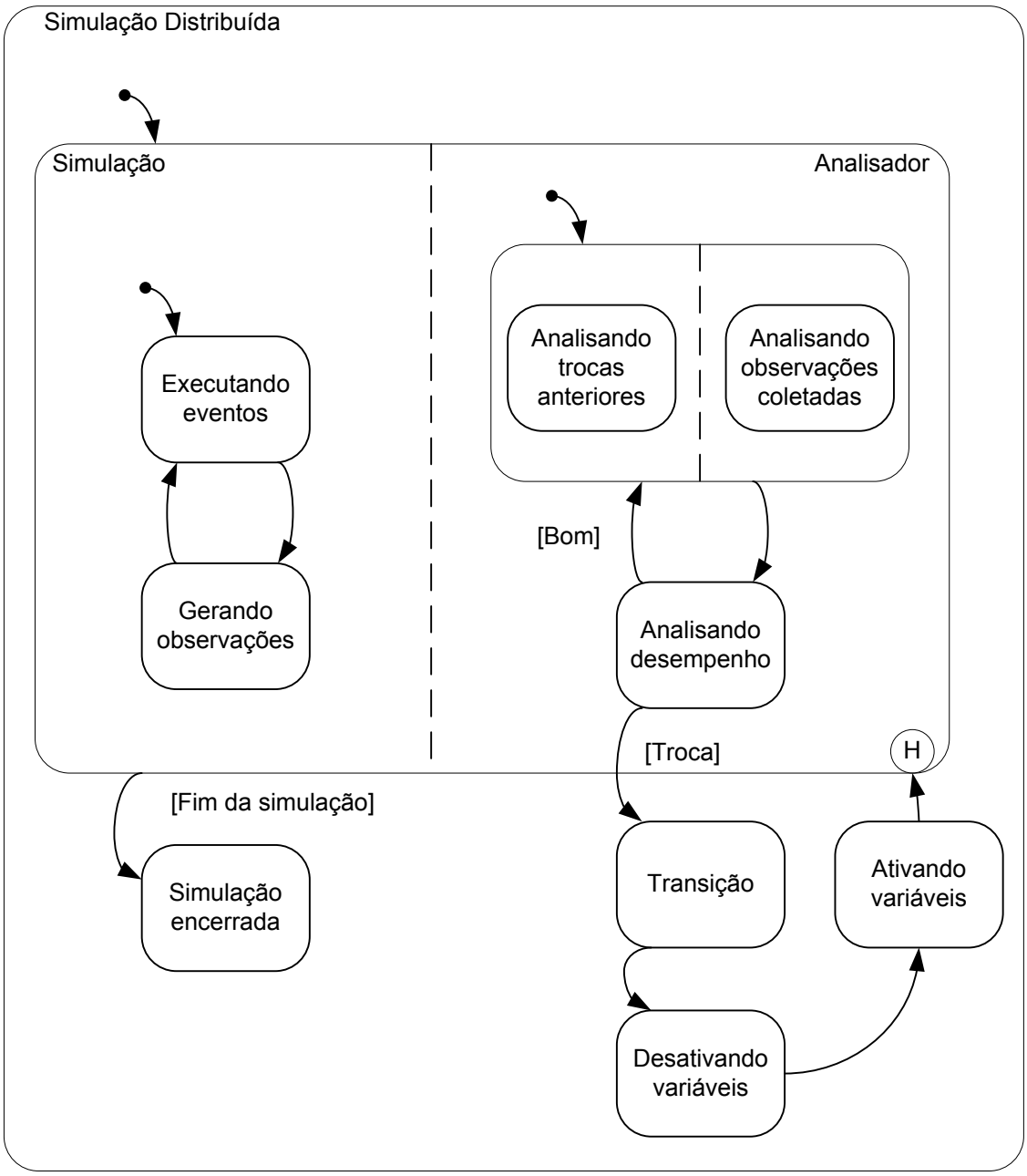


Figura 6.4: Statechart representando o mecanismo de troca de protocolo parcial.

A Figura 6.4 mostra o diagrama *statechart* que representa o mecanismo de troca de protocolo parcial. O diagrama representa as ações que ocorrem dentro de cada processo lógico, ou seja, cada processo lógico possui, dentro de si, o mecanismo, e cada processo lógico age de forma independente do outro processo lógico. Isso implica que cada *pl* pode estar em um estado diferente dentro do *statechart* se a simulação for congelada em um instante de tempo qualquer.

Esse diagrama representa os acontecimentos dentro do processo lógico. Quando a simulação começa, é iniciada tanto a simulação propriamente dita (*executando eventos* – dentro do módulo simulação) quanto o módulo analisador (*analisando trocas anteriores* e *analisando observações coletadas* concorrentemente – dentro do módulo analisador). Na simulação, os eventos são executados e são geradas observações que são enviadas ao módulo analisador. O módulo analisador continuamente recebe essas informações e as processa, confrontando os dados obtidos com a simulação com os dados das trocas de protocolo efetuadas anteriormente (isso é necessário para que ocorra com menor probabilidade a troca do protocolo e um desempenho resultante pior do que o protocolo anterior à troca). Esses dados permitem uma análise de desempenho, que caso seja detectado como bom desempenho, a análise continua sendo efetuada (*analisando desempenho*), e caso seja ruim, a troca de protocolo é efetuada.

A fase de análise da simulação (analisando trocas anteriores) verifica se uma troca anterior já foi efetuada e qual era o desempenho da simulação antes da troca. Caso o desempenho esteja ruim, mas melhor do que o desempenho apresentado pelo processo lógico antes da troca, o protocolo de sincronização corrente é mantido, pois uma nova troca não traria benefícios para a simulação. Se, por outro lado, o desempenho esteja pior que o desempenho apresentado antes de uma troca efetuada, uma nova troca deve ser proposta.

Essa troca implica no processo lógico entrar em um estado de transição (*transição*) para que a simulação entre em um estado consistente para que a troca possa ser efetuada. Nesse ponto, tanto a simulação quanto o analisador ficam em suspenso. Uma vez que a simulação fique em um estado consistente, as variáveis do protocolo são desativadas (*desativando variáveis*) e as variáveis do protocolo alvo são ativadas (*ativando variáveis*) para o reinício da simulação no ponto em que ela parou para a troca. Nesse ponto, tem-se no diagrama a letra H, que indica *history*, ou seja, tanto a

simulação quanto o analisador irão continuar a partir do ponto em que foram interrompidos e não recomeçar.

Quando for detectado fim da simulação, a simulação do processo lógico e o mecanismo de troca são encerrados (*simulação encerrada*).

Como pode ser observado pelo diagrama, o ambiente de simulação que será criado com a implementação do mecanismo de troca pode ser dividido em duas partes, o ambiente de simulação propriamente dito e o módulo analisador.

Para a implementação do ambiente de simulação basicamente deve-se ter diversas funções que podem ser separadas de acordo com a sua funcionalidade:

- Núcleo de funções comuns: essas funções são utilizadas pelos dois protocolos e serão utilizadas para a execução da simulação. Exemplos dessas funções são: iniciar as variáveis do ambiente, verificar o próximo evento a ser executado, liberar evento, escalonar evento, enviar mensagem completa, dentre outras.
- Núcleo de funções CMB: são as funções específicas do CMB que serão utilizadas somente quando o protocolo CMB está sendo executado. Exemplos dessas funções são: iniciar variáveis CMB, enviar mensagem nula, etc.
- Núcleo de funções TW: são funções específicas do TW que serão utilizadas somente quando o protocolo TW estiver sendo executado. Exemplos: enviar antimensagem, iniciar variáveis TW, executar *rollback*, etc.
- Núcleo de funções que geram as observações: essas funções têm por objetivo extrair informações sobre desempenho da simulação, ou seja, são as funções que coletam, por exemplo, tempo executando, tempo bloqueado, tempo de *rollback*, número de mensagens nulas, número de mensagens completas, etc.
- Núcleo de funções para conversão do protocolo: são as funções que irão interromper a simulação, verificar a sua consistência e/ou executar a simulação até um estado de consistência e colocar o novo protocolo em execução. As funções que compõem esse módulo são: executar transição TW/CMB, etc.

O módulo analisador tem a função de colher as informações coletadas durante a execução da simulação e analisar o desempenho da simulação. Esse módulo basicamente terá:

- Núcleo de funções de análise CMB: responsável por verificar quantidade de mensagens nulas, completas, tempo bloqueado, tempo executando, entre outras variáveis de forma a definir se o protocolo CMB está tendo bom desempenho ou a troca de protocolo para TW irá melhorar a simulação.
- Núcleo de funções de análise TW: verifica o tempo que a simulação fica em *rollbacks* em relação ao tempo executando, verifica porcentagem de mensagens que foram efetivamente executadas em relação a mensagens que tiveram que ser desfeitas, dentre outras.

6.7 Considerações finais

Neste capítulo foi proposta uma taxonomia para o mecanismo de troca de protocolo onde se tem o tipo de análise de desempenho (local ou global) e a troca propriamente dita (local ou global). Essa taxonomia permitiu a definição de quatro tipos distintos de mecanismos: TGAL, TGAG, TLAL e TLAG.

O mecanismo TGAG é o mecanismo proposto por Morselli (2000). Os testes efetuados e apresentados no capítulo 5 mostraram a viabilidade de um mecanismo local.

De acordo com os resultados apresentados e as vantagens associadas à abordagem TLAL, uma proposta gráfica dessa abordagem foi apresentada e mais algumas considerações podem ser feitas para detalhar a implementação do mecanismo.

Para implementar o mecanismo local algumas modificações são necessárias na estrutura dos dois protocolos que modificam o funcionamento básico da simulação. Essas modificações incluem: verificação da origem das mensagens quando elas chegam a um processo lógico, necessidade de mais de uma fila de eventos para o *Time Warp* (originalmente ela possui uma só), dentre outras modificações que serão abordadas no próximo capítulo.

7 Mecanismo de troca parcial de protocolo

Este capítulo apresenta uma proposta para a implementação do mecanismo de troca parcial de protocolo.

7.1 Considerações iniciais

O capítulo 6 mostrou a taxonomia proposta para o mecanismo de troca de protocolos baseada nos dois momentos que podem ser considerados: a tomada de decisão e a troca propriamente dita. Essa taxonomia foi proposta de acordo com os resultados apresentados no capítulo 5, onde se pôde observar que uma mesma simulação pode ter características em que protocolos diferentes para partes da simulação são mais adequados do que um só protocolo para toda a simulação. Essas observações permitiram a definição da taxonomia, e pôde-se concluir, por meio da análise de cada uma das abordagens, que a estratégia TLAL é a que apresenta uma menor sobrecarga a simulação e que dessa forma deve ser detalhada para que se mostre a viabilidade de sua implementação.

Assim, neste capítulo serão abordados detalhes do funcionamento da abordagem TLAL e quais são as modificações que devem ser feitas nos protocolos de forma a acomodar as características dos dois protocolos e, além disso, como será a troca de mensagens entre processos lógicos executando protocolos diferentes.

Durante a execução da simulação distribuída, independente de qual protocolo esteja sendo utilizado, é necessário que as informações relativas às variáveis que medem o desempenho da simulação sejam obtidas e analisadas para que se possa decidir se existe a necessidade da troca de protocolos.

No mecanismo de troca de protocolos de sincronização original, proposto por Morselli (2000), foram definidos três processos lógicos para a avaliação de desempenho. Esses processos não têm nenhuma participação no contexto da simulação de um modelo qualquer.

O Capítulo 3 e em particular a Figura 3.4 apresenta a estrutura dos processos utilizados no mecanismo original, onde a troca de protocolo é feita por todos os processos lógicos. Na abordagem proposta nesta tese, como a decisão de troca de protocolo é local, os processos gerenciador e observador são fundidos em um único processo, denominado processo analisador, como pode ser visto na Figura 7.1.

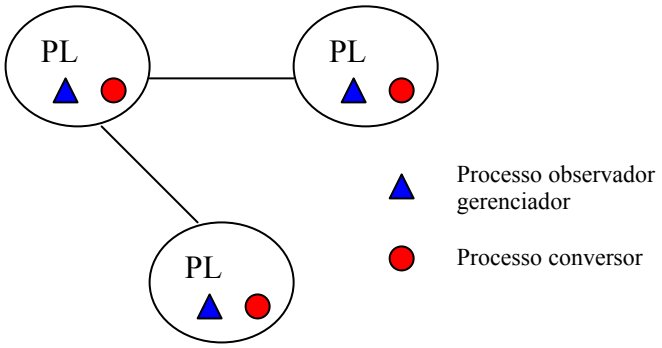


Figura 7.1: Estrutura dos processos envolvidos no sistema de simulação utilizando a troca dinâmica de protocolos, onde o processo observador e gerenciador são representados em um mesmo processo lógico.

A junção dos processos observador e gerenciador é possível, pois cada processo lógico será analisado independente dos demais. Esse processo deve colher e analisar os dados referentes ao desempenho da simulação.

Dessa forma, de acordo com a proposta desta tese, o processo analisador colhe e analisa os dados, e caso seja detectado um baixo desempenho, o processo conversor é ativado para que esse processo lógico migre para o novo protocolo de sincronização.

Nas próximas seções são detalhados os passos necessários à conversão de protocolo, que é composto pela garantia de consistência da simulação (seção 7.2) e o mecanismo de conversão das estruturas de dados (seção 7.3).

7.2 Garantia da Consistência

As regras que regem os dois protocolos considerados para o mecanismo são diferentes. Enquanto o CMB sempre se encontra em um estado consistente, ou seja, a simulação, em qualquer ponto de execução, está sempre correta, o *Time Warp* pode estar em uma situação em que *rollback* é necessário para desfazer uma execução errônea. Dessa forma, para que a simulação possa ser executada com os dois protocolos de sincronização diversas considerações devem ser feitas e analisadas para que a simulação possa avançar de forma correta.

Garantia de consistência na conversão CMB para *Time Warp*

Uma simulação distribuída sincronizada por um protocolo conservador estará sempre consistente, desta forma a utilização do mecanismo de mensagens nulas garante a consistência de uma simulação distribuída sincronizada pelo protocolo CMB em qualquer momento durante sua execução, o que faz com que a troca de protocolo possa ser feita em qualquer instante da simulação. Nesse caso, a simulação é interrompida e após a troca de protocolo o processo lógico deve informar os demais processos sobre a troca.

Garantia de consistência na conversão *Time Warp* para CMB

O paradigma otimista possui como característica a execução de eventos sem a preocupação se estão em ordem cronológica ou não. Essa característica faz com que a ocorram períodos de inconsistências na simulação. Isso torna necessária a adoção de mecanismos que garantam a consistência da simulação na troca do protocolo para CMB, que aceita apenas eventos seguros para serem processados e não possui mecanismos que possam reverter casos de infração na propriedade de causa e efeito.

Na proposta de Morselli (2000), a garantia de consistência na troca de protocolo *Time Warp* para CMB possuía duas abordagens possíveis. (Lembrando que aquela proposta faz a troca de protocolo para todos os processos lógicos.) A primeira abordagem utilizaria o maior LVT dos processos lógicos utilizando-o como um marco para ser atingido por todos os demais processos lógicos. Ou seja, os processos lógicos executariam a simulação (com mensagens, antimensagens e *rollbacks*) até que todos atingissem o valor do LVT.

A segunda abordagem propõe o cálculo do GVT para que todos os processos lógicos voltem a esse valor, que é o último valor em que se pode garantir que todos os processos lógicos estão consistentes.

Essas abordagens aplicam-se mais a troca total de protocolo de sincronização. Na proposta em que cada processo lógico poderá ter seu protocolo trocado por outro em instantes diferentes, uma garantia de consistência diferente se faz necessária.

A solução mais simples seria fazer um comunicado de cálculo de GVT para todos os processos lógicos de forma a se obter o valor atual do GVT. Nessa mensagem o processo lógico também comunica que está modificando seu protocolo. Uma vez atualizado o valor do GVT, inicia-se um processo de *rollback* do processo a ser convertido para CMB até esse ponto do GVT. Retornando o processo lógico para o tempo GVT, que é o mínimo dentre todos os LVTs, pode-se garantir que nenhuma mensagem chegará a esse *pl* com evento a ser executado antes de seu LVT. Essa característica garante que não haverá erros de causa e efeito. Essa abordagem pode levar a existência de diversos *rollbacks* em cascata, causando uma sobrecarga a mais e possivelmente desnecessária na simulação. Feito isso o processo lógico muda o seu protocolo de sincronização para CMB e continua o seu processamento.

A sobrecarga do *rollback* para o último GVT torna essa abordagem pouco atrativa.

Uma outra solução seria enviar uma mensagem solicitando o cálculo do GVT e uma vez com esse valor, o processo lógico manda uma mensagem para todos os processos lógicos comunicando que está modificando o seu protocolo. O processo lógico irá ficar em um estado de transição que irá durar desde o GVT até o seu LVT, que era o tempo de simulação do *pl* quando se iniciou o processo de troca de protocolo.

Nesse estado de transição o *pl* apenas verifica se o processamento já efetuado está correto. Ou seja, o *pl* continua recebendo mensagens e verifica se o processamento local de cada uma das mensagens foi feita na ordem correta. Caso ocorra um caso de processamento errôneo é acionado o mecanismo de *rollback*. Esse algoritmo pode ser visto na Figura 7.2.

```
Tempo_simulação ← GVT
Enquanto tempo_simulação < LVT faça:
Início
        Continua recebendo mensagem
        Atualiza clock dos canais
        Se tempo_evento_processado ≤ menor_clock_dos_canais então
                Evento foi processado em ordem correta
                Passa para próximo evento
                Atualiza tempo_simulação
        Senão
        Se tempo_evento_processado < tempo_msg_chegou então
                Rollback
Fim do enquanto
```

Figura 7.2: Algoritmo de transição *Time Warp* para CMB

Essa abordagem é mais atrativa, pois pode não ocorrer *rollbacks*, minimizando possíveis sobrecargas que a troca de protocolo poderia infligir na simulação. Essa abordagem evita ainda a ocorrência de *rollbacks* em cascata desnecessários. Por outro lado, existe a possibilidade de ser necessário o retorno do *pl* para o GVT calculado no momento da decisão de troca. Neste caso, o mecanismo proposto vai apresentar os seguintes problemas: o retorno ao GVT, a espera para verificar se será necessário o *rollback* e uma maior possibilidade de *rollbacks* em cascata, uma vez que os outros *pl*s já adiantaram ainda mais suas simulações.

7.3 Mecanismo de Conversão

Esta seção analisa os fatores que devem ser considerados para a conversão do protocolo de forma independente em cada processo lógico. Para isso, para cada classe de protocolos existe um conjunto de características que devem ser consideradas no momento da troca de protocolo, sendo que quatro pontos devem ser tratados na conversão:

a) As diferenças entre as estruturas de dados necessárias nos dois protocolos considerados (seção 7.3.1);

b) A adaptação da topologia lógica da rede de comunicação utilizada pelos protocolos de sincronização (seção 7.3.2);

c) A necessidade da compatibilização das variáveis de tempo empregadas para manter a ordem cronológica da execução dos eventos (seção 7.3.3);

d) A forma em que o sincronismo será garantido em uma simulação com os dois protocolos (seção 7.3.4).

7.3.1 Estruturas de Dados

Uma característica importante com relação à implementação dos protocolos de sincronização refere-se às estruturas de dados que auxiliam no gerenciamento das mensagens utilizadas na simulação. Para cada protocolo de sincronização existe um conjunto diferente de variáveis e estruturas de dados, ou seja, um determinado protocolo possui certas estruturas que não são similares ao outro protocolo.

A troca de protocolos durante a execução da simulação exige que estruturas que não existiam anteriormente em um processo lógico CMB, por exemplo, passem a fazer parte deste processo e vice e versa. De forma geral, esta tese propõe o emprego de um mecanismo complementar de utilização de estruturas de dados, assim como definido em (Morselli 2000), porém com algumas adaptações para a execução dos dois protocolos simultaneamente. A maior parte das estruturas de dados foi reunida em apenas um conjunto de estruturas, e outras estruturas foram criadas. Um processo *Time Warp*, por

exemplo, irá conter além de suas estruturas de dados originais, as estruturas de dados de um processo lógico CMB. Estas estruturas deverão ser declaradas inicialmente e ativadas (utilizadas) em tempo de execução no momento da troca de protocolos. Os dados que irão fornecer o valor inicial destas estruturas podem fazer parte de outras estruturas utilizadas pelo protocolo de origem ou, em caso contrário, são mantidos especificamente para a troca de protocolos.

A seguir é analisada cada estrutura de dados utilizada nos protocolos considerados, verificando-se como obtê-las no caso da conversão.

- **Lista de Eventos Futuros (EVL)**

A lista de eventos futuros contém os eventos escalonados (local ou remotamente) a serem executados pelo processo lógico. Em uma simulação distribuída cada processo lógico contém sua própria lista de eventos. Nela os eventos estão ordenados pelo seu tempo de ocorrência e a cada iteração da simulação o evento com o menor tempo de ocorrência é retirado da lista e processado.

CMB → *Time Warp*

A estrutura da EVL é igual nos dois protocolos e como o CMB sempre está em estado de consistência, não há necessidade de nenhuma modificação no caso da troca do protocolo. Desta forma, a lista de eventos futuros deverá manter, após a troca de protocolos, o conjunto de eventos armazenados durante a execução no momento anterior ao processo de troca.

***Time Warp* → CMB**

O protocolo de sincronização *Time Warp* é marcado por sucessivos períodos de inconsistência na simulação. Ou seja, após um período de estabilidade pode ocorrer um período de inconsistência provocado pela execução de um *rollback* em função da chegada de uma mensagem defasada.

Um *rollback* é executado quando chega uma mensagem com tempo menor que o LVT do processo lógico. Nesse caso, uma execução errônea foi feita e deve-se voltar a

simulação até o último ponto consistente. Além de poder existir na lista de eventos futuros, antimensagens, que quando executadas, deixam a simulação em um estado inconsistente.

Para que a troca de *Time Warp* para CMB seja realizada em um processo lógico é necessário que ele chegue a um estado consistente. Esse estado consistente poderia ser traduzido como sendo o último valor de GVT disponível. Nesse ponto sabe-se que a simulação anterior ao tempo do GVT está correta. Porém, o processo lógico não pode simplesmente voltar a esse ponto e desprezar todo o processamento feito, pois diversas mensagens foram enviadas e cada uma dessas mensagens precisa ter um destino para que a simulação como um todo continue estável. A garantia de consistência já foi definida na seção 7.2.

Algumas modificações são necessárias nos protocolos de forma a garantir o funcionamento do algoritmo descrito e do mecanismo como um todo. Durante o capítulo serão abordadas todas as modificações necessárias e na seção 7.3.4 será abordado com detalhes como os processos lógicos irão interagir entre si com protocolos diferentes.

Com relação à estrutura da mensagem utilizada pelo mecanismo de troca deve-se considerar que se por um lado as mensagens do protocolo CMB possuem informações sobre o identificador da tarefa, o tempo em que a tarefa deverá ser executada (tempo de ocorrência) e o número do evento a ser processado, por outro lado as mensagens que fazem parte da implementação do protocolo *Time Warp* possuem, além destas informações, um campo para o sinal de identificação (mensagem ou antimensagem), identificador do processo remetente e identificador do processo destinatário. Por esta razão a estrutura de mensagens utilizada pelo protocolo *Time Warp* foi utilizada para a implementação do mecanismo de troca e foi necessário também acrescentar a informação de qual protocolo o processo remetente está executando e o menor tempo *Time Warp* do processo origem (Figura 7.3).

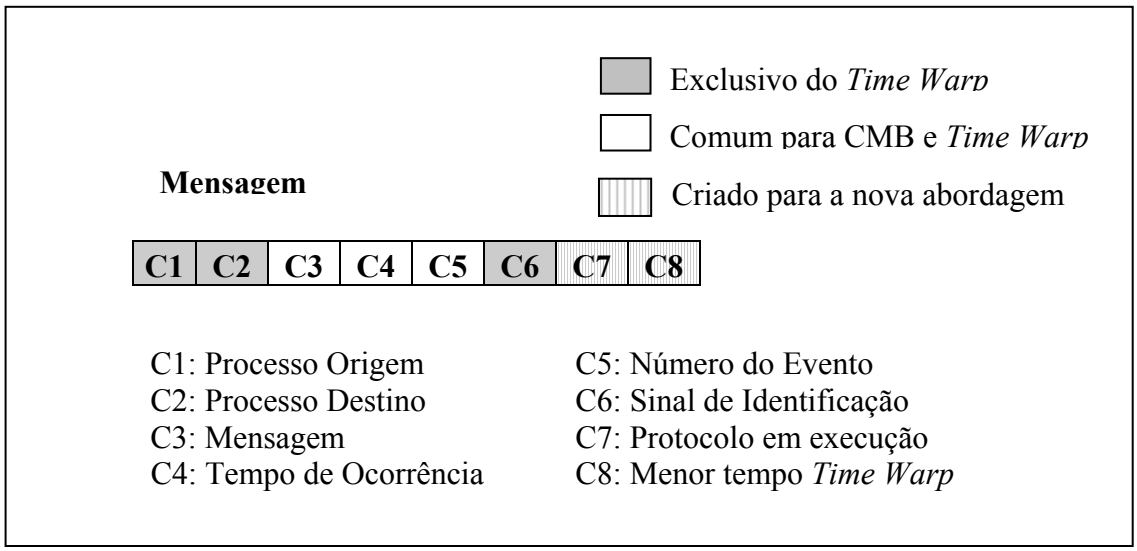


Figura 7.3: Mensagem Utilizada pelo Mecanismo de Troca de Protocolos de Sincronização.

O campo C7 serve então para informar ao *pl* destinatário qual protocolo está sendo executado no *pl* de onde a mensagem está chegando. Isso é necessário para que o *pl* saiba quais as ações devem ser tomadas em relação a essa mensagem, como por exemplo, qual será o tempo utilizado para atualizar a lista de *clocks* dos canais.

Como os processos lógicos precisam saber quais são os protocolos que cada um dos processos lógicos está executando, é necessário criar uma estrutura não existente em nenhum dos dois protocolos que armazena o tipo de protocolo sendo executado. Essa estrutura armazena o número do processo lógico e o protocolo que está em execução nesse processo lógico.

Já o campo C8 informa, caso o *pl* remetente seja *Time Warp*, o menor tempo de *clock* dos canais. A implementação original do *Time Warp* não possui a estrutura que armazena os *clocks* dos canais, somente a implementação do CMB possui essa estrutura. Nesse caso, foi necessário utilizar essa estrutura no *Time Warp* também para que seja possível a coexistência dos dois protocolos. Mais detalhes de como será essa estrutura será visto na seção 7.3.3. Esse menor clock dos canais informa ao *pl* destinatário qual o tempo que deve ser armazenado no *clock* do canal, ou seja, apesar da mensagem possuir um *timestamp*, o tempo que será colocado no *clock* do canal (e que é utilizado para executar as mensagens da fila de eventos futuros) não será do *timestamp* da mensagem e sim, do campo C8, que é o um tempo menor ou igual ao *timestamp* da mensagem que

chegou. Essa informação garante que uma mensagem que chega de um *pl Time Warp* para um *pl* CMB não seja executado fora da ordem cronológica. Esse tempo é coletado no *pl* remetente fazendo-se uma varredura de todos os *clocks* dos canais e pegando-se o menor valor.

Os *clocks* dos canais armazenam o *timestamp* das mensagens que chegam, no caso da mensagem vir de um *pl* CMB ou o menor tempo *Time Warp,* caso venha de um *pl Time Warp*. Assim, a forma de atualização dos *clocks* dos canais garante um limitante mínimo de que as ações efetuadas até aquele ponto do *pl* remetente é seguro.

A troca de mensagem entre os *pl's* sempre dá a informação de qual protocolo o remetente está executando. Com essa informação, a mensagem que chega deve ser tratada de forma diferente em cada caso. Assim:

- Uma mensagem de um *pl* CMB para outro *pl* CMB:

  A mensagem que chega de um *pl* CMB para outro *pl* CMB deve ser tratado como no protocolo original sem modificações, ou seja, a mensagem deve ser colocada na fila de eventos futuros na ordem de *timestamp* aguardando para ser executado.

- Uma mensagem de um *pl* CMB para um *pl Time Warp*:

  Quando chega uma mensagem de um *pl* CMB para um *pl Time Warp*, o *Time Warp* sempre sabe que a mensagem é segura e não há problemas em executar essa mensagem. Mesmo que seja executada em ordem não cronológica, não afetará a simulação, pois no *Time Warp* pode ser executado *rollback*. Dessa forma, o tratamento para uma mensagem de um *pl* CMB deve ser tratado da mesma forma que uma mensagem de um *pl Time Warp*.

- Uma mensagem de um *pl Time Warp* para um *pl Time Warp*:

  A mensagem que chega de um *pl Time Warp* para outro *pl Time Warp* deve ser tratado da mesma forma como no protocolo original *Time Warp*, pois qualquer mensagem que seja executada no *Time Warp*, caso seja executada na ordem de *timestamp* incorreta, pode ser desfeita sem problemas.

- Uma mensagem de um *pl Time Warp* para um *pl* CMB:

  Quando um *pl* CMB recebe uma mensagem, ele deve analisar se é uma

mensagem de outro *pl* CMB ou de um *pl Time Warp*. Quando se tratar de uma mensagem de um *pl Time Warp*, o *pl* CMB deve colocar essa mensagem na fila de eventos e atualizar a lista de *clock* dos canais com o valor do menor tempo *Time Warp* (campo C8, Figura 7.3). Isso é necessário, pois o tempo da mensagem do *Time Warp* pode ser um valor que necessita de *rollback*. Já o menor tempo é como um tempo mínimo que garante que os eventos processados até esse tempo são seguros. Nesse caso, para um evento ser executado, ele deve ter tempo menor ou igual ao menor valor de *clock* de canal armazenado na lista de *clock* dos canais. Como o canal do processo lógico *Time Warp* é atualizado com o valor do menor tempo *Time Warp*, se a mensagem que chegou está fora de ordem cronológica provavelmente essa mensagem receberá uma antimensagem e será eliminada.

A EVL (lista de eventos futuros) sempre deverá conter os valores das variáveis. Apesar do CMB não precisar dos valores do processo origem, do processo destino e do sinal de identificação, como os dois protocolos podem estar em execução simultaneamente, o CMB deverá preencher esses valores, independente se o protocolo *Time Warp* estiver ativo em algum processo lógico. Nesse caso deve-se ter:

- *Processo Origem.* Esta informação não está disponível, normalmente, no CMB, que deverá ser alterado para mantê-lo;
- *Processo Destino.* Esta informação é necessária no *Time Warp* para o envio de anti-mensagens;
- *Sinal de Identificação.* Todos os eventos da EVL devem ser identificados como positivos.

- **Portas de Entrada e Saída de Mensagens**

As estruturas do protocolo CMB *input buffer (IB[i])* e *output buffer (OB[j])*, também chamadas de *portas*, são responsáveis pelo armazenamento temporário (em ordem cronológica) das mensagens recebidas/enviadas por um determinado canal de comunicação. A variável *IB[i]* armazena as mensagens recebidas pelo processo por

meio do canal i e a variável *OB[j]* armazena as mensagens que deverão ser enviadas pelo canal j. No *Time Warp* estas variáveis são substituídas pelo par de estruturas *input buffer* (*IB*) e *output buffer* (*OB*). Todas as mensagens recebidas pelo processo *Time Warp* são inseridas em IB e as mensagens que o processo *Time Warp* deve enviar são armazenadas em OB (não necessariamente em ordem cronológica). Nota-se que essas estruturas, tanto no protocolo CMB como no protocolo *Time Warp*, exercem as mesmas funções (armazenar temporariamente as mensagens que chegam ou partem do processo lógico). Porém há uma diferença no número de estruturas de cada um dos protocolos.

Para o protocolo proposto nesta tese, a estrutura a ser utilizada é o *IB[i]* e *OB[j]* do protocolo CMB fazendo com que o *Time Warp* utilize essa estrutura. Isso permite uma flexibilidade muito maior em caso de troca de protocolo. O *Time Warp* deve, dessa forma, separar as mensagens pelos canais e armazená-los em ordem cronológica, como o CMB, compatibilizando totalmente o armazenamento dessas mensagens temporárias.

O que deverá ser modificado é a forma como o *Time Warp* executa essas mensagens. Como havia somente uma fila de entrada dos eventos, o *Time Warp* executava o primeiro da fila. Com a modificação proposta é necessário que o algoritmo de execução do *Time Warp* seja modificado para que ele faça uma varredura na estrutura *IB[i]* e pegue a mensagem com menor tempo e execute. Isso torna necessário também que ele atualize o valor de CC[i] (*channel clock*) e o LVTH (*Local Virtual Time Horizon*). Essas estruturas serão abordadas nas próximas seções.

CMB → *Time Warp*

No momento da troca CMB → TW todas as mensagens armazenadas na estrutura *IB[i]* devem ser mantidas pois a simulação irá continuar do ponto em que o CMB parou de simular. Nenhuma alteração é necessária nos dados armazenados nessas estruturas.

***Time Warp* → CMB**

Como o processo *Time Warp* fica em um estado de transição, passado esse período as estruturas *IB[i]* e *OB[j]* já estarão no ponto para serem utilizadas pelo CMB.

- **Estruturas IQ (*Input Queue*) e OQ (*Output Queue*)**

Todas as mensagens enviadas ou recebidas por um processo *Time Warp* são armazenadas por um determinado período de tempo nas estruturas IQ e OQ. Isto ocorre porque quando uma mensagem cronologicamente defasada é recebida por um processo *Time Warp*, este processo deve transmitir uma ou mais antimensagens para que as alterações provocadas pelas mensagens emitidas antes do recebimento da mensagem defasada sejam processadas na ordem cronológica correta. As mensagens mantidas nas estruturas IQ e OQ são aquelas com tempo de ocorrência entre o GVT e o LVT do processo. As mensagens com tempo inferior ao GVT podem ser desprezadas visto que até o GVT garante-se que a simulação foi executada corretamente.

Desta forma, o armazenamento das mensagens recebidas e transmitidas permite a correção de um possível erro de causalidade causado pelo processamento de uma mensagem fora da ordem cronológica.

CMB → *Time Warp*

As estruturas IQ e OQ não fazem parte do processo lógico no modo CMB, dessa forma, o processo Conversor deverá habilitar essas estruturas após a troca para o protocolo *Time Warp*. A partir do momento em que essas estruturas estejam habilitadas as estruturas IQ e OQ passam a receber as mensagens contidas nas estruturas IB e OB do processo *Time Warp*, levando-se em consideração que elas devem ser armazenadas em uma estrutura apenas. Ou seja, apesar de estarem separadas por canal na estrutura IB[i] e OB[j], na transição deve-se pegar todos os valores de IB[i] e passá-los para IQ e todos os valores de OB[j] e passá-los para OQ.

***Time Warp* → CMB**

Na troca de protocolos *Time Warp* → CMB as mensagens, após a troca, são retiradas das estruturas IB[i] e OB[j] e escalonadas diretamente para a lista de eventos futuros sem a necessidade de serem armazenadas temporariamente, sendo assim, as estruturas IQ e OQ são desativadas pelo processo Conversor. Isso deve ocorrer após o período de transição do processo lógico.

- **S - Controle de Variáveis da Simulação**

A estrutura de dados *S* (conjunto de registros) armazena os valores das variáveis locais e dos dados estatísticos que possibilitam a análise da simulação quando do seu término. Exemplos de informações armazenadas são: tempo médio de utilização de um recurso, tamanho médio das filas, quantidade de tempo simulado e número de preempções.

Como são variáveis de análise estatística, elas existem nos dois protocolos e, portanto fazem parte da implementação dos dois protocolos e não precisam ser manipuladas no caso de troca de protocolo. Devem ser mantidas e com o mesmo conteúdo.

- **SS – Estrutura de Controle de Contexto do *Time Warp***

Essa estrutura permite que seja possível a realização do mecanismo de *rollback*, armazenando os valores que descrevem o contexto da simulação *Time Warp* durante o período de tempo considerado instável (eventos ocorridos entre o GVT e o LVT). Esses dados incluem *clock* do processo, elementos da lista de eventos futuros e a estrutura que armazena as estatísticas da simulação. O sistema de *garbage collection* do *Time Warp* é responsável pela eliminação das instâncias mais antigas consideradas seguras. Essas instâncias seguras são definidas como aquelas que possuem *timestamp* menor que o GVT (*clock* global).

CMB → *Time Warp*

O protocolo CMB não tem problemas com a propriedade de causa e efeito, pois executa apenas eventos seguros. Dessa forma, não utiliza a estrutura SS que não precisa estar ativa durante a execução com o protocolo CMB. Apenas no momento da troca para *Time Warp* essa estrutura deve ser ativada com os valores de LVT, EVL e S.

Uma vez que a conversão será realizada em um estado consistente, os valores de LVT, EVL e S, presentes no processo lógico CMB, podem ser utilizados pelo protocolo *Time Warp*.

***Time Warp* → CMB**

Quando a troca de protocolos for no sentido oposto, a estrutura SS, típica do protocolo *Time Warp*, poderá ser desativada após o período de transição, uma vez que no protocolo CMB não poderá ocorrer uma violação da ordem cronológica de execução das mensagens. Após o período de transição a estrutura SS provavelmente estará vazia, uma vez que o valor do GVT deve alcançar o valor do LVT na fase de transição.

7.3.2 Canais de Comunicação

O protocolo de sincronização CMB clássico, proposto por Chandy e Misra (1979), não prevê um esquema de suporte para alocação dinâmica de canais de comunicação. Estudos posteriores (Jha & Bagrodia, 1994) mostram que também é possível uma conectividade dinâmica para os protocolos conservadores com algumas restrições.

Apesar da possibilidade de topologia dinâmica para o protocolo CMB, a implementação mais comum é a com topologia estática, dessa forma, para que os dois protocolos de sincronização possam operar simultaneamente no mecanismo proposto é preciso que o *Time Warp* também esteja trabalhando com topologia estática.

Assim, em relação ao uso do canal de comunicação, um dos requisitos necessários para a existência dos dois protocolos simultâneos é a execução em topologia estática. Ou seja, independente do protocolo que inicie a execução da simulação, a topologia dos canais de comunicação deverão já estar alocados de acordo com o modelo de aplicação que foi implementado.

CMB → *Time Warp*

No protocolo CMB todo processo lógico possui um conjunto de canais de comunicação pelos quais as mensagens são recebidas e armazenadas em suas portas de entrada (IB[i]) e cada porta de saída (OB[j]) possui um canal de comunicação que levará a mensagem até os outros processos. No momento da troca CMB → *Time Warp* o canal

de comunicação será mantido, uma vez que foi modificada a estrutura original do *Time Warp* de forma a utilizar a mesma estrutura que o CMB.

Essa modificação foi necessária para simplificar a transição entre os protocolos, porém, ela traz uma sobrecarga a mais, pois a variável passa a ser bidimensional. Porém, como a estrutura já era necessária para a transição *Time Warp* / CMB, a manutenção dessa variável não prejudica a simulação.

***Time Warp* → CMB**

A estrutura original do *Time Warp* foi modificada de forma que ele utiliza as estruturas IB[i] e OB[j] ao invés de apenas IB e OB. Isso faz com que seja necessário apenas voltar a simulação no ponto do GVT calculado como é feita na fase de transição.

7.3.3 Variáveis de Tempo

Os protocolos de sincronização utilizam variáveis para armazenar os valores de tempo que controlam as execuções dos eventos da simulação em sua ordem cronológica correta e para representar o avanço do tempo real do modelo simulado. As variáveis utilizadas pelos protocolos CMB e *Time Warp* estão representadas na Figura 7.4.

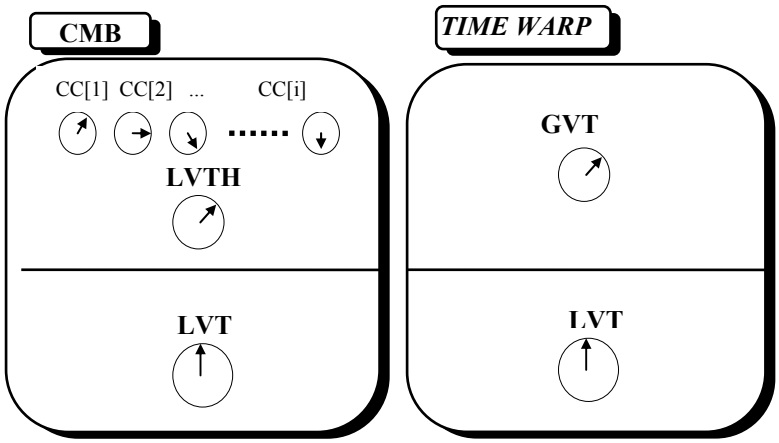


Figura 7.4: Variáveis de Tempo Utilizadas nos Protocolos CMB e *Time Warp* (Morselli, 2000)

- **Estruturas de Tempo Clock Channel (CC[i])**

As estruturas de dados IB[i] do protocolo CMB mantém as mensagens que foram recebidas ordenadas pelo seu tempo de ocorrência na simulação Desta forma, a mensagem que ocupa a primeira posição da fila é aquela que possui o menor valor do tempo de ocorrência recebida por aquele canal de comunicação. Este valor é armazenado em uma das estruturas CC[i].

Para que o controle local do processo possa decidir qual mensagem deverá ser escalonada (inserida na lista de eventos futuros) é escolhida a mensagem com menor valor entre as estruturas CC[i].

As estruturas CC[i] não são utilizadas pelo protocolo *Time Warp*. Isso ocorre porque no *Time Warp* as mensagens recebidas por um processo lógico são inseridas em apenas uma estrutura (IQ), assim basta ao controle local do processo *Time Warp* escalonar a mensagem que ocupa a primeira posição da estrutura IQ.

Para a abordagem de múltiplos protocolos executando simultaneamente é necessário criar a estrutura CC[i] para o *Time Warp*. Como a estrutura OB[i] já foi modificada para o *Time Warp*, a forma de atualização da estrutura CC[i] será pelos mesmos princípios do CMB, ou seja, como as mensagens serão colocadas em ordem cronológica em OB[i], a mensagem que ocupa o primeiro lugar na fila é o valor que será armazenado em CC[i].

A criação dessa estrutura no *Time Warp* é necessária para que o *Time Warp* possa informar a um processo lógico CMB qual o menor clock atual dele. Esse valor segue junto a mensagem (Figura 7.3).

Esse tempo é calculado pegando-se o menor valor entre os armazenados em CC[i]. Esse tempo assegura que até esse ponto a simulação do *Time Warp* foi segura. Informação que o CMB precisa para avançar com a simulação e saber se uma mensagem proveniente de um processo lógico *Time Warp* é seguro para ser executado.

A estrutura CC[i], apesar de causar uma sobrecarga no *Time Warp*, é necessária pois é a forma como o *Time Warp* pode informar para o CMB qual é o tempo seguro em que as mensagens podem ser processadas sem a ocorrência de erros de causa e efeito. Essa estrutura é atualizada de acordo com a chegada de mensagens, completas e nulas, dos canais de comunicação. Caso a mensagem que chega é de um *pl* CMB, o *clock* do

canal é atualizado com o *timestamp* da mensagem, porém, caso a mensagem que chega é de um *pl Time Warp*, o *clock* do canal é atualizado com o valor do campo C8 (menor valor *Time Warp*) da mensagem.

Apesar de representar uma sobrecarga a mais no *Time Warp*, como essa estrutura já existe no CMB, o espaço de memória ocupado por ela já estará alocada e não precisará ser modificada no caso de troca de protocolo, facilitando a transição de protocolo tanto no sentido CMB / *Time Warp* quanto *Time Warp* / CMB, pois esses valores não necessitarão de atualização no momento da troca.

Outro detalhe que deve ser implementado no *Time Warp* é o envio de mensagens nulas para processos lógicos que estejam executando o protocolo CMB. Isso é necessário para que o *pl* CMB atualize o *clock* do canal para avançar na simulação.

Essa mensagem nula contém o LVT e o menor valor dos CC[i] e deve ser enviada de acordo com a regra já existente no protocolo CMB. Assim, após cada execução de evento dentro do *pl Time Warp* ou a cada intervalo predefinido uma mensagem nula deve ser enviada.

Uma outra consideração importante a ser feita diz respeito ao envio de mensagens nulas para processos lógicos *Time Warp*. Apesar do *pl Time Warp* não necessitar das mensagens nulas para avançar na simulação, ele precisa receber esse valor para atualizar a estrutura CC[i] para cálculo do LVTH, garantindo que a simulação avance no tempo.

CMB → *Time Warp*

As estruturas IB[i] e CC[i] estão sendo tratadas da mesma forma nos dois protocolos. Dessa forma não são necessárias modificações.

***Time Warp* → CMB**

Quando a troca de protocolos for no sentido de *Time Warp* para CMB as mensagens já estarão armazenadas em IB[i] e as estruturas de tempo CC[i] já estarão com os valores corretos. Durante a fase de transição alguns *rollbacks* poderão ocorrer, mas não afetarão essas variáveis.

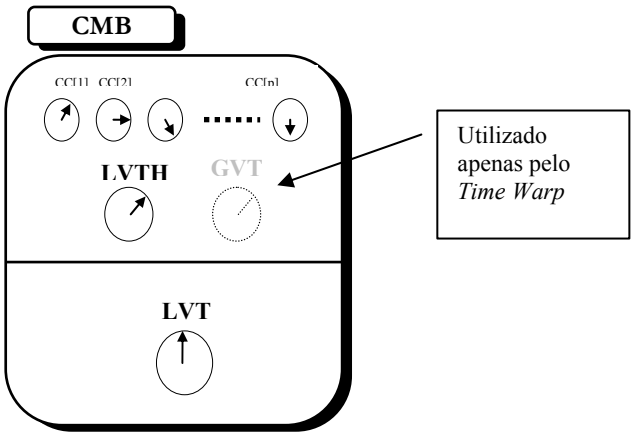


Figura 7.5: Mecanismo de Troca: Variáveis de Tempo Utilizadas no Modo CMB

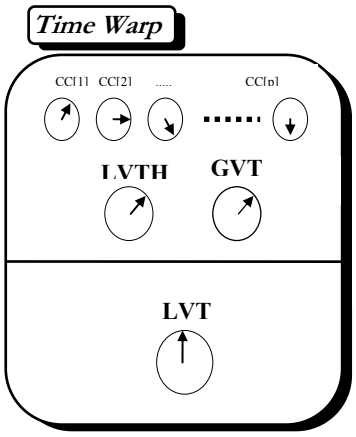


Figura 7.6: Mecanismo de Troca: Variáveis de Tempo Utilizadas no Modo *Time Warp*.

A Figura 7.5 mostra as variáveis que são utilizadas no mecanismo para o protocolo CMB. Em relação ao protocolo original, apenas a variável GVT foi adicionada, porém, essa variável fica desativada durante a vigência do protocolo CMB, sendo ativada apenas quando ocorrer a troca de protocolo para *Time Warp*.

Já a Figura 7.6 mostra as variáveis que são utilizadas no protocolo *Time Warp*. De acordo com as modificações propostas, o *Time Warp* passa a utilizar todas as variáveis relacionadas ao CMB para que a simulação possa ser executada com os dois protocolos simulataneamente.

- **Variável de Tempo *Local Virtual Time Horizon* (LVTH)**

A variável LVTH, utilizada exclusivamente pelo protocolo CMB, armazena o *menor* valor entre as mensagens que estão armazenadas nas portas de entrada (CC[i])

para cada canal de comunicação mencionado no item anterior. O evento que possui esta marca de tempo deverá ser escalonado na lista de eventos futuros.

Como a estrutura original do *Time Warp* precisou ser modificada com a utilização da estrutura CC[i] a variável LVTH passa a ter significado no protocolo *Time Warp* pois irá armazenar o valor que será enviado junto a cada mensagem em C8 (Figura 7.3).

- **LVT (Local Virtual Time)**

A variável LVT, tanto para o protocolo CMB como para o protocolo *Time Warp* corresponde à marca de tempo do último evento processado naquele processo lógico. Este valor, denominado *clock do processo*, possui uma grande importância, pois representa a quantidade de tempo virtual que o processo lógico simulou do processo físico correspondente. Além disto, é por meio do valor de LVT que se determina quando uma mensagem está fora da sua ordem cronológica.

A forma de calcular os LVT's nos dois protocolos considerados é diferente, mas seu significado e seu valor são o mesmo. Desta forma esta variável será mantida após o momento da troca de protocolos não importando qual protocolo esteja sincronizando a simulação no instante anterior a troca.

- **Variável de Tempo Global Virtual Time (GVT)**

Pode-se afirmar que um sistema de simulação simulou o comportamento de um sistema real até o valor de tempo armazenado em GVT. Este valor é obtido por meio do cálculo do menor LVT entre todos os processos do sistema de simulação.

No protocolo *Time Warp,* o *garbage collection* (eliminação dos estados da simulação armazenados para a necessidade de efetuar um *rollback*) é realizada por meio do valor do GVT disponível no processo lógico, ou seja, qualquer alteração efetuada no estado da simulação antes do valor de GVT é considerada estável.

O protocolo CMB, ao contrário, não prevê a existência deste valor armazenado em cada processo lógico (Figura 3.2 e Figura 3.3).

7.3.4 A simulação sendo executada com os dois protocolos

Diversas variáveis foram modificadas no esquema do *Time Warp* original para que fosse possível a execução simultânea dos dois protocolos.

Essa execução é possível graças a possibilidade de cálculo de GVT atualizado dentro de cada processo lógico que está executando *Time Warp*. O GVT caracteriza-se como o menor tempo entre todos os LVTs dos processos lógicos. Como a implementação atual permite a utilização da estrutura CC[i], o *pl Time Warp* possui a informação de tempo dos processos lógicos que fazem parte da simulação, dessa forma, o menor tempo entre os CC[i]'s será o LVTH, valor esse que sempre será menor ou igual ao valor do GVT da simulação.

Para garantir que sempre seja atualizado esse valor, é necessário criar um mecanismo de envio de mensagens nulas por parte do processo *Time Warp* quando ele detecta a presença de processos lógicos CMB na simulação.

Isso é possível, pois os processos lógicos terão uma estrutura que armazena o *status* (estrutura TP – tipo de protocolo que está sendo executado) dos processos lógicos quanto ao tipo de protocolo que ele está executando (Figura 7.7).

Para saber quais são as mensagens seguras de serem executadas no CMB, existe uma lista de *clock* dos processos lógicos que pode indicar qual o menor *clock* e quais mensagens podem ser executadas de acordo com esse *clock*.

Isso possibilita a utilização dessa lista para o cálculo do GVT do processo lógico. Criando-se uma regra onde caso o processo lógico seja *Time Warp* e um outro seja CMB, quando um processo TW mandar uma mensagem para o CMB, além de enviar junto à mensagem o *timestamp* do evento, o TW envia também o tempo mínimo local calculado.

A Figura 7.7 mostra o esquemático do processo lógico que deve ser utilizado na abordagem de troca de protocolo local. Nessa figura é possível observar que estão presentes todas as variáveis necessárias para a execução do protocolo CMB e todas para a execução do protocolo *Time Warp*, além da variável TP (tipo do protocolo) que define qual protocolo está ativo em cada um dos processos lógicos que compõe a simulação.

De acordo com as modificações que foram propostas para o mecanismo de troca, as variáveis dos dois processos lógicos são necessárias para o andamento da simulação. Quando o protocolo CMB estiver sendo executado, as variáveis relacionadas ao protocolo CMB estarão todas ativas. Na ocorrência de troca de protocolo, além do uso das variáveis do CMB, o *Time Warp* também promoverá o uso das variáveis que estão relacionadas ao *rollback*, como as estruturas SS, IQ e OQ.

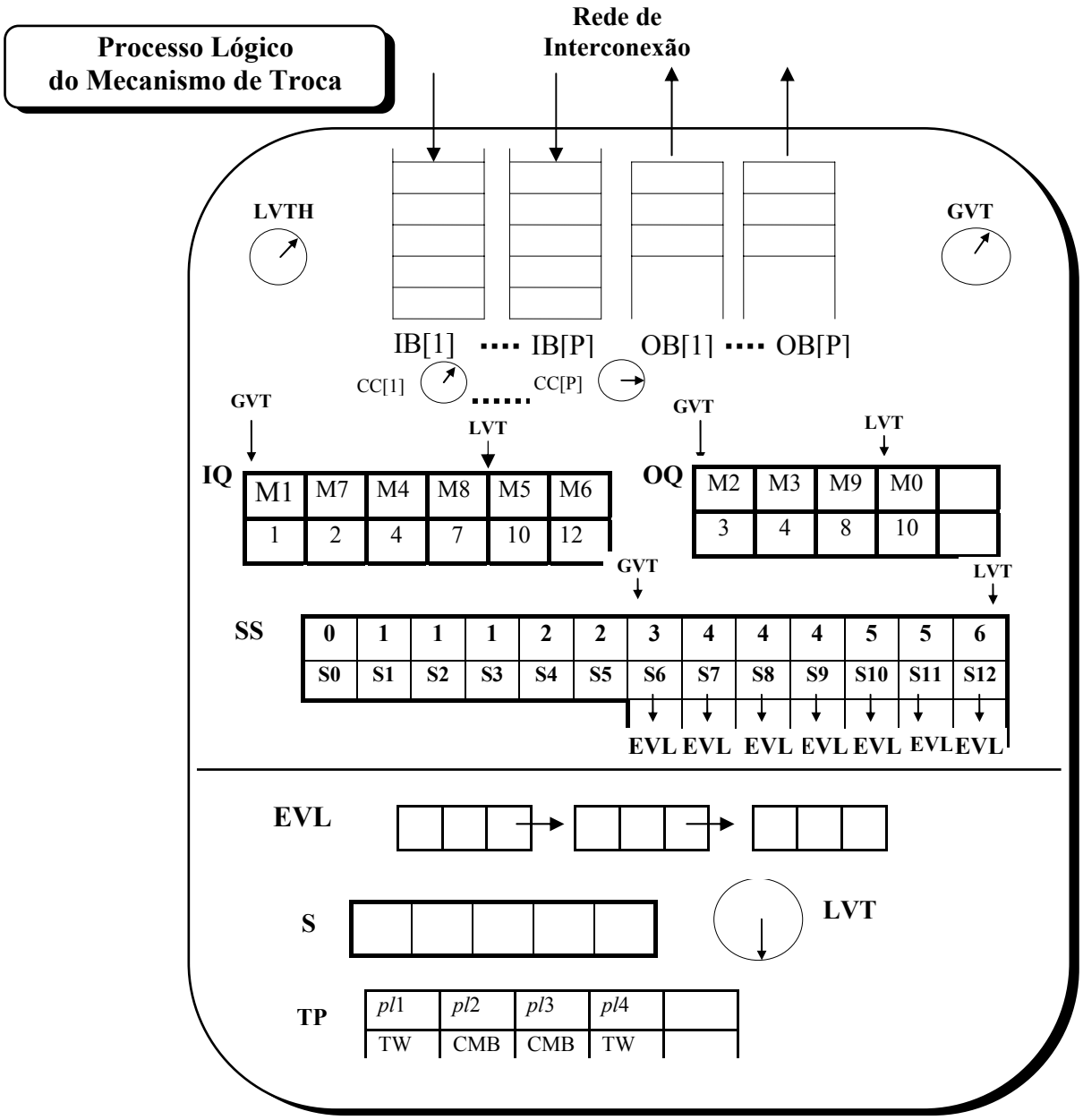


Figura 7.7: Processo lógico utilizado pelo mecanismo de troca de protocolo.

Detalhando-se os passos que ocorrem na simulação, é possível ter-se para cada protocolo ações diferentes de acordo com o tipo de protocolo que está sendo executado e a mensagem que chega. Dessa forma, é necessário analisar cada um dos casos.

- **Processo lógico executando CMB recebendo mensagem CMB**

Neste caso, a mensagem deve ser tratada da mesma forma como se somente estivesse sendo executado apenas CMB em todos os *pls* da simulação. A mensagem pode ser completa ou nula, mas as ações não serão diferentes da simulação CMB convencional.

- **Processo lógico executando TW recebendo mensagem TW**

As mensagens provenientes de um *pl* TW para outro TW devem ser tratadas da mesma forma que a execução do TW tradicional. Essa mensagem que chega pode ser uma mensagem ou antimensagem. Caso seja uma mensagem, deve ser colocada na lista de eventos e caso seja uma antimensagem, tanto pode causar um cancelamento de mensagem da lista de eventos quanto um *rollback*. Nos dois casos o tratamento deve seguir os passos definidos na implementação convencional do TW.

- **Processo lógico executando CMB recebendo mensagem TW**

Quando um pl CMB recebe uma mensagem TW essa mensagem pode ser:

Mensagem completa: a mensagem completa contém um evento que deve ser escalonado na EVL em ordem cronológica e o *pl* deve atualizar o valor de CC[i]. Como o valor de CC[i] é atualizado com o menor valor *Time Warp* e não com o *timestamp* da mensagem, assegura-se que essa mensagem só será executada quando o menor tempo da mensagem *Time Warp* avançar até o *timestamp* dessa mensagem. Quando isso acontecer sabe-se que não irão ocorrer erros de causa e efeito. Essas considerações já foram discutidas na seção 7.3.1.

Mensagem nula: a mensagem nula que chega de um *pl* TW serve apenas para atualizar o valor de CC[i] (com o valor do campo C8 da mensagem – menor tempo entre os *clocks* dos canais do remetente) e dessa forma, eventos podem ser desbloqueados para serem executados.

Anti-mensagem: as antimensagens são mensagens de eventos que foram executados em ordem cronológica errada. Porém, no caso dessa mensagem, como o CMB sempre executa eventos seguros, para que essa mensagem tivesse sido executada erroneamente, o valor de LVTH deveria ser maior que o tempo dessa mensagem. Porém, como todos os *clocks* de canais são vasculhados para ter-se o menor tempo entre todos os canais, a mensagem referente a antimensagem que chegou só pode estar na fila de eventos futuros, pois não existe a possibilidade dela ter sido executada. Dessa forma, com a chegada de uma antimensagem, a única providência que o *pl* CMB deve tomar é cancelar a mensagem referente à antimensagem que chegou.

- **Processo lógico executando TW recebendo mensagem CMB**

  Essa mensagem pode ser de dois tipos:

Mensagem completa: essa mensagem deve ser armazenada normalmente na lista de eventos futuros.

Mensagem nula: a mensagem nula que chega de um *pl* CMB serve apenas para atualizar o valor de CC[i]. Esse valor deve ser atualizado para que no envio de mensagens do TW para o CMB ele sempre esteja com um tempo atualizado.

A Figura 7.8 e a Figura 7.9 resumem o comportamento dos protocolos com a chegada de mensagens.

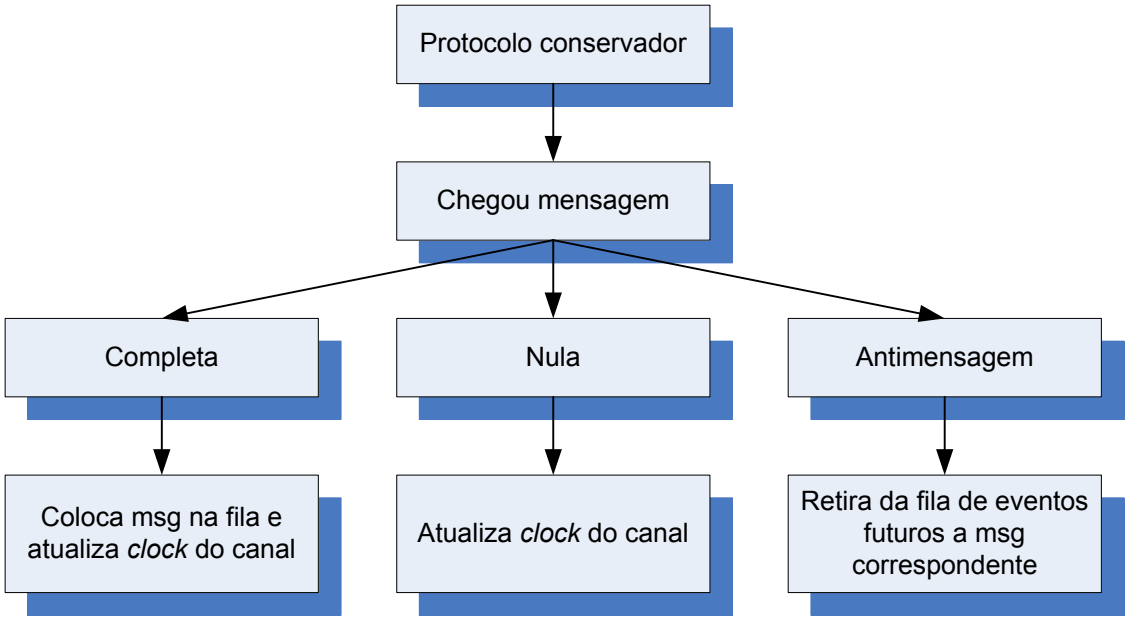


Figura 7.8: Comportamento do protocolo conservador com a chegada de mensagem.

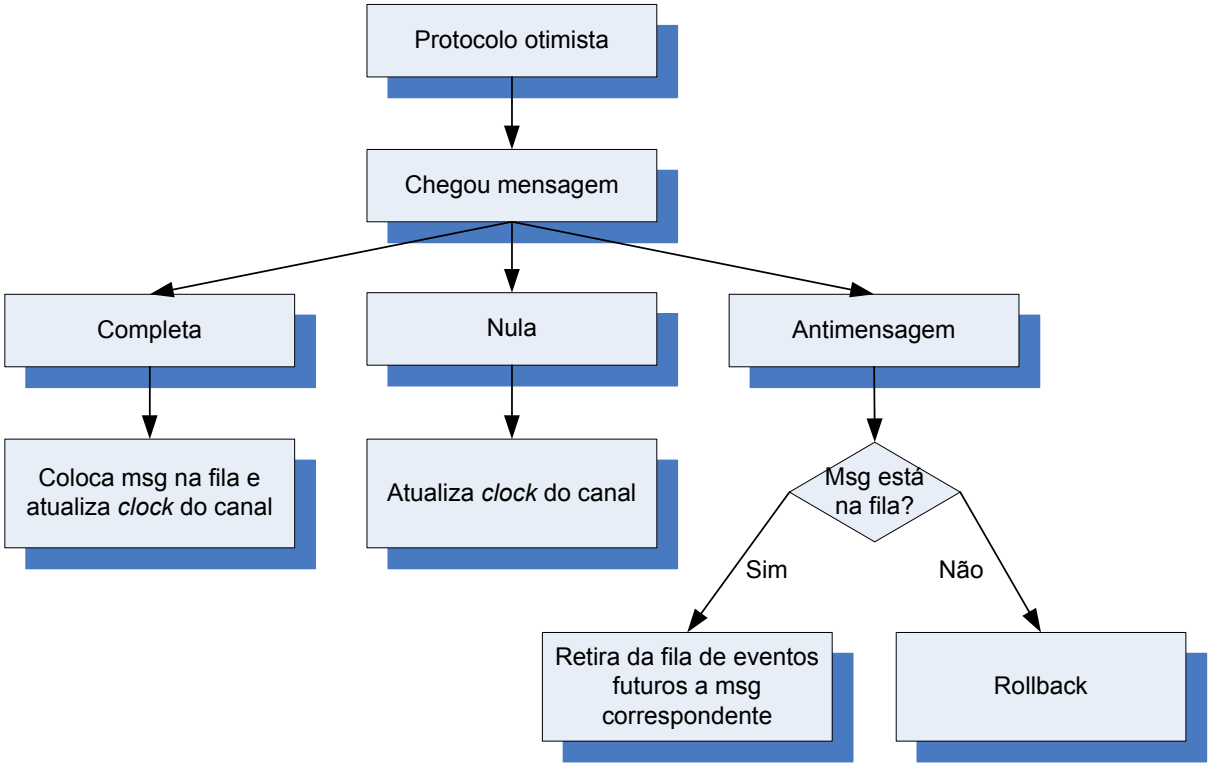


Figura 7.9: Comportamento do protocolo otimista com a chegada de mensagem.

A abordagem descrita torna possível a execução da simulação com os dois protocolos. É possível que ocorra uma queda de desempenho durante um intervalo na simulação (o intervalo em que ocorre a troca de protocolo), porém com o andamento da simulação diversas situações podem ocorrer. Essas situações estão descritas a seguir.

Supondo uma simulação iniciada com o protocolo CMB. Caso exista um processo lógico que sempre está bloqueado e possui eventos que poderiam ter sido executados quando da chegada de mensagens nulas ou completas, é visível que uma mudança de protocolo poderia surtir melhor desempenho. Dessa forma, uma troca de protocolo deve ser considerada.

Com a proposta de execução simultânea dos dois protocolos sendo possível, apenas um processo lógico muda para TW e continua sua execução. Caso os demais processos lógicos passem a verificar que ficam bloqueados grande parte do tempo com mensagens que poderiam ter sido executadas, esses processos lógicos devem também mudar o protocolo. Essa situação pode fazer com que a simulação troque totalmente de protocolo. Nesse caso, do protocolo CMB para o *Time Warp*.

Uma outra possibilidade também seria que a simulação melhore seu desempenho apenas com a troca de um processo lógico, ou seja, a abordagem de vários protocolos sendo executados simultaneamente contempla os modelos de simulação que possuem características em que parte do modelo possui desempenho melhor com um ou outro protocolo.

Supondo agora que a simulação iniciou a sua execução em *Time Warp*. Um processo lógico que executa muitos *rollbacks* em relação a execução de eventos da simulação não está tendo um bom desempenho com o protocolo TW. Nesse caso, a mudança de protocolo é feita como descrito nesse capítulo. Caso os demais processos lógicos também estejam executando muitos *rollbacks*, um a um os demais processos lógicos podem migrar para CMB.

A possibilidade de que a simulação não melhore com uma troca de protocolo também é possível, e dessa forma, um mecanismo que impeça que um processo lógico fique migrando de protocolo diversas vezes deve ser implementado e, principalmente, a análise do desempenho do protocolo deve estar bem definido de forma que a simulação realmente tenha melhor desempenho com uma troca de protocolo.

Apesar de que o pior caso seria um processo lógico migrar de um protocolo e logo depois migrar novamente para o protocolo inicial, a abordagem descrita nesta tese, em que cada processo lógico decide a troca e faz a troca localmente, causa um impacto menor na simulação do que a abordagem original em que todos os processos lógicos são migrados para outro protocolo.

Com base nos resultados apresentados no capítulo 5 é possível verificar se o protocolo que está em execução está com um desempenho não satisfatório e deve portanto ser alterado e é possível também determinar se a distribuição não é adequada para a simulação em andamento. Essa informação pode ser útil para evitar essa troca constante de protocolos pois esses detalhes podem indicar que os protocolos de sincronização distribuídas não estão tendo bom desempenho e a utilização de outras abordagens, como por exemplo, MRIP ou simulação seqüencial são mais indicadas.

Porém, pode ser detectado também que uma abordagem seqüencial não é a mais indicada. Nesse caso, é possível obter-se métricas que indicam qual foi o protocolo que teve um desempenho melhor, considerando-se que ambos os protocolos tiveram um desempenho ruim. Assim, procura-se manter o protocolo que teve um desempenho aquém do esperado, mas que é melhor que a outra abordagem.

7.3.5 A simulação com os dois protocolos: outra abordagem

Visando melhorar o desempenho da simulação, minimizando a sobrecarga gerada para possibilitar a execução de protocolos simultâneos, alguns melhoramentos podem ser considerados no mecanismo descrito. Um dos pontos a ser analisado refere-se aos *clocks* dos canais.

Apesar do uso do tempo mínimo dos *clocks* dos canais na proposta descrita garantir o andamento da simulação, a utilização de mais uma variável na mensagem que trafega pela rede poderia não ocorrer. Inclusive, o cálculo desse valor para o envio junto à mensagem também poderia ser suprimida.

Essas considerações podem ser válidas se algumas premissas puderem ser feitas em relação a uma simulação em andamento:

- Uma mensagem *Time Warp* com tempo menor sempre vai chegar antes que uma mensagem com tempo maior em um processo lógico CMB. Essa garantia faz parte das premissas da simulação conservadora.
- A topologia continua sendo estática e conhecida desde o início da simulação.

Essas considerações limitam a implementação original do protocolo *Time Warp*, pois são necessárias garantias de entrega da rede em ordem de envio de mensagens. Uma rede em um ambiente controlado com um protocolo adequado garante essa ordem de recebimento. Uma limitação que essa abordagem causa é a não existência de possibilidade de uso de uma rede baseada na internet, pois mensagens podem chegar fora de ordem e assim, mensagens não seguras podem ser executadas erroneamente.

Em relação à topologia estática, ela é necessária para garantir o conhecimento dos tempos de todos os canais, que é a forma como o CMB monitora a execução dos eventos. Se fosse utilizada topologia dinâmica não seria possível ter-se controle sobre os canais de comunicação e assim, poderiam ocorrer violações na propriedade de causa e efeito.

Com essas premissas, o valor mínimo local não é mais necessário, pois a forma como o CMB faz a execução da simulação garante que os eventos não serão executados em ordem cronológica errada, pois o *pl* CMB analisa todos os *clocks* de canal para avançar na simulação.

As mensagens que vão chegando do *pl* TW ficam na fila de eventos, mas somente depois que o evento é processado é que o *clock* do canal é atualizado com o próximo *timestamp* da mensagem que se encontra na fila.

Utilizando a premissa que uma mensagem com tempo menor sempre chega antes que uma de tempo maior, não é possível que o *pl* CMB execute um evento com *timestamp* maior, que caracterizaria uma violação na propriedade de causa e efeito. Dessa forma, o processo lógico não precisa da informação do valor mínimo de *clock* para garantir que as mensagens serão executadas na ordem correta. Se ocorrer de um canal ficar sem mensagens, o processo lógico fica bloqueado até a chegada de uma mensagem ou mensagem nula.

7.4 Considerações finais

O mecanismo original proposto por Morselli não abrangia aspectos relativos à sobrecarga que o mecanismo impunha a simulação. Esses aspectos incluem:

- A existência de um processo a mais na simulação (processo gerenciador);
- A sobrecarga de tráfego de dados de desempenho dos processos lógicos além do tráfego da simulação.
- A troca de protocolo de processos lógicos que podem estar com bom desempenho, mas estão com a simulação como um todo com um desempenho insatisfatório;
- O atraso que a troca total de protocolo possui intrinsecamente pela necessidade de colher informações de todos os processos lógicos, o processamento dessas informações, a decisão de troca e o envio da ordem de troca;
- A necessidade de paralização da simulação em todos os processos lógicos para efetuar a troca de protocolo.

Da análise desses aspectos verificou-se que esses podem ser minimizadas com uma proposta de modificação do mecanismo, que inclui a troca parcial do protocolo e decisão de troca descentralizada.

Essas modificações foram detalhadas neste capítulo e um conjunto diferencial pode ser visualizado nessa nova abordagem:

- A simulação não tem mais a sobrecarga de um processo a mais na simulação, ou seja, o processo gerenciador não é mais necessário, pois não há a necessidade de um processo que centralize todas as informações e tome a decisão de troca.
- Cada processo lógico colhe as infomações de desempenho independentemente e verificam o desempenho localmente, não impondo nenhuma sobrecarga de tráfego de dados na rede, além do tráfego normal associado à simulação.
- Modelos que possuem parte dele mais adequados a um protocolo e parte a outro são contemplados com essa abordagem. E, todas as modificações que são necessárias foram abordadas neste capítulo.

- Na ocorrência da necessidade de troca de protocolo, a troca é mais simples e mais rápida com a abordagem proposta, pois não há a necessidade de trocas de informações com outros processos lógicos. O *pl* apenas verifica que seu desempenho não está bom e efetua a troca, comunicando aos demais processos lógicos a sua nova condição.
- O mecanismo proposto independe da arquitetura que se está utilizando. O sistema computacional pode ser homogêneo ou heterogêneo e o mecanismo continua sendo útil, pois o desempenho de cada processo lógico irá se adequar à realidade da máquina em que está sendo executado.

Considerando esses dados, a proposta apresentada atinge os objetivos de propor um melhor aproveitamento da simulação distribuída por meio da proposta de um ambiente com protocolos diferentes sendo executados simultaneamente. Esse capítulo mostrou a viabilidade de implementação dessa abordagem e detalhes de funcionamento desse sistema.

8 Conclusões

8.1 Conclusões gerais

Esta tese de doutorado propõe uma nova abordagem para a execução de simulação distribuída visando melhorar o desempenho e facilitar a utilização deste tipo de simulação. O trabalho desenvolvido está inserido em uma linha de pesquisa (em desenvolvimento tanto no Grupo de Sistemas Distribuídos e Programação Concorrente do ICMC como em outros grupos, por exemplo, o Grupo *UCLA Parallel Computing Laboratory* nos Estados Unidos (Bagrodia 1999, Bagrodia 2000), o *Grupo PARADISE Research Laboratory* no Canadá (Boukerche & Tropper 2001) e o Grupo Parallel *and Distributed Simulation* do *Georgia Institute of Technology* nos Estados Unidos (Perumalla 2005, Fujimoto 2003)), cujo objetivo é facilitar a utilização de simulação distribuída por meio de mecanismos que facilitem ao usuário a escolha do protocolo, de ferramentas, de particionamentos de modelos, etc. mais adequados.

O trabalho descrito nesta tese teve início com um estudo sobre o comportamento de uma simulação distribuída. O ParSMPL, uma extensão funcional para a linguagem C foi instrumentado para avaliar o andamento da simulação distribuída durante sua execução. Com os dados resultantes desta instrumentação pode-se tirar várias conclusões em relação tanto a simulação como um todo quanto a cada processo lógico individualmente.

Os resultados obtidos mostraram que a simulação distribuída pode ter características em que parte dela se adequa melhor a um protocolo e parte a outro protocolo. Nesse sentido, verificou-se que a possibilidade de se ter protocolos diferentes para uma mesma simulação torna-se interessante para melhorar o desempenho.

Os estudos efetuados mostraram que durante a execução de um modelo tem-se processos lógicos executando com bom desempenho no protocolo conservador (poucos

eventos estão bloqueados de forma desnecessária e se fossem executados provocariam *rollbacks*), e processos lógicos onde a troca para o protocolo otimista levaria a um melhor desempenho (eventos bloqueados de forma desnecessária).

Os testes mostraram também que em alguns casos, os processos lógicos não atingiram um bom desempenho com o protocolo conservador, e que a troca para o protocolo otimista também poderia não atingir um bom desempenho devido à granulosidade baixa, particionamento desfavorável, dentre outras características. Nestes casos, tem-se muitos eventos bloqueados, dificultando o andamento da simulação distribuída mas que, se fossem executados ocasionariam *rollbacks*. Dessa forma, como os protocolos de simulação distribuídos não oferecem bom desempenho, uma solução para esses casos é a utilização da simulação seqüencial ou MRIP onde diversas instâncias são simuladas em paralelo.

Um dos objetivos da instrumentação da simulação distribuída é determinar quais métricas são adequadas para verificar se a simulação está caminhando de forma apropriada com o protocolo escolhido para iniciá-la. Com os resultados obtidos pode-se concluir que nenhuma métrica utilizada individualmente é suficiente para concluir-se sobre o andamento da simulação. Por outro lado, utilizando-se um conjunto de métricas e relacionando-se essas métricas é possível concluir se o protocolo em uso é apropriado e se não for, se a troca poderia levar melhores resultados. As métricas que são propostas neste trabalho são: relação executando / bloqueado, porcentagem de tempo bloqueado necessariamente, porcentagem de tempo bloqueado desnecessariamente, porcentagem de mensagens nulas que causaram bloqueios desnecessários e porcentagem de mensagens completas que causaram bloqueios necessários.

Todas essas métricas são obtidas localmente nos processos lógicos e a combinação dessas métricas tornou possível a postulação de regras que indicam qual o desempenho que o processo lógico está apresentado.

De acordo com as métricas obtidas pode-se chegar às seguintes conclusões:

- O desempenho está bom e o protocolo deve ser mantido.
- O desempenho não está bom e o protocolo deve ser alterado.
- O desempenho não está bom e a troca não é indicada, sendo mais adequada a simulação seqüencial ou MRIP.

Por meio da análise das métricas mencionadas para cada processo lógico de uma simulação, pode-se observar que na maioria dos casos coexistem processos onde a

conclusão sobre a troca ou não de protocolos são distintas. Essa situação dificulta a possibilidade de troca global de todos os processos lógicos e fortalece a idéia de troca parcial envolvendo apenas os processos lógicos que apresentam desempenho não adequado, pois uma troca de protocolo total pode significar uma melhora no desempenho em alguns processos lógicos e uma piora em outros. Nesse sentido, é possível que a troca de protocolo possa manter o desempenho ruim e em alguns casos, até piorar o desempenho da simulação.

Essas conclusões levaram a mais estudos sobre a dificuldade na troca de protocolo total, e a investigação sobre onde estavam os pontos de maior sobrecarga da troca de protocolo total. Verificou-se que o mecanismo de troca pode ser separado de forma natural em dois momentos: a decisão de troca e a troca propriamente dita.

Esses dois momentos podem ser analisados separadamente e podem ser efetuados de forma local ou global, dependendo da forma como se deseja analisar os dados e/ou consolidar a troca de protocolo. Essa separação propiciou uma proposta de taxonomia em que são considerados se a análise é local ou global e se a troca é local ou global.

A análise global com troca global refere-se ao mecanismo de troca de protocolo proposto por Morselli (2000). As principais desvantagens dessa abordagem são a sobrecarga que a fase de análise (sobrecarga de comunicação) apresenta e a troca global que necessita da paralisação da simulação para se efetuar a troca. A abordagem de análise local com troca local pode minimizar essas sobrecargas, levando-se a obtenção de um melhor desempenho.

Assim, os resultados obtidos mostram que a coexistência de protocolos de sincronização pode levar a obtenção de melhor desempenho na execução de uma simulação distribuída. No entanto, essa abordagem é nova e torna-se necessário demonstrar sua viabilidade e avaliar quais as vantagens e desvantagens que apresenta. Um estudo realizado nesta tese mostra não apenas a viabilidade, mas as vantagens apresentadas pela abordagem local.

Nesse sentido foi realizada uma avaliação das modificações que devem ser efetuadas nos protocolos de forma a tornar possível a existência de diferentes protocolos sendo executados na mesma simulação: criou-se uma proposta de funcionamento do mecanismo TLAL, foi feita uma análise das estruturas de dados e uma análise da forma

de comunicação entre os processos lógicos de forma a compatibilizar o funcionamento dos dois protocolos dentro da mesma simulação.

Na proposta desenvolvida criou-se um processo lógico que une variáveis dos dois protocolos (conservador e otimista) e cujas variáveis foram compatibilizadas, inclusive no seu funcionamento, para que a troca de protocolo possa ser efetuada tão logo seja detectado um baixo desempenho.

Como a comunicação entre os processos lógicos nos protocolos conservadores e otimistas é diferente, procurou-se compatibilizar o formato da mensagem que trafega de forma que ela sempre carregue as informações pertinentes e necessárias aos dois protocolos.

Resumindo, esta tese de doutorado mostra, por meio da monitoração de simulações distribuídas em execução, que a utilização de mais de um protocolo de sincronização em uma mesma simulação pode levar a ganhos em seu desempenho. Mostra ainda que a coexistência de protocolos é viável e apresenta diversas vantagens em relação a outras opções.

A possibilidade de trocas de protocolo em tempo de execução facilita a utilização de simulação distribuída por usuários que utilizam a simulação como uma ferramenta e que não possuem o conhecimento necessário para escolher o melhor protocolo de sincronização. Assim, o trabalho desenvolvido nesta tese de doutorado contribui de forma significativa para tornar a simulação distribuída uma opção viável para o usuário de outras áreas que utilizam simulação para avaliar o desempenho de sistemas.

8.2 Contribuições desta tese

A principal contribuição desta tese é a demonstração de que em uma mesma simulação, os processos lógicos possuem desempenho diferenciado indicando a necessidade de criar uma forma de contemplar os modelos de simulação com um mecanismo capaz de melhorar o desempenho da simulação, extrapolando a troca de protocolo quando detectado baixo desempenho. Nesse sentido, esta tese de doutorado propõe o uso de mais de um protocolo dentro de uma mesma simulação, propondo, a cada processo lógico, o protocolo mais adequado, de acordo com as métricas que podem

ser extraídas durante a execução da simulação. Todo o esforço efetuado para a proposição desta tese pode ser resumido nas contribuições a seguir:

1. Realização de testes no ambiente de simulação distribuída conservadora ParSMPLX:

- A análise de desempenho descrita nesta tese é efetuada em tempo de execução. As análises encontradas na literatura mostram avaliações de desempenho que coletam informações de desempenho que podem ser empregadas para antever o desempenho de um modelo ou para avaliar o desempenho da simulação já efetuada. A proposta do estudo efetuado no ambiente ParSMPLX foi definir métricas que podem ser utilizadas durante tempo de execução e que permeiem a decisão de troca de protocolo.

- Foram feitas análises sobre desempenho da simulação conservadora em uma ambiente de simulação desenvolvido e estabelecido. Nesse ambiente foram definidas métricas que podem ser obtidas durante a execução da simulação de forma que dados possam ser obtidos para uma análise de desempenho.

- A análise de desempenho desenvolvida, verificando cada processo lógico isoladamente mostrou que cada processo lógico possui um desempenho diferente que é passível de ser medido por meio de monitores de software. As métricas que podem ser extraídas são: relação tempo executando / tempo bloqueado, porcentagem de mensagens completas que geraram bloqueios necessários e porcentagem de mensagens nulas que geraram bloqueios desnecessários. De acordo com os valores que se obtém com essas métricas aplicadas a uma simulação é possível definir o desempenho da simulação e propor uma mudança de protocolo caso seja detectado um baixo desempenho.

- Processos lógicos que possuem uma relação executando / bloqueado acima de um, ou seja, passam mais tempo processando do que bloqueados podem ser considerados com bom desempenho, e dessa forma, devem manter o protocolo corrente. Nesses casos, verificou-se

que a porcentagem de mensagens completas que causaram bloqueios necessários é alta.

- Processos lógicos que possuem uma relação executando / bloqueado abaixo de um não estão tendo bom desempenho. Nesse caso, uma análise das outras métricas é necessária. Assim caso seja detectado que a porcentagem de mensagens completas que causaram bloqueios necessários é baixa (abaixo de 55%) e a porcentagem de mensagens nulas que causaram bloqueios desnecessários é alta (acima de 50%) deve-se trocar o protocolo para otimista.
- Processos lógicos que apresentem uma relação executando / bloqueado abaixo de uma porcentagem de mensagens completas que causaram bloqueios necessários alta (acima de 55%) não irá melhorar seu desempenho com o protocolo otimista, pois ele fica muito tempo bloqueado necessariamente. Nesse caso, uma solução é o uso da simulação seqüencial ou MRIP.

2. Definição de uma taxonomia de acordo com os momentos que podem ser extraídos do mecanismo: decisão (local ou global) e troca (local ou global):

- Foi proposta uma taxonomia que leva em consideração os dois momentos que podem ser extraídos de um mecanismo de troca de protocolo: a fase de análise de desempenho e a troca propriamente dita. Esses momentos podem ser separados e cada um deles pode ser efetuado localmente ou globalmente. Dessa forma, quatro abordagens podem ser seguidas e as vantagens e desvantagens de cada uma dessas abordagens foi discutida.
- A abordagem TGAG é a abordagem original e que motivaram os estudos efetuados no doutoramento. Essa abordagem é a que faz a análise global e a troca de protocolo para todos os processos lógicos. Essa abordagem foi avaliada e dela surgiram as demais propostas, que foram motivadas pelos testes realizados.
- A abordagem TLAL é a abordagem que possui uma menor sobrecarga e que contempla modelos em que parte é melhor simulado com um

protocolo e parte com outro e, um detalhamento dessa abordagem foi feita e foram descritas suas vantagens e desvantagens.

3. Detalhamento da abordagem TLAL:

- Foi proposto um mecanismo que contempla modelos em que parte da simulação possui melhor desempenho com um protocolo e parte com outro protocolo, pois a possibilidade de coexistência de protocolos diferentes para uma mesma simulação é viável e foi definida.
- Esse mecanismo foi definido com as modificações necessárias em relação a estruturas de dados, consistência da simulação, troca de mensagens, e andamento da simulação como um todo.
- O mecanismo modificado torna a simulação mais flexível, pois cada processo lógico pode ser executado com o protocolo que mais se adequa às características de simulação desse processo. Com esse mecanismo, plataformas heterogêneas podem ser melhor exploradas pois de acordo com o desempenho apresentado pelo processo lógico sendo executado, é possível adequar o melhor protocolo às características da máquina que está sendo utilizada.

8.3 Sugestões para trabalhos futuros

Os experimentos que foram conduzidos nesta tese mostram que a simulação distribuída é um campo bastante amplo e mais pesquisas precisam ser produzidas de forma que se adquira o desempenho esperado para a simulação efetiva de sistemas mais complexos.

Como sugestões para trabalhos futuros podem-se citar:

I. Aprimoramento dos resultados e métricas para avaliação de desempenho de simulações distribuídas em tempo de execução.

a) Os resultados apresentados nesta tese advêm da simulação de modelos. Esses modelos, sua parametrização e divisão em processos lógicos foram escolhidos com o objetivo de cobrir o maior número possível de casos. No entanto, algumas opções podem ainda ser exploradas. Assim, trabalhos futuros devem

considerar novos modelos de simulação com o objetivo de refinar e ampliar os resultados obtidos.

b) A alta complexidade de sistemas que devem ser avaliadas por meio de simulações distribuídas foi "simulado" aumentando-se a granulosidade dos modelos. Devido a elevada quantidade de modelos avaliados neste trabalho, não foi possível considerar ainda modelos reais e complexos. A avaliação deste tipo de modelo em trabalhos futuros poderá auxiliar no refinamento dos resultados.

c) Análise de novas métricas pra avaliação das simulações. Este trabalho sugere a utilização de diversas métricas. Novas métricas podem ser sugeridas e avaliadas.

II. Implementação da Troca de Protocolos em Tempo de execução e com a coexistência de protocolos. Com o mecanismo proposto implementado pode-se avaliar com maior precisão o seu desempenho. Essa implementação possibilitaria ainda a comparação de diferentes mecanismos considerando modelos com características distintas.

III. Avaliação da possibilidade de trocar uma simulação SRIP para MRIP em tempo de execução. Como foi mostrado que em diversos casos a simulação MRIP é a mais adequada deve-se investigar a possibilidade desta opção ser considerada durante a execução da simulação.

IV. Execução dos modelos descritos nesta tese no protocolo otimista para a definição de métricas que possam ser utilizadas para avaliar o desempenho durante a execução da simulação distribuída no protocolo otimista. Em trabalhos futuros poderiam ser avaliadas quais métricas podem ser utilizadas no protocolo otimista para definir se a troca de protocolo deve ou não ser efetuada.

Referências

AGARWAL, A.; HYBINETTE, M.; XIONG, Y. (2005). Merging parallel simulation programs. In: **THE 18TH WORKSHOP ON PARALLEL AND DISTRIBUTED SIMULATION – PADS 2005**. Proceedings… Monterrey, CA, USA.

ALMASI, G. S.; GOTTIEB, A. (1994). **Highly Parallel Computing**. The Benjamin/Cummings Publishing Company.

ALONSO, J. M. et al. (1998). An evaluation of implementations of the CMB parallel simulation algorithm on distributed memory multicomputers. **Journal of systems architecture**, v.44: (6-7) p.519-545.

ALONSO, J. M.; FRUTOS, A. A.; PALACIO, R. B. (1994). Conservative and optimistic distributed simulation in massively parallel computers: a comparative study, In: **THE 1ST INTERNATIONAL CONFERENCE ON MASSIVELY PARALLEL COMPUTING SYSTEMS**. Proceedings… p.528-532.

AVRIL, H.; TROPPER, C. (2001). On rolling back and checkpointing in Time Warp, **IEEE transactions on parallel and distributed systems**, v.12: (11) p.1105-1121.

BAGRODIA, R. et al. (1999). Performance prediction of large parallel applications using parallel simulations. In: **SYMPOSIUM ON PRINCIPLES AND PRACTICE OF PARALLEL PROGRAMMING**. ACM SIGPLAN. Atlanta. Georgia. USA.

BAGRODIA, R.; TAKAI, M. (2000). Performance evaluation of conservative algorithms in parallel simulation languages. **IEEE transactions on parallel and distributed systems**, v11: (4), p.395-411.

BALAKRISHNAN, V. et al. (2001). A performance and scalability analysis framework for parallel discrete event simulators. **Simulation practice and theory**, v. 8: (8), p.529-553.

BAUER, D. et al. (2005). Seven-O'Clock: a new distributed GVT algorithm using network atomic operations. In: **THE 18TH WORKSHOP ON PARALLEL AND DISTRIBUTED SIMULATION – PADS 2005**. Proceedings… Monterrey, CA, USA.

BAUSE, F.; EICKHOFF, M. (2003). Truncation point estimation using multiple replications in parallel. In: **THE 2003 WINTER SIMULATION CONFERENCE**. Proceedings… p.414-421.

BONONI, L. et al. (2005). Concurrent replication of parallel and distributed simulations. In: **THE 18TH WORKSHOP ON PARALLEL AND DISTRIBUTED SIMULATION – PADS 2005**. Proceedings… Monterrey, CA, USA.

BOUKERCHE, A.; TROPPER, C. (2001). Local versus global lookahead in conservative parallel simulations. **Parallel computing**, v27: (8), p.1033-1055.

BRUSCHI, S. M. (2003). **ASDA - Ambiente de simulação distribuída automático**. Tese (Doutorado) - Instituto de Ciências Matemáticas e de Computação, Universidade de São Paulo, São Carlos. 2003.

BRUSCHI, S. M. et al. (2004). An automatic distributed simulation environment. In: **2004 WINTER SIMULATION CONFERENCE – WSC 2004**. Proceedings… Washington, USA.

BYRANT, R. E. (1977). **Simulation of packet communication architecture computer systems**. MS Dissertation - Massachusetts Institute of Technology, Cambridge, Massachusetts. 1977. USA.

CHANDY, K. M.; MISRA, J. (1979). Distributed simulation: a case study in design and verification of distributed programs. **IEEE transactions on software engineering**, v.SE-5, n.05, p.440-452.

CHANDY, M.; MISRA, J. (1981). Asynchronous distributed simulation via a sequence of parallel computations. **Communication ACM**, v.24, n.4, p.198-205.

CHEN, G.; SZYMANSKI, B. K. (2002). Lookback: a new way of exploiting parallelism in discrete event simulation. In: **THE 16TH WORKSHOP ON PARALLEL AND DISTRIBUTED SIMULATION**. Proceedings… p.153-162.

CHEN, G.; SZYMANSKI, B. K. (2002b). Lookahead, rollback and lookback, searching for parallelism in discrete event simulation. In: **SUMMER COMPUTER SIMULATION CONFERENCE**. Proceedings…

CHOE, M.; TROPPER, C. (2001). Flow control and dynamic load balancing in Time Warp. **Transactions of the society for computer simulation international**, v.18: (1), p.9-23.

CHOI, E.; CHUNG, M. J. (1995). An important factor for optimistic protocol on distributed systems: granularity. In: **THE 1995 WINTER SIMULATION CONFERENCE**. Proceedings… p.642-649.

CURRY, R. et al. (2005) Sequential performance of asynchronous conservative PDES algorithms. In: **THE 18TH WORKSHOP ON PARALLEL AND DISTRIBUTED SIMULATION – PADS 2005**. Proceedings… Monterrey, CA, USA.

DEELMAN, E. (2001). Improving lookahead in parallel discrete event simulations of large-scale applications using compiler analysis. In: **THE 15TH WORKSHOP ON PARALLEL AND DISTRIBUTED SIMULATION - PADS 2001**. Proceedings… p.5-13.

EWING, G. C.; PAWLIKOWSKI, K.; MCNICKLE, D. (1998). **Akaroa v2.5 user's manual**. Department of Computer Science, University of Caterbury, New Zeland.

EWING, G. C.; PAWLIKOWSKI, K.; MCNICKLE, D. (2002). Akaroa-2: Exploiting network computing by distributing stochastic simulation. In: EUROPEAN SIMULATION MULTICONFERENCE - ESM99. Proceedings… Varsóvia. Polônia. p.175-181.

EWING, G.; MCNICKLE, D.; PAWLIKOWSKI, K. (2002) Spectral analysis for confidence interval estimation under multiple replications in parallel. In: **EUROPEAN SIMULATION SYMPOSIUM – ESS 2002**. Proceedings… ISCS Press. Dresden. Germany. p. 52-55&61.

FERSCHA, A. (1995). Parallel and distributed simulation of discrete event systems. P**arallel and distributed computing handbook**, ch.35. McGraw-Hill.

FERSCHA, A.; JOHNSON, J.; TURNER, S. J. (2001). Distributed simulation performance data mining. **Future generation computer systems,** v.18: (1), p.157-174.

FERSCHA, A.; MALONY, M. (2001). Performance data mining: automated diagnosis, adaptation and optimization. **Journal of future generation computing systems**. v.18, Issue 1, p.127-130. North Holland.

FOSTER, I. (1995). **Designing and building parallel programs**. Addison-Wesley Publishing Company.

FOSTER, I. et al. (2002) **Sourcebook of parallel computing**. Morgan Kaufmann Publishing.

FUJIMOTO, R. M. (1990). Parallel discrete event simulation. **Communications of the ACM**, v.33, n.10.

FUJIMOTO, R. M. (1995). Parallel and distributed simulation. In: **THE 1995 WINTER SIMULATION CONFERENCE**. Proceedings…

FUJIMOTO, R. M. (1999). Exploiting temporal uncertainty in parallel and distributed simulations. In: **WORKSHOP ON PARALLEL AND DISTRIBUTED SIMULATION**.

FUJIMOTO, R. M. (2000). Parallel and distributed simulation systems. **Wiley series on parallel and distributed computing**. Interscience.

FUJIMOTO, R. M. (2001). Parallel and distributed simulation systems. In: **THE 2001 WINTER SIMULATION CONFERENCE**. Proceedings… p.147-157.

FUJIMOTO, R. M. (2003). Distributed simulation systems. In: **THE 2003 WINTER SIMULATION CONFERENCE**. Proceedings… p.124-134.

GLYNN, P. W.; HEIDELBERGER, P. (1992). Analysis of initial transient deletion for parallel steady-state simulations. **SIAM J. Sci. Stat. Comput**, v.13, n.4, p.904-922.

HAREL, D. (1988). On visual formalism. **Communications of the ACM**, v.31, n.5, p.171-187.

HEIDELBERGER, P. (1988). Discrete event simulation and parallel processing statistical properties. **SIAM J. Stat. Comput.,** v.9, n.6, p.1114-1132.

HWANG, K. (1993). **Advanced computer architecture: parallelism, scalability and programmability**. MCGraw-Hill, New York. USA.

HYBINETTE, M. (2004). Just-in-time cloning. In: **THE 18TH WORKSHOP ON PARALLEL AND DISTRIBUTED SIMULATION – PADS 2004**. Proceedings… Kufstein, Austria. p.45-51.

JEFFERSON, D. R. (1985). Virtual time. **ACM transactions on programming language and systems**, v.7, n.3, p.406-425.

JEONG, H. J. et al. (2005). **Distributed steady-state simulation of telecommunication networks with self-similar teletraffic**. Simulation Modelling Practice and Theory, v.13. p.233-256.

JHA, V.; BAGRODIA, R. (1994). A unified framework for conservative and optimistic distributed simulation. In: **THE 8TH WORKSHOP ON PARALLEL AND DISTRIBUTED SIMULATIONS – PADS 1994**. Proceedings… Edinburgh. Scotland. U.K. p.12-19.

KALANTERY, N. (2004). Time Warp – connection oriented. In: **THE 18TH WORKSHOP ON PARALLEL AND DISTRIBUTED SIMULATION – PADS 2004**. Kufstein, Austria. Proceedings… p.71-77.

KUMAR, V. et al. (2003). **Introduction to parallel computing design and analysis of algorithms**. 2.ed. USA: Addison Wesley.

LEE, J. S. (2005). Design of a satellite cluster system in distributed simulation. **Simulation 2005**. Society for Computer Simulation International. San Diego, CA, USA. v.81, p.57-66.

LEE, S.; LEANEY, J.; O'NEILL, T.; HUNTER, M. (2005). Performance benchmark of a parallel and distributed network simulator. In: **THE 18TH WORKSHOP ON PARALLEL AND DISTRIBUTED SIMULATION – PADS 2005**. Proceedings… Monterrey, CA, USA.

LI, L.; TROPPER, C. (2004). Event reconstruction in time warp. In: **THE 18TH WORKSHOP ON PARALLEL AND DISTRIBUTED SIMULATION – PADS 2004**. Proceedings… Kufstein, Austria. p.37-44.

LILJENSTAM, M.; RONNGREN, R.; AYANI, R. (2001). Partitioning WCN models for parallel simulation of radio resource management. **Wireless networks**. v.7: (3), p.307-324.

MACDOUGALL, M. H. (1987). **Simulating computing systems - techniques and tools**. The MIT Press.

MEYER, R. A.; BAGRODIA, R. L. (1998). Improving lookahead in parallel wireless network simulation. In: **SIXTH IEEE INTERNATIONAL SYMPOSIUM ON MODELING, ANALYSIS, AND SIMULATION OF COMPUTER AND TELECOMMUNICATIONS SYSTEMS - MASCOTS98**. Proceedings… Montreal. Quebec. Canada. p.262.

MISRA, J. (1986). Distributed discrete-event simulation. **ACM computing surveys**.

MORSELLI Júnior, J. C. M. (2000). **Um mecanismo para troca de protocolos de sincronização de simulação distribuída em tempo de execução**. Tese (Doutorado) - Instituto de Física de São Carlos, Universidade de São Paulo, São Carlos. 2000.

MORSELLI Júnior, J. C. M. et al. (2000a). An approach for dynamic swapping of distributed simulation synchronization protocols. In: **12TH EUROPEAN SIMULATION SYMPOSIUM**. Proceedings… SCS - Simulation Computer of Society. Summer Conference Simulation Society - SCSC. Hamburg. v.1, p.103-107.

MOTA, E.; WOLISZ, A.; PAWLIKOWSKI, K. (1999). Sequential batch means techniques for mean value analysis in distributed simulation. In: **EUROPEAN SIMULATION MULTICONFERENCE – ESM 1999**. Proceedings… Varsóvia. Polônia. p.129-134.

NICOL, D. M.; FUJIMOTO, R. (1995). **Parallel Simulation Today**. Technical Report. Disponível em <http://www.cs.wm.edu/~subhas/parsim.html>. Acesso em out/2004.

PARK, A.; FUJIMOTO, R. M.; PERUMALLA, K. S. (2004). Conservative syncronization of large-scale network simulations. In: **THE 18TH WORKSHOP ON PARALLEL AND DISTRIBUTED SIMULATION – PADS 2004**. Proceedings… Kufstein, Austria. p.153-161.

PAWLIKOWSKI, K. (2003). Towards credible and fast quantitative stochastic simulation (invited paper). In: **THE INTERNATIONAL SCS CONFERENCE ON DESIGN, ANALYSIS AND SIMULATION OF DISTRIBUTED SYSTEMS - DASD03**. Proceedings… Orlando, Florida, USA.

PERUMALLA, K. (2005). μsik - A Micro-Kernel for parallel/distributed simulation systems. In: **THE 18TH WORKSHOP ON PARALLEL AND DISTRIBUTED SIMULATION – PADS 2005**. Proceedings… Monterrey, CA, USA.

QUAGLIA, F.; BERALDI, R. (2004). Space uncertain simulation events: some concepts and an application to optimistic synchronization. In: **THE 18TH WORKSHOP ON PARALLEL AND DISTRIBUTED SIMULATION – PADS 2004**. Proceedings… Kufstein, Austria. p.181-188.

QUAGLIA, F.; CORTELLESSA, V.; CICIANI, B. (1999). Tradeoff between sequential and time warp based parallel simulation. **IEEE transactions on parallel and distributed systems**, v.10, n.8, p.781-794.

QUINN, M. (1994). **Parallel computing - theory and practice**. McGraw-Hill Inc.

RAJAEI, H.; AYANI, R.; THORELLI, L. E. (1993). The local time warp approach to parallel simulation. In: **THE 7TH WORKSHOP ON PARALLEL AND DISTRIBUTED SIMULATION**. Proceedings… San Diego, CA. USA. p.119-126.

REED, D. A.; FUJIMOTO, R. M. (1987). **Multicomputer network: message-based parallel processing**. The MIT Press.

RILEY, G. F. (2004). Space-parallel network simulations using ghosts. In: **THE 18TH WORKSHOP ON PARALLEL AND DISTRIBUTED SIMULATION – PADS 2004**. Proceedings… Kufstein, Austria. p.170-177.

ROBINSON, S. (2005). Distributed simulation and simulation practice. **Simulation**, v.81, p.5-13.

RÖNNGREN, R. et al. (1996). A comparative study of state saving mechanisms for time warp synchronized parallel discrete event simulation. In: **THE 29TH ANNUAL SIMULATION SYMPOSIUM - ASS96**. Proceedings… New Orleans. USA.

RÖNNGREN, R. et al. (1996). Transparent incremental state saving in time warp parallel discrete event simulation. In: **PARALLEL AND DISTRIBUTED SIMULATION – PADS 1996**. Proceedings... Philadelphia. USA. p.70-77.

SANTORO, A.; QUAGLIA, F. (2005). Transparent state management for optimistic synchronization in the high level architecture. In: **THE 18TH WORKSHOP ON PARALLEL AND DISTRIBUTED SIMULATION – PADS 2005**. Proceedings… Monterrey, CA, USA.

SILVA, M. P. (2004). **Método para escolha de abordagem para simulação distribuída utilizando inteligência artificial**. Monografia apresentada para o exame de qualificação de Mestrado - Instituto de Ciências Matemáticas e de Computação, Universidade de São Paulo, São Carlos.

SOARES, L. F. G. (1992). **Modelagem e Simulação Discreta de Sistemas**. Editora Campus Ltda.

SOLCONY, V.; SAFARIK, J. (2002) The lookahead in a user-transparent conservative parallel simulator. In: **Workshop on Parallel and Distributed Simulation** – PADS 2002. Proceedings… p.11-16.

SPOLON, R. (2001). **Um método para avaliação de desempenho de protocolos de sincronização otimistas para simulação distribuída**. Tese (Doutorado) **-** Instituto de Física de São Carlos, Universidade de São Paulo, São Carlos. 2001.

SPOLON, R.; SANTANA, M. J.; SANTANA, R. C. (1999). Distributed simulation, time warp and its variants: taxonomy and performance evaluation issues. In: **THE 13TH EUROPEAN SIMULATION MULTICONFERENCE**. Proceedings... The society for computer simulation international. Polônia. p.220-227.

STEINMAN, J. (2005). The WarpIV simulation kernel. In: **THE 18TH WORKSHOP ON PARALLEL AND DISTRIBUTED SIMULATION – PADS 2005**. Proceedings… Monterrey, CA, USA.

SUPPI, R.; CORES, F.; LUQUE, E. (2000). Improving optimistic PDES in PVM environments. **Lecture notes in computer science**, v. 1908, p.304-312.

TANG, Y. et al. (2005). Optimistic parallel discrete event simulations of physical systems using reverse computation. In: **THE 18TH WORKSHOP ON PARALLEL AND DISTRIBUTED SIMULATION – PADS 2005**. Proceedings… Monterrey, CA, USA.

TATSUMI, E. S. (2003). **Avaliação e aprimoramento de uma implementação para simulação distribuída conservativa visando utilização em um ambiente automático**. Dissertação (Mestrado) - Instituto de Ciências Matemáticas e de Computação, Universidade de São Paulo, São Carlos. 2003.

TAYLOR, S. J. E. et al. (2005). A comparison of CMB- and HLA-based approaches to type I interoperability reference model problems for COTS-based distributed simulation. **Simulation**, v.81, n.1, 2005. Society for Computer Simulation International. San Diego, CA, USA. p. 33-43.

TEO, P.; TURNER, S. J.; JUHASZ, Z. (2005) Optimistic protocol analysis in a performance analyser and prediction tool. In: **THE 18TH WORKSHOP ON PARALLEL AND DISTRIBUTED SIMULATION – PADS 2005**. Proceedings… Monterrey, CA, USA.

TEO, Y. M.; NG, Y. K.; ONGGO, B. S. S. (2002). Conservative simulation using distributed-shared memory. In: **THE 16TH ACM/IEEE/SCS WORKSHOP ON PARALLEL AND DISTRIBUTED SIMULATION.** Proceedings... IEEE Computer Society Press. Washington. USA. p.3-10.

TEO, Y. M.; ONGGO, B. S. S. (2004). Formalization and strictness of simulation event orderings. In: **THE IEEE/ACM WORKSHOP ON PARALLEL AND DISTRIBUTED SIMULATION**. Proceedings… IEEE Computer Society Press. Kufstein. Austria. p.89-96.

TEO, Y. M.; TAY, S. (1999). Performance evaluation of a parallel simulation environment. In: **THE 32ND ANNUAL SIMULATION SYMPOSIUM**. Proceedings… IEEE Computer Society Press. San Diego. USA. p.86-93.

TEO, Y. M.; WANG, H.; TAY, S. C. (1999). A Framework for analyzing parallel simulation performance. In: **THE 32ND ANNUAL SIMULATION SYMPOSIUM**. Proceedings… IEEE Computer Society Press. San Diego. USA. p.102-109.

ULSON, R. S. (1999). **Simulação distribuída em plataformas de portabilidade: viabilidade de uso e comportamento do protocolo CMB**. Tese (Doutorado) - Instituto de Física de São Carlos, Universidade de São Paulo, São Carlos. 1999.

ULSON, R. S. et al. (1999). Conservative distributed simulation on portability platforms: the CMB protocol behavior. In: T**HE IASTED INTERNATIONAL CONFERENCE APPLIED MODELING AND SIMULATION,** Sep. 1-3. Proceedings… Cairns. Australia.

VERBRAECK, A. (2004). Component-based distributed simulation. The way forward?. In: **THE 18TH WORKSHOP ON PARALLEL AND DISTRIBUTED SIMULATION – PADS 2004**. Proceedings… Kufstein, Austria. p.141-148.

WILMARTH, T. et al. (2005). Performance prediction using simulation of large-scale interconnection networks in POSE. In: **THE 18TH WORKSHOP ON PARALLEL AND DISTRIBUTED SIMULATION – PADS 2005**. Proceedings… Monterrey, CA, USA.

XU, J.; CHUNG, M. J. (2004). Predicting the performance of synchronous discrete event simulation. **IEEE transactions on parallel and distributed systems**, v.15, n.12, p.1130-1137.

XU, Q.; TROPPER, C. (2005). XTW, a parallel and distributed logic simulator. In: **THE 18TH WORKSHOP ON PARALLEL AND DISTRIBUTED SIMULATION – PADS 2005**. Proceedings… Monterrey, CA, USA.

ZENG, Y.; CAI, W.; TURNER, S. J. (2004). Batch based cancellation: a rollback optimal cancellation scheme in time warp simulations. In: **THE 18TH WORKSHOP**

ON PARALLEL AND DISTRIBUTED SIMULATION – PADS 2004. Proceedings… Kufstein, Austria. p.78-86.

APÊNDICE A –
Resultados das simulações

Os resultados das simulações dos modelos considerados nos capítulos 4 e 5 estão apresentados neste apêndice.

Modelo 1 (servidor central) – configuração 1

GRAN	Tempo de execução (Sequencial)	Tempo de execução (Paralelo)	Speedup
0	6,59	176,39	0,037
25	174,85	279,70	0,625
50	1060,15	1022,36	1,037
75	3252,36	2873,73	1,132
100	7320,48	6303,40	1,161

Resultado para o processo lógico 0:

GRAN	Executando/ bloqueado (pl0)	% tempo bloqueado (pl0)	% tempo bloqueado nec	% tempo bloqueado desn	% msg compl bloq nec (pl0)	%msg compl bloq desn (pl0)	%msg nulas bloq desn (pl0)
0	0,29	82,82	60,52	17,72	73,79	16,81	80,53
25	0,76	56,85	31,32	23,66	75,66	6,77	77,05
50	1,62	45,28	17,06	27,63	75,67	6,77	77,03
75	2,06	42,55	13,89	28,57	75,66	6,78	77,03
100	2,23	41,86	12,97	28,89	75,67	6,77	77,03

Resultados para o processo lógico 1:

GRAN	Executando/ bloqueado (pl1)	% tempo bloqueado	% tempo bloqueado nec	% tempo bloqueado desn	% msg compl bloq nec (pl1)	%msg compl bloq desn (pl1)	%msg nulas bloq desn (pl1)
0	0,14	87,96	34,38	52,10	51,36	0,00	98,50
25	0,51	66,37	17,67	47,80	51,03	0,00	95,28
50	0,78	56,29	8,86	47,20	51,05	0,00	95,36
75	0,85	54,04	6,87	47,13	51,05	0,00	95,36
100	0,87	53,34	6,27	47,06	51,05	0,00	95,36

Modelo 1 (servidor central) – configuração 2

GRAN	Tempo de execução (Sequencial)	Tempo de execução (Paralelo)	Speedup
0	6,58	262,06	0,025
25	174,81	367,01	0,476
50	1060,02	1279,51	0,828
75	3251,92	3583,20	0,908
100	7319,93	7831,48	0,935

Resultado para o processo lógico 0:

GRAN	pl0 (executando /bloqueado)	% tempo bloqueado	% tempo bloqueado nec	% tempo bloqueado desn	% msg compl bloq nec (pl0)	%msg compl bloq desn (pl0)	%msg nulas bloq desn (pl0)
0	0,114	89,74	64,94	22,45	77,62	5,10	56,59
25	0,474	67,83	35,19	31,22	73,01	2,45	50,41
50	0,771	56,45	18,21	37,91	72,66	1,36	47,12
75	0,846	54,18	14,45	39,68	72,65	1,36	47,13
100	0,877	53,28	13,38	39,89	72,65	1,36	47,29

Resultado para o processo lógico 1:

GRAN	pl1 (executando /bloqueado)	% tempo bloqueado	% tempo bloqueado nec	% tempo bloqueado desn	% msg compl bloq nec (pl1)	%msg compl bloq desn (pl1)	%msg nulas bloq desn (pl1)
0	0,038	96,38	24,37	69,13	43,54	0,00	59,47
25	0,143	87,46	18,68	66,37	43,62	0,00	60,87
50	0,209	82,70	15,20	65,59	43,45	0,20	61,12
75	0,223	81,74	14,37	65,57	43,45	0,21	61,12
100	0,228	81,44	14,23	65,41	43,45	0,21	61,12

Resultado para o processo lógico 2:

GRAN	pl2 (executando /bloqueado)	% tempo bloqueado	% tempo bloqueado nec	% tempo bloqueado desn	% msg compl bloq nec (pl2)	%msg compl bloq desn (pl2)	%msg nulas bloq desn (pl2)
0	0,112	89,95	20,79	63,12	43,00	0,52	62,15
25	0,424	70,23	12,09	54,05	42,92	0,83	63,04
50	0,790	55,86	7,20	46,83	42,94	0,97	63,03
75	0,905	52,48	6,11	45,18	42,94	0,97	63,02
100	0,934	51,70	5,80	44,89	42,94	0,97	63,02

Modelo 3 – sistema computacional simplificado

GRAN	Tempo de execução (Sequencial)	Tempo de execução (Paralelo)	Speedup
0	19,83	35,92	0,552
25	43,44	53,06	0,819
50	166,35	161,29	1,031
75	472,19	425,48	1,110
100	1036,41	917,64	1,129

Resultado para o processo lógico 0:

GRAN	pl0 (executando / bloqueado)	% tempo bloqueado	% tempo bloqueado nec	% tempo bloqueado desn	% msg compl bloq nec (pl0)	%msg compl bloq desn (pl0)	%msg nulas bloq desn (pl0)
0	0,654	60,44	18,22	38,88	71,56	0,00	97,74
25	1,444	40,91	10,32	28,41	71,19	0,00	98,46
50	3,399	22,73	3,54	18,47	71,46	0,00	98,54
75	4,693	17,57	1,51	15,79	71,46	0,00	98,54
100	5,346	15,76	0,83	14,80	71,52	0,00	98,56

Resultado para o processo lógico 1:

GRAN	pl1 (executando / bloqueado)	% tempo bloqueado	% tempo bloqueado nec	% tempo bloqueado desn	% msg compl bloq nec (pl1)	%msg compl bloq desn (pl1)	%msg nulas bloq desn (pl1)
0	0,045	95,72	2,51	53,07	9,90	0,00	43,45
25	0,151	86,86	1,71	53,52	9,80	0,00	43,73
50	0,277	78,33	1,12	54,26	9,84	0,00	44,17
75	0,323	75,58	0,95	54,38	9,84	0,00	44,17
100	0,337	74,79	0,89	54,51	9,78	0,00	44,23

Modelo 4 – sistema computacional com CPU e dois discos

GRAN	Tempo de execução (Sequencial)	Tempo de execução (Paralelo)	Speedup
0	6,77	121,10	0,056
25	138,42	202,27	0,684
50	839,06	761,96	1,101
75	2572,77	2153,16	1,195
100	5805,85	4727,13	1,228

Resultado para o processo lógico 0:

GRAN	pl0 (executando / bloqueado)	% tempo bloqueado	% tempo bloqueado nec	% tempo bloqueado desn	% msg compl bloq nec (pl0)	%msg compl bloq desn (pl0)	%msg nulas bloq desn (pl0)
0	0,201	83,27	64,42	16,22	84,69	0,002	80,36
25	0,927	51,90	27,25	23,22	84,53	0,001	83,64
50	1,673	37,41	10,18	26,83	84,53	0,001	83,65
75	1,926	34,18	6,39	27,68	84,52	0,001	83,63
100	2,012	33,20	5,27	27,91	84,52	0,001	83,66

Resultado para o processo lógico 1:

GRAN	pl1 (executando / bloqueado)	% tempo bloqueado	% tempo bloqueado nec	% tempo bloqueado desn	% msg compl bloq nec (pl1)	%msg compl bloq desn (pl1)	%msg nulas bloq desn (pl1)
0	0,174	85,18	33,64	50,06	52,69	0,001	99,35
25	0,493	66,98	17,76	48,31	51,87	0,001	95,97
50	0,648	60,68	10,87	49,56	51,87	0,001	95,97
75	0,686	59,30	9,36	49,90	51,87	0,001	95,98
100	0,697	58,92	8,92	50,00	51,87	0,001	95,97

Modelo de redes de fila 1 (modelo fechado)

GRAN	Tempo de execução (Sequencial)	Tempo de execução (Paralelo)	Speedup
0	1,85	49,35	0,038
25	85,51	105,77	0,808
50	524,83	453,57	1,157
75	1612,16	1315,69	1,225
100	3628,10	2909,86	1,247

Resultado para o processo lógico 0:

GRAN	pl0 (executando / bloqueado)	% tempo bloqueado	% tempo bloqueado nec	% tempo bloqueado desn	% msg compl bloq nec (pl0)	%msg compl bloq desn (pl0)	%msg nulas bloq desn (pl0)
0	0,08	92,50	4,00	48,63	9,26	1,14	55,23
25	0,43	70,02	1,89	43,27	9,18	1,18	55,50
50	0,64	61,10	1,06	41,55	9,18	1,18	55,51
75	0,68	59,37	0,89	41,25	9,18	1,18	55,51
100	0,70	58,85	0,84	41,14	9,18	1,18	55,51

Resultado para o processo lógico 1:

GRAN	pl1 (executando / bloqueado)	% tempo bloqueado	% tempo bloqueado nec	% tempo bloqueado desn	% msg compl bloq nec (pl1)	%msg compl bloq desn (pl1)	%msg nulas bloq desn (pl1)
0	0,10	91,30	25,79	61,77	64,83	0	98,71
25	0,89	52,95	16,72	34,51	65,02	0	99,41
50	1,65	37,73	13,91	23,41	65,03	0	99,41
75	1,88	34,66	13,34	21,19	65,03	0	99,41
100	1,96	33,78	13,18	20,56	65,03	0	99,41

Modelo de redes de fila 2 (modelo aberto) – configuração 1

GRAN	Tempo de execução (Sequencial)	Tempo de execução (Paralelo)	Speedup
0	3,22	10,51	0,306
25	22,61	27,95	0,809
50	123,68	123,74	1,000
75	373,85	360,81	1,036
100	837,24	800,73	1,046

Resultado para o processo lógico 0:

GRAN	pl0 (executando / bloqueado)	% tempo bloqueado	% tempo bloqueado nec	% tempo bloqueado desn	% msg compl bloq nec (pl0)	%msg compl bloq desn (pl0)	%msg nulas bloq desn (pl0)
0	0,32	75,49	15,52	50,30	85,76	0,00	93,00
25	1,52	39,72	4,92	31,80	85,05	0,00	95,58
50	2,62	27,59	1,22	25,68	85,05	0,00	95,58
75	2,96	25,27	0,52	24,51	85,05	0,00	95,58
100	3,07	24,58	0,32	24,15	85,05	0,00	95,58

Resultado para o processo lógico 1:

GRAN	pl1 (executando / bloqueado)	% tempo bloqueado	% tempo bloqueado nec	% tempo bloqueado desn	% msg compl bloq nec (pl1)	%msg compl bloq desn (pl1)	%msg nulas bloq desn (pl1)
0	0,12	89,03	15,48	71,74	41,64	0,00	99,96
25	0,25	80,17	9,47	70,02	42,04	0,00	99,87
50	0,30	76,84	7,03	69,65	42,04	0,00	99,87
75	0,31	76,18	6,57	69,56	42,04	0,00	99,87
100	0,32	76,01	6,44	69,55	42,04	0,00	99,88

Modelo de redes de fila 2 (modelo aberto) – configuração 2

GRAN	Tempo de execução (Sequencial)	Tempo de execução (Paralelo)	Speedup
0	3,22	8,86	0,363
25	22,61	27,48	0,823
50	123,68	127,91	0,967
75	373,85	376,27	0,994
100	837,24	838,19	0,999

Resultado para o processo lógico 0:

GRAN	pl0 (executando / bloqueado)	% tempo bloqueado	% tempo bloqueado nec	% tempo bloqueado desn	% msg compl bloq nec (pl0)	%msg compl bloq desn (pl0)	%msg nulas bloq desn (pl0)
0	0,59	62,86	17,58	32,92	86,45	0,00	97,14
25	3,56	21,93	5,31	12,76	87,58	0,00	98,55
50	11,25	8,16	1,35	5,96	87,58	0,00	98,60
75	16,67	5,66	0,60	4,77	87,58	0,00	98,60
100	19,24	4,94	0,39	4,41	87,58	0,00	98,60

Resultado para o processo lógico 1:

GRAN	pl1 (executando / bloqueado)	% tempo bloqueado	% tempo bloqueado nec	% tempo bloqueado desn	% msg compl bloq nec (pl1)	%msg compl bloq desn (pl1)	%msg nulas bloq desn (pl1)
0	0,04	96,22	4,12	82,79	15,75	0,00	57,94
25	0,06	94,65	3,25	84,48	16,09	0,00	58,60
50	0,06	94,24	2,88	84,90	16,14	0,00	58,73
75	0,06	94,15	2,81	84,95	16,14	0,00	58,73
100	0,06	94,13	2,79	84,98	16,14	0,00	58,73

Modelo de redes de fila 2 (modelo aberto) – configuração 3

GRAN	Tempo de execução (Sequencial)	Tempo de execução (Paralelo)	Speedup
0	3,22	9,62	0,335
25	22,61	26,47	0,854
50	123,68	118,85	1,041
75	373,85	347,37	1,076
100	837,24	770,26	1,087

Resultado para o processo lógico 0:

GRAN	pl0 (executando / bloqueado)	% tempo bloqueado	% tempo bloqueado nec	% tempo bloqueado desn	% msg compl bloq nec (pl0)	%msg compl bloq desn (pl0)	%msg nulas bloq desn (pl0)
0	0,43	170,78	420,22	602,03	92,34	0,00	91,56
25	2,19	50,96	230,43	734,03	92,23	0,00	95,85
50	4,33	11,32	169,21	901,41	92,23	0,00	95,85
75	5,09	3,90	157,96	962,18	92,23	0,00	95,85
100	5,35	1,78	154,68	982,84	92,23	0,00	95,85

Resultado para o processo lógico 1:

GRAN	pl1 (executando / bloqueado)	% tempo bloqueado	% tempo bloqueado nec	% tempo bloqueado desn	% msg compl bloq nec (pl1)	%msg compl bloq desn (pl1)	%msg nulas bloq desn (pl1)
0	0,16	86,40	17,64	67,16	43,78	0,00	99,92
25	0,21	82,44	12,91	68,94	44,07	0,00	99,92
50	0,23	81,35	11,02	70,19	44,07	0,00	99,92
75	0,23	81,08	10,66	70,37	44,07	0,00	99,92
100	0,23	80,98	10,56	70,40	44,07	0,00	99,92