PROTOCOLO CONSERVATIVO

CMB PARA SIMULAÇÃO DISTRIBUÍDA

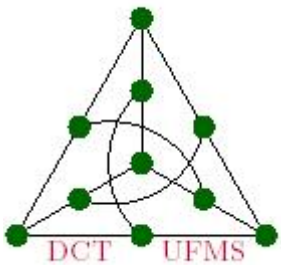
Marta Oliveira da Silva Balieiro

Dissertação de Mestrado

Orientação: Prof. Dra. Renata Spolon Lobato

Área de Concentração: Simulação Distribuída

Dissertação apresentada ao Departamento de Computação e Estatística da Universidade Federal de Mato Grosso do Sul, como parte dos requisitos para obtenção do título de **Mestre em Ciência da Computação.**



Departamento de Computação e Estatística
Centro de Ciências Exatas e Tecnologia
Universidade Federal de Mato Grosso do Sul
Fevereiro de 2005

Aos meus pais Airton e Orelina
Aos meus filhos Airton e Aline
Ao meu esposo Marcos

Agradecimentos

A Deus, por estar sempre ao meu lado e me guiar em todos os momentos da minha vida.

A minha orientadora, Profª. Drª Renata Spolon Lobato, pela dedicação, pela paciência e excelente orientação, que tornou este trabalho possível. Obrigada.

Aos meus pais e esposo que mesmo nas horas mais difíceis me passaram confiança e nunca deixaram de acreditar em mim.

Ao Professor Marcelo Henriques Carvalho, que tanto se dedicou para tornar realidade o Mestrado em Ciência da Computação da UFMS.

Aos amigos Bianca Dantas e Rodrigo Sacchi, por todas as vezes que estudamos juntos, por todos os esclarecimentos fornecidos. Sempre foram as primeiras pessoas a que eu procurava, nos momentos mais difíceis, e sempre me ajudaram, quando a conclusão deste trabalho parecia impossível. Muito Obrigada.

Agradeço a Gabriela por me ajudar com meus filhos, para que eu pudesse me ausentar e fazer a coleta de estatísticas, as quais foram realizadas em horários restritos.

E finalmente a todos que contribuíram para este trabalho direta ou inderetamente.

Conteúdo

Lista de Tabelas

Lista de Figuras

Lista de Siglas e Abreviaturas

Apostle	*A Parallel Object – Oriented Simulation Language*
ACM	*Association for Computing Machinery*
ASDA	Ambiente de Simulação Distribuída Capital
ASIA	Ambiente de Simulação Automático
bps	Bytes por Segundo
CMB	*Chandy, Misra e Bryant*
DCT	Departamento de Computação e Estatística
FEL	*Future Event List*
FCFS	*First-come First-Servedt*
FIFO	*First Come First-Served*
GVT	*Global Virtual Time*
ICMC	Instituto de Ciências Matemáticas e de Computação
IS	*Infinite Server*
kbps	*K bytes por segundo*
LCFS	*Last-come first-served*
LEF	Lista de Eventos Futuros
LIFO	*Last In First Out*
LVT	*Local Virtual Time*
LVTH	*Local Virtual Time Horizon*
MIM	*Multiple Instruction, Multiple Data*
MIPS	*Multiple Instruction Segund*
MPI	*Message Passing Interace*
Moose	*Maise-based Object- Oriented Simulation Environment*
NF	Número de Enfileiramentos
NL	Número de Liberações
NMC	Número de Mensagens Completas
NMN	Número de Mensagens Nulas
NMD	Número de Mensagens Nulas na Demanda
NP	Número de Preempções
Parsec	Paralell simulation environment for complex systems
PMO	Período Médio Ocupado
PRTY	*Nonpreemptive Priority*
PRTYPR	*Preemptive –Restime Priority*
PS	*Processor Sharing*
PL	Processo Lógico
RR	*Round-Robin*
SMPL	Simulation Language Program
SPMD	*Single Program Multiple Data*
SpN	*Speedup na* CMB-Simulation *Nula*
SpD	*Speedup na* CMB-Simulation *sob Demanda*
tc	Tempo entre Chegadas de Clientes
TMF	Tamanho Médio da Fila
ts	Tempo de Serviço do Servidor
UCP	Unidade Central de Processamento
UFMS	Universidade Federal de Mato Grosso do Sul
USP	Universidade de São Paulo

Resumo

Este trabalho apresenta a ferramenta CMB-*Simulation*, uma extensão da linguagem C para sincronização de processos na simulação distribuída conservativa. CMB-*Simulation* permite o uso da abordagem de mensagens nulas ou da abordagem de mensagens nulas sob demanda na simulação de redes de filas. A ferramenta foi desenvolvida com base na estrutura de SMPL, uma extensão da linguagem C que implementa simulação seqüencial orientada a eventos, e utiliza o MPI para troca de mensagens entre os processos que compõem a simulação distribuída (no sistema operacional Linux). Os testes efetuados com diversos modelos de filas mostram a utilização da ferramenta na avaliação da simulação distribuída conservativa.

Abstract

This work presents the CMB-*Simulation* tool, an extension for the C programming language, that can be used for synchronization of processes in the conservative distributed simulation. CMB-*Simulation* allows the use of null-messages or on-demand null-messages approach for queue nets simulation. The tool was developed based on the SMPL structure and uses the LAM-MPI on Linux operating system for message exchange between the processes that implements the distributed simulation. The tool was tested with several queue nets for conservative distributed simulation evaluation.

Capítulo 1

Introdução

1.1 Motivação

O desenvolvimento da área de simulação vem acentuando-se gradativamente, principalmente devido ao aumento da complexidade dos problemas a serem resolvidos e à constante necessidade por ferramentas de avaliação de desempenho. O objetivo da avaliação de desempenho é determinar o grau de qualidade de um sistema, podendo ser utilizada na análise de sistemas existentes ou não [Fuj90a]. Os custos de projetos e desenvolvimento de sistemas podem ser reduzidos através do uso de uma técnica de análise de desempenho que antecipe algumas alterações para que o sistema se comporte de acordo com alguma métrica de avaliação.

A avaliação de desempenho de sistemas computacionais pode ser efetuada através das técnicas de aferição e das técnicas de solução de modelos. As técnicas de aferição (prototipação, *benchmarks* e coleta de dados) são utilizadas para a análise através do próprio sistema ou de um protótipo do mesmo. Nas técnicas de modelagem é necessário o desenvolvimento de um modelo do sistema a ser avaliado. Esse modelo pode ser resolvido através de solução analítica ou por simulação. Na solução analítica a modelagem é feita em termos de equações, e o acréscimo de novas características no modelo pode aumentar consideravelmente a complexidade da solução. Na simulação, o modelo é transformado em um programa que representa o sistema real [San94].

A simulação vem sendo utilizada nas mais diversas áreas, como por exemplo na simulação de sistemas urbanos, sistemas de controle aeroespaciais, sistemas de treinamento

militar, sistemas ecológicos, de sistemas de redes de comunicação de computadores, etc. [Fuj02, Hus04, Mac01, Zho04]. Devido ao avanço na complexidade dos sistemas, o tempo gasto com o processamento na simulação seqüencial pode ser muitas vezes inviável, favorecendo mais pesquisas na área de simulação distribuída. Na simulação seqüencial, a sincronização entre os eventos é garantida com a utilização da *LEF* (Lista de Eventos Futuros). Na simulação distribuída deve ser utilizado algum mecanismo de sincronização para garantir a execução dos eventos na ordem correta. Os métodos utilizados para realizar a sincronização de eventos podem ser otimistas ou conservativos.

Essa classificação baseia-se na forma adotada para evitar ou corrigir os erros obtidos por uma situação de inconsistência do sistema. Os métodos conservativos evitam situações de inconsistência através de bloqueio de processos. Alguns autores propõem abordagens mistas, nas quais há submodelos que executam de modo conservativo ou otimista [Bag94b]. Nos métodos otimistas a inconsistência é tratada através da execução de *rollbacks,* ou seja, a computação executada de forma errônea é desfeita e a simulação volta para um estado seguro.

1.2 Objetivos

O objetivo desse trabalho é o desenvolvimento de uma ferramenta para o protocolo conservativo CMB (Chandy, Misra e Bryant) original e a variante com troca de mensagens nulas sob demanda, para a simulação de redes de filas. Essa ferramenta, denominada CMB-*Simulation*, foi implementada na linguagem C, no sistema *Linux* em máquinas *IBM-PC*. É composta por um conjunto de processos, cada um representando um processo lógico da simulação distribuída conservativa. A troca de mensagens entre os processos é efetuada através de primitivas bloqueantes do MPI (*Message Passing Interface*), pela razão do mesmo procurar padronizar as plataformas de portabilidadade.

Essa ferramenta visa condições para realização de estudos e análises comparativas entre os protocolos conservativos e otimistas, enfatizando o CMB e *Time Warp*, assim como identificação de classes de aplicação que melhor se adaptam em cada caso. A CMB-*Simulation* vem de encontro com a necessidade de um ambiente propício para a implementação do mecanismo de troca dinâmica entre os protocolos CMB e *Time Warp*, proposto por Morselli [Mor00].

1.3 Organização do Texto

No Capítulo 2, a simulação seqüencial é apresentada como uma técnica para avaliação de sistemas, abordando tipos de simulação, vantagens e desvantagens, tipos de modelos existentes, fases de elaboração de um modelo. Além disso, nesse capítulo descreve-se o comportamento e rotinas do SMPL, uma extensão funcional para simulação discreta. Também são apresentadas algumas linguagens e pacotes utilizados na simulação seqüencial.

No Capítulo 3, são discutidos conceitos básicos da simulação distribuída, incluindo o protocolo CMB. São descritas as fases de desenvolvimento da simulação distribuída, assim como as variantes do protocolo conservativo CMB. Também são descritas algumas linguagens para simulação distribuída.

O Capítulo 4 descreve o funcionamento da ferramenta CMB-*Simulation*. A funcionalidade de cada módulo dessa ferramenta é devidamente descrita. A implementação é validada através da comparação dos resultados com a solução seqüencial, implementada em SMPL [Mac87].

O Capítulo 5 apresenta análise de modelos de filas, com a ferramenta CMB-*Simulation* com troca de mensagens nulas e sob demanda. São apresentados cinco modelos, e para cada modelo é feita uma análise referente ao *speedup*, ao número de mensagens nulas, à capacidade de *lookahead* e ao tempo de execução da simulação com relação à simulação seqüencial. Também são estudadas situações com diferentes granulosidades da execução de eventos. Com o intuito de alcançar o valor de *speedup*, acrescentou-se ao modelo granulosidades fina, média e grossa.

O Capítulo 6 aborda as conclusões desse trabalho, enfatizando as contribuições e as propostas para trabalhos futuros.

Capítulo 2

Simulação Seqüencial

2.1 Considerações Iniciais

Simular é o processo de desenvolver um modelo matemático ou lógico de um sistema real com o intuito de realizar experimentos que forneçam uma prévia do comportamento desse sistema [Men99]. A partir de um programa computacional, pode-se exercitar o modelo de alguma maneira. Essa tarefa pode ser realizada com a utilização de dados obtidos do sistema ou através de valores aleatórios gerados a partir de distribuições de probabilidade, com a finalidade de representar as variáveis do sistema [Soa92]. A simulação é uma técnica que pode ser utilizada na experimentação e avaliação de sistemas, de modo a prever as conseqüências causadas por mudanças de políticas, de condições ou métodos, sem a necessidade de implementá-las no sistema real, evitando grandes gastos e riscos de obter resultados inesperados. Além disso, ela pode ser utilizada como um meio de projeção do futuro, isto é, como ferramenta de previsão e planejamento quantitativo [Ban98].

A simulação vem sendo utilizada para analisar sistemas de controle de tráfego aéreo, de redes de comunicação de computadores, bem como sistemas de produção, de serviços, financeiros, governamentais, ambientais e sociais [Fuj02, Hus04, Mac01, Zho04]. Sua aceitação deve-se à sua versatilidade, flexibilidade e baixo custo, dado que fornece a possibilidade de obtenção de respostas rápidas diante de modificações no modelo [Bru03]. Um modelo é a descrição de um sistema, o qual pode ser representado visualmente através de Redes de Filas, Redes de Petri ou *Statecharts*, por exemplo. Um modelo pode ser resolvido

analiticamente ou por simulação, alternativa na qual se utiliza um programa que representa o sistema real [Spo94].

Um sistema é visto dinamicamente como uma coleção de processos interativos, cada um composto por diversas atividades, controladas e coordenadas pela ocorrência de eventos. Os modelos de um sistema podem ser classificados como modelos de mudança discreta e modelos de mudança contínua. Mudanças são alterações nas variáveis de estado do sistema, as quais são as bases para a classificação dos modelos. As variáveis incluídas na simulação são funções do tempo. No modelo de mudança discreta essas variáveis mudam em pontos específicos do tempo simulado, pontos esses conhecidos como tempo de evento. Em contraste, no modelo de mudança contínua as variáveis podem variar continuamente ao longo do tempo simulado.

Para descrever o comportamento dinâmico de sistemas discretos utilizam-se três entidades: atividade, processo e evento. A relação entre essas entidades é ilustrada na Figura 2.1 [Soa92]. Uma entidade é um objeto de interesse no sistema. Os atributos são propriedades das entidades existentes, ou seja, cada entidade possui seu conjunto de atributos.

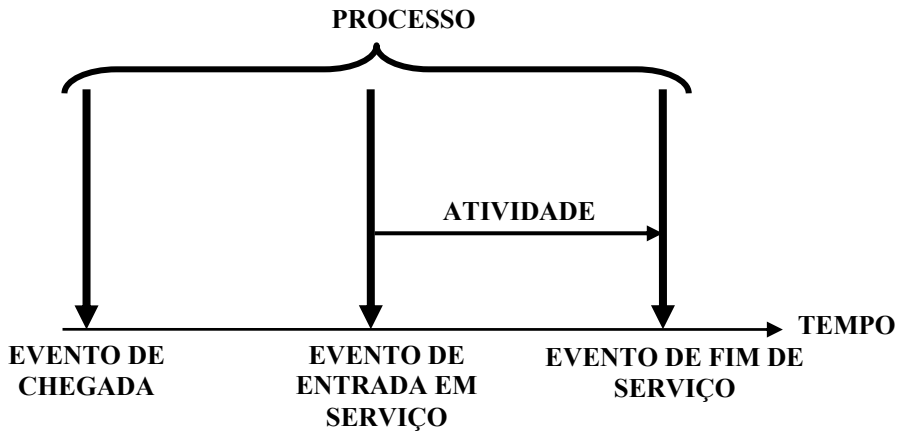


Figura 2.1: Atividades, processos e eventos.

A atividade é a menor unidade de trabalho, tendo associada a ela um tempo de execução. Um conjunto de atividades logicamente relacionadas constitui um processo. A definição de processo de uma simulação depende do nível de abstração adotado, um processo pode ser um programa inteiro ou apenas uma função desse programa. Um evento causa uma mudança de estado de alguma entidade do sistema. Como exemplo de eventos pode-se citar o início ou término de uma atividade [Ban01].

A simulação discreta utiliza três estruturas de dados [Mis86, Nic96]:

- as variáveis que descrevem o estado do sistema;
- uma lista de eventos, denominada lista de eventos futuros (*LEF*), a qual contém todos os eventos pendentes para execução;
- um relógio global que controla o progresso da simulação.

Cada evento a ser executado possui uma marca de tempo, que determina quando uma mudança de estado deve ocorrer. A *LEF* é ordenada ascendentemente pelo tempo de ocorrência dos eventos. O programa de simulação repetidamente remove o evento com a menor marca de tempo do início da *LEF* para executá-lo. A ativação desses eventos é controlada por um relógio global, que determina o tempo de simulação. Quando um programa completa todo o processamento em um determinado instante de tempo, o evento do início da lista (com a menor marca de tempo associada) é retirado, o relógio é avançado para o tempo de ocorrência do mesmo e o programa de simulação inicia a execução do evento [Uls99]. Esse mecanismo garante que os eventos no sistema físico sejam simulados em ordem cronológica no tempo de simulação.

2.2 Tipos de Orientação de uma Simulação

Pode-se organizar um modelo de simulação discreta levando-se em consideração três entidades. Dependendo dessa organização, a simulação discreta pode ser orientada a evento, orientada a processo ou orientada a atividade [Soa92].

Na simulação orientada a evento o sistema é modelado de acordo com a ocorrência de mudanças no sistema em pontos discretos do tempo. O modelador deve determinar os eventos que ocasionam mudanças no estado do sistema e desenvolver a lógica associada aos mesmos. A simulação do sistema consiste na execução da lógica associada a cada evento, em uma seqüência ordenada no tempo [Spo94].

Para ilustrar o funcionamento de uma simulação orientada a eventos considera-se, como exemplo, a estrutura de um sistema de casa lotérica. A decisão sobre o numero de caixas a serem utilizados pode ser resolvida, por exemplo, através de um modelo de simulação. Ao chegar à lotérica, se os funcionários em serviço estiverem ocupados, o cliente

entra espera pelo atendimento em uma fila, caso contrário ele é atendido imediatamente e, após o atendimento, deixa a lotérica. As mudanças de estado nesse sistema ocorrem devido a algum dos seguintes eventos:

- chegada de um cliente para efetuar apostas e/ou operações bancárias;
- final do atendimento a um cliente e sua saída do sistema.

A lógica associada ao evento chegada é descrita na Figura 2.2. A Figura 2.3 descreve a lógica associada ao evento final de atendimento.

EVENTO CHEGADA DE CLIENTE

```
Escalone a próxima chegada
    Se os funcionários estiverem ocupados então
    Aumente o numero de clientes na fila
    Senão
      Torne o estado de um funcionário ocupado
      Escalone o fim de atendimento para tempo corrente + tempo de atendimento
```

Figura 2.2: Exemplo do evento chegada de um cliente.

EVENTO TÉRMINO DE ATENDIMENTO

```
Se existe algum cliente na fila então
  Retire um cliente da fila de atendimento
  Escalone o fim de atendimento para tempo corrente + tempo de atendimento
Senão
  Torne o estado do funcionário desocupado
```

Figura 2.3: Exemplo do evento término do atendimento.

Na simulação orientada a atividade, deve-se definir quais são as atividades relacionadas a cada entidade e quais são as condições de início e fim de cada atividade. Os eventos que iniciam ou terminam uma atividade não são escalonados pelo modelador, mas são iniciados a partir das condições especificadas para a atividade. Conforme o tempo de simulação avança, as condições para início e fim de uma atividade são examinadas.

Devido à necessidade de escalonar as atividades a cada avanço do tempo, o método é relativamente ineficiente quando comparado com a simulação orientada a evento. Por isso, a maioria das linguagens de simulação não utiliza essa abordagem [Soa92, Spo94].

No exemplo onde os clientes são atendidos pelos funcionários da casa lotérica, como mostra a Figura 2.4, podem ser identificadas as seguintes atividades:

- chegada de um cliente;
- término de atendimento.

```
INÍCIO
Verifica o Tempo de simulação
Enquanto não fim da simulação faça
        Verificar o conjunto de atividades
        Se chegada de clientes então
                Processa Chegada
        Se término de atendimento então
                Processa Término
                Tempo de simulação = tempo de simulação + 1
 Fim Enquanto
        Imprima Dados Estatísticos
```

Figura 2.4: Exemplo da orientação a atividade.

A simulação orientada a processo combina as facilidades da simulação orientada a evento e da simulação orientada a atividade. Essa simplicidade reside no fato da lógica dos eventos associados às informações estar contida na linguagem de simulação. Porém, como existem restrições ao conjunto de padrões fornecidos pela linguagem, a flexibilidade não é relevante quando comparada com a simulação orientada a evento. Essa abordagem deve seguir os seguintes passos [Spo94]:

- definir as entidades do sistema;
- criar um processo para cada entidade, descrevendo suas etapas;
- executar os processos concorrentemente.

Segundo esse tipo de orientação, no exemplo da lotérica, o processo cliente e chegada de cliente são organizados como ilustra as Figuras 2.5 e 2.6.

PROCESSO CHEGADA DE CLIENTE

Indique chegada de um novo cliente

Escalone o processo para tempo corrente + tempo entre chegadas

FIM

Figura 2.5: Exemplo do processo chegada de um cliente.

PROCESSO CLIENTE

Se o funcionário está desocupado então

  Torne o estado do funcionário ocupado.

    O cliente ocupa o funcionário pelo tempo de atendimento (escalone o processo para tempo corrente + tempo de atendimento)

  Torne o estado do funcionário desocupado

  O cliente parte do sistema

Senão

  O cliente entra na fila de espera

FIM

Figura 2.6: Exemplo do processo cliente.

Para a implementação de uma simulação segundo uma das orientações apresentadas, existem linguagens e pacotes de simulação que possibilitam ao modelador escrever programas de simulação orientados a evento ou a processo (a orientação à atividade não é utilizada).

2.3 Fases de Desenvolvimento da Simulação

A simulação corresponde à realização de experimentos sobre um modelo, matemático ou lógico, com a intenção de prever o comportamento real do sistema [Men99]. Para realizar uma simulação algumas fases devem ser observadas, como pode ser visto na Figura 2.7 [Mac87], as quais são descritas de acordo com Banks *et al* [Ban01].

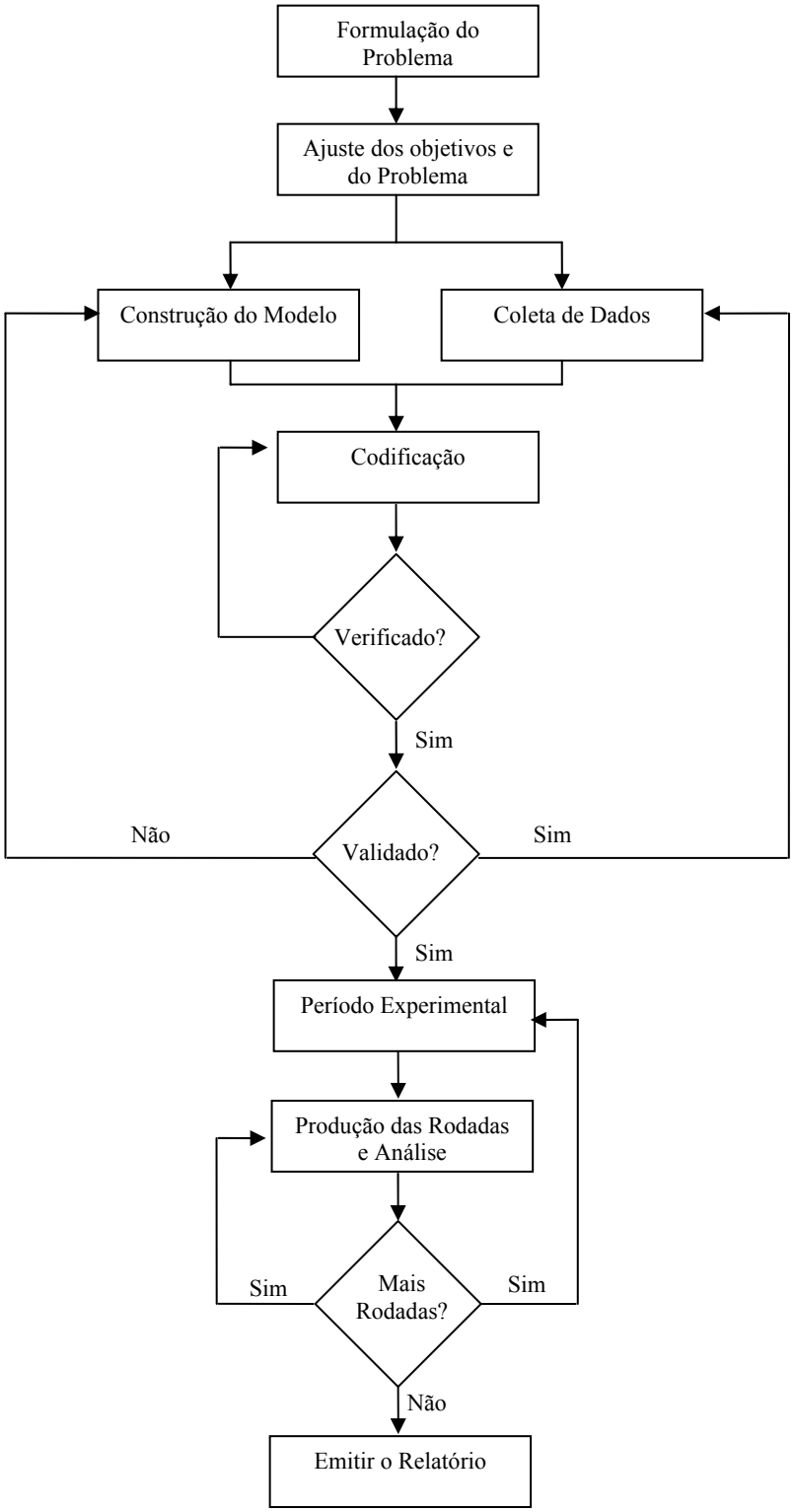


Figura 2.7: Fases de desenvolvimento da simulação seqüencial.

Formulação do Problema

Para descrever o sistema deve-se ter um bom entendimento do mesmo, para que se possa apresentá-lo sem dubiedade, sem negligenciar características relevantes e nem considerar detalhes desnecessários, o que poderia resultar em erros.

Ajuste dos Objetivos e Planos

Essa fase consiste na definição dos objetivos da simulação. Com base nessa descrição, pode-se estabelecer o nível de detalhamento, os tipos de dados de entrada e o tipo de ambiente de simulação. Percebe-se, dessa forma, que uma elaboração inadequada dos objetivos pode comprometer a avaliação de desempenho do sistema computacional em questão, dado que essa etapa compromete o processo como um todo.

Construção do Modelo

Uma vez que o sistema a ser simulado está bem definido, é necessário abstraí-lo através da construção de um modelo, o qual irá representar uma visão particular do sistema. Um modelo pode ser uma representação de alto ou baixo nível do sistema.

Em um modelo de alto nível, tem-se uma visão geral do sistema, correspondendo a uma "síntese" do sistema, na qual a representação de alguns elementos dos sistemas pode ser julgada desnecessária. Em um modelo de baixo nível, o sistema é descrito através de uma decomposição, ou seja, um detalhamento sucessivo. Nessa técnica a descrição do sistema real deve ser feita primeiramente em alto nível, fazendo-se posteriormente os detalhamentos sucessivos, até que seja alcançado o nível de detalhamento desejado. A descrição do modelo fica a critério do modelador, porém recomenda-se a utilização da decomposição em sistemas de grande complexidade.

Coleta de Dados

Ao gerar a descrição do modelo surge a necessidade de definir quais os tipos de dados que irão fluir no sistema. Esses dados de entrada servirão como parâmetros do modelo. Os dados podem ser valores hipotéticos (quando não se tem um sistema de referência) ou baseados em uma análise preliminar de um sistema de referência. Essa etapa pode ser realizada paralelamente à construção do modelo.

Codificação

Essa fase consiste em traduzir o modelo para uma forma aceitável pelo computador. O programa de simulação pode ser desenvolvido utilizando uma das seguintes opções. Com base no tipo de orientação da simulação, que pode ser a evento ou a processo, deve-se escolher dentre umas dessas alternativas de ferramentas (ambientes computacionais). Além disso, deve-se efetuar tal escolha considerando fatores como complexidade de desenvolvimento, restrições da linguagem, facilidade de programação, flexibilidade e portabilidade. Esse assunto será abordado de maneira mais aprofundada na Seção 2.4.

Verificação

O passo seguinte no processo de modelagem é verificar se a implementação do modelo representa exatamente a descrição e especificação conceitual do programador, isto é, se o programa de simulação é uma implementação válida do modelo [Hu01]. Para modelos pequenos, essa tarefa pode ser realizada por inspeção, contudo, para modelos maiores uma análise mais detalhada torna-se necessária. A técnica normalmente utilizada consiste em uma revisão cuidadosa do programa e do modelo [San94, Spo94].

A simulação é um processo amostral, isso significa que é gerado um grande número de valores amostrais e utiliza-se a média amostral como uma estimativa da média real. Conseqüentemente, algumas diferenças entre os resultados analíticos e da simulação podem ser resultado da variação amostral [Mac87].

Para resolver esse problema, pode-se conduzir uma análise adicional: coletar dados adicionais, computar limites de confiança para a estimativa da simulação e determinar se esse limite inclui o resultado obtido.

Validação

O modelo de simulação deve ser uma representação exata do sistema real, de acordo com objetivo do modelo, ou seja, o modelo deve reproduzir o comportamento do sistema com fidelidade, para satisfazer aos objetivos da análise. Durante a fase de validação compara-se o modelo com o sistema e, se o resultado não for satisfatório, a lógica do modelo, bem como as suposições e simplificações efetuadas e os dados coletados devem ser analisados cuidadosamente [Hu01, San94].

Existem dois casos de validação a considerar. No primeiro caso, o sistema modelado existe e pode ser medido. O objetivo da análise é avaliar uma mudança proposta para o sistema, e a validação é baseada na comparação dos resultados do modelo com os dados medidos a partir do sistema. Caso os resultados coincidam, assume-se que a simulação do sistema modificado produzirá estimativas válidas dos efeitos da mudança proposta [Mac87].

No segundo caso, existe apenas o projeto do sistema modelado, e o objetivo da análise é estimar o desempenho desse sistema projetado ou avaliar projetos alternativos. Quando o sistema modelado não existe, a validação é feita, inicialmente, através de uma revisão; o modelo é examinado em termos do projeto do sistema, e cada hipótese e abstração são justificadas [Mac87]. A validação pode também basear-se em comparações com outros modelos de sistemas já validados, como por exemplo, os modelos analíticos [San94, Spo94].

Uma das técnicas utilizadas para validar um modelo é *face validity* a qual consiste em verificar se o modelo é razoável para os usuários. Nesse contexto, ser razoável significa estar de acordo com o que se espera do modelo, ou seja, os resultados obtidos devem ser consistentes com o sistema modelado. Uma maneira de realizar *face validity* é a utilização de uma análise de sensibilidade sobre as variáveis de entrada [Ban01]. Uma estrutura para verificação e validação é proposta por Carson[Car02]:

- teste do modelo para *Face Validity;*
- teste do modelo para um número de parâmetros de entrada;
- quando possível, pode-se comparar as predições do modelo através de um modelo base representando o sistema, ou por desempenhos obtidos no sistema atual. Já no projeto de um novo sistema, pode-se usar hipóteses e especificações para efetuar a comparação com relação ao comportamento do modelo.

Período Experimental e Produção das Rodadas e Análise

Uma questão importante em simulação é por quanto tempo, ou quantas vezes, executar o programa de simulação. As entradas da simulação são obtidas através da geração de variáveis aleatórias de distribuições de probabilidades, as saídas da simulação são funções dessas variáveis e, portanto, a análise de saída é um problema estatístico. Isto significa que os

resultados das execuções de uma simulação devem ser cuidadosamente analisados para que se possa entender o seu significado.

Existem métodos descritos na literatura para determinar quando parar um programa de simulação, com base no cálculo de um intervalo de confiança para a média de uma variável de saída. Como exemplo pode-se citar replicação e *Batch Means* [Ban01, Bru03].

Outro conceito importante na simulação é a análise de sensibilidade, que consiste em determinar a sensibilidade das respostas em função da variação dos parâmetros de entrada. A análise de sensibilidade é realizada através da variação sistemática dos valores dos parâmetros em algum intervalo de interesse, observando esse efeito nas respostas do modelo.

É importante ressaltar que os resultados obtidos em simulações estocásticas são funções das variáveis de entrada, as quais são baseadas na geração de números aleatórios. Dessa forma, a análise baseia-se em estimar o intervalo de confiança dos valores médios obtidos na simulação [Mac87], significando que a simulação encerra quando a estimativa da variância da média está dentro de uma tolerância especificada. Se a precisão não for atingida devem ser realizadas mais rodadas.

Emissão de Relatório

Os resultados obtidos com a simulação devem ser fornecidos ao usuário de forma clara e concisa, permitindo análise de resultados de experimentos.

2.4 Tipos de Simulação

Ao desenvolver um modelo para simulação deve-se selecionar o enfoque que será utilizado para descrever as relações funcionais entre os elementos do sistema.

Ao utilizar uma linguagem de simulação, normalmente, o enfoque está implícito na linguagem. No entanto, ao utilizar uma linguagem de programação convencional de propósito geral, como Pascal ou C, o enfoque utilizado na organização da descrição do sistema é de responsabilidade do modelador [Soa92].

As alterações nas variáveis de estado do sistema são as bases para a classificação dos modelos. O tempo é a variável independente. As outras variáveis incluídas na simulação são

funções do tempo e, portanto, são variáveis dependentes. Os adjetivos discreto e contínuo na classificação dos modelos referem-se ao comportamento destas variáveis dependentes [Soa92]:

- modelo de mudança discreta: é aquele em que as variáveis dependentes variam discretamente em pontos específicos do tempo, referidos como tempo de evento;

- modelo de mudança contínua: é aquele em que as variáveis podem variar continuamente ao longo do tempo.

Com base nessa classificação, tem-se que a simulação discreta é o exercício de um modelo discreto, enquanto a simulação contínua é o exercício de um modelo contínuo.

A fase de desenvolvimento do programa de simulação consiste em traduzir o modelo para uma forma aceitável pelo computador. Essa tradução pode ser feita com a utilização de um dos seguintes enfoques:

- desenvolvimento do programa de simulação em uma linguagem convencional como por exemplo, C. Com a utilização dessa abordagem, o programador é responsável por criar o ambiente necessário para a simulação;

- utilização de uma extensão funcional de uma linguagem de programação convencional. As extensões são bibliotecas feitas a partir de uma linguagem como, por exemplo, C ou Pascal, e que unida a ela, compõem um ambiente completo de simulação. Exemplos de extensões funcionais da linguagem C são *SimPack* [Cub95] e SMPL [Mac87];

- utilização de linguagens de simulação de uso geral, as quais possuem todas as estruturas necessárias para simulação. Essas linguagens são classificadas como orientadas a evento ou a processo;

- utilização de pacotes de uso específico.

Para auxiliar o usuário, existem ambientes de simulação denominados ambientes automáticos (que incluem editores gráficos, interpretadores, compiladores e otimizadores), que afastam o usuário da tarefa de transcrição de um modelo em um programa de simulação, fornecendo recursos para a elaboração do mesmo.

Nos ambientes de simulação automáticos, o programa de simulação é gerado e executado automaticamente a partir de uma representação gráfica do modelo do sistema a ser avaliado. A saída da simulação também pode ser gráfica e o usuário observa o comportamento do sistema através de gráficos ou animações.

A simulação pode ser classificada como determinística ou estocástica, de acordo com os dados de entrada. A simulação determinística envolve problemas para os quais os parâmetros de entrada sempre ocorrem sob determinado conjunto de condições. Na simulação estocástica os dados de entrada são gerados aleatoriamente, a partir de uma distribuição de probabilidade.

A simulação estocástica é um processo amostral, ou seja, nela são gerados valores amostrais da distribuição e utiliza-se a média amostral como uma estimativa da média real. Isso implica na necessidade de validar o modelo para determinar o grau de confiança e a corretude dos resultados. Algumas diferenças entre os resultados obtidos por solução analítica e por simulação podem ser resultantes dessa variação amostral. Dessa forma, a análise de saída é de fundamental importância na solução de modelos por simulação, pois as saídas geradas são valores médios que exigem uma análise estatística.

Em casos de modelos complexos que demandam grande esforço computacional, a simulação é executada repetidamente até o momento em que a análise estatística dos resultados indicar que a precisão desejada foi obtida, o que pode consumir um tempo de processamento inviável [Mac87, Soa92,] com a utilização dos processadores disponíveis. Para amenizar esse problema, pode-se utilizar simulação distribuída, abordada no Capítulo 3.

2.5 Redes de Filas e Modelos Baseados em Redes de Filas

Modelagem para simulação é assunto de interesse de projetistas e analistas de sistemas. Os projetistas de sistemas desejam ser capazes de prever o comportamento de um novo sistema proposto e selecionar o melhor projeto a partir de uma série de alternativas,

enquanto os analistas de sistemas estão preocupados com o efeito de mudanças em um sistema existente e se esses sistemas podem suportar um aumento de carga de trabalho.

O compartilhamento de recursos limitados dentro de um sistema é de fundamental importância na sua operação. Se não há disputa, a análise de desempenho torna-se mais fácil. Porém, essa suposição não é realista. Para ilustrar esse compartilhamento, tem-se a situação em que *jobs* em um computador competem pela UCP (Unidade Central de Processamento), unidades de entrada e saída, canais, entre outros; mensagens em redes de comunicação competindo por enlaces, *buffers*, permissão, janelas; e tarefas em uma linha de montagem competindo por ferramentas, áreas de estocagem, robôs, mecanismos de transportes, entre outros [Soa92].

Basicamente, uma rede de filas é constituída de entidades, os centros de serviços, e um conjunto de entidades chamadas clientes, os quais recebem serviços nos centros. Um centro de serviço é constituído de um ou mais servidores, representando os recursos do sistema, que prestam serviço aos usuários, e uma linha de espera, denominada fila, para os clientes que estão esperando pelo serviço. Como exemplo de centro de serviço pode-se considerar a Unidade Central de Processamento (UCP) de um sistema computacional. A Figura 2.8 mostra os tipos de sistemas de redes de filas, onde a Figura 2.8(a) apresenta um modelo com apenas uma fila e um servidor. A Figura 2.8(b) apresenta uma fila e vários servidores. A rede de filas exemplificada na Figura 2.8(c) é constituída por várias filas e um servidor. Já a Figura 2.8(d) mostra um sistema constituído por múltiplas e filas e servidores.

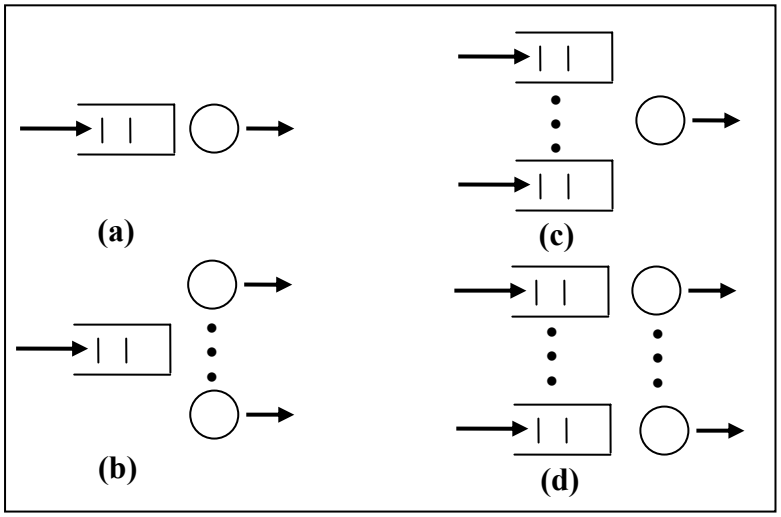


Figura 2.8 – Tipos de Redes de Filas.

É interessante observar a notação gráfica utilizada para modelar essas redes. O círculo representa um servidor e o quadrado uma fila. As setas representam a chegada e a partida dos clientes do sistema. Os clientes chegam ao servidor, e se o servidor não está ocupado o cliente é atendido, caso contrário, une-se à fila e espera por sua vez. Um cliente é selecionado para atendimento de acordo com a disciplina da fila. O serviço requisitado é executado pelo servidor e ao seu término o cliente deixa o sistema [Nas97].

Pelo fato de serem redes de filas, deve haver uma política de escalonamento para escolha do usuário que será o próximo a ser atendido pelo centro de serviço. Existem diversas disciplinas, e as mais utilizadas são *FIFO* (*First In*, *First Out*) e *LIFO* (*Last In*, *First Out*).

A Figura 2.8(a) apresenta o exemplo de um centro de serviço constituído de uma única fila que alimenta vários servidores. Como exemplo desse tipo de modelo pode-se considerar uma agência bancária, onde vários caixas atendem aos clientes que formam uma fila única. Pode-se ter também modelos constituídos de centros de serviços formados por múltiplas filas associadas a um único servidor, como mostra a Figura 2.8 (b). Um exemplo desse tipo de sistema pode ser um guichê e de uma repartição pública que atende a usuários que formam duas filas diferentes. Uma das filas se destina aos aposentados, e a outra atende às demais pessoas. O funcionário dá preferência para a fila de aposentados, somente atendendo a outra quando essa estiver vazia.

Outra possibilidade é a de um modelo constituído de múltiplas filas e múltiplos servidores, situação ilustrada na Figura 2.8(d). Como exemplo, considera-se uma agência bancária, onde vários caixas atendem a clientes que formam duas filas, uma para os clientes em geral, e outra para idosos, gestantes e portadores de deficiências físicas. Os clientes da fila especial são atendidos com prioridade.

Alguns sistemas de fila são compostos por um número de subsistemas interligados em rede. Em tais sistemas os fregueses recebem serviço em mais de um subsistema antes de terem seus requisitos de serviço totalmente atendidos. Tais sistemas são denominados redes de fila.

A notação padrão para sistemas baseados em redes de filas é: A/S/c/k/m, onde A representa a Distribuição do tempo de chegada, S representa a Distribuição de tempo de serviço, c é o número de servidores, k é o número máximo de clientes no sistema e m é o número de clientes disponíveis na fonte [Mac87].

As densidades de probabilidade A e S são colhidas a partir do conjunto, entre outros:

- M: indica função densidade de probabilidade exponencial (M significa *Markov*);
- D: todos os clientes têm o mesmo valor (D quer dizer determinística);
- G: geral (isto é, função densidade de probabilidade arbitrária);
- Ek: função densidade de probabilidade *Erlang* de ordem k.

Um sistema de filas pode ser caracterizado pelos seguintes componentes [San94]: padrão de chegada, distribuição do serviço, capacidade de serviço, disciplina da fila, número de servidores e tamanho da área de espera [Soa92].

Padrão de Chegada

O padrão de chegada é a freqüência regular, com que os clientes chegam ao sistema. Essa freqüência pode ser regular ou aleatória, os clientes chegam a instantes igualmente espaçados. No entanto, a chegada regular é irreal em muitas situações. Na freqüência aleatória, os dados podem ser obtidos através de tabelas ou pela utilização de uma distribuição de probabilidade. Algumas distribuições importantes são [Soa92]:

- Distribuição Uniforme;
- Distribuição Triangular;
- Distribuição Exponencial;
- Distribuição de *Poisson;*
- Distribuição Normal;
- Distribuição de *Erlang*;

Distribuição do serviço e a capacidade de serviço

O serviço solicitado por um cliente é denominado demanda de serviço ou simplesmente trabalho. A unidade de serviço varia de acordo com a natureza do servidor e dos clientes.

Pode ser uma instrução, se o servidor for uma UCP e o cliente um programa. Se o servidor é uma linha de transmissão, os clientes são mensagens e a unidade de serviço pode ser bit ou byte. A demanda de serviço pelos clientes segue uma distribuição denominada distribuição de serviço.

Deve-se especificar também a capacidade do servidor, a qual representa o quão rápido o servidor realiza o serviço. A unidade da capacidade também depende do servidor e dos clientes. No caso da UCP, a unidade pode ser instruções/seg ou MIPS e no caso da linha de transmissão a unidade pode ser bps (bytes por segundo) ou kbps (kilo bytes por segundo).

Se a demanda S de serviço de um cliente (unidade de serviço) e o correspondente serviço possui uma capacidade C (unidade de serviço/seg), então a razão S/C (seg) é chamada de tempo de serviço. Em outras palavras, é o tmpo necessário para que o servidor execute um serviço. Normalmente é especificado por uma distribuição de probabilidade.

Disciplina da Fila

A disciplina da fila descreve a ordem com que os clientes são retirados da fila. Existem vários algoritmos, denominados algoritmos de escalonamento, usados para decidir qual usuário deve entrar em serviço quando um servidor se encontra disponível, entre esses cabe destacar:

- FCFS (*first-come first-served*): consiste em uma fila padrão, a ordem de chegada é a ordem de atendimento no centro de serviço;
- LCFS (*last-come first-served*): o último usuário a chegar ao centro é o que entrará em serviço assim que houver disponibilidade do servidor;
- RR (*round-robin*): cada usuário é atendido por um pequeno intervalo de tempo denominado *quantum*. Caso não tenha sido suficiente, o usuário volta para o final da fila e o usuário seguinte entra para receber seu *quantum* de

serviço. O processo se repete até que o serviço requisitado pelo usuário se complete;

- PS (*Processor Sharing*): todos os usuários dividem a capacidade do centro de serviço, como se executassem em paralelo;

- IS (*Infinite Server*): nunca existe fila. Todos os usuários são servidos assim que chegam ao sistema;

- PRTY (*Nonpreemptive Priority*) e PRTYPR (*Preemptive-Restime Priority*): o escalonamento com prioridade é utilizado para dar preferência a alguns usuários. Com prioridade sem *preempção*, o usuário em atendimento não é afetado quando um usuário de prioridade mais alta chega ao centro. Já na prioridade com *preempção*, a chegada de um usuário com prioridade mais alta tira de serviço o usuário de prioridade mais baixa, que só entrará em serviço assim que todos os usuários com prioridades maiores forem atendidos.

Número de Servidores

Pode-se ter um ou mais servidores no centro de serviço. Quando mais de um servidor está disponível, deve-se escolher qual deles irá atender o cliente. O servidor selecionado pode ser o que está desocupado a mais tempo, ou pode ser escolhido aleatoriamente ou, ainda, segundo algum tipo de prioridade.

Tamanho da Área de Espera

Nem todos os sistemas têm uma área de espera infinita para as filas. Quando clientes estão enfileirados em número excessivo para um espaço finito de área de espera, alguns clientes são perdidos ou rejeitados.

2.6 Pacotes e Linguagens para Simulação

O intenso uso de simulação como uma abordagem de análise de desempenho de sistemas deu origem a uma série de ferramentas específicas projetadas para esse fim. Essas

diversas linguagens e pacotes de modelagem impõem uma certa estruturação nos modelos e simplificam suas soluções [Soa92].

A simulação é uma ferramenta útil para a avaliação de desempenho de sistemas, porém, a sua utilização não é simples, pois implica em aprendizado de linguagens de simulação, de modelagem de sistemas e de análise dos dados. Desse modo, para facilitar o uso da simulação, existem ambientes de simulação automáticos que afastam o usuário da tarefa árdua de transcrição de um modelo em um programa de simulação tais como *Asia* [Bru97], *ARENA* [Swe01], e *OMNeT++* [Var01].

Um requisito importante para as linguagens de simulação é a presença de mecanismos para a representação do tempo. Enquanto o sistema sendo modelado executa em tempo real, a simulação trabalha com um relógio próprio, que marca a passagem do tempo no programa de simulação. Outro requisito importante é a capacidade de fornecer facilidades para coletas de estatísticas e emissão de relatórios [Bru97].

Pode-se usar os seguintes enfoques para a implementação de um modelo de simulação:

- linguagens de programação convencional;
- linguagens de simulação;
- pacotes de uso específico;
- extensões funcionais.

2.6.1 Linguagem de Programação Convencional

Em princípio, toda linguagem de programação é uma candidata a linguagem de simulação. Porém, para desenvolver um programa de simulação em uma linguagem convencional (FORTRAN, C++, Pascal) o programador deve criar todo o ambiente necessário para a simulação, pois essas linguagens não oferecem todas as ferramentas (estruturas de dados, facilidades para a manipulação de listas, abstração de dados, coletas de estatísticas e emissão de relatórios) necessárias para um ambiente de simulação.

A desvantagem de se utilizar linguagens convencionais é que o programador deve ter conhecimento da linguagem para poder criar um ambiente de simulação.

Uma vantagem é que o programador não necessita aprender uma nova linguagem, já que tem a liberdade de utilizar uma que já é de seu conhecimento. Esta abordagem oferece ainda uma grande flexibilidade uma vez que o usuário pode utilizar todos os recursos oferecidos pela linguagem.

2.6.2 Linguagens para Simulação

As linguagens para simulação são projetadas para a modelagem de vários tipos de sistemas [Soa92]. Essas linguagens já contêm todas as estruturas necessárias para a criação de um ambiente de simulação, livrando o programador da necessidade de executar tal tarefa. As linguagens são classificadas em orientadas a evento, atividade ou processo e a escolha irá depender se o modelo é orientado a evento, atividade ou processo.

Linguagens Orientadas a Evento

Nas linguagens orientadas a evento, o programa de simulação é organizado como um conjunto de rotinas ou seções de eventos. Essas linguagens tendem a impor uma visão global e de alto nível do sistema ao modelador, o que as tornam mais adequadas a modelos de pequeno e médio porte. Com o aumento da complexidade, sugere-se a estruturação do modelo para a utilização de uma linguagem orientada a processo. Exemplos de linguagem de simulação orientada a evento são SimJAVA [Kre97], *SIMSCRIPT* II. 5, Slam II, Siman V e ModSim II [Ban01].

Linguagens Orientadas a Processo

Uma linguagem orientada a processo é mais utilizada em análise de sistemas complexos, como por exemplo sistemas na área de robótica, projeto de sistemas computacionais e redes de comunicação [Bru97].

Nesta abordagem o sistema é visto como uma coleção de processos interativos. Um processo tem as seguintes características:

- pode ser ativado, ficar em estado suspenso ou encerrar sua execução em certo instante de tempo ou baseado em alguma condição;
- uma vez ativado, repete seu comportamento até ser colocado no estado suspenso ou terminar sua execução.

Programas de simulação escritos nesta linguagem são construídos de uma forma mais natural, tornando o sistema e o modelo similares. Exemplos de linguagens de simulação orientadas a processo são RESQ [Soa92], GPSS/H [Cra99], Simula e Slam II [Ban01].

2.6.3 Pacotes de Uso Específico

Os pacotes de uso específico são voltados para a avaliação de sistemas particulares. Devido ao fato de serem muitos específicos para uma dada aplicação, esses pacotes oferecem facilidades em pontos particulares da aplicação para a qual foram desenvolvidos, mas são pouco flexíveis a mudanças [San94]. Nestes pacotes, a formulação do modelo é constituída na própria ferramenta, sendo que os parâmetros do modelo são especificados de forma definida pelo pacote. Alguns exemplos são Arena [Roh02, Sad99], AutoMod [Roh02, Sta01], ProModel [Har03] e SIMFACTORING II [Gob91].

2.6.4 Extensões Funcionais

Para facilitar o trabalho do programador, algumas bibliotecas (conjunto de rotinas disponíveis ao programador) podem ser inseridas em linguagens convencionais (chamadas hospedeiras), compondo assim um ambiente completo de simulação. O trabalho do programador é facilitado, pois ele não precisa aprender uma nova linguagem [Bru00]. Algumas extensões são:

- extensões da linguagem C: SMPL [Mac87] (orientada a evento), CSIM [Hla02] e EFC (orientada a processo);
- extensões da linguagem C++: SIMPACK [Fis92], C++Sim [Lit93] (orientadas a evento e processo) e SimKit [Gom95] (orientada a evento);
- extensões da linguagem Modula 2: EFM2 [Spo92] e HPSIM [Sha88] (orientadas a processo).

2.7 Extensão Funcional SMPL

SMPL constitui uma extensão funcional da linguagem de Programação C, para simulação discreta orientada a eventos em plataformas compatíveis com o *IBMP-PC* [Uls99]. A biblioteca “smpl.h” contém as declarações externas para as funções do SMPL, o que facilita

o trabalho do programador, que não precisa aprender uma nova linguagem de programação, pois pode trabalhar com a que está habituado. Está implícito em SMPL o enfoque orientado a evento que é dado ao programa de simulação.

2.7.1 Estrutura SMPL

Na extensão funcional SMPL existem três entidades básicas para a representação dos modelos: recursos, *tokens* e *eventos* [Mac87].

- **recurso**: um recurso representa algum recurso do sistema modelado, tais como a UCP em um sistema computacional ou um barramento em um modelo de rede local. SMPL provê funções para realizar as tarefas de requisição, liberação e *preempção* de eventos no recurso*;*

- ***tokens***: os *tokens* representam as entidades ativas do sistema. O comportamento dinâmico do sistema é modelado pelo movimento dos *tokens* através de um conjunto de recursos. Um *token* pode representar, por exemplo, uma tarefa em um modelo de sistema computacional ou um pacote em um modelo de rede de comunicação;

- **eventos**: uma mudança de estado de qualquer entidade do sistema é um evento. Suponha que um processo encerra sua execução na UCP e a libera. Dois eventos ocorreram: o processo terminou e a UCP mudou de estado (de ocupada para disponível). SMPL provê funções para escalonar eventos e para selecionar eventos pela ordem de seus tempos de ocorrência.

Um programa de simulação em SMPL compreende uma rotina de iniciação, uma rotina de controle e algumas rotinas de evento. A rotina de controle seleciona o número do próximo evento a ocorrer e transfere o controle para a rotina de evento apropriada que, tipicamente, escalona um ou mais outros eventos e então retorna a rotina de controle, e por fim são realizadas operações para a coleta de informações estatísticas. O trecho de código apresentado na Figura 2.9 apresenta a estrutura de um programa da simulação utilizando SMPL [Uls99].

```
#include <smpl.h>
main ()

{

 //Declaração das Variáveis
 //Inicialização das Variáveis
  facility();
  schedule();
  while()
  {
    cause(evento, token)
    {
      switch(evento)
      {
        evento 1:

          evento 2:

           evento 3:
       }
     }
   }
    report();
}
```

Figura 2.9 – Estrutura de um programa em SMPL.

2.7.2 Primitivas do SMPL

O SMPL fornece ao usuário um conjunto de primitivas que auxiliam o desenvolvimento de programas de simulação com orientação a eventos. As principais primitivas dessa extensão são [Uls99]:

- *smpl:* responsável por iniciar o sistema para que a simulação possa ser executada. Atribui os valores iniciais às estruturas de dados e ao tempo de simulação (relógio);

- *facility*: essa primitiva cria e nomeia um descritor para cada recurso do modelo. O descritor é utilizado para qualquer operação a ser realizada com o recurso, como a requisição ou o encerramento dos serviços. Cada recurso possui um número de servidores e uma fila de acesso. Quando a reserva do recurso é executada, os servidores são examinados e se algum estiver disponível, esse será reservado para o *token* solicitante;

- *request*: essa primitiva é utilizada para requisitar o atendimento de um dos servidores de um recurso. Se um servidor está livre, esse servidor é reservado para o *token* solicitante. Se não existir servidor disponível, a primitiva retorna um valor indicando que o recurso está ocupado. Nesse caso, a requisição é inserida na fila do recurso;

- *release*: libera um servidor de um determinad recurso. Após liberar o servidor, o sistema atualiza as estatísticas e decrementa o número de servidores ocupados do recurso. Caso exista algum evento pendente na fila do recurso, esse evento é inserido no início da lista de eventos futuros para execução imediata;

- *schedule*: responsável pelo escalonamento dos eventos da simulação. Todo evento a ser escalonado é inserido na lista de eventos futuros em ordem crescente do tempo de ocorrência do evento;

- *cause*: essa primitiva remove o elemento do início da lista de eventos futuros e avança o tempo da simulação (relógio) para o tempo de ocorrência do evento a ser executado.

O SMPL possui também diversas primitivas para a manipulação das filas de eventos dos recursos e da lista de eventos futuros, além de primitivas para análise estatística [Nas97].

2.8 Análise de Saída

Após a definição do ambiente computacional a ser usado, constrói-se o programa de simulação, que ao ser executado gera um conjunto de resultados que deve representar corretamente as características do sistema [Orl95]. Para assegurar a validade dos resultados de uma simulação, além dos cuidados que devem ser tomados na fase que antecede a execução do programa de simulação, deve-se fazer uma análise desses resultados, para que se possa verificar a precisão dos mesmos.

2.8.1 Período Transiente ou Período de *Warm-Up*

O período transiente equivale ao período em que o sistema ainda não está em equilíbrio. Isto implica que os dados coletados pelo programa não são significativos para uma avaliação do período de equilíbrio. Um sistema fica em equilíbrio ao atingir um estado estacionário, como por exemplo, o sistema tem a capacidade de fornecer o serviço solicitado sem a necessidade de alterar o padrão entre chegadas. A análise quanto à precisão dos resultados da simulação é averiguada após atingir o equilíbrio, isto é, a questão do *warm-up* estar resolvida [Orl95]. Caso o sistema seja instável, a simulação nunca ultrapassará o período

transiente e os resultados permanecerão numa trajetória não estacionária, alterando seu comportamento continuamente [Bru03].

2.8.2 Métodos para Análise de Saída

Uma questão importante é o período de tempo, ou quantas vezes executar o programa de simulação. As entradas do programa são obtidas através da geração de variáveis aleatórias de distribuições de probabilidades, as saídas da simulação são funções dessas variáveis e, portanto, a análise de saída é um problema estatístico. Isso significa que os resultados das execuções de uma simulação devem ser cuidadosamente analisados para total compreensão de seu significado.

Existem vários métodos descritos na literatura para determinar quando parar a execução de um programa de simulação, os quais se baseiam no cálculo de um intervalo de confiança para a média de uma variável de saída. Várias são as técnicas utilizadas para definir um intervalo de confiança, dentre as quais se destacam as seguintes [Bru03, Mac87, Paw90]:

- técnica de replicação: o intervalo de confiança é definido através dos dados provenientes de um número específico de execuções do programa de simulação. Assim, são feitas N execuções, denominadas replicações, usando-se diferentes sementes de números aleatórios em cada uma dessas execuções. Tendo-se os resultados das N execuções (ou replicações), define-se o intervalo de confiança através da média desses resultados. Essa é a técnica mais simples de ser aplicada, no entanto, na maioria dos casos, pode exigir um elevado número de replicações;

- técnica de regeneração: nessa técnica, o modelo de simulação retorna a um ponto previamente determinado, a partir do qual é reiniciada automaticamente a simulação. O intervalo de confiança é construído através dos dados coletados a cada um desses ciclos. A grande dificuldade dessa técnica consiste na definição desses pontos de retorno;

- técnica das médias por lotes (*batch means*): divide-se uma execução longa em um conjunto de N sub-execuções, denominadas lotes, cada um determinado comprimento m. O intervalo de confiança é calculado ao término de cada lote. Esse valor do intervalo de confiança deve ser comparado com o valor requerido. Caso a precisão requerida já tenha sido alcançada, o processo é

encerrado. Caso contrário, determina-se um novo lote e repete-se o processo de comparação até que a precisão requerida tenha sido atingida. Essa técnica é mais complexa que a de replicação, no entanto, mais eficiente, já que o tratamento do *warm-up* é feito uma única vez, enquanto que na replicação o *warm-up* é tratado N vezes (N replicações).

Os sistemas simulados, em geral, são compostos de um ou mais elementos, os quais possuem certo nível de incerteza associada a eles. Esses sistemas evoluem no tempo de um modo imprevisível, e são chamados de sistemas estocásticos. A simulação de sistemas estocásticos requer que a variabilidade dos elementos no sistema seja caracterizada utilizando-se conceitos probabilísticos.

Mesmo quando se decide construir um modelo de simulação, é uma boa idéia ter um modelo analítico mais simples, que possa ser resolvido, e que seja uma aproximação do modelo simulado. O modelo analítico pode fornecer um meio de constatação parcial de que o modelo de simulação está funcionando corretamente [Soa92].

Em sistemas que possuem clientes, servidores e filas, ou seja, em sistemas baseados na teoria de redes de filas, as medidas de desempenho mais utilizadas são [Soa92]:

- utilização do servidor: fração de tempo que o servidor está ativo;
- *throughput* ou vazão: número de serviços completos por unidade de tempo;
- tamanho médio da fila: número médio de usuários que esperaram para serem atendidos pelo servidor;
- tempo médio de espera na fila: razão entre o tempo total de espera na fila e o número de clientes na fila;
- tempo médio de serviço: razão entre o tempo que o servidor foi utilizado e o número total de clientes servidos;
- tempo médio no sistema: tempo médio que os clientes gastaram no sistema, contando o tempo de espera na fila e o tempo de serviço.

Técnica da Replicação/Deleção

Em simulações terminantes, o problema do período transiente pode ser solucionado de acordo com uma das seguintes abordagens [Mac87]:

- prevenindo a sua ocorrência, estabelecendo as condições iniciais que representam o comportamento do sistema no estado de equilíbrio (*steady-state*);
- eliminando os seus efeitos, através da deleção das primeiras l observações iniciais;
- executando a simulação por um tempo razoavelmente longo, com o intuito de reduzir os efeitos a um nível pouco significativo.

A primeira opção é praticamente inviável devido à impossibilidade de prever os valores do estado de equilíbrio. Na abordagem de replicação, utiliza-se um método para identificar o tamanho do período transiente e eliminá-lo.

O problema da correlação entre as observações pode ser solucionado com a realização de replicações independentes da simulação. A simulação é repetida um determinado número de vezes e cada replicação utiliza uma seqüência independente e diferente de números pseudo-aleatórios. Supondo que k replicações independentes são executadas e que cada replicação produz n observações (X_{i1}, X_{i2},..., $\boldsymbol{X}_{in}$), e que as primeiras l observações são descartadas, tem-se que as médias amostrais

$$Y_i = \frac{1}{n - l} \sum_{j = l+1}^{n} \boldsymbol{X}_{ij} \tag{4.1}$$

são variáveis aleatórias independentes e identicamente distribuídas e se

$$Y_i = \frac{1}{k} \sum_{i=1}^{n} \bar{kY_i} \tag{4.2}$$

é um estimador não viciado da média μ e a variância amostral dos Y_i´s

$$S_k^2(Y) = \frac{1}{k-1}\sum_{i=1}^{k}(Y_i - \bar{Y}_k)^2 \qquad (4.3)$$

é um estimador não viciado da variância de (X) (*n*). Para *k* e *n* suficientemente grandes e um nível de confiança de (1 - σ), a média μ estará contida no intervalo de confiança:

$$\bar{Y}_k \pm t_{k-1,1-\sigma/2}\sqrt{S_k^2/k} \qquad (4.4)$$

Esse método é mais sensível em relação às observações coletadas no período transiente do que os métodos baseados em uma única execução, dado que a cada replicação, um novo período transiente ocorre [Paw90].

Algumas observações importantes com relação aos valores de *l*, *n* e *k* são [Ale01]:

- se *l* aumenta, para um valor de *n* fixo, o erro sistemático em cada *Y*, devido às condições iniciais, diminui, mas o erro amostral aumenta devido ao pequeno número de observações (a variância de *Y* é proporcional a *1/n-1*);
- se *n* aumenta, para um valor fixo de *l*, os erros sistemáticos e amostrais em *Y* diminuem;
- o erro sistemático na média amostral *Y* não é reduzido, aumentando o número de replicações *k*.

2.8.3 Geração de Números Aleatórios

O método utilizado para a geração de amostras aleatórias a partir de uma dada distribuição, consiste em primeiro gerar uma ou mais amostras uniformemente distribuídas, entre 0 e 1 e, posteriormente, transformá-las nas novas amostras desejadas. Amostras independentes, uniformemente distribuídas no intervalo de 0 a 1, são chamadas números aleatórios.

O método mais utilizado para a obtenção de números aleatórios consiste em empregar uma equação recursiva, que gera o *i*-ésimo número aleatório a partir do *(i -1)*-ésimo. Uma vez que a seqüência de números é produzida deterministicamente, a partir de uma semente, por uma equação, os números não são verdadeiramente aleatórios, e por isso são usualmente chamados de números pseudo-aleatórioss [Soa92]. Essa semente é necessária para dar início ao processo de geração de números aleatórios. No contexto da simulação, entretanto, os números pseudo-aleatórios são denominados de aleatórios por uma questão de simplificação da linguagem [L'e98].

Os geradores de números pseudo-aleatórioss devem possuir as seguintes propriedades:

- os números devem ser uniformemente distribuídos no intervalo [0,1];
- os números devem ser independentes;
- muitos números devem ser gerados antes que o mesmo número seja obtido. O total de números gerados antes da repetição é chamado ciclo ou período do gerador;
- deve permitir a reprodução da seqüência de números aleatórios, bem como diferentes valores de início, denominados sementes, de forma a permitir que diferentes seqüências sejam geradas.

A técnica que melhor satisfaz essas propriedades e também a mais usada é chamada método congruente. Esse método emprega as seguintes equações recursivas [Soa92]:

$$z_{i+1} = (az_i + b)\ (mod\ c),\ \mathrm{i} = 0,1,2,3\ \ldots \tag{4.5}$$

$$r_{i-1} = z_{i-1}/c \tag{4.6}$$

onde z_0 é a semente e r_i é o vigésimo número pseudo-aleatórios. Os melhores valores para *a*, *b* e *c* têm sido objetos de intensas pesquisas.

2.9 Considerações Finais

A avaliação de desempenho de sistemas computacionais pode ser realizada com a utilização de diversas técnicas, cada uma com vantagens e desvantagens inerentes. A simulação é uma técnica amplamente utilizada, devido a seu baixo custo e flexibilidade, uma vez que as mudanças no sistema são facilmente refletidas no modelo. A validade dos resultados obtidos através de uma simulação deve ser assegurada na fase de análise de saída. É nessa fase que os resultados obtidos com as execuções da simulação são analisados e a precisão dos mesmos é verificada. Se os resultados não forem significativos o programa de simulação não pode ser considerado representativo do sistema real.

O problema da simulação de modelos complexos é o tempo necessário para processar o programa de simulação, uma vez que o tempo geralmente aumenta com a complexidade do modelo, podendo tornar a simulação inviável. Técnicas de simulação distribuída podem ser utilizadas para resolver esse problema, pois proporcionam uma redução potencial do tempo de execução da simulação [San97]. O Capítulo 3 aborda os conceitos que envolvem o uso da simulação distribuída, descrevendo os aspectos do esquema básico da simulação distribuída e da modelagem do sistema físico a ser simulado, além de discutir o mecanismo de sincronização conservativa entre os processos lógicos para a simulação distribuída.

Capítulo 3

Simulação Distribuída

3.1 Considerações Iniciais

A simulação de sistemas complexos nas mais diversas áreas exige grande esforço computacional e elevado tempo de execução em máquinas seqüenciais, para que se possam obter resultados realistas. Uma tendência natural é tentar utilizar vários processadores para acelerar o processo de simulação. Nesse contexto, a simulação distribuída vem, gradativamente, substituindo a simulação seqüencial, permitindo o uso de máquinas e/ou ambientes paralelo com vários processadores e alta capacidade de comunicação [Bru03].

A maior parte das pesquisas em simulação distribuída tem como objetivo o desenvolvimento de novos algoritmos e a obtenção de melhor desempenho em aplicações específicas. Isso se deve ao fato de que a maioria dos usuários tem acesso somente a máquinas com apenas um processador, enquanto a utilização de máquinas com processamento paralelo fica restrita, principalmente, à comunidade científica, além de exigir experiência no desenvolvimento de aplicações paralelas e considerável esforço de programação para migrar de uma simulação seqüencial para o paradigma distribuído [Low99].

A natureza do mecanismo da *LEF* imposta pela simulação seqüencial, na qual a cada ciclo da simulação apenas um item é removido da lista de eventos, não permite a execução concorrente do algoritmo em máquinas com mais de um elemento de processamento, uma vez que a *LEF* não pode ser particionada. A existência de uma lista de eventos, ordenados cronologicamente, impõe uma organização seqüencial na simulação, onde apenas um evento é

executado por vez. Para possibilitar a utilização de diversos processos na execução de uma simulação é necessário adaptar a simulação seqüencial para essa nova situação, uma vez que as estruturas de dados utilizadas não podem ser compartilhadas na simulação distribuída e cada processo deve ter suas próprias estruturas de dados, como por exemplo, a *LEF* [Nic96].

Na simulação distribuída, o sistema a ser simulado é dividido em *n* subsistemas ou em *n* processos lógicos PL_0, PL_1,... , PL_n, e cada um representando um processo do sistema real. Cada PL pode trabalhar com a evolução de seu tempo de processamento de forma independente de outros PLs, mantendo assim seu próprio relógio *LVT* (*Local Virtual Time*). Quando um evento é processado, o *LVT* é automaticamente avançado para a marca de tempo desse evento [Fuj00]. O conjunto de *LVTs* de todos os PLs determina o valor do *GVT* (*Global Virtual Time*), que representa o tempo geral da simulação, isto é, pode-se afirmar que um sistema de simulação simulou o comportamento de um sistema real até o valor de tempo armazenado em *GVT*.

Todas as interações entre os processos físicos são modeladas por mensagens trocadas entre os processos lógicos correspondentes, obedecendo à ordem seqüencial imposta pelo sistema real. As mensagens são trocadas entre os processos por intermédio dos canais de comunicação.

Na simulação, se dois eventos são escalonados para execução em um mesmo processo, o evento com a menor marca de tempo deve ser executado primeiro. Se a execução de um evento em um processo resulta no escalonamento de outro evento em um processo diferente, o evento local deve ser executado antes do evento remoto. Todo o processamento de eventos deve corresponder ao comportamento do sistema real, os relacionamentos de causa e efeito dos eventos nunca devem ser violados. O relacionamento de causa e efeito determina que, se em um sistema real sendo simulado, o evento E_1 ocorre antes do evento E_2, na simulação o mesmo deve ocorrer, ou seja, E_1 deve ser tratado antes de E_2.

Uma simulação distribuída é constituída por processos lógicos que interagem por via da troca de mensagens, as quais possuem uma marca de tempo associada. Um processo obedece ao relacionamento de causa e efeito, se e somente se cada processo lógico executa os eventos em ordem não decrescente de marca de tempo. Como exemplo, pode-se considerar a seguinte situação: o processo PL_1 executa o evento E_1 com marca de tempo 10 e o processo PL_2 executa o evento E_2 com marca de tempo 20, como mostra a Figura 3.1(a). Se E_1

escalonar um novo evento E_3 para execução em PL_2 com marca de tempo menor que 20, E_3 pode afetar o evento E_2, o que implica na necessidade de execução seqüencial dos três eventos, como mostra a Figura 3.1(b) [Fuj00].

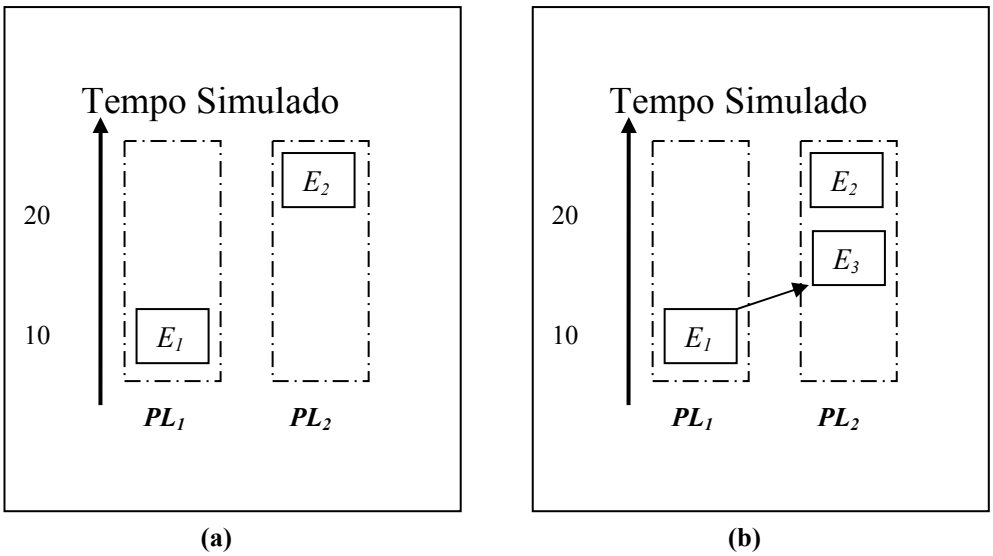


Figura 3.1: Ocorrência de erros de causa e efeito.

Existem dois grupos de protocolos de sincronização baseados em troca de mensagens: síncronos e assíncronos. Os protocolos síncronos possuem um mecanismo global para o controle da simulação, através do qual os processos compartilham o mesmo relógio global, processando todos os eventos que ocorrem em um determinado tempo da simulação. Devido aos mecanismos de sincronização utilizados, esses protocolos são mais adequados para arquiteturas com memória compartilhada. Janela de Tempo Conservativo [Aya94] e Evento Condicional são exemplos desses protocolos [Fuj90, Fuj93].

Nos protocolos assíncronos cada processo possui um relógio local e sincroniza com os processos com os quais se comunica. Os eventos são processados pela ordem crescente de tempo de ocorrência, assim, os processos podem assumir valores de relógios distintos em um determinado ponto da simulação [Fer95]. Esses protocolos dividem-se em conservativos [Mis86], otimistas [Jef85] e mistos [Jha96], conforme ilustrado na Figura 3.2. Nessa última classe estão incluídos os modelos que podem ser executados utilizando o modo conservativo ou o modo otimista. Alguns autores citam ainda outra classe de protocolos, denominados protocolos adaptativos, os quais possuem a capacidade de modificar seu comportamento dinamicamente, em resposta às mudanças no estado da simulação [Bag94b].

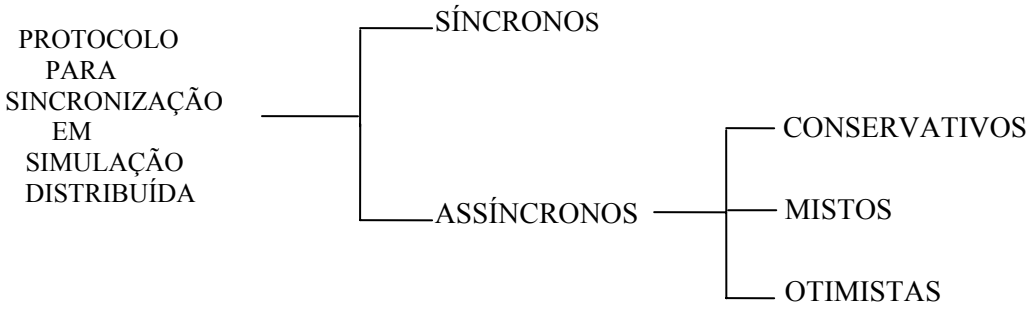


Figura 3.2: Classificação dos protocolos em simulação distribuída.

Para solucionar o problema de sincronismo entre os processos e a troca de informações na simulação distribuída, foram desenvolvidos protocolos de troca de mensagens entre os PLs. As mensagens enviadas devem carregar, juntamente com seu conteúdo, o valor da marca de tempo. O recebimento de mensagens com marca de tempo menor que o *LVT* (conhecidas como *straggler*s) ocasiona erros de causa e efeito os quais devem ser evitados ou tratados [Uls99].

Os erros de causa e efeitos são caracterizados por situações onde o futuro afeta o passado [Fuj00, Mis86] e somente ocorrem quando são utilizados protocolos otimistas. O principal representante dessa classe de protocolos é o *Time Warp* [Fuj00]. Os protocolos conservativos evitam a possibilidade da ocorrência de erros de causa e efeito, determinando quando é seguro processar um evento. Já os protocolos otimistas são caracterizados pela utilização de uma estratégia de detecção e recuperação, na qual o erro de causa e efeito é detectado e um mecanismo de *rollback* é utilizado para recuperação [Spo00, Uls99].

3.2. Fases de Desenvolvimento de uma Simulação Distribuída

As fases necessárias para o desenvolvimento de um programa de simulação distribuída são basicamente as mesmas utilizadas para a simulação seqüencial. Entretanto, há ainda o acréscimo de duas novas fases, o particionamento e a configuração da arquitetura, as quais são descritas a seguir.

Particionamento

Nessa fase deve-se decidir qual é o particionamento em processos lógicos que resulta em melhor desempenho dos centros de serviço do modelo a ser simulado, pois a forma como

os processos são particionados pode acarretar diferenças no balanceamento e comunicação entre os processos. Uma análise sobre as possíveis formas de realizar esse particionamento é apresentada em Ulson [Uls99]:

- o número de processos lógicos envolvidos na partição do modelo deve ser determinado em função das características da plataforma utilizada;
- para evitar comunicação desnecessária entre os processos, a partição deve agrupar os recursos em uma seqüência lógica;
- em sistemas com meio de comunicação de alto de desempenho, a partição deve ser realizada procurando maximizar o balanceamento da plataforma. Quando se tem meio de comunicação de baixo desempenho deve procurar reduzir a comunicação, ou seja, a partição deve ser feita em função das características da plataforma de *hardware* e *software* utilizada;
- em modelos com recursos estruturados em série a partição deve adotar a granulosidade grossa, reduzindo o número de processos e consequentemente, a comunicação entre eles;
- um melhor desempenho é obtido em modelos com recursos formando estrutura de *feedback* (realimentação), agrupando os recursos envolvidos na realimentação em um mesmo processo lógico, devido à redução na comunicação.

Configuração da Arquitetura

Para executar uma simulação distribuída há a necessidade de se estabelecer a arquitetura em que a simulação será executada [Uls99]. No caso da utilização de um ambiente no quais os computadores estão distribuídos através de uma rede de interconexão, devem-se estabelecer quais são as máquinas participantes da computação, bem como o ambiente de troca de mensagens utilizado.

3.3 Protocolo Conservativo CMB

Nas simulações conservativas, um processo lógico só trata um evento se puder garantir que não chegará um outro com marca de tempo menor do que a do evento a ser tratado. O problema básico desses protocolos consiste em determinar quando é seguro processar um evento, evitando rigorosamente que ocorra em algum momento um erro de causa e efeito. Os processos que não contêm eventos seguros ficam bloqueados, o que pode levar a situação de *deadlock* se não forem tomadas às precauções necessárias [Fuj00, Mis86]. *Deadlock* é uma situação em que um determinado número de processos forma um ciclo de espera uns pelos outros, para poderem continuar executando, o que acarreta uma espera infinita.

O tratamento do *deadlock* é um dos principais enfoques dos protocolos conservativos. Alguns protocolos previnem o *deadlock*, enquanto outros permitem sua ocorrência, para depois detectá-la e utilizar algum mecanismo de recuperação [Mis86].

Os primeiros algoritmos para simulação distribuída foram desenvolvidos por Chandy e Misra [Cha79] e Bryant [Bry77], dentre eles destaca-se o protocolo CMB (conhecido como sinônimo de protocolo conservativo). O CMB define que um evento só pode ser executado se possui a menor marca de tempo em todo ambiente de simulação. O PL deve ficar bloqueado enquanto não possuir o evento com a menor marca de tempo, esperando que o evento no topo de sua *LEF* seja o escolhido e possa ser executado. A comunicação entre os diversos PLs deve ser realizada ponto-a-ponto, através de canais previamente definidos (topologia estática), com *buffers* associados, para armazenamento de mensagens recebidas e enviadas. Essa configuração provoca situações em que todos os processos fiquem bloqueados (*deadlock*), sem que se saiba qual deles deve seguir a execução [Mis86].

O problema do *deadlock* nesse protocolo é demonstrado na Figura 3.3. Nesse exemplo, mesmo existindo mensagens aguardando para serem processadas em outras filas de entrada de cada processo, os três processos estão bloqueados, pois cada um espera indefinidamente a chegada de uma mensagem no canal de entrada que contém o menor valor de relógio, uma vez que a fila desse canal está vazia. [Fuj00].

As mensagens que chegam a cada canal de entrada são armazenadas em ordem FIFO, que também é a ordem das marcas de tempo, para determinar quando é seguro processar um evento. Cada canal possui um relógio que armazena o valor da marca de tempo

da mensagem que está no início da fila. Se a fila estiver vazia, o valor do relógio será o valor da marca de tempo da última mensagem recebida, já no caso onde nenhuma mensagem foi recebida pelo canal o valor do relógio é zero. O menor valor entre os relógios dos canais de comunicação utilizados pelo processo é denominado *LVTH* (*Local Virtual Time Horizon*).

O valor do relógio da simulação *GVT* é o menor valor do relógio de todos os processos lógicos. Pode-se dizer que, em um determinado ponto da simulação, o sistema físico tenha sido simulado até o tempo indicado por *GVT*, mesmo que algum processo lógico tenha simulado seu processo físico correspondente até um tempo maior que *GVT*.

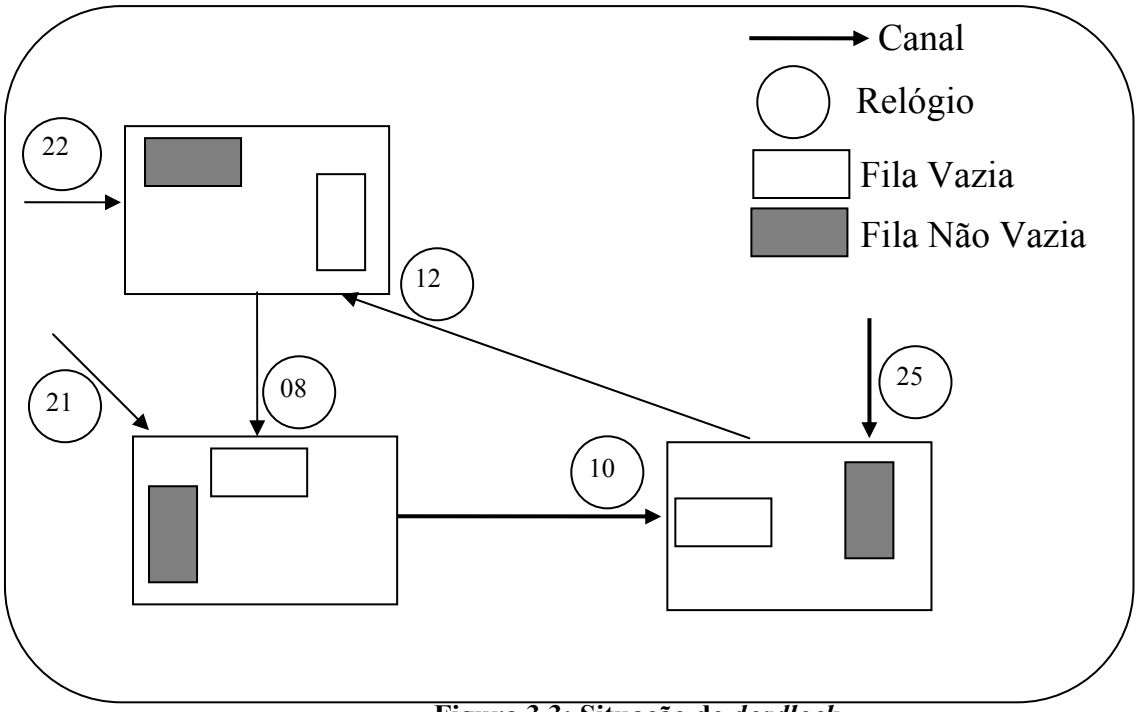


Figura 3.3: Situação de *deadlock*.

3.3.1 Prevenção de *Deadlock* Usando Mensagens Nulas

Uma solução para eliminar ou prevenir a ocorrência de situações de *deadlock* é o envio de mensagens nulas entre os processos [Mis86]. As mensagens nulas servem somente para sincronização, não correspondendo a qualquer atividade do sistema físico. Uma mensagem nula com marca de tempo *T*, enviada de PL_1 para PL_2, indica que PL_1 não irá enviar mensagens a PL_2 com marca de tempo entre o valor corrente do relógio e o tempo T. Por essa razão, qualquer mensagem futura de PL_1 para PL_2 terá a marca de tempo superior a T.

Quando o processo PL_1 recebe uma mensagem, atualiza seu *LVT* e avança a simulação até o valor de *LVT*, enviando esse novo valor para os processos com os quais se comunica. Dessa forma, PL_1 pode predizer que, após a transmissão dessas mensagens, o processo físico correspondente a PL_1 não irá enviar mensagens ao processo PL_2 até o tempo T_2.

Uma vez que PL_1 conhece o estado do processo físico correspondente até o tempo LVT_1, pode predizer todas as mensagens no mínimo até o tempo LVT_1. As mensagens nulas são tratadas como qualquer mensagem quando recebidas por um processo, ou seja, a recepção de uma mensagem nula faz com que o processo atualize seu *LVT* e, possivelmente, envie novas mensagens [Mis86].

Uma desvantagem dessa abordagem é a sobrecarga provocada pelo envio de mensagens nulas após a execução de cada evento. O envio de mensagens nulas sob demanda [Cha81] é uma técnica desenvolvida para diminuir a quantidade dessas mensagens.

3.3.2 Mensagens Nulas Sob Demanda

A freqüência da demanda no envio de mensagens nulas sob demanda pode ser dada por um *timeout* ou também quando o menor relógio dos canais for referente a um canal onde a fila é vazia, indicando que o processo está bloqueado. Quando isso ocorre, a próxima mensagem é requisitada (pode ser uma mensagem nula ou não) do processo que envia para esse canal. O processo continua sua execução quando a resposta a esse pedido for recebida.

Suponha que um processo PL_1 requisita a um processo PL_2 que aumente o valor do relógio do canal que conecta os dois processos. Esse avanço é possível se o valor do relógio desse canal for menor que o valor atual do *LVT* de PL_2 (mínimo entre os relógios dos canais de entrada de PL_2). Nesse caso, PL_2 envia uma mensagem avançando o relógio do canal (PL_2, PL_1). A Figura 3.4 ilustra o exemplo, onde o processo PL_2 pode enviar uma mensagem nula ao processo PL_1 incrementando o relógio do canal (PL_2, PL_1) para 12. Com isso, o processo PL_1 pode computar o novo valor de seu *LVT* (igual a 9) e prosseguir sua execução.

Porém, se o valor do relógio do canal (PL_2, PL_1) for igual ao valor do *LVT* de PL_2, nenhum avanço será possível, como mostra a Figura 3.5. O processo PL_2 tem que avançar seu próprio *LVT*, fazendo o mesmo tipo de pedido a todos os processos que comunicam com PL_2, cujos relógios dos canais sejam iguais ao *LVT* de PL_2**.** Esse é o caso do canal de entrada 2 do processo PL_2. Nesse caso, o pedido será efetivamente propagado por PL_2.

O processo PL_2 só poderá enviar uma mensagem para PL_1 incrementando o valor do relógio do canal, quando o valor do *LVT* de PL_2 for maior que o valor do relógio do canal (PL_2, PL_1). As propagações dos pedidos de avanço do relógio podem formar um ciclo, gerando uma situação de *deadlock*.

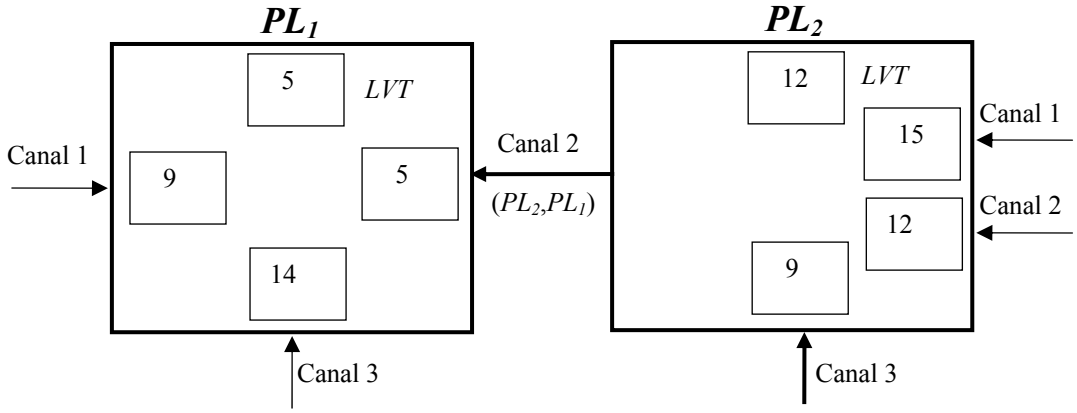


Figura 3.4: Transmissão de mensagens nulas sob demanda (a).

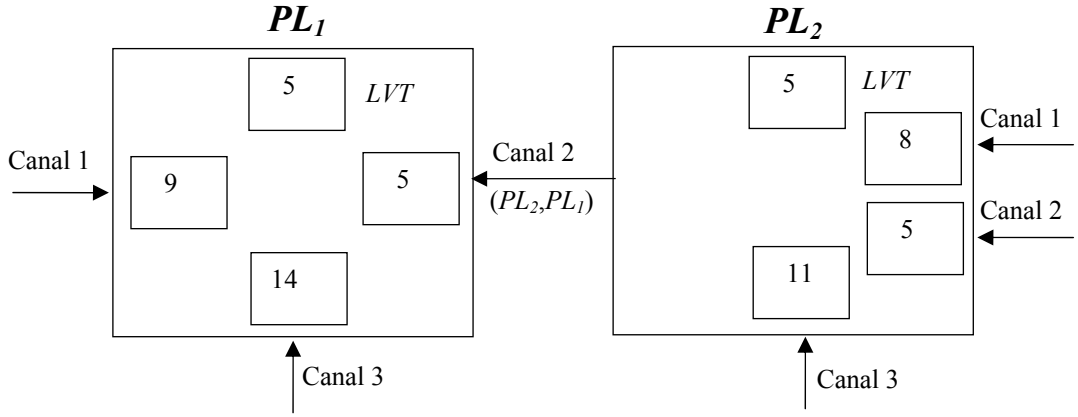


Figura 3.5: Transmissão de mensagens nulas sob demanda (b).

3.3.3 Detecção e Recuperação de *Deadlock*

Em situações em que a ocorrência de *deadlock* é rara têm-se uma sobrecarga de envio de mensagens nulas. Chandy e Misra [Cha81] desenvolveram um protocolo alternativo que elimina o uso dessas mensagens. Essa abordagem utiliza um mecanismo de detecção e outro de recuperação de *deadlock*.

Esse protocolo utiliza um marcador que circula continuamente pela rede de processos lógicos, devendo percorrer todos os canais da rede, em algum tempo durante um ciclo. Esse marcador é apenas um tipo especial de mensagem. Quando um processo recebe o marcador,

deve enviá-lo para sua rota designada, depois de certo tempo em que o processo se tornou ocioso (sem mensagens para enviar).

Cada processo possui um *flag* de 1 bit que indica se o mesmo recebeu ou enviou uma mensagem desde a última vez que recebeu o marcador. Um processo é branco se não recebeu nem enviou mensagens desde a última passagem do marcador, caso contrário, o processo é preto. Inicialmente, todos os processos são pretos. A situação de *deadlock* é detectada quando os *n* últimos processos visitados pelo marcador estejam brancos (*n* é o número de canais na rede).

O marcador armazena o número de processos brancos que visitou desde o último processo preto encontrado, além de conter informações sobre os próximos tempos de eventos dos processos brancos que visitou. Quando o marcador detecta o *deadlock*, o processo pode ser reiniciado, pois o marcador possui indicações do próximo tempo de evento e do processo em que ocorreu o *deadlock* [Mis86, Fuj90].

3.3.4 *Lookahead*

O conceito de *lookahead*, desde que foi introduzido por Chandy e Misra [Cha79], tem sido interpretado de diferentes maneiras. A inclusão do *lookahead* (intervalo de tempo em que os processos não podem agendar novos eventos) foi uma das alterações no protocolo CMB original que proporcionou redução do tempo de execução do protocolo. Misra [Mis86] define *lookahead* como sendo um intervalo de tempo em que um processo pode inferir o futuro com absoluta certeza. Já Fujimoto [Fuj90] define *lookahead* como a capacidade do processo em prever o que acontecerá, ou mais importante, o que não acontecerá, no tempo de simulação futuro. Se um processo no tempo de simulação T pode prever com segurança todos os eventos que ele poderá gerar até o tempo de simulação $T + L$, pode-se dizer que o processo tem *lookahead* L [Bru03].

Lookahead é uma técnica fundamental da qual os processos conservativos fazem uso, a qual cria virtualmente uma barreira no tempo. Antes dessa barreira, os processos podem ser livremente executados sem preocupação com a ocorrência de erros de causa e efeito. A definição do valor dessa fatia de tempo varia de uma aplicação para outra, devendo ser cuidadosamente definida a fim de não atribuir um intervalo muito pequeno ocasionando muitas sincronizações e, conseqüentemente, baixo desempenho ou muito grande, causando computações errôneas ou incoerência com o modelo real simulado [Bag99].

O valor do *lookahead* pode ser calculado a cada sincronização através de fórmulas matemáticas [Laz90].

Uma outra abordagem, adotada neste trabalho, é a utilização de um mesmo *lookahead* em todos os PL*s*, dado que simulam o mesmo mundo real [Por97]. Uma vez que os sistemas do mundo real podem ser totalmente diferentes em comportamento, o valor do *lookahead* deve ser escolhido de acordo com o sistema a ser simulado.

Para melhor compreensão do conceito de *lookahead* pode-se considerar um exemplo de um sistema simples de fila, ilustrado na Figura 3.6, no qual as tarefas chegam através de um canal de entrada com uma fila que obedece a disciplina *FIFO*. As tarefas utilizam o Servudor, cujo tempo de serviço pode ser obtido através de uma distribuição exponencial [Mor00].

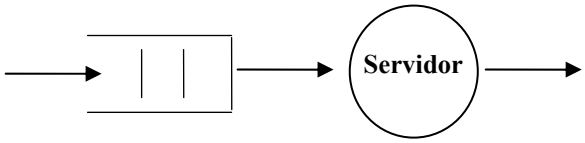


Figura 3.6: Sistema de fila FIFO.

Servidor

Como mostra a Tabela 3.1, a tarefa A com tempo de duração de 12 unidades de tempo inicia seu processamento no tempo 30. As tarefas B, C e D aguardam na fila para serem processadas, com tempos de serviço s iguais a, respectivamente, 3, 5 e 11. No tempo 32 chega uma nova tarefa (E) ao sistema com tempo de serviço 9. A Tabela 3.1 mostra os tempos de ocorrência, de duração e de término do processamento de cada uma das tarefas.

Tabela 3.1: Tempos de Serviço.

Tarefa	Inicio	Duração	Término
A	30	12	42
B	42	3	45
C	45	5	50
D	50	11	61
E	61	9	70

Através do comportamento desse processo lógico simulando o sistema de fila em relação à tarefa E, podem-se discutir questões associadas ao *lookahead.* A Figura 3.7 mostra a situação do sistema no instante em que a tarefa *E* é recebida (T = 32). Após o processo lógico

atingir o tempo de simulação 32, o próximo evento a ser processado é o tratamento da chegada da mensagem associada com a tarefa *E*.

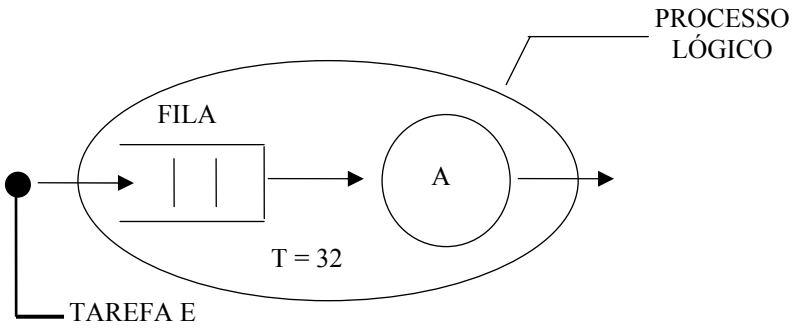


Figura 3.7: Processo lógico do exemplo de fila FIFO.

Uma vez que o processo já tenha avançado para o tempo 32 e conhece os valores do tempo de execução para as tarefas que chegaram anteriormente, é possível calcular a marca de tempo com valor 70 como o término do processamento da tarefa *E* [Mor00].

Segundo Fujimoto [Fuj00], uma desvantagem do protocolo conservativo é que ele não explora totalmente o paralelismo disponível na simulação. No pior caso, o protocolo conservativo é pessimista, exigindo uma execução seqüencial quando a mesma não é necessária. Convém destacar que o desempenho dos protocolos conservativos está diretamente ligado à capacidade de "olhar à frente" no tempo de simulação futuro.

3.4 Linguagens e Ambientes para Simulação Distribuída

O principal atrativo na utilização da simulação distribuída é a redução do tempo de processamento. A tarefa de desenvolvimento dos programas não é trivial, pois exige conhecimentos do sistema a ser modelado, das técnicas de modelagem e dos paradigmas de programação paralela e/ou distribuída a serem utilizados. O uso da simulação distribuída é dificultado pela falta de ferramentas adequadas para o seu desenvolvimento.

Para motivar a adoção do paradigma de simulação distribuída, têm sido desenvolvidas diversas linguagens e bibliotecas. A área de linguagens e ambientes de simulação paralelos está em desenvolvimento. Muitos estudos devem ser feitos para o projeto de ferramentas paralelas de propósito geral, sua aplicação em domínios específicos, e na

identificação de características da aplicação que possam ser exploradas por protocolos de sincronização específicos [Spo01].

As linguagens de simulação paralelas diferem no modo como fornecem suporte aos protocolos de sincronização. Muitas linguagens fornecem suporte somente a protocolos otimistas como :

- *ModsimII* : é uma linguagem orientada a objetos baseada na linguagem Modula [Ric91];
- *Apostle* (A Parallel Objetc-Oriented Simulation Language): é uma linguagem baseada no protocolo *Time Warp* [Won96].

Outras linguagens podem ser utilizadas tanto em protocolos otimistas como conservativos [Bag94a]:

- *Maise*: implementa vários protocolos conservativos e otimistas [Bag94a]*;*
- *Moose* (*Maise-based Object-Oriented Simulation Environment*): é uma versão orientada a objetos da *Maise* [Fis96]
- *Parsec* (*Parallel Simulation Environment for Complex Systems*): é uma evolução da linguagem *Maise* [Bag98];
- *YADDES*: linguagem que implementa diversos protocolos de sincronização [Gho96].

Uma biblioteca (ou extensão funcional) para simulação distribuída fornece apenas o conjunto de primitivas que devem ser utilizadas em conjunto com a linguagem de programação hospedeira e que possibilitam o desenvolvimento do programa de simulação. Como exemplo de extensões funcionais tem-se [Bag99]:

- *ParSMPL* : é uma ferramenta de simulação distribuída que utiliza o protocolo CMB para efetuar a sincronização entre os processos lógicos, desenvolvida no Grupo de Sistemas Distribuídos e Programação Concorrente do ICMC-USP. Essa ferramenta é baseada no SMPL, uma extensão funcional da linguagem C e executa em ambiente UNIX [Uls99];

- *Parasol, PSK, Simkis*, *SPDDES* e *WARPED* : são bibliotecas acadêmicas orientadas a objeto desenvolvidas em C++, que implementam protocolos otimistas [Low99, Ril03];

- *GTW e TWOS* : bibliotecas que utilizam a linguagem C e implementam o protocolo otimista *Time Warp* [Car00].

De acordo com Bruschi [Bru03], ambientes automáticos para simulação distribuída são ambientes constituídos de editores gráficos, interpretadores, compiladores e otimizadores, ou seja, o usuário não precisa se preocupar com os protocolos de sincronização, pois com apenas a representação gráfica do modelo o programa de simulação é gerado automaticamente e os resultados obtidos são ilustrados através de gráficos e/ou animações. Ainda há poucos ambientes automáticos para simulação distribuída, dentre eles pode-se citar *ASDA* [Bru03], *UCLA* [Bag94b], Akaroa 2 [Ewi99] e *DISplay* [Mas95].

3.5 Considerações Finais

Neste capítulo foram apresentados os principais conceitos relacionados à simulação distribuída, a qual tem atraído interesse considerável nos últimos anos devido ao crescimento da utilização da simulação em sistemas de grande porte. O grande problema da simulação distribuída consiste em lidar com a ocorrência de erros de causa e efeito. Os protocolos conservativos e otimistas solucionam este problema a partir de diferentes abordagens, os conservativos utilizam a técnica de bloquear os processos que na possuem eventos seguros para serem processados, pois pode levar a situação de *deadlock,* enquanto os protocolos otimistas permitem a situação de *deadlock* e utilizam um mecanismo de recuperação.

O estudo efetuado da literatura sobre o protocolo CMB original e sua variante com mensagens nulas sob demanda permitem a implementação de uma ferramenta para simulação de modelo de filas, denominada CMB-*Simulation*. O Capítulo 4 descreve a funcionalidade dessa ferramenta, apresentando as estruturas dos processos e todos os aspectos relacionados ao projeto e implementação dessa ferramenta.

Capítulo 4

Projeto e Implementação da Ferramenta CMB-*Simulation*

4.1 Considerações Iniciais

O projeto e implementação da ferramenta CMB-*Simulation* foram desenvolvidos com base no protocolo conservativo CMB original e a variante com mensagens nulas e sob demanda, implementada na linguagem C, em plataformas *IBM-PC* com sistema operacional Linux, o qual utiliza as primitivas não bloqueantes do LAM-MPI (*Message Passing Interface*) como ambiente de troca de mensagens.

O MPI é um padrão para os ambientes de troca de mensagens com plataforma portáteis. O padrão MPI tenta definir a sintaxe e semântica de uma biblioteca de um conjunto de rotinas que, utilizando a linguagem C++ ou Fortran, podem ser empregadas no desenvolvimento de aplicações paralelas executadas em arquiteturas MIMD (*Multiple Instruction, Multiple Data)*. A principal vantagem de estabelecer um padrão é a portabilidade, pois o MPI foi desenvolvido por um comitê MPI (*commitee*) composto por cerca de oitenta pesquisadores representando mais de quarenta organizações que trabalham com processamento paralelo em todo mundo, especialmente na Europa e Estados Unidos [Gro99]. Foi desenvolvido com o intuito de atingir os seguintes objetivos [Lam04]:

- Portabilidade do código fonte, ou seja, o código pode ser utilizado em outros ambientes heterogêneos sem alterações;
- Possibilidade de ser utilizado tanto em sistemas com memória distribuída e compartilhada;

- Prover uma interface de comunicação confiável, de maneira que o usuário não precise se preocupar com falhas na comunicação.

A implementação da CMB-*Simulation* é composta por um conjunto de processos, onde cada é representado por um processo lógico. Como o protocolo CMB não permite a criação dinâmica de processos, os mesmos processos são criados no início da execução e destruídos no final da execução da simulação.

Outro conceito importante na simulação é a análise dos resultados obtidos com uso da CMB-*Simulation*. Essa análise consiste em determinar se os resultados obtidos com a simulação distribuída através da CMB-*Simulation* correspondem aos resultados esperados. Essa análise é realizada com base na comparação dos resultados obtidos na simulação distribuída com os resultados obtidos com a aplicação da simulação seqüencial usando SMPL.

A Seção 4.2 apresenta a descrição das rotinas implementadas na ferramenta CMB-*Simulation*. Na Seção 4.3 é feita uma análise do comportamento da simulação distribuída usando a CMB-*Simulation*, com base em dois modelos. Na Seção 4.4 é exemplificado o funcionamento da CMB-*Simulation* na abordagem com troca de mensagens nulas e sob demanda. A Seção 4.5 apresenta as considerações finais do capítulo.

4.2 Estrutura e Implementação da CMB-*Simulation*

A implementação da simulação distribuída é composta por um conjunto de processos, cada um representando um processo lógico. Todos os processos são criados no início da execução da simulação e destruídos no final da mesma, ou seja, não existe a criação dinâmica de processos. Cada um dos processos que compõem a simulação possui a estrutura interna apresentada na Figura 4.1, na qual a criação de todas as estruturas de dados é executada automaticamente pela CMB-*Simulation.*

Cada processo possui um número pré-determinado de canais bidirecionais, através dos quais as mensagens são recebidas e enviadas. De acordo com o particionamento do modelo entre os processos, cada processo possui no mínimo um recurso (*Facility*) com um número pré-determinado de servidores (*Servidores da Facility*).

Todo processo possui um vetor CC de tamanho *n*, onde *n* é o número de processos que constituem a simulação. Cada posição (0 a *n*) desse vetor guarda o tempo da última mensagem recebida do processo *i (*relógio do canal *i*).

As interações entre os processos são realizadas através de troca de mensagens. Cada processo possui seu próprio relógio local (*LVT*), e o *LVTH* que é o mínimo entre os relógios dos canais. O *LVT* controla o tempo de simulação, e uma lista de eventos na qual ficam armazenadas (em ordem cronológica) as mensagens recebidas de outros processos.

Inicialmente, o processo verifica se há alguma mensagem nos canais de comunicação para serem armazenadas no *buffer* do canal. Em seguida, verifica se há algum evento na *LEF* a ser executado, em caso afirmativo esse evento é retirado da fila. Caso contrário, a mensagem do canal com menor relógio é retirada. Em ambos os casos o *LVT* é atualizado e propagado através da troca de mensagens para os demais processos da simulação.

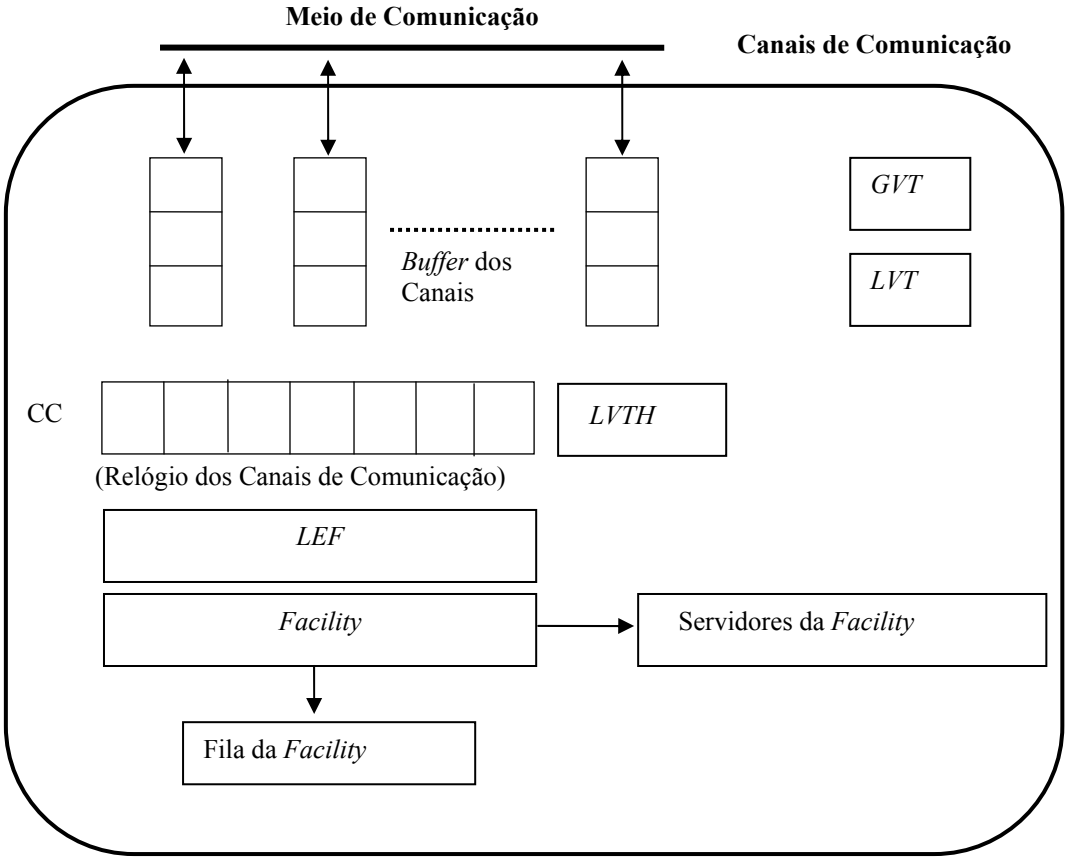


Figura 4.1: Estrutura do processo na CMB-*Simulation.*

Nas subseções seguintes são apresentadas as rotinas implementadas na CMB-*Simulation* com troca de mensagens nulas e sob demanda, sendo as rotinas comuns especificadas apenas na primeira subseção.

4.2.1 CMB-*Simulation* com Troca de Mensagens Nulas

A implementação da CMB-*Simulation* com troca de mensagens nulas é constituída pelos módulos ilustrados na Figura 4.2, que têm a função de iniciar as primitivas do MPI, iniciar as variáveis de simulação, estabelecer o controle da simulação por tempo ou por ciclo de tarefas, definir os recursos, o número de processos, os canais de comunicação e o valor do *lookahead.*

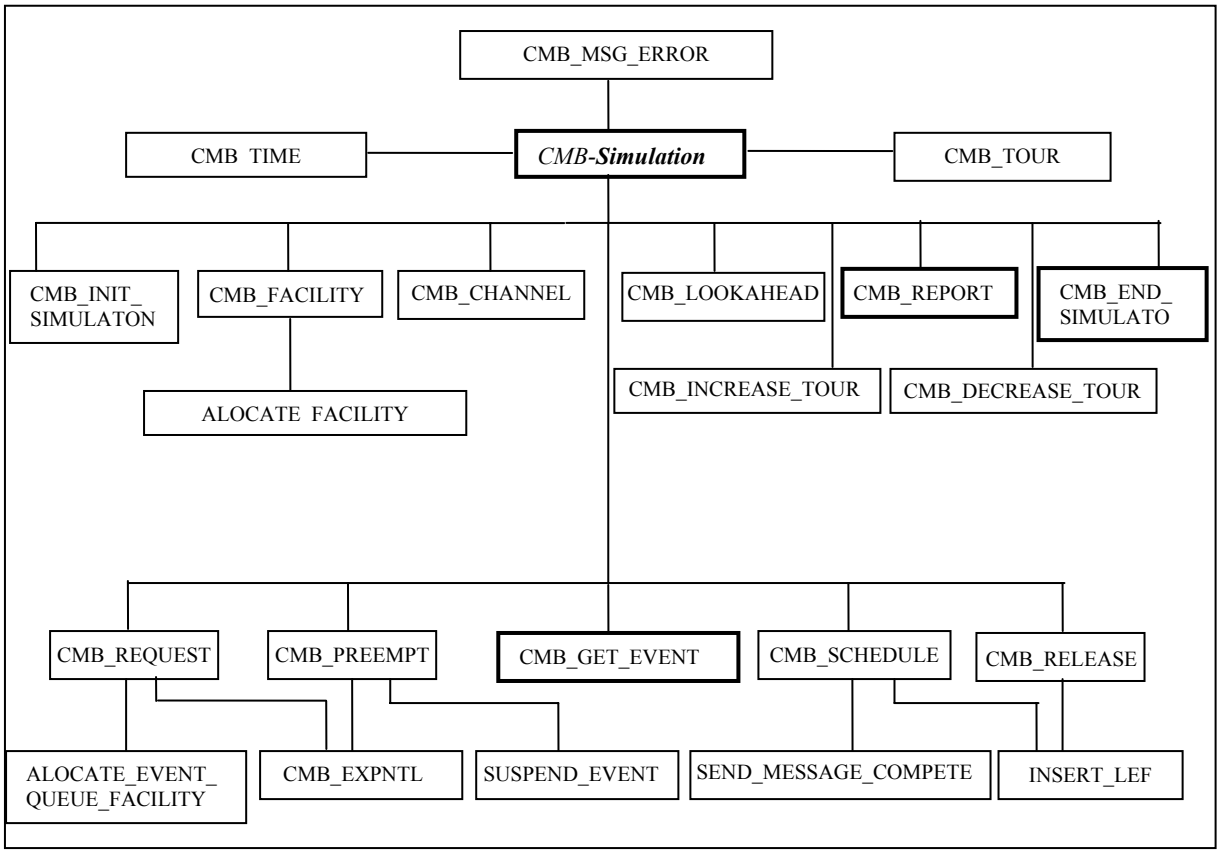


Figura 4.2: Estrutura de rotinas da CMB-*Simulation.*

CMB_INIT_SIMULATION

Responsável por iniciar o sistema para execução da simulação. Inicia as primitivas do *MPI* e atribui valores iniciais às estruturas de dados e ao *LVT*.

CMB_FACILITY

Aloca os recursos para cada processo. O número de servidores para cada recurso é especificado como parâmetro e realizado através da função ALOCA_SERVER_FACILITY.

ALOCATE _SERVER_ FACILITY

Aloca os servidores para o recurso do processo.

CMB_CHANNEL_ PROCESS

Estabelece canais bidirecionais de comunicação entre os processos especificados e atribui o tempo zero ao relógio desses canais.

CMB_LOOKAHEAD

Atribui o valor recebido como parâmetro ao *lookahead* do modelo.

CMB_SCHEDULE

Verifica se o evento deve ser escalonado localmente ou em outro processo (escalonamento remoto). Se o evento deve ser executado localmente, o mesmo é inserido na *LEF* do processo, caso contrário, as informações necessárias para execução desse evento são enviadas ao processo destino, através do envio de uma mensagem completa.

CMB_TIME

Retorna o *LVT* atual do processo.

CMB_REQUEST

Requisita o atendimento de um dos servidores do recurso. Se um servidor estiver livre, o mesmo é reservado para o *token* solicitante. Se não existir servidor disponível, o valor um é retornado, indicando que o recurso está ocupado. Nesse caso, a requisição é inserida na fila do recurso.

CMB_PREEMPT

Requisita o atendimento de um dos servidores da *facility*. Se houver servidor livre, o mesmo é reservado para o *token* solicitante. Se não existir servidor disponível, verifica-se se a prioridade do *token* solicitante é maior do que a maior prioridade dentre todos *tokens* nos

servidores ocupados. Em caso afirmativo, o *token* que ocupa o servidor é sofre preempção, a liberação do servidor é suspensa para esse *token* e inserida na fila do recurso. O *token* solicitante passa a ocupar o servidor.

ALOCA_ EVENT_ QUEUE_ FACILITY

Aloca os eventos requisitantes ou *preemptados* na fila da facilidade de acordo com sua prioridade. Atualiza as estatísticas e incrementa o tamanho da fila do recurso.

SUSPEND_ EVENT

Percorre a *LEF*, elimina o evento especificado pelo *token* e o retorna.

CMB_ ERLANG

Gera um número aleatório seguindo a distribuição de *Erlang*, com média especificada por parâmetro.

CMB_ EXPNTL

Gera um número aleatório seguindo a distribuição exponencial, com média especificada por parâmetro.

CMB_STREAM

Seleciona uma semente para geração de números aleatórios.

CMB_RANF

Gera números aleatórios no intervalo de zero a um, utilizando uma distribuição uniforme.

CMB_RANDOM

Gera um número aleatório inteiro contido no intervalo cujos extremos são passados como parâmetro.

CMB_RELEASE

Libera um servidor de um determinado recurso. Após a liberação, o sistema atualiza as estatísticas e decrementa o número de servidores ocupados. Caso exista algum evento

pendente na fila do recurso, esse é inserido no início da lista de eventos futuros para execução imediata ou não.

SEND_ MESSAGE _COMPLETE

Envia uma mensagem completa para o processo destino, contendo informações sobre o evento a ser escalonado, seu tempo de ocorrência, a identificação do *token* e do processo destino.

A Figura 4.3 aborda os módulos que retiram e tratam um evento da *LEF*, além de enviar mensagens nulas e verificar se há mensagens para receber nos canais de comunicação.

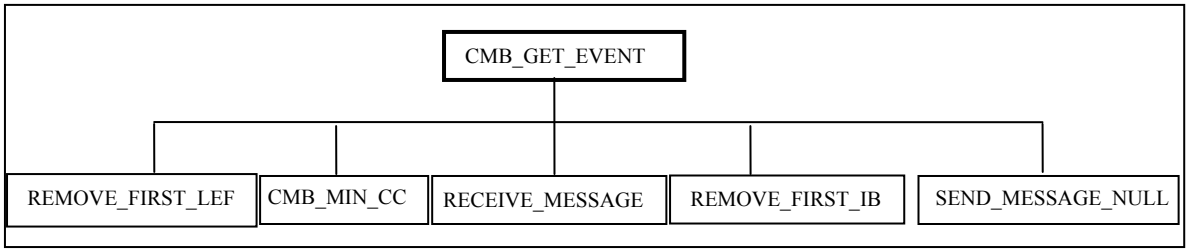


Figura 4.3: Estrutura da rotina CMB_GET_EVENT.

CMB_GET_EVENT

Acionada sempre que um evento deve ser executado. Sua execução consiste em verificar se o tempo de ocorrência do próximo evento é menor ou igual ao relógio do processo. Em caso afirmativo, remove o elemento do início da *LEF* e avança o *LVT* para o tempo de ocorrência do evento a ser executado. Caso contrário, verifica se há mensagem no *buffer* do canal com menor relógio, se houver remove essa mensagem do *buffer* do canal e atualiza *LVT*, senão o processo fica bloqueado. Caso *LVT* do processo tenha sido atualizado, o mesmo envia mensagens nulas com o tempo *LVT + lookahead* para os processos com os quais se comunica.

CMB_MIN CC

Retorna o menor tempo entre os relógios dos canais de comunicação do processo e o índice numérico referente a esse canal.

RECEIVE_ MESSAGE

Verifica se o processo tem alguma mensagem para ser recebida. de acordo com o tipo da mensagem, essa pode ser inserida na LEF ou no *Buffer* do canal, através das funções INSERT_LEF e INSERT_IB, respectivamente.

INSERT_LEF

Insere o evento na *LEF* em ordem cronológica de tempo.

INSERT_IB

Insere a mensagem recebida na estrutura de *buffer* de canais, atribuindo ao relógio do canal a marca de tempo da mensagem.

REMOVE _FIRST_ LEF

Remove o evento do início da lista de eventos futuros.

REMOVE_ FIRST_ IB

Remove a mensagem que está no início do *buffer* do canal e atualiza o relógio do mesmo.

SEND_MESSAGE_NULL

Envia mensagens nulas para os demais processos com os quais o processo se comunica.

Os módulos responsáveis pela visualização dos resultados da simulação distribuída estão ilustrados na Figura 4.4, ou seja, esses módulos informam as estatísticas calculadas, e o número de mensagens nulas e completas enviadas pelos processos.

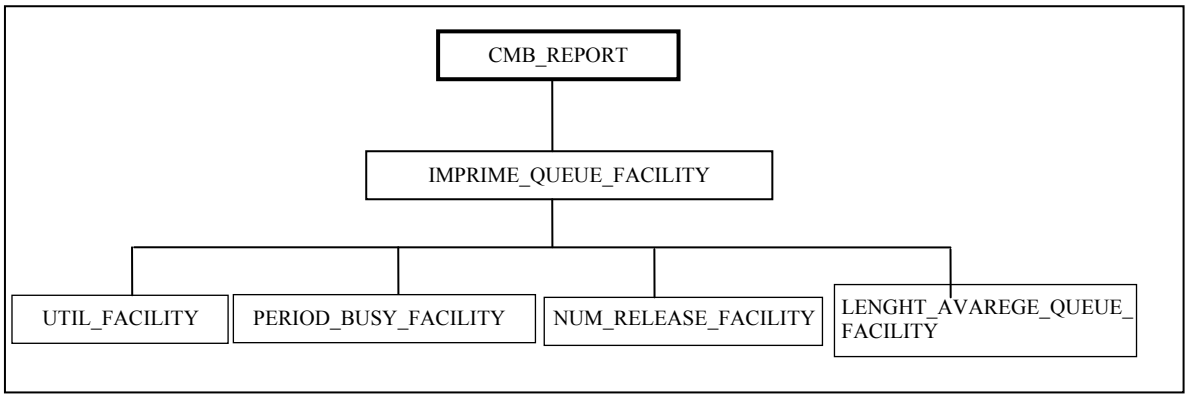


Figura 4.4: Estrutura da rotina CMB_REPORT.

CMB_ REPORT

Imprime os resultados estatísticos do modelo de simulação.

UTIL_ FACILITY

Obtém a taxa de utilização do recurso.

PERIOD_ BUSY_ FACILITY

Retorna o período em que o recurso permaneceu ocupado.

NUM_RELEASE_FACILITY

Retorna o número de liberações feitas pelo recurso.

LENGHT_AVEREGE_ QUEUE _FACILITY

Retorna o tamanho médio da fila do recurso.

A Figura 4.5 mostra os módulos responsáveis pelo término da execução da simulação, pela destruição das estruturas dos processos envolvidos na simulação e suas estruturas alocadas.

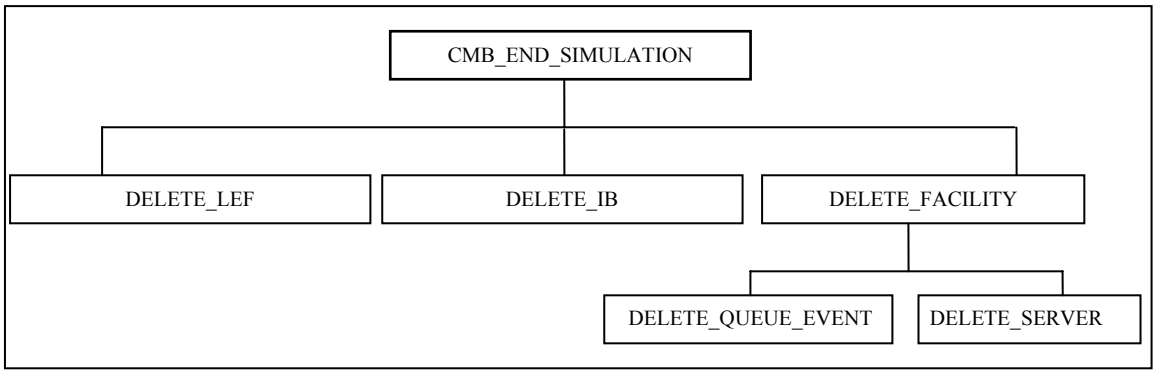


Figura 4.5: Estrutura da rotina CMB_END_SIMULATION

CMB_END_SIMULATION

Realiza a chamada das funções responsáveis pela eliminação das estruturas de dados e finaliza o *MPI*.

DELETE_LEF

Exclui a estrutura *LEF* alocada ao processo.

DELETE_IB

Exclui a estrutura de canais alocada para o processo.

DELETE_FACILITY

Exclui as facilidades alocadas ao processo.

DELETE_QUEUE_EVENT

Exclui a estrutura de fila de eventos do(s) recurso(s) do processo.

DELETE_SERVER

Exclui a estrutura da lista de servidores alocados para o(s) recurso(s) do processo.

4.2.2 CMB-*Simulation* com Mensagens Nulas sob Demanda

As rotinas utilizadas pelo CMB com mensagens nulas sob demanda são as mesmas rotinas utilizadas na abordagem com troca de mensagens nulas, descritas anteriormente. Entretanto, com a utilização da demanda, não existe *lookahead* e as rotinas CMB_GET_EVENT e RECEIVE_MESSAGE sofreram algumas modificações, havendo a necessidade de inclusão de novas rotinas. A Figura 4.6 ilustra essas alterações.

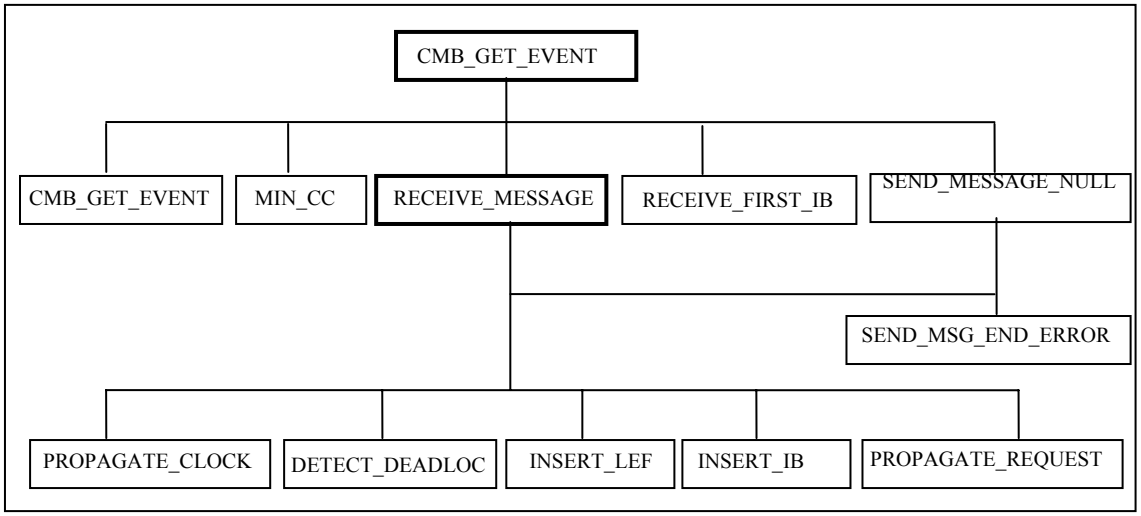


Figura 4.6: Estrutura da rotina CMB_GET_EVENT sob demanda.

CMB_GET_EVENT

Verifica se há alguma mensagem para ser recebida e se há algum evento na *LEF* com tempo menor ou igual ao *LVTH* (relógio do canal). Se houver algum evento, o mesmo é executado, caso contrário, verifica-se a existência de mensagem no canal. Se houver, essa mensagem é retirada do canal e o *LVT* é atualizado. Se não houver evento na *LEF* e mensagem no canal com valor de relógio menor, o processo envia uma mensagem de requisição para o processo relacionado a esse canal e permanece bloqueado até a chegada de uma resposta. Isso pode acarretar a situação de *deadlock*, o qual é resolvido com base em troca de mensagens e cálculo do tempo de ocorrência do próximo evento *NEXT_TIME_EVENT*, isto é, o tempo do próximo evento na simulação, e esse avanço é propagado para os demais processos.

RECEIVE_MESSAGE

Verifica a existência de mensagem nos canais de comunicação e, em caso afirmativo, recebe e trata a mensagem. Os tipos de mensagens e o tratamento dado a cada um desses tipos são:

- mensagem de erro: propagada para todos os processos para os quais há um canal de comunicação, relatando alguma inconsistência de dados, solicitando o encerramento da simulação;

- mensagem de requisição de avanço do relógio: nesse caso há duas possibilidades, de acordo com o *LVT* do processo;

  – O *LVT* do processo é maior do que o especificado pela mensagem: envia-se uma mensagem de resposta contendo o valor de *LVT*;

  – O *LVT* é menor ou igual ao avanço especificado pela mensagem: verifica se há ocorrência de *deadlock*. Se ocorrer, calcula-se o tempo de ocorrência do próximo evento (*NEXT_TIME_EVENT*) e envia-se uma mensagem de resposta para o processo emissor da mensagem. Não havendo *deadlock*, a mensagem é armazenada no *buffer* do canal e a requisição é propagada;

- mensagem de término: envia outras mensagens de término para os processos com os quais se comunica e o canal de comunicação com o processo origem é anulado;

- mensagem completa: armazena-se a mensagem na *LEF*.

O recebimento de qualquer tipo de mensagem implica no desbloqueio do processo receptor, com exceção da mensagem de requisição de avanço do relógio do canal.

PROPAGATE_REQUEST

Solicita o envio de uma mensagem nula de requisição de avanço do relógio do canal para todos os processos que possuam relógio do canal igual ao *LVT* do processo.

PROPAGATE_CLOCK

Solicita o envio de uma mensagem nula de resposta, contendo o avanço do *LVT* da simulação, para o processo informado por parâmetro.

SEND_MSG_ NULL_END_ERROR

Envia mensagem nula (de erro ou término) para os processos com os quais o processo se comunica.

SEND_MSG_NULL

Envia mensagem nula, para o processo especificado por parâmetro, requisitando um avanço ou contendo o avanço do relógio do canal.

NEXT_TIME_EVENT

Determina o tempo do próximo evento a ser executado, com base nos relógios dos canais e na *LEF* dos processos.

4.3. Análise de Comportamento da CMB-*Simulation*

Para realizar a análise de comportamento da ferramenta CMB-*Simulation* com troca de mensagens nulas e por demanda foram realizadas diversas execuções nos modelos MM1 e Sistema Computacional Simplificado, descritos nas Seções 4.4.1 e 4.4.2, respectivamente. Todos os tempos gerados e obtidos na simulação seqüencial e distribuída são em unidades de segundos. Os resultados obtidos com a execução da CMB-*Simulation* foram comparados com os resultados obtidos na simulação seqüencial usando SMPL. Para propósitos de validação, a cada execução um conjunto distinto de sementes foi pré-fixado para a geração de números aleatórios.

4.3.1 Modelo Sistema de Fila MM1

Esse modelo constitui uma fila M/M/1 básica, com um servidor, que atende vários clientes que chegam ao sistema, como mostra a Figura 4.7. Os clientes requisitam a utilização do servidor e após o término do atendimento deixam o sistema.

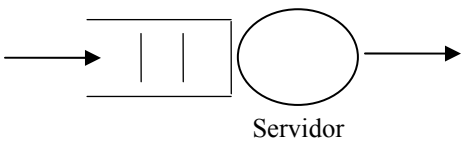


Figura 4.7: Modelo de filas M/M/1.

Como esse modelo possui apenas um recurso, ele foi particionado em um processo lógico PL_0, como mostra a Figura 4.8.

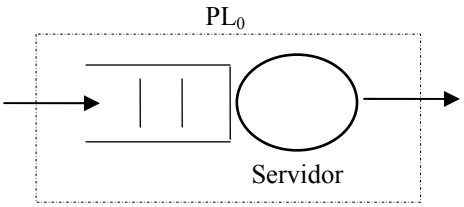


Figura 4.8: Particionamento do modelo de filas M/M/1.

A implementação do sistema de Filas MM1 na versão seqüencial em SMPL, é ilustrada na Figura 4.9. Na linha 1 a biblioteca "smpl.h" é chamada, na linha quatro os parâmetros de entrada são definidos: tempo entre chegadas igual a 1,0, tempo de serviço do servidor igual a 2,0 e o tempo de simulação igual a 10.000. Na linha seis inicia-se a simulação, na linha oito declara-se a facilidade do modelo, que é o servidor, e na linha seguinte escalona-se o primeiro evento igual a 1 com tempo zero.

```
1 #include "smpl.h"
2 main()
3 {
4  real tc = 1.0,ts = 2.0, te = 10000;
5 int  Customer = 1,Token,Event,Servidor;
6 smpl(0, "Modelo M/M/1 ");
7 Servidor  = facility("Servidor", 1);
8 schedule(1, 0.0, Customer);
9 while(TIME() <= te)
10{
11   cause(&Event, &Token);
12   switch(Event)
13   {
14    case 1:  /* Chegada de cliente */
15        schedule(2, 0.0, Token);
16        schedule(1, expntl(tc,0), ++Customer);
17        break;
18    case 2:  /* Requisição do servidor */
19        if(request(Servidor, Token, 0) == 0)
20         schedule(3,expntl(ts,0), Token);
21         break;
22    case 3:  /* Fim de servico */
23         release(Servidor, Token);
24        break;
25  }
26 }
27 report();
28 }
```

Figura 4.9: Solução SMPL para o modelo de filas M/M/1.

O laço da linha nove controla a execução da simulação até que o tempo de execução do evento seja maior que o tempo de simulação pré-definido. Nas linhas quatorze, dezoito e vinte e dois, são tratados os três tipos de eventos, respectivamente:

- evento 1 : chegada de cliente;
- evento 2 : requisição do servidor;
- evento 3 : para liberação do servidor.

A rotina que apresenta dos dados estatísticos coletados durante a execução da simulação é chamada na linha vinte e sete.

Com o intuito de facilitar a utilização da CMB-*Simulation*, as suas rotinas foram definidas seguindo o padrão utilizado no SMPL, onde foram acrescentadas as rotinas CMB_LOOKAHEAD, CMB_CHANNEL e CMB_END_SIMULATION. A rotina CMB_INIT_SIMULATION é equivalente á rotina smpl do SMPL, só que acrescida de cinco parâmetros. Esses parâmetros informam o nome do programa fonte, o nome do programa destino (arquivo para guardar os resultados estatísticos da simulação), o número de processos envolvidos na simulação, o número de ciclos a serem executados e o tempo estipulado para execução da simulação, respectivamente, além de iniciar as primitivas do MPI.

As rotinas CMB_REQUEST e CMB_SCHEDULE se diferem das rotinas do SMPL apenas em um parâmetro acrescido, o qual informa a identificação do processo, no qual o evento deve ser escalonado ou recurso em questão está alocado. O uso das rotinas da ferramenta CMB-*Simulation* para o modelo descrito acima é exemplificado na Figura 4.10, onde na linha 16 tem-se uma função CMB_SCHEDULE (0, 2, 0.0, *token*), que escalona para o processo 0 um evento 2 com tempo 0.0 segundos, para o *token* em questão.

Como o modelo sistema de fila MM1 tem apenas um recurso, a inserção das rotinas CMB_LOOKAHEAD e CMB_CHANNEL não foi necessária. Devido à ausência de comunicação, a simulação é constituída por apenas um processo lógico.

Para usar alguma distribuição na geração de números aleatórios ou tempo serviço deve-se incluir a biblioteca “CmbRand.h”. Para que o usuário tenha facilidade ao efetuar a adaptação da implementação da abordagem com troca de mensagens nulas para a implementação sob demanda, as rotinas utilizadas são as mesmas, sendo necessária apenas uma alteração na linha um, a substituição da biblioteca inicial “CmbNula.h” para “CmbDemanda.h”.

Na linha 7 a rotina CMB_END_SIMULATION inicia a simulação com os parâmetros que indicam o arquivo de execução, arquivo de saída, o nome do modelo a ser simulado, o número de ciclos de tarefas e o tempo de simulação. Na linha 8 o recurso servidor é declarado. Na linha 9 é escalonado o primeiro evento (chegada de cliente) com tempo zero.

No final da simulação a rotina CMB_REPORT é chamada para imprimir os dados estatísticos coletados, assim como a CMB_END_SIMULATION para encerrar a simulação.

```
1 #include "CmbNula.h"
2 #include "CmbRand.h"
3 int main(int argc, char* argv[])
4 {
5  double tc = 1.0,ts = 2.0,te = 10000.0;
6  int   Servidor,customer = 1,event,token=1;
7  CMB_INIT_SIMULATION(argc,argv,1,"Modelo M/M/1",0,te);
8  Servidor = CMB_INIT_FACILITY(0,"Servidor",1);
9  CMB_SCHEDULE(0,1,0.0,customer);
10 while(CMB_TIME() <= te)
11 {
12   CMB_GET_EVENT(event,token);
13   switch(event)
14   {
15    case 1:  /* Chegada de cliente */
16       CMB_SCHEDULE(0,2,0.0,token);
17       CMB_SCHEDULE(0,1,CMB_EXPNTL(tc,0),++customer);
18       break;
19    case 2:  /* Requisição do servidor */
20     if(CMB_REQUEST(0,Servidor,event,token,0)==0)
21       CMB_SCHEDULE(0,3,CMB_EXPNTL(ts,0),token);
22      break;
23    case 3:  /* Fim de servico */
24       CMB_RELEASE(0,Servidor,token);
25       break;
26  }
27 }
28 CMB_REPORT();
29 CMB_END_SIMULATION(0);
30 }
```

Figura 4.10: Solução na CMB-*Simulation* para o modelo de filas MM1.

Foram selecionadas, para ilustração, seis execuções do total de quinze execuções, sendo que as quinze execuções são apresentadas no apêndice B, com tempo de simulação (te) de 10.000, onde a cada execução os parâmetros de entrada como, tempo entre chegadas de clientes (tc) e o tempo de serviço do servidor (ts) foram distintos, para verificar o comportamento o modelo. Algumas médias estatísticas foram obtidas como, período médio ocupado (PMO), tamanho médio da fila (TMF), número de liberações (NL), número de preempções (NP) e número de enfileiramentos (NE) realizados pelo recurso, como mostra a Tabela 4.1. Essa tabela apresenta os resultados das execuções do modelo na versão seqüencial, que por sua vez, são idênticos aos resultados obtidos na versão distribuída com o uso da CMB-*Simulation* com troca de mensagens nulas e sob demanda, em todas as diferentes situações dos parâmetros do modelo. Por essa razão apenas uma tabela é apresentada.

Os resultados obtidos com a variação dos parâmetros de entrada, tais como o tempo entre chegadas e os tempos de serviços do servidor foram analisados. Nessa análise, constatou-se que na execução da simulação seqüencial e distribuída, o modelo comportou-se da maneira esperada, em situações como, por exemplo, no aumento do tempo entre chegadas (os clientes demoram mais para chegar ao sistema, portanto forma-se uma fila menor) ou na

diminuição do tempo de serviço, mantendo-se constantes os demais parâmetros (se o servidor é mais rápido, clientes saem do sistema mais cedo, portanto o comprimento médio da fila diminui.

Como por exemplo, pode-se analisar o comportamento da simulação quando o tempo entre chegadas aumentou de 1,0 para 1,5 (primeira e segunda execução), mantendo-se constantes os demais valores. Nesse caso, o tamanho médio da fila diminuiu, como esperado. Outro comportamento a ser constatado, foi que ao aumentar o tempo de serviço do servidor de 1,0 para 1,5 (quinta e sexta execução), onde houve o aumento significativo da utilização, do período médio ocupado, do tamanho médio ocupado, ou seja, servidor mais lento.

O comportamento do modelo da terceira para quarta execução como para da quinta para sexta execução foi o esperado, onde o tempo de serviço aumentou de 2,5 para 3,0, que ocasionou um aumento considerável no período médio em que o servidor ficou ocupado e no tamanho médio da fila do servidor, o que gerou um número menor de liberações efetuados pelo servidor.

Tabela 4.1: Resultados da simulação do modelo M/M/1.

Tempo de Execução = 10000 (s)						
1ª Execução (tc=1.0 – ts = 2.0) LVT = 10000.027 (s)						
Recurso	**Utilização**	**PMO**	**TMF**	**NL**	**NP**	**NE**
Servidor	0.9999	1.985	2544.271	5038	0	5038
2ª Execução (tc=1.5 – ts = 2.0) LVT = 10000.606 (s)						
Recurso	**Utilização**	**PMO**	**TMF**	**NL**	**NP**	**NE**
Servidor	0.9999	1.986	868.632	5034	0	5034
3ª Execução (tc=1.5 – ts = 2.5) LVT = 10003.494 (s)						
Recurso	**Utilização**	**PMO**	**TMF**	**NL**	**NP**	**NE**
Servidor	0.9996	2.511	1374.225	3983	0	3983
4ª Execução (tc=1.5 – ts = 3.0) LVT = 10004.752 (s)						
Recurso	**Utilização**	**PMO**	**TMF**	**NL**	**NP**	**NE**
Servidor	1.000	3.000	1732.225	3334	0	3334
5ª Execução (tc=2.0 – ts = 1.0) LVT = 10001.352 (s)						
Recurso	**Utilização**	**PMO**	**TMF**	**NL**	**NP**	**NE**
Servidor	0.4932	0.993	0.492	4969	0	2463
6ª Execução (tc=2.0 – ts = 1.5) LVT = 10000.191 (s)						
Recurso	**Utilização**	**PMO**	**TMF**	**NL**	**NP**	**NE**
Servidor	0.7296	1.478	1.755	4937	0	3614

4.3.2 Modelo Sistema Computacional Simplificado

Esse modelo apresenta uma visão macroscópica de um sistema computacional [Uls99], composto por uma Unidade Central de Processamento (UCP) e uma unidade de disco. Por meio dos testes efetuados com o modelo, efetuou-se a variação de diversos parâmetros, tais como tempo de execução de uma tarefa e tempo entre chegadas. O modelo de redes de filas para esse sistema computacional é apresentado na Figura 4.11.

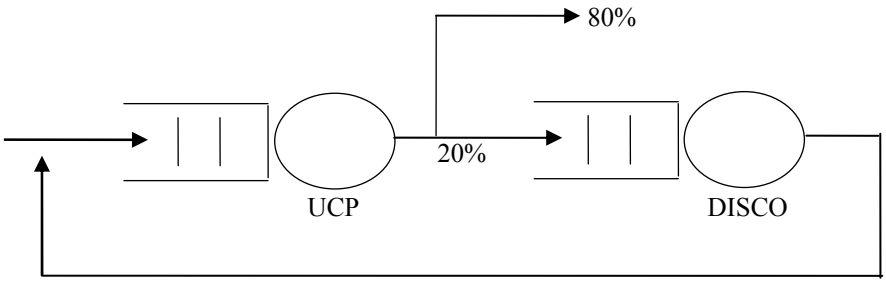


Figura 4.11: Modelo de redes de filas para o sistema computacional simplificado.

Inicialmente, uma tarefa é escalonada para a utilização da UCP; se essa não estiver disponível, a requisição é inserida na fila do recurso e, quando a UCP estiver disponível , uma requisição é retirada da fila e é executada. Após a liberação da UCP, a tarefa tem a probabilidade de 80% de deixar o sistema, e 20% de requisitar a utilização da unidade de disco. Ao liberar a unidade de disco a tarefa volta a requisitar a utilização da UCP. Os parâmetros utilizados foram definidos apenas para a realização de testes da simulação distribuída, não sendo representativos de um sistema real. O modelo foi particionado em dois processos, como mostra a Figura 4.12. A implementação desse modelo usando as rotinas do SMPL e da CMB-*Simulation* com troca de mensagens nulas é ilustrada na Figura 4.13 e na Figura 4.14, respectivamente.

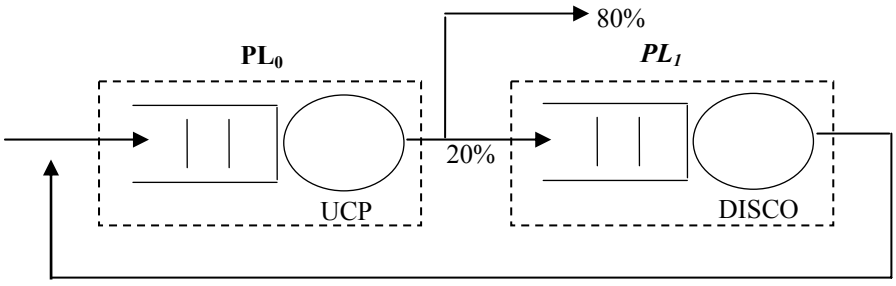


Figura 4.12: Particionamento do modelo do sistema computacional simplificado.

```
1 #include "smpl.h"
2 #define Taxa 80
3 int main(void)
4 {
5  real tc = 1.0,ts1 = 1.5,ts2 = 2.0,te = 5000.0;
6  int  customer = 1,event,token,CPU,Disco,prob;
7  smpl(0, "Sistema Computacional Simplificado");
8  CPU = facility ("CPU", 1); Disco = facility ("Disco", 1);
9  schedule(1, 0.0, customer);
10 while(TIME() <= te)
11 {cause (&event, &token);
12   switch(event)
13   {
14    case 1: /* Chegada de cliente */
15         schedule(2, 0.0, token);
16         schedule(1, expntl(tc,1), ++customer);break;
17   case 2: /* Requisição da UCP */
18       if(request(CPU, token, 0) == 0)
19         schedule(3, expntl(ts1,2), token);break;
20  case 3:  /* Fim de serviço */
21        release(CPU, token);
22        prob = random(1, 100,3);
23       if( prob >= Taxa )
24         schedule(4, 0.0, token);
25        break;
26   case 4: /* Requisição de serviço */
27         if(request(Disco, token, 0) == 0)
28          schedule(5, expntl(ts2,4), token); break;
29   case 5: /* Fim de serviço */
30         release(Disco, token);
31         schedule(2, 0.0, token);break;
32  }
33 }
34 report();
35}
```

Figura 4.13: Solução SMPL para o sistema computacional simplificado.

De acordo com a Figura 4.13, na linha 2 são definidos os seguintes parâmetros; tempo entre chegadas (tc) igual a 1,0, tempo de servido da UCP igual a 1,5, tempo de serviço do disco igual a 2,0 e o tempo de simulação igual a 5.000. Nesse modelo são tratados cinco tipos de eventos: evento 1 (chegada de um cliente), evento 2 (requisição da UCP), evento 3 (liberação da UCP), evento 4 (requisição do disco) e evento 5(liberação do disco).

Como mostra a Figura 4.14, nas linhas 9 e 10 os recursos são declarados e alocados, a UCP no processo PL_0 e o disco no processo PL_1. O modelo em questão foi particionado em dois processos lógicos, e o valor do *lookahead* é necessário para que a comunicação prossiga de acordo com o funcionamento do protocolo conservativo CMB. O uso das rotinas CMB_CHANNEL e CMB_LOOKAHEAD foi necessário, como mostram as linhas onze e doze, onde é pré-estabelecido um canal estático de comunicação entre o processo 0 e 1, e a capacidade do *lookahead* permitida pelo modelo foi de 0,0001.

```
1 #include "CmbNula.h"
2 #include "CmbRand.h"
3 #define Taxa 80
4 int main(int argc, char* argv[])
5 {
6  double tc = 1.0,ts1 = 1.5, ts2 = 2.0,te = 5000.0;
7  int Cpu,Disco,customer =1,event,token,prob;
8  CMB_INIT_SIMULATION(argc,argv,2,"Sistema Computacional Simplificado",0,te);
9  Cpu   = CMB_INIT_FACILITY(0,"CPU",1);
10 Disco = CMB_INIT_FACILITY(1,"DISCO",1);
11 CMB_CHANNEL_PROCESS(0,1);
12 CMB_LOOKAHEAD(0.001);
13 CMB_SCHEDULE(0,1,0.0,customer);
14 while(CMB_TIME() <= te)
15 {
16   CMB_GET_EVENT(event,token);
17   switch(event)
18   {
19     case 1: /* Chegada de cliente */
20       CMB_SCHEDULE(0,2,0.0,token);
21       CMB_SCHEDULE(0,1,CMB_EXPNTL(tc,1),++customer);
22       break;
24     case 2: /* Requisição da UCP */
25       if(CMB_REQUEST(0,Cpu,event,token,0)==0)
26        CMB_SCHEDULE(0,3,CMB_EXPNTL(ts1,2),token);
27       break;
28    case 3: /* Fim de serviço */
29      CMB_RELEASE(0,Cpu,token);
30      prob = CMB_RANDOM(1, 100, 3);
31      if( prob >= Taxa )
32         CMB_SCHEDULE(1,4,0.0,token);
33      break;
34    case 4: /* Requisição de serviço */
35      if( CMB_REQUEST(1,Disco, event,token, 0) == 0)
36            CMB_SCHEDULE(1,5,CMB_EXPNTL(ts2, 4), token);
37      break;
38    case 5: /* Fim de serviço */
39      CMB_RELEASE(1,Disco,token);
40      CMB_SCHEDULE(0,2,0.0,token);
41      break;
42  }
43 }
44 CMB_REPORT();
45 CMB_END_SIMULATION(0);
46 return 0;
}
```

Figura 4.14: Solução na CMB-*Simulation* para o sistema computacional simplificado.

A adaptação do código para a abordagem sob demanda é simples, sendo necessário realizar a alteração da biblioteca "CmbNula.h" para biblioteca "CmbDemanda.h" e a exclusão da rotina CMB_LOOKAHEAD, a qual não é utilizada nessa abordagem.

Na Tabela 4.2 são apresentados os resultados das execuções com base no SMPL e CMB-*Simulation* (versão com troca de mensagens nulas e na sob demanda). O Tempo de Execução (te) utilizado foi de 5.000. São utilizadas as abreviações Tempo entre Chegadas (tc), Tempo de Serviço da UCP (tucp), Tempo de Serviço do Disco (tdisco), Período Médio Ocupado (PMO), Tamanho Médio da Fila (TMF), Número de Liberações (NL), de Preempções (NP) e de Enfileiramentos (NE). Verificou-se que os valores obtidos com a execução do modelo foram iguais, tanto utilizando a execução seqüencial (SMPL), quanto à

execução distribuída utilizando a CMB-*Simulation* nas abordagens com troca de mensagens nulas e sob demanda, que justifica o uso de apenas uma tabela.

Tabela 4.2: Resultados da simulação do sistema computacional simplificado.

Tempo de Execução = 5000 (s)						
1ª Execução (tc=1.0 – tucp = 1.5 – tdisco = 2.0) LVT = 5001.478 (s)						
Recurso	**Utilização**	**PMO**	**TMF**	**NL**	**NP**	**NE**
UCP	0.9994	1.519	1222.985	3291	0	3290
DISCO	0.2767	2.035	0.116	680	0	194
2ª Execução (tc=0.5 – tucp = 1.5 – tdisco = 2.0) LVT = 5000.002 (s)						
Recurso	**Utilização**	**PMO**	**TMF**	**NL**	**NP**	**NE**
UCP	0.9997	1.519	3729.470	3291	0	3291
DISCO	0.2768	2.035	0.116	680	0	194
3ª Execução (tc=1.0 – tcpu = 1.0 – tdisco = 2.0) LVT = 5000.422 (s)						
Recurso	**Utilização**	**PMO**	**TMF**	**NL**	**NP**	**NE**
UCP	1.000	0.996	555.199	5020	0	5019
DISCO	0.4076	1.963	0.276	1038	0	408
4ª Execução (tc=1.0 – tcpu = 1.5 – tdisco = 1.5) LVT = 5001.478 (s)						
Recurso	**Utilização**	**PMO**	**TMF**	**NL**	**NP**	**NE**
UCP	0.9994	1.519	1223.106	3291	0	3290
DISCO	0.2075	1.526	0.064	680	0	144
5ª Execução (tc=1.0 – tcpu = 1.5 – tdisco = 1.0) LVT = 5001.478 (s)						
Recurso	**Utilização**	**PMO**	**TMF**	**NL**	**NP**	**NE**
UCP	0.9994	1.519	1223.211	3291	0	3290
DISCO	0.1384	1.018	0.028	680	0	102
6ª Execução (tc=1.0 – tcpu = 1.5 – tdisco = 2.5) LVT = 5001.478 (s)						
Recurso	**Utilização**	**PMO**	**TMF**	**NL**	**NP**	**NE**
UCP	0.9994	1.519	1222.846	3291	0	3290
DISCO	0.3459	2.544	0.185	680	0	228

Com o intuito de verificar o comportamento da implementação da CMB-*Simulation,* os resultados obtidos com a variação dos parâmetros de entrada, tais como o tempo entre chegadas e os tempos de serviços na UCP e nos discos, foram analisados. Nessa análise, constatou-se que a execução da simulação do modelo comportou-se da maneira esperada, conforme apresentado na tabela. Como exemplo, pode-se considerar a variação do comportamento da simulação quando o tempo de serviço do disco aumentou de 1,0 para 2,5 (quinta e sexta execução), mantendo-se constantes os demais valores. Nesse caso, o tamanho médio da fila dos discos aumentou consideravelmente, como esperado, pois os mesmos ficaram mais lentos. Conseqüentemente, suas utilizações e tamanho médio da fila da UCP aumentaram de forma significativa.

O modelo também se comportou como o esperado, da primeira para segunda execução, onde o tempo entre chegadas passou de 1,0 para 0,5 que gerou um aumento no tamanho médio da fila da CPU. Da terceira para quarta execução, houve um aumento no tempo de serviço no tempo de serviço da CPU ocasionado um aumento relevante no tamanho médio da fila, também houve uma alteração no tempo de serviço do DISCO de 2,0 para 1,5 que diminuiu o valor do período médio ocupado, do tamanho médio da fila e do número de liberações do DISCO.

O valor do período médio ocupado e o tamanho médio da fila aumentou devido a alteração do tempo serviço do DISCO de 1,0 para 2,5, conforme mostra da quinta para sexta execução na Tabela 4.2. Os testes mostraram a correspondência de resultados entre a simulação distribuída (com troca de mensagens nulas e troca de mensagens nulas sob demanda) e a simulação seqüencial. Além disso, a análise de comportamento da simulação distribuída mostrou que a solução do modelo segue o comportamento esperado. Com esses resultados considera-se que ambas as abordagens da CMB-*Simulation* estão validadas.

4.4. Funcionamento da Ferramenta CMB-*Simulation*

Para exemplificar o funcionamento da CMB-*Simulation* foi utilizado o modelo apresentado na Seção 4.4.2, considerando a partição ilustrada na Figura 4.12, ou seja, a simulação é constituída por dois processos, PL_0 e PL_1. Os parâmetros utilizados são: tempo da simulação igual a 1,5, tempo entre chegadas igual a 1,0, tempo de serviço da UCP igual a 1,5 e tempo de serviço do disco igual a 2,0. Todo processo possui o seu *LVT* e sua *LEF,* a qual é representada por um vetor, o qual possui uma seqüência de pares com a identificação e o tempo de evento, como mostra a Figura 4.15.

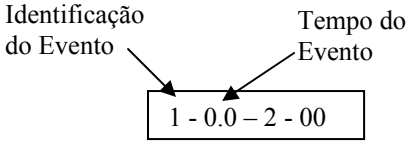


Figura 4.15: Estrutura da *LEF* no processo lógico.

O código executado por cada processo é o apresentado na Figura 4.14, e apenas para esse exemplo a geração dos tempos de chegadas e serviços utiliza tempos fixos, e não distribuição exponencial (CMB_EXPNTL). Nas subseções a seguir será ilustrado o funcionamento da CMB-*Simulation* com troca de mensagens nulas e sob demanda.

Para melhor compreender a execução da ferramenta, na Figura 4.16 são apresentado todos os eventos gerados em ordem cronológica, associados ao seu processo lógico. O primeiro evento 1 (chegada de cliente ao sistema) é escalonado com tempo 0,0, esse evento é tratado e escalona um evento 2 (requisição da UCP) e em seguida escalona o evento 1 com tempo 1, 0, referente à chegada de próximo cliente no sistema. O evento 2 é tratado e escalona uma liberação de recurso (evento 3) com tempo 1,5 o qual ao ser tratado escalona um evento remoto, ou seja, escalona um evento 4 (requisição do disco) no processo PL_1, e esse por sua vez escalona um evento 5 com tempo 3,5. O término da simulação é efetuado pelo processo PL_0 ao executar o evento 1 com tempo 2,0, pois o tempo de simulação é 1,5.

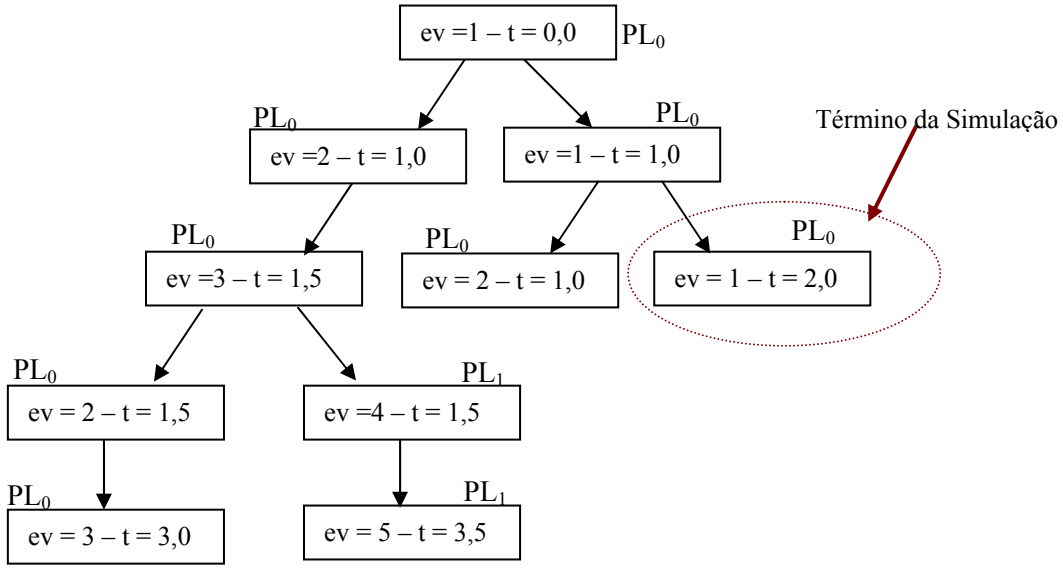


Figura 4.16: Ordem cronológica dos eventos.

4.4.1 CMB-*Simulation* com Troca de Mensagens Nulas

Nessa abordagem é necessário estabelecer o valor do *lookahead* para que a troca de mensagens entre os processos não ocasionem erros de causa e efeito. A capacidade de *lookahead* adotada foi de 0,5. Como mostra a Figura 4.17 (a), inicialmente o processo PL_0 escalona o primeiro evento com tempo 0,0. Como o *LVT* também é 0,0, o primeiro evento é

executado. Após a execução desse evento o processo envia uma mensagem nula, onde a marca de tempo é o valor do *LVT* acrescido do valor do *lookahead* (0,5), para PL_1.

No próximo passo (Figura 4.17 (b)) o processo PL_0 retira o evento da *LEF*, enquanto em paralelo o processo PL_1 recebe a mensagem e atualiza o valor do *LVT* e em seguida envia uma mensagem nula com marca de tempo *LVT* + *lookahead* (1,0) para o processo PL_0.

No próximo passo (Figura 4.17(c)) PL_0 recebe essa mensagem, atualiza o *LVT* e executa o evento da *LEF* com o tempo igual ao *LVT* (evento 1 - tempo 1,0). Em seguida envia uma mensagem nula incrementando o tempo para o processo PL_1. Esse ciclo se segue até que um dos dois processos execute um evento maior que o tempo de simulação pré-fixado, como mostra a Figura 4.17 (h).

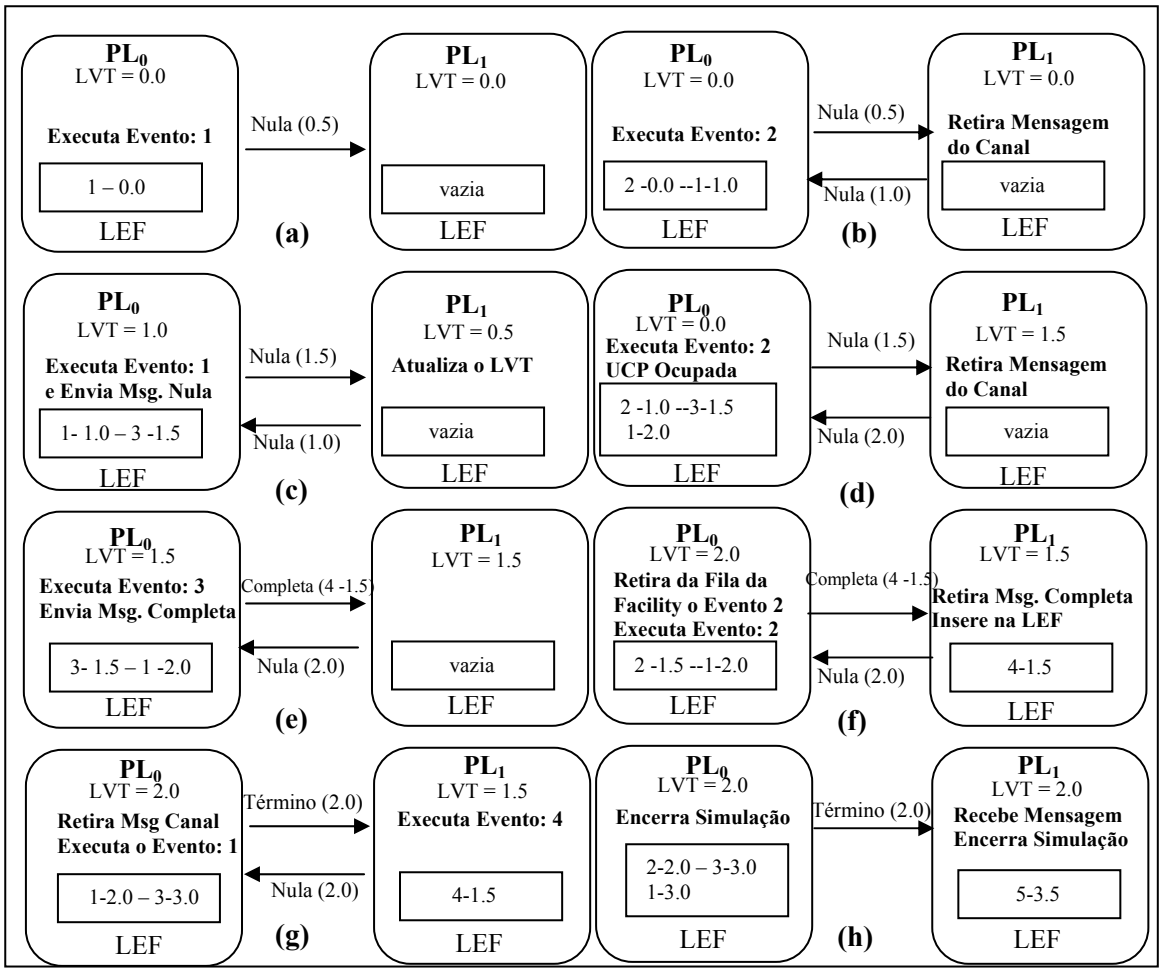


Figura 4.17: Funcionamento da CMB-*Simulation* na abordagem nula.

4.4.2 CMB-*Simulation* com Troca de Mensagens Nulas Sob Demanda

A abordagem sob demanda não utiliza a declaração do valor do *lookahead*, o ciclo de execuções é o mesmo, a diferença está na troca de mensagens, pois um processo só envia mensagem nula para requisitar, responder e/ou propagar um avanço no relógio da simulação, como mostra a Figura 4.18.

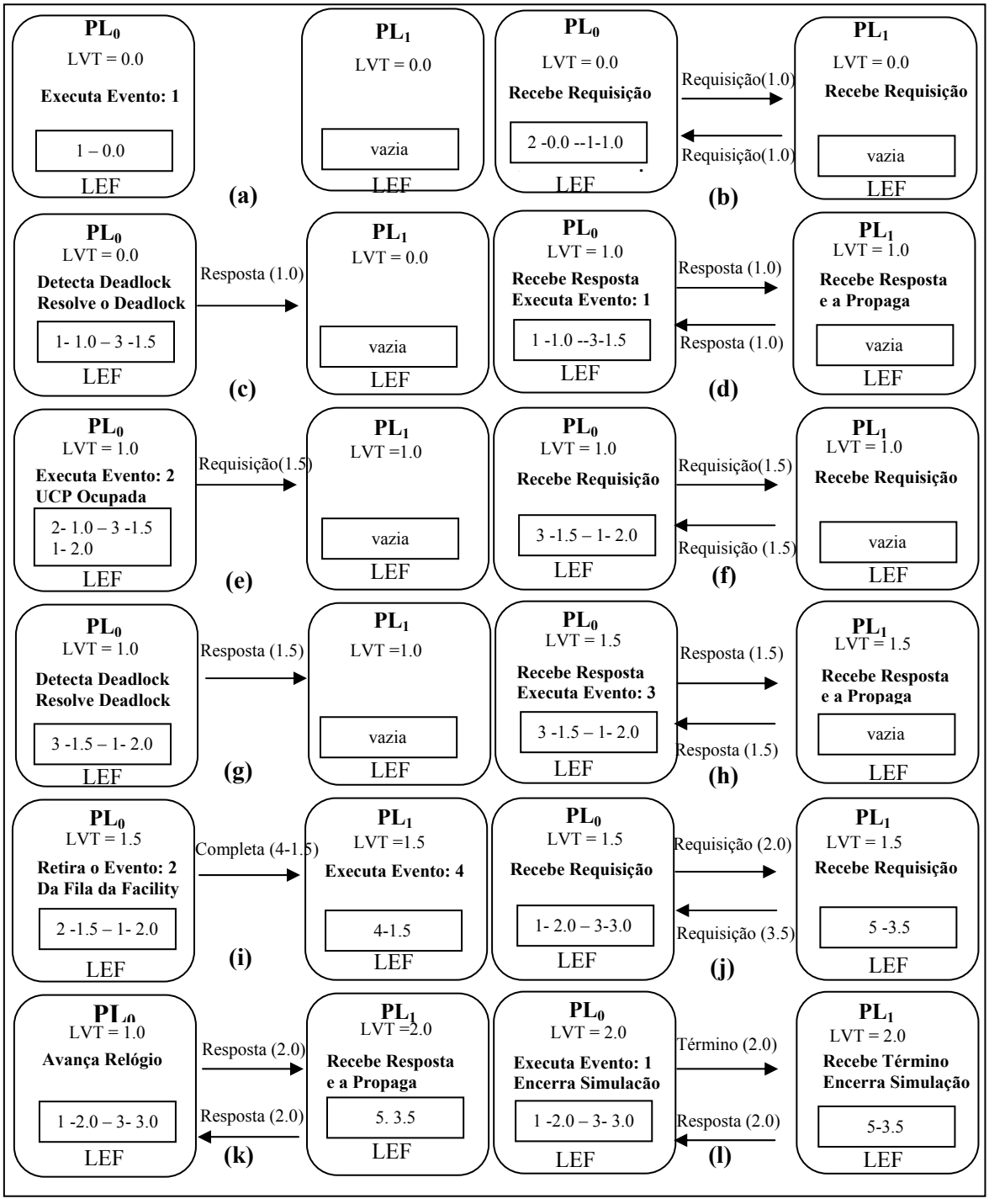


Figura 4.18: Funcionamento da CMB-*Simulation* na abordagem Sob Demanda

Na Figura 4.18(a), inicialmente o processo PL_0 executa o evento 1 e o evento 2 com tempo 0,0. Em seguida PL_0 envia uma requisição de avanço do relógio para o processo PL_1. Como PL_1 não conseguiu avançar o relógio do canal, porque o tempo é igual ao seu *LVT*, ele propaga essa requisição para PL_0, o qual detecta *deadlock* (Figura 4.18(b)).

Com base nos dados captados nas mensagens de requisição, PL_0 resolve a situação de *deadlock*, com isso avança o relógio do canal para 1,0 e envia esse avanço para o processo PL_1. PL_1 propaga essa resposta para PL_0, o qual executa o evento 1 com tempo 1,0 (Figura 4.18(d), e esse ciclo se repete até o final da simulação ilustrado na Figura 4.18(l)).

4.5 Considerações Finais

Nesse capítulo foi apresentado o mecanismo de uma simulação distribuída, mais especificamente do protocolo CMB, alvo de estudo deste trabalho, através da descrição da ferramenta CMB-*Simulation* e suas estruturas e componentes.

Para realizar a validação da ferramenta CMB-*Simulation* os resultados fornecidos pela simulação distribuída através da CMB-*Simulation* foram comparados com os resultados obtidos com a execução do mesmo modelo com a utilização do SMPL. A análise da CMB-*Simulation* foi realizada também através de uma inspeção dos resultados obtidos com a variação dos parâmetros do modelo. Cada variação efetuada nesses parâmetros acarretou as modificações esperadas nos resultados de saída, de acordo com a lógica associada ao modelo em questão. Como por exemplo, tem-se o aumento do tempo entre chegadas, mantendo-se inalterados os demais parâmetros. Em um sistema de filas, é esperado que isso provoque a diminuição do comprimento médio da fila do recurso.

Duas abordagens para o envio de mensagens foram consideradas, com o envio regular ou sob demanda das mensagens nulas especificadas pelo protocolo. A diferença entre o funcionamento dessas abordagens foi ilustrada na seção 4.4. As diferenças de desempenho de ambas as abordagens na execução de modelos simples são apresentadas no Capítulo 5, o qual apresenta os tempos de execução obtidos para cada um desses modelos, bem como a descrição dos mesmos.

Capítulo 5

Estudo de Modelos de Filas com CMB-*Simulation*

5.1 Considerações Iniciais

Avaliar a simulação distribuída conservativa não é uma tarefa fácil, dado que diversos fatores afetam o seu desempenho, como por exemplo, o modelo simulado, a divisão em processos lógicos, o valor de *lookahead* utilizado, além das plataformas de *hardware* e *software* utilizadas.

Para realizar um estudo quanto ao desempenho da simulação distribuída usando a CMB-*Simulation*, foram abordados cinco modelos, estudando quais os fatores que mais afetam o desempenho da simulação. Os tempos de execução obtidos com a CMB-*Simulation* foram comparados com os tempos obtidos com a utilização da simulação seqüencial (SMPL) [Mac87]. Os tempos de execuções nesse trabalho são todos em segundos. Para a coleta desses tempos foi utilizada a função *gettimeofday*, a qual obtém os tempos em microssegundos, os quais por sua vez são convertidos em segundos. Com o intuito de obter dados estatisticamente corretos, todos os testes foram executados quinze vezes. As seguintes métricas foram utilizadas nesse estudo:

- tempo de execução seqüencial: tempo de execução (em segundos) do programa de simulação em um único elemento de processamento;
- tempo de execução em paralelo: tempo de execução (em segundos) do programa de simulação com a utilização de mais de um elemento de

- processamento, utilizando o protocolo CMB com troca de mensagens nulas e nulas sob demanda para sincronização entre os processos;

- *peedup*: relação entre os tempos de execução da simulação seqüencial e da simulação distribuída;

- número de mensagens nulas geradas: para cada modelo exemplificado foi determinado o total de mensagens geradas por processo lógico, diferenciando as mensagens nulas das mensagens completas, com o intuito de verificar a influência dessas mensagens no desempenho da simulação.

Para cada modelo foram consideradas diferentes configurações, parâmetros da simulação e a inserção de granulosidades na execução dos eventos. A utilização de diferentes configurações de particionamento em processos lógicos foi realizada com o intuito de verificar qual a configuração ideal para cada modelo e analisar o impacto da comunicação no desempenho da simulação distribuída, pois o aumento do número de processos lógicos pode levar a um aumento considerável da comunicação [Le04].

Na Seção 5.2 são apresentados modelos de sistemas computacionais e hipotéticos utilizados na análise de desempenho e os resultados obtidos. Na Seção 5.3 foram realizados testes de hipóteses, os quais mostram se os valores obtidos se adequam aos níveis de precisão e de confiança especificados. A Seção 5.4 apresenta as considerações finais do capítulo.

5.2 Modelos em Estudo e Resultados Obtidos

Três elementos principais exercem influência no desempenho de uma simulação distribuída utilizando a sincronização conservativa [Uls99]:

- necessidade de conhecimento dos detalhes do protocolo de comunicação, principalmente quando são consideradas arquiteturas paralelas que possuem sistema de comunicação de alto desempenho;

- impossibilidade de se explorar integralmente o paralelismo do problema, além da sobrecarga introduzida pelo protocolo;

- partição do modelo em processos lógicos;

- capacidade de *lookahead* do modelo.

A necessidade de que o usuário tenha experiência em simulação distribuída e conhecimento da aplicação para obtenção de bom desempenho está relacionada à possibilidade de maximizar o balanceamento de carga do programa distribuído e de minimizar a comunicação necessária [Uls99].

Os computadores utilizados na coleta de resultados possuem processadores Intel Pentium 4 de 1,5 GHz, com 256 MB de memória RAM e encontram-se conectados por uma rede *Ethernet* de 10 Mbps, instalada no laboratório de Linux no Departamento de Ciência da Computação da UFMS. Para garantir a precisão dos dados, todos os testes foram executados quinze vezes e analisados com o objetivo de garantir dados estatisticamente corretos. As simulações foram executadas com a rede isolada, evitando, assim, a concorrência com aplicações de outros usuários.

Nas subseções seguintes são analisados cinco modelos visando o estudo das diferentes características de modelos que podem vir a afetar o desempenho da simulação distribuída, tais como pontos de tomada de decisão, recursos estruturados em série e realimentação (*feedback)*, partição do modelo e capacidade de *lookahead*.

5.2.1 Modelo de Filas em Série

Esse modelo representa um sistema aberto, como mostra a Figura 5.1, composto por dois recursos. Tarefas requisitam o recurso REC1 e, após o término do processamento,, requisitam o REC2 disco e deixam o sistema após o fim do serviço.

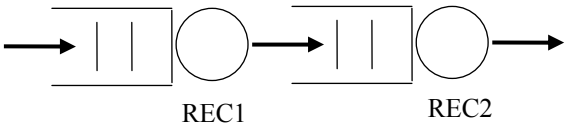


Figura 5.1: Modelo filas em série.

Devido ao número limitado de recursos no sistema apenas uma configuração foi analisada para esse modelo, isto é, houve apenas a partição do modelo em dois processos lógicos, como ilustrado na Figura 5.2. Em todos os modelos abordados neste trabalho os seguintes padrões foram adotados para a simulação com diferentes granulosidades:

- para simular a granulosidade média, a inserção de um laço com 175000 iterações, após a execução de cada evento;
- para representar a granulosidade grossa, o laço inserido realiza 1750000 iterações;
- a granulosidade fina é representada sem a inserção do laço.

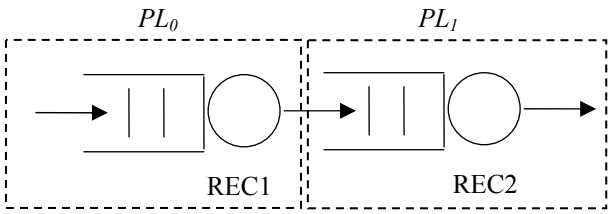


Figura 5.2: Partição do modelo filas em série.

Outro fator importante a ressaltar é que na abordagem com troca de mensagens nulas a inserção da granulosidade ao modelo não interfere no número de mensagens nulas geradas, como mostra a Figura 5.3. A implementação da CMB-*Simulation* na abordagem sob demanda é um algoritmo não determinístico, o que resulta em muitas variações no número de mensagens nulas. Mas o acréscimo da granulosidade ao modelo reduziu o número de mensagens nulas, pois a comunicação entre os processos é retardada (retardo esse justificado pelo aumento no tempo de processamento de cada processo). Por causa disso, houve também o aumento do tempo de execução (Figura 5.4).

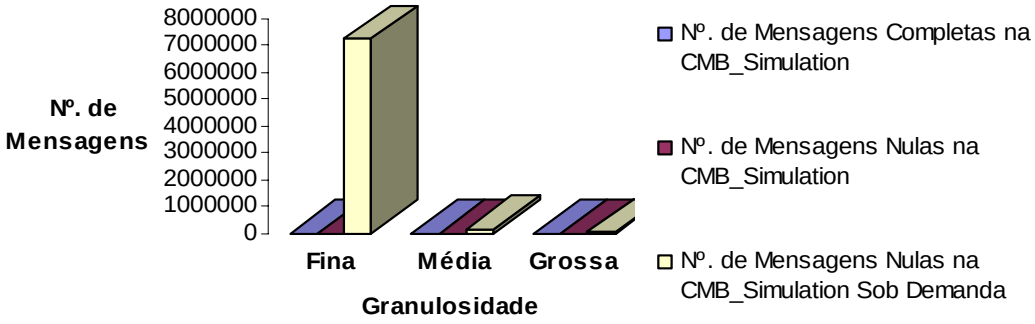


Figura 5.3: Modelo filas em série: granulosidade x nº. de mensagens nulas.

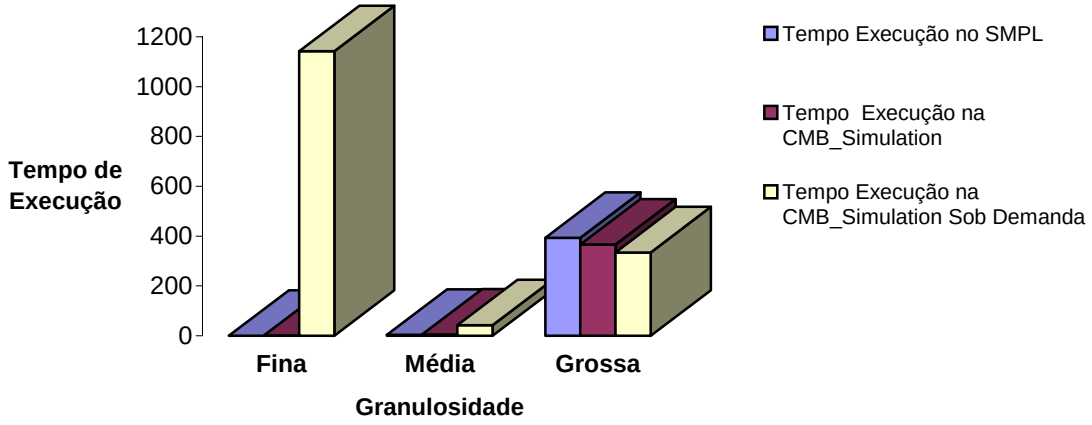


Figura 5.4: Modelo filas em série: granulosidade X tempo de execução.

Constatou-se que na granulosidade fina e média, a abordagem com troca de mensagens nulas apresentou um desempenho superior à abordagem com o envio de mensagens nulas sob demanda, entretanto, ambos não conseguiram equiparar-se à execução seqüencial, como mostrado na Figura 5.5. Essa degradação no desempenho da CMB-*Simulation* na abordagem com troca de mensagens nulas sob demanda está diretamente relacionada ao alto valor do *lookahead* aceitável por esse modelo, que conseqüentemente provoca a diminuição do número de mensagens nulas. Já na granulosidade grossa o modelo apresentou desempenho melhor, na abordagem sob demanda, devido ao fato do retardo do envio das nulas requisitando o avanço do relógio do canal.

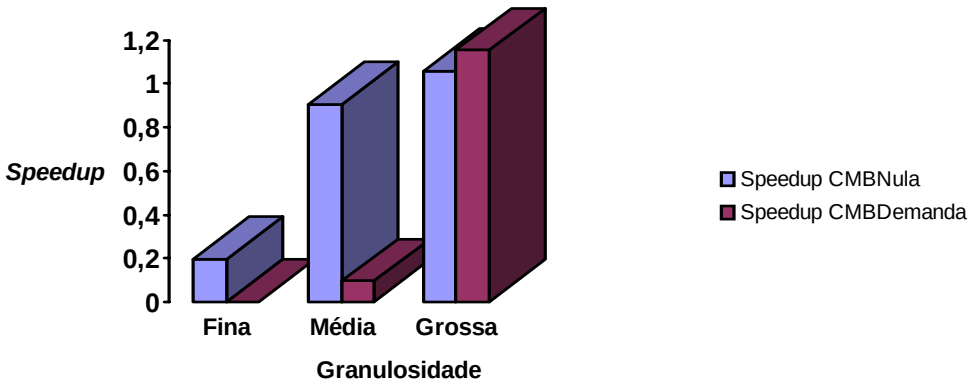


Figura 5.5: Modelo filas em série: granulosidade X *speedup*.

5.2.2 Modelo Servidor Central

Esse modelo representa sistema computacional constituído por um elemento de processamento (UCP) e quatro unidades de disco. Como mostra a Figura 5.6, as tarefas requisitam a utilização do elemento de processamento e, após o término do processamento, uma das unidades de disco é selecionada [Mac87]. Esse tipo de modelo é utilizado na literatura para avaliação da simulação seqüencial e distribuída [Mac87, Mor00 e Uls99].

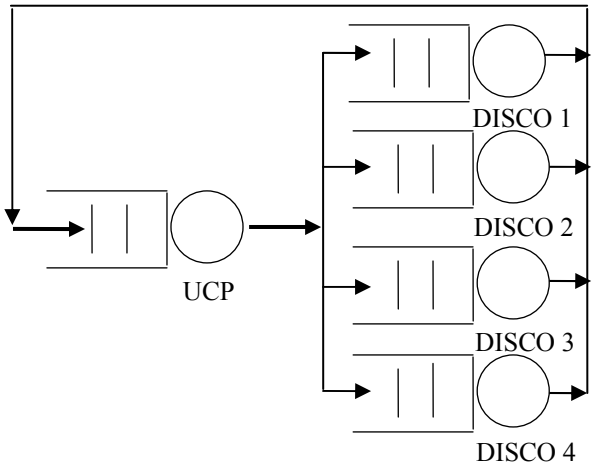


Figura 5.6: Modelo servidor central.

Nesse modelo são utilizados cinco recursos, os quais representam a UCP e as unidades de disco. Como o modelo é fechado, um número fixo de clientes (tarefas) utiliza os recursos do sistema. Inicialmente as tarefas são escalonadas para execução. Uma tarefa requisita a utilização da UCP, que executa o processamento quando estiver disponível. Após o término do processamento, uma das unidades de disco é determinada aleatoriamente e essa requisição é escalonada. Encerrado o processamento na unidade de disco, uma nova requisição para utilizar a UCP é escalonada e o ciclo de processamento se repete.

Para esse modelo foram analisadas quatro formas distintas de particionamento em processos lógicos, isto é, as partições têm tamanhos variados, com dois processos na Configuração 1 (Figura 5.7), com 3 processos na Configuração 2 (Figura 5.8), com quatro processos na Configuração 3 (Figura 5.9) e com cinco processos na Configuração 4 (Figura 5.10). Através da utilização dessas diferentes partições, pôde-se verificar a influência das configurações dos processos lógicos no desempenho da simulação, uma vez que o aumento no número de processos pode levar ao aumento da comunicação, além de identificar qual a configuração mais adequada para esse tipo de modelo.

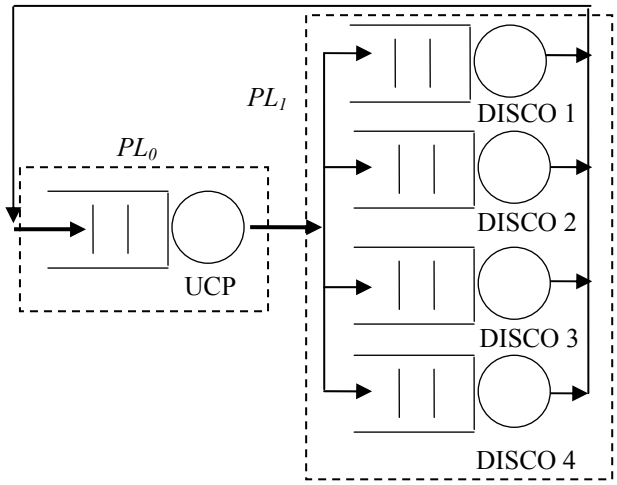


Figura 5.7: Modelo servidor central na configuração 1.

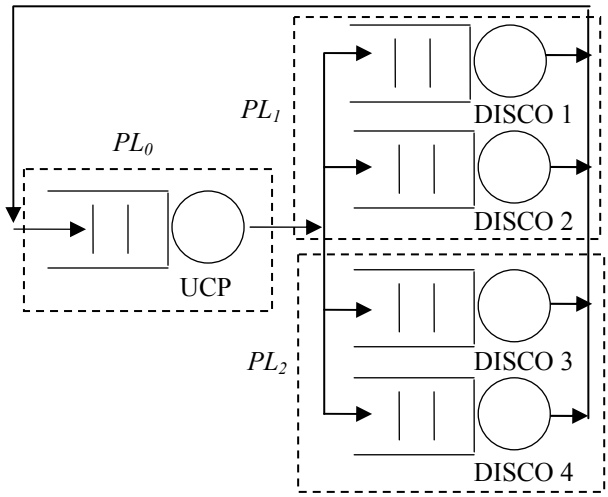


Figura 5.8: Modelo servidor central na configuração 2.

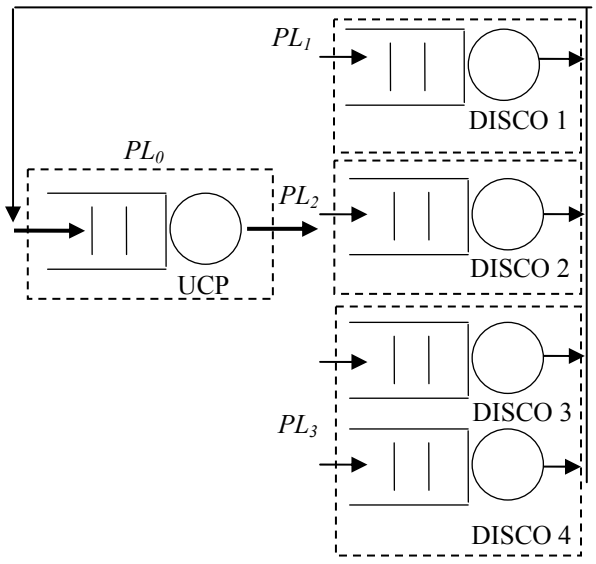


Figura 5.9: Modelo servidor central na configuração 3.

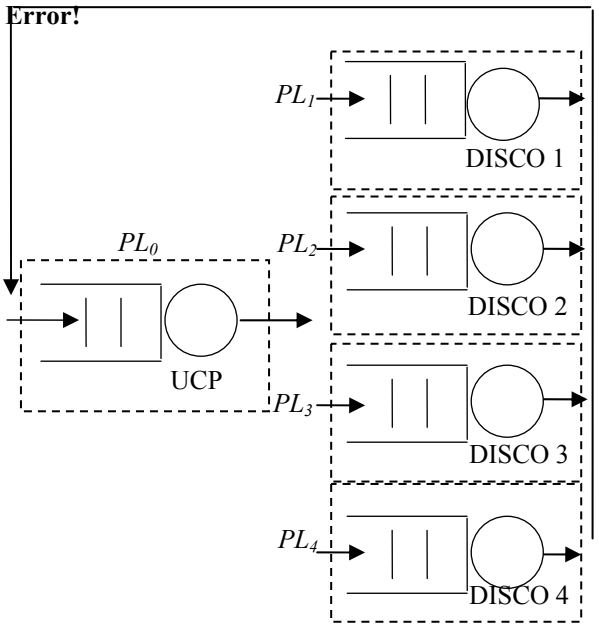


Figura 5.10: Modelo servidor central na configuração 4.

A Figura 5.11 e 5.12 mostram os resultados obtidos referentes às quatro configurações utilizadas, parametrizadas com o número de ciclos igual a 500 e *lookahead* igual a 0,01. Como pode ser observado nas duas abordagens implementadas, devido à sobrecarga na comunicação, o número de mensagens nulas cresce proporcionalmente ao aumento no número de processos envolvidos na simulação, e conseqüentemente aumenta o tempo de execução na CMB-*Simulation* (com mensagens nulas e nulas sob demanda). Esses tempos superam o tempo da execução seqüencial com uso do SMPL.

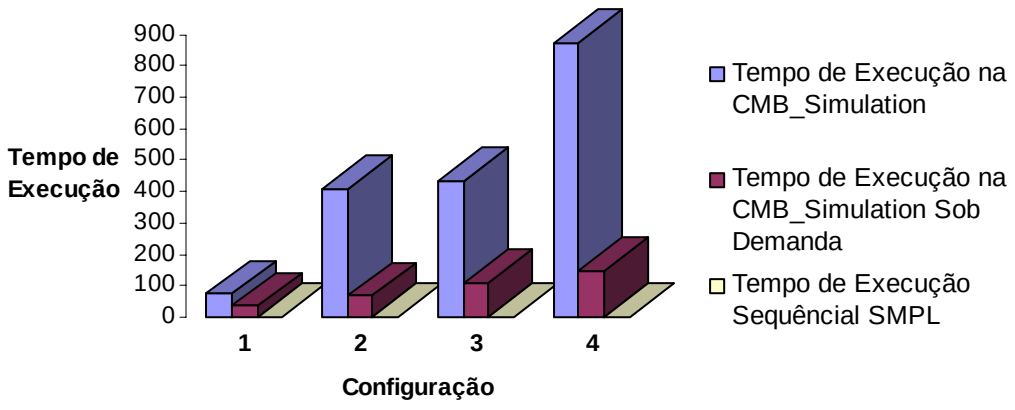


Figura 5.11: Modelo servidor central: tempo de execução x configuração.

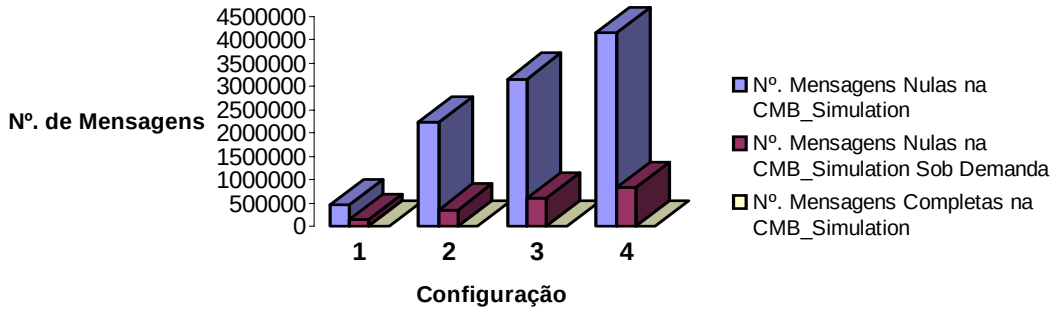


Figura 5.12: Modelo servidor central: configuração x nº. de mensagens .

Na Figura 5.13 nota-se que, devido à baixa capacidade de *lookahead* utilizada na CMB-*Simulation* com troca de mensagens nulas, a CMB-*Simulation* com troca de mensagens nulas sob demanda obteve um melhor desempenho, quando comparada à versão com troca de mensagens nulas.

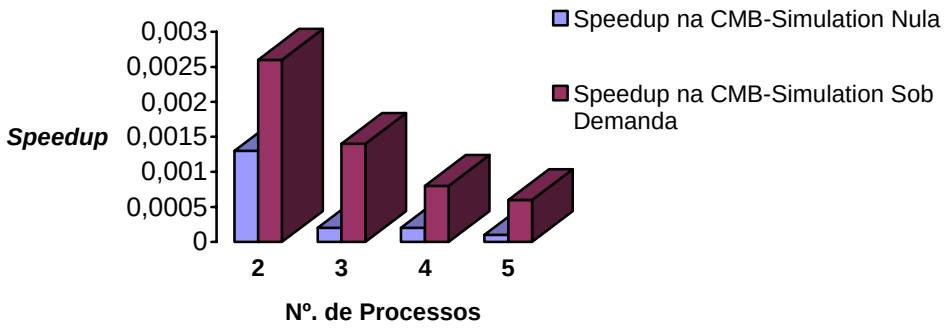


Figura 5.13: Modelo servidor central: nº. de processos x *speedup.*

Os resultados das execuções em diferentes granulosidades, com as configurações 1, 2 3 e 4, são apresentados na Figura 5.14, na Figura 5.15, na Figura 5.16 e na Figura 5.17, respectivamente. Através desses resultados, foi constatado que a partição do modelo em mais de dois processos não favorece o desempenho da simulação distribuída, ou seja, o aumento no número de processos gera uma sobrecarga de comunicação devido ao aumento do número de mensagens nulas.

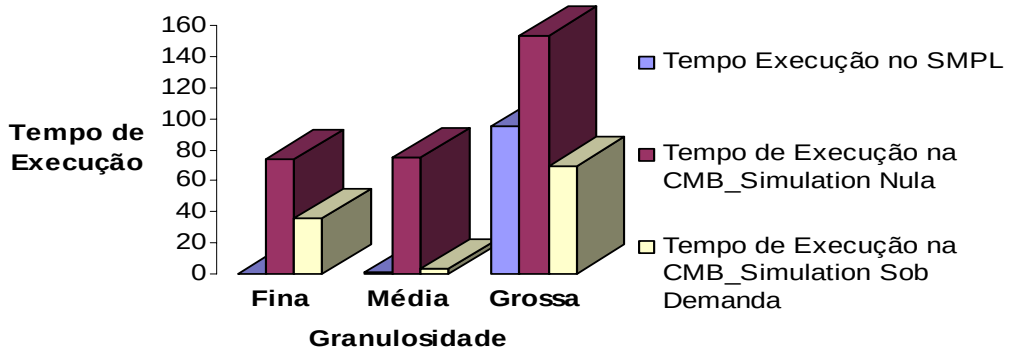


Figura 5.14: Modelo servidor central configuração 1: granulosidade x tempo de execução.

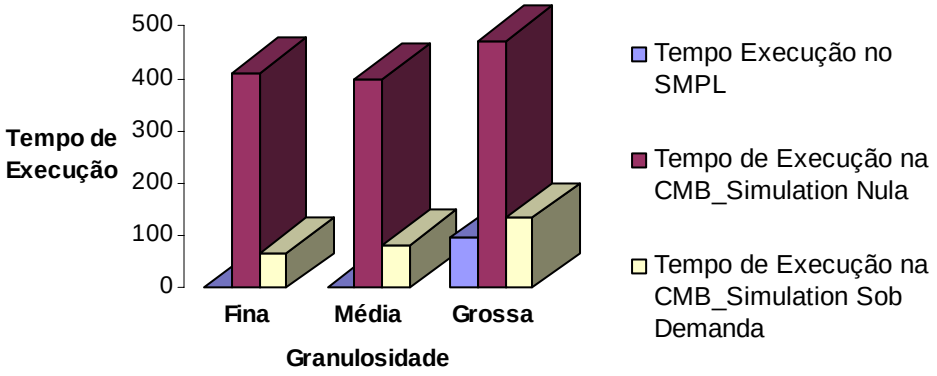


Figura 5.15: Modelo servidor central configuração 2: granulosidade x tempo de execução.

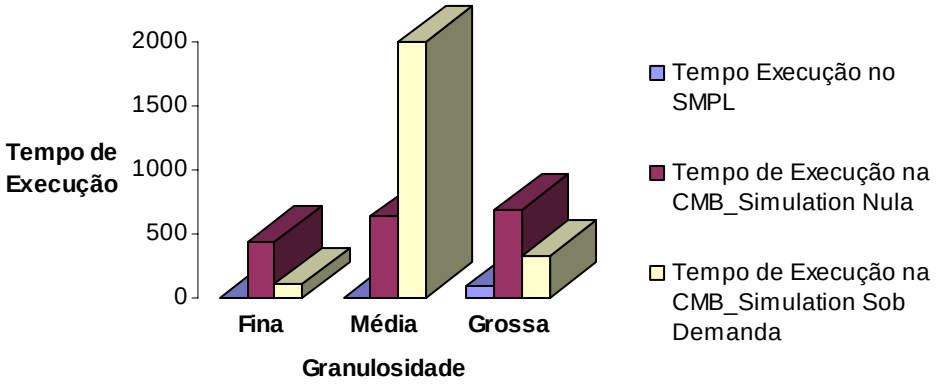


Figura 5.16: Modelo servidor central configuração 3 : granulosidade x tempo de execução.

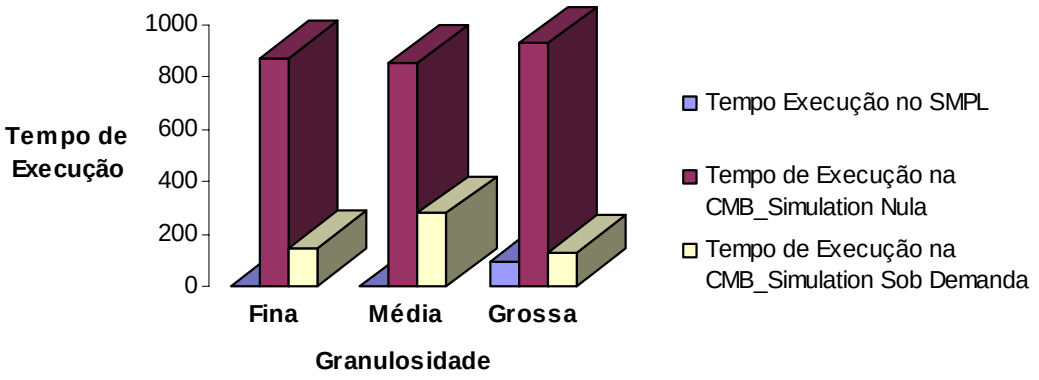


Figura 5.17: Modelo servidor central configuração 4: granulosidade x tempo de execução.

Como pode ser observado na Figura 5.18, na Configuração 1 a CMB-*Simulation* obteve êxito com a granulosidade grossa, na abordagem sob demanda, em relação às demais situações. Mas nas demais granulosidades a simulação distribuída em ambas as abordagens não superou o desempenho da simulação seqüencial. Nas outras configurações não houve ganho de desempenho. Portanto, a Configuração1 é ideal para esse modelo, devido ao retorno da tarefa ao sistema após a utilização do disco, a qual conseqüentemente atribui uma pequena capacidade de *lookahead* ao modelo.

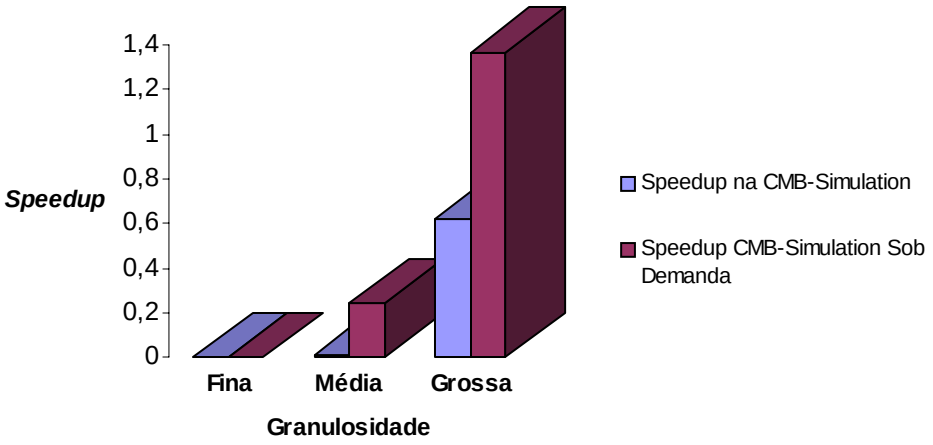


Figura 5.18: Modelo servidor central (configuração 1) : granulosidade x *speedup*.

5.2.3 Modelo Sistema Computacional

Esse modelo representa um sistema computacional composto por uma unidade de processamento e duas unidades de disco, apresentados na Figura 5.19. As requisições (tarefas) são escalonadas para a utilização da UCP de acordo com a taxa de chegada. Essas requisições são inseridas na fila da unidade de processamento se esse recurso estiver ocupado. Quando disponível, a UCP retira uma requisição da fila e a executa. Após o processamento, a tarefa pode deixar o sistema ou solicitar os serviços de uma das unidades de disco disponíveis de acordo com taxas de probabilidade. Nesse caso, uma requisição para a utilização é escalonada e, após a execução, a tarefa retorna para utilização da UCP.

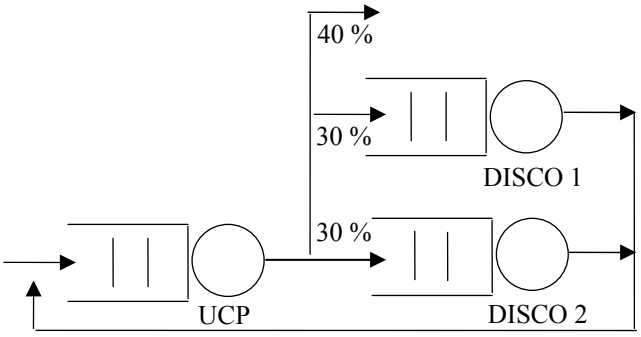


Figura 5.19: Modelo sistema computacional.

Foram analisadas duas formas distintas de configuração dos processos lógicos com alteração na carga de trabalho dos processos. A Figura 5.20 e a Figura 5.21 apresentam o modelo de redes de filas para o Sistema Computacional com diferentes configurações, com dois e três processos lógicos, respectivamente. A escolha das unidades de disco é feita com igual probabilidade, ou seja, se a probabilidade da tarefa deixar o sistema é 40%, a probabilidade de solicitar a utilização de cada uma das unidades de disco é 30%.

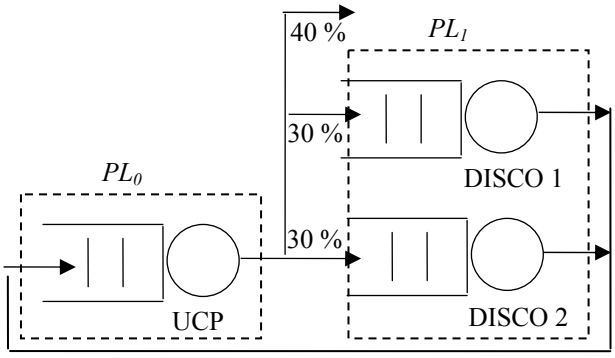


Figura 5.20: Modelo sistema computacional na configuração 1.

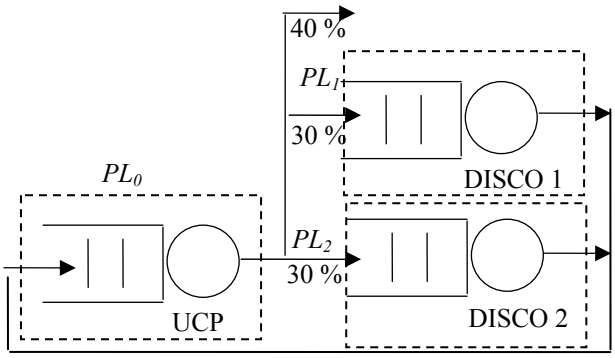


Figura 5.21: Modelo sistema computacional na configuração 2.

Os resultados obtidos das execuções com base nas duas configurações utilizadas estão ilustrados na Tabela 5.1, sendo que modelo foi parametrizado com um tempo de execução igual a 2.000,0 e 0,00001 como valor do *lookahead*. Na Configuração 1 a CMB-*Simulation* com troca de mensagens nulas obteve um baixo desempenho com relação à abordagem sob demanda. Isso se deve ao pequeno valor do *lookahead* adequado ao modelo. O valor do *lookahead* é justificado pela realimentação existente no modelo, isto é, os recursos envolvidos na realimentação não estão agrupados em um mesmo processo lógico. Nessa Tabela são utilizadas as seguintes abreviações: Tempo de Execução da CMB-*Simulation* com troca de mensagens nulas (TN) e demanda (TD). Também há abreviações para o número de mensagens completas (NMC) e número de mensagens nulas na CMB-*Simulation* com troca de mensagens nulas (NMN) e sob demanda (NMD) e o valor de *speedup* para as abordagens CMB com troca de mensagens nulas (SpN) e sob demanda (SpD).

Tabela 5.1: Resultados da simulação do modelo sistema computacional.

Tseq = 0,1739	**TN**	**TD**	**NNM**	**NMD**	**NMC**	**SpN**	**SpD**
Configuração 1	1149,81	156,8768	20013177,73	674088,73	1561	0,00015	0,00111
Configuração 2	---------	112,5524	----------	630197,46	1561	--------	0,00154

Ao efetuar a partição do modelo de acordo com a Configuração 2, percebeu-se que, devido ao valor do *lookahead,* a CMB-*Simulation* com a abordagem com troca de mensagens nulas não é viável, devido ao grande número de mensagens nulas geradas. Por isso as informações da tabela 5.1, referentes à execução com mensagens nulas, não são apresentadas. A configuração 1 é a mais indicada para esse modelo.

A Figura 5.22 apresenta os tempos de execução da Configuração 1, nas abordagens nula e sob demanda. A abordagem com mensagens nulas apresenta tempos de execução

muito maiores do que o tempo de execução seqüencial, por causa do grande número de mensagens nulas.

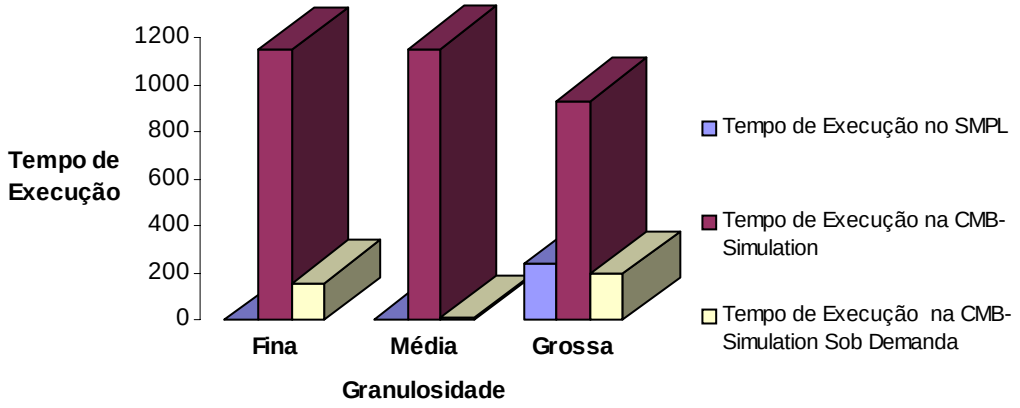


Figura 5.22: Sistema computacional - configuração 1: granulosidade x tempo de execução.

A abordagem sob demanda, na granulosidade grossa, permite que a simulação distribuída tenha um bom *speedup*, como mostra a Figura 5.23. Isso é justificado pelo fato de que um número menor de mensagens nulas circula entre os processos lógicos, quando comparada à abordagem com mensagens nulas..

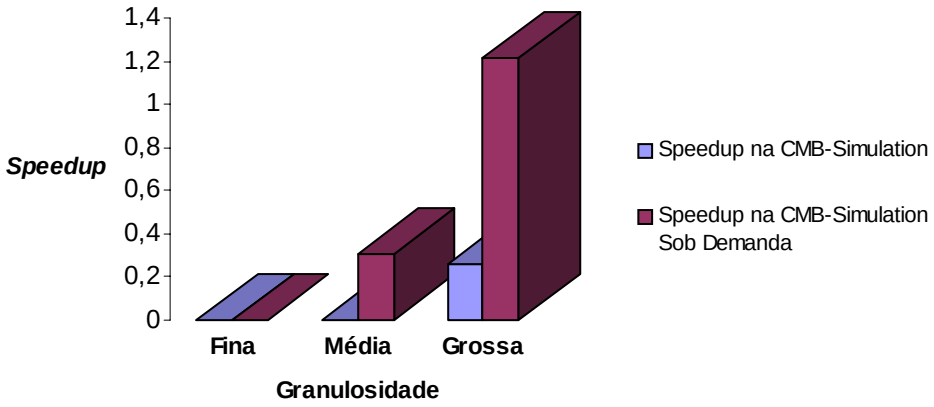


Figura 5.23: Modelo sistema computacional -configuração 1: granulosidade x *speedup*.

5.2.4 Modelo Hipotético 1

O modelo de redes de filas apresentado na Figura 5.24 descreve um sistema hipotético constituído por sete recursos [Uls99]. Três desses recursos estão organizados de forma a constituir uma estrutura em série e os demais recursos estão relacionados com pontos de decisão no sistema.

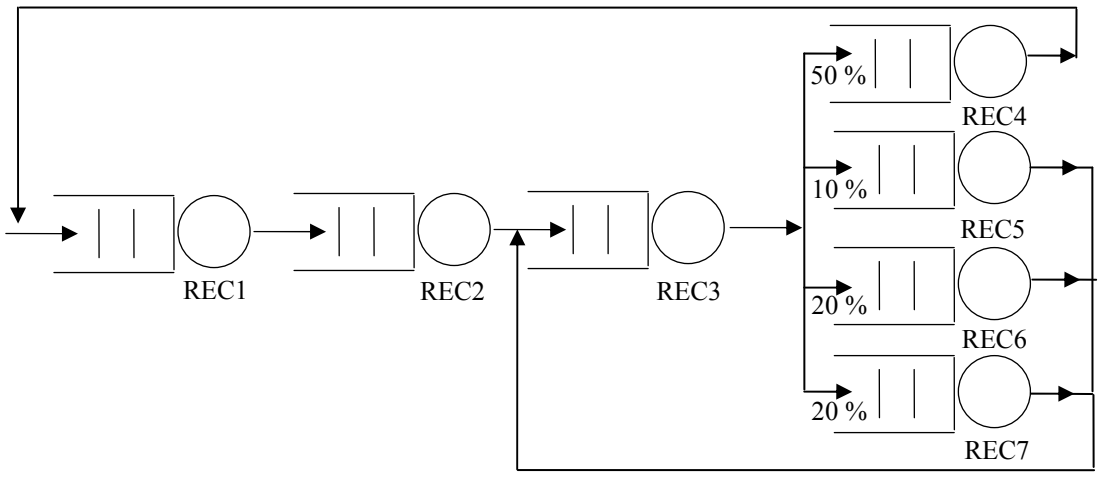


Figura 5.24: Modelo hipotético 1.

Um número fixo de tarefas circula pelo sistema. Quando uma tarefa chega ao sistema, utiliza os serviços dos recursos identificados por REC1, REC2 e REC3. Após processamento no recurso REC3, ocorre um ponto de decisão no sistema e por meio de escolha probabilística, determina-se qual o próximo recurso a ser reservado para a utilização. Em função desse recurso, a tarefa pode retornar para o recurso REC3 ou retornar ao início do sistema para utilizar os serviços oferecidos pelo recurso REC1.

Foram analisadas duas formações distintas para a configuração dos processos. As configurações do modelo de redes de filas para o Modelo Hipotético 1 são apresentadas na Figuras 5.25 e na Figura 5.26, considerando diferentes formas para a partição dos processos lógicos. Na Configuração 1 apenas dois recursos que constituem uma estrutura em série foram agrupados em um mesmo processo lógico. O terceiro recurso (REC3) foi agrupado juntamente com os demais recursos em um processo lógico distinto. Na Configuração 2 os três recursos que constituem uma estrutura em série são agrupados em um processo lógico, e os demais recursos em outro processo lógico.

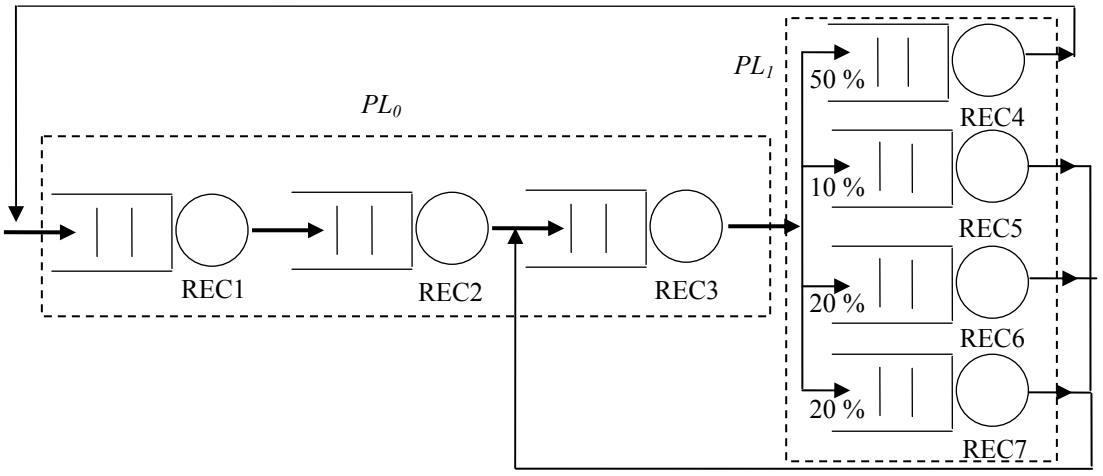


Figura 5.25: Modelo hipotético 1 na configuração 1.

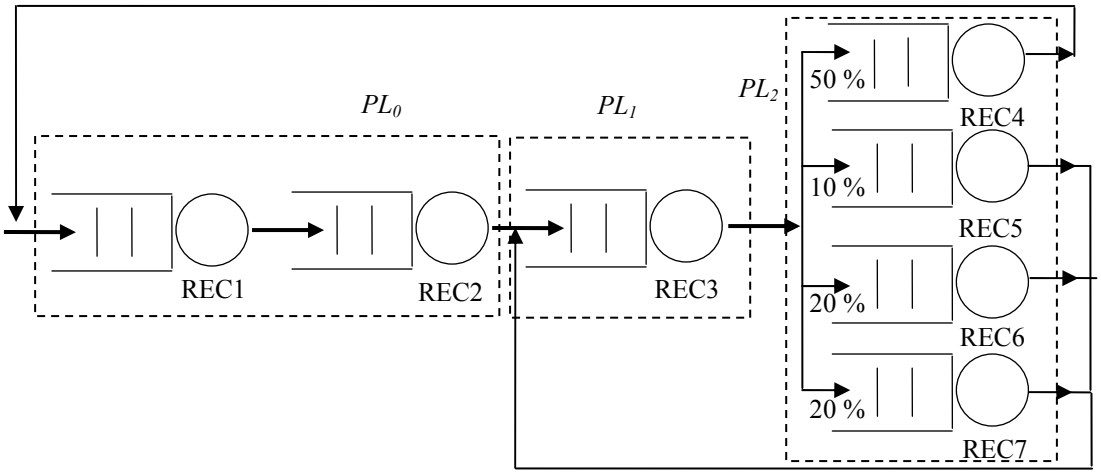


Figura 5.26: Modelo hipotético 1 na configuração 2.

Os tempos de execução do modelo nas duas configurações podem ser visualizados na Figura 5.27, obtidos a partir de quinze execuções com conjuntos de sementes distintos em cada um dos 1200 ciclos. Na Configuração 2, o tempo de execução na abordagem com mensagens nulas é superior à abordagem sob demanda. Isso está relacionado ao baixo valor do *lookahead* exigido por esse modelo, pois a Configuração 2 não permitiu a execução da CMB-*Simulation* com *lookahead* maior que 0,0002. Esse valor é justificado pelo fato dos recursos envolvidos na realimentação estarem alocados em processos distintos. Por isso o grande número de mensagens nulas necessárias (Figura 5.28).

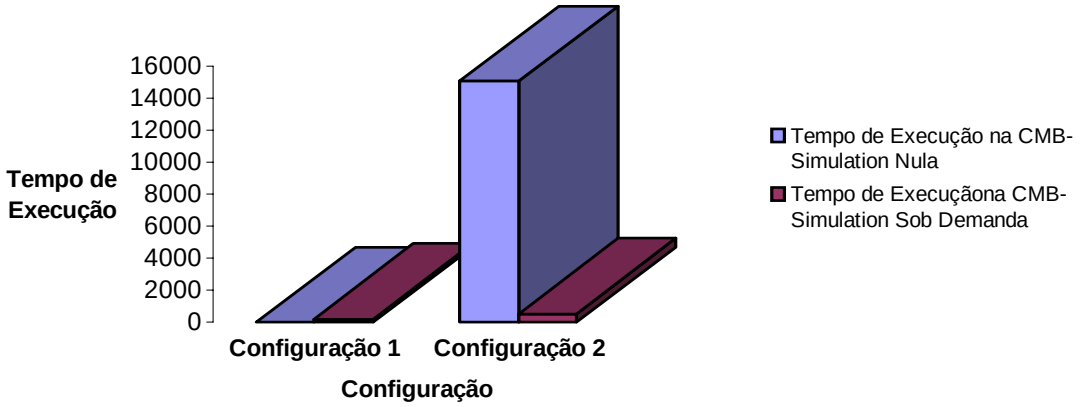


Figura 5.27: Modelo hipotético 1: configuração x tempo de execução.

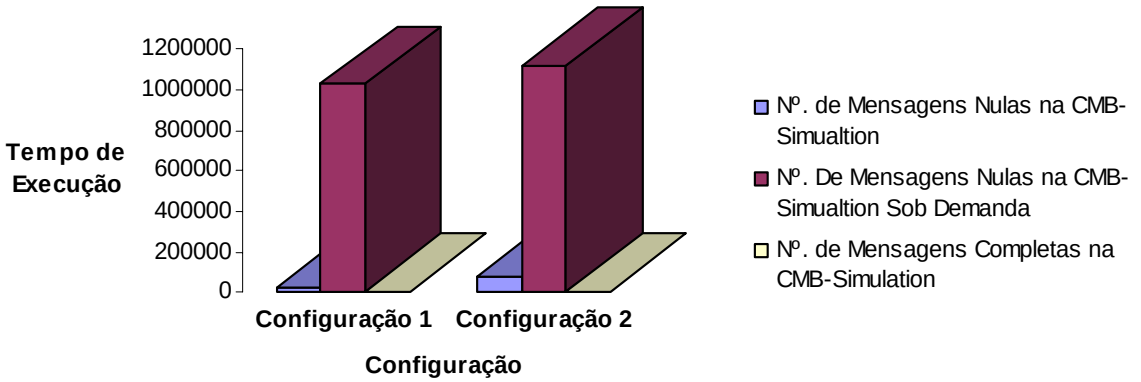


Figura 5.28: Modelo hipotético 1: configuração x nº. de mensagens.

Como a configuração 1 é a mais indicada para esse modelo, foram efetuados testes com diferentes granulosidades, apresentados na Figura 5.29. Na granulosidade grossa, por causa do balanceamento entre carga de trabalho e comunicação, o tempo de execução da simulação distribuída, nas duas abordagens, é menor do que o tempo de execução seqüencial.

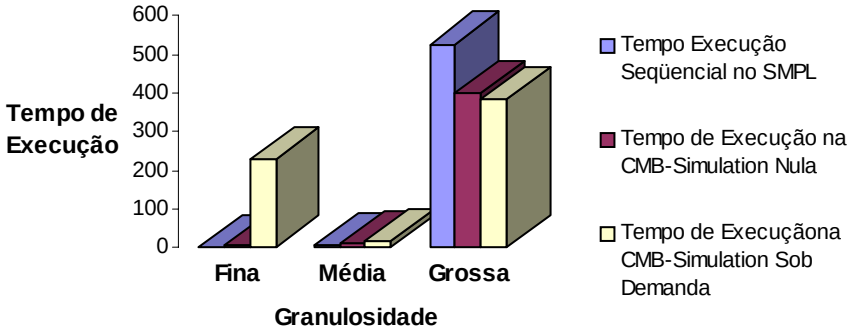


Figura 5.29: Modelo hipotético 1-configuração 1: granulosidade x tempo de execução.

Com base nos valores ilustrados na Figura 5.30, ressalta-se o fato de que na granulosidade grossa a CMB-*Simulation* obteve um melhor desempenho que a versão em SMPL. Isso pode ser constatado pelos valores de s*peedup* medidos, portanto situações que apresentam uma simulação com granulosidade grossa reagem melhor à execução da simulação distribuída através da execução da CMB-*Simulation*, para esse modelo de filas.

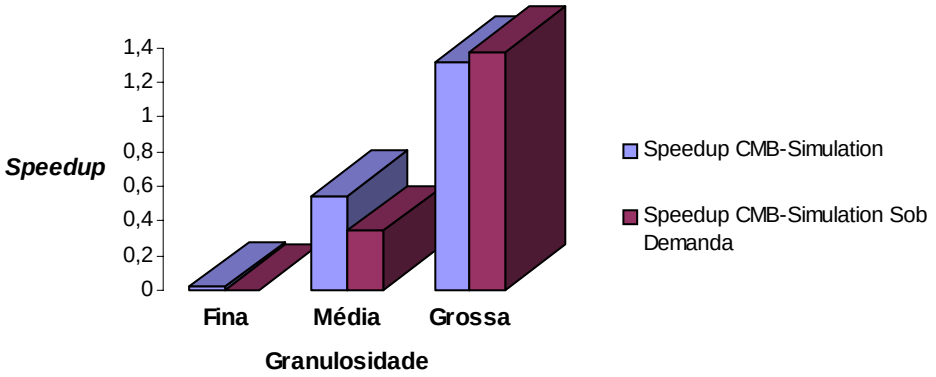


Figura 5.30: Modelo hipotético 1-configuração 1: granulosidade x s*peedup.*

5.2.5 Modelo Hipotético 2

O modelo Hipotético 2 é constituído por quatro recursos formando uma estrutura em série, como mostra a Figura 5.31 [Uls99]. Quando uma tarefa chega ao sistema, utiliza os serviços dos recursos identificados por REC1, REC2 e REC3. Encerrado o processamento no recurso REC3, ocorre um ponto de decisão no sistema e por meio de taxa de probabilidade a tarefa pode retornar para novo processamento no recurso REC2 ou prosseguir a execução. Se

a tarefa continua a execução, solicita os serviços do recurso REC4. Encerrado o processamento, a tarefa deixa o sistema.

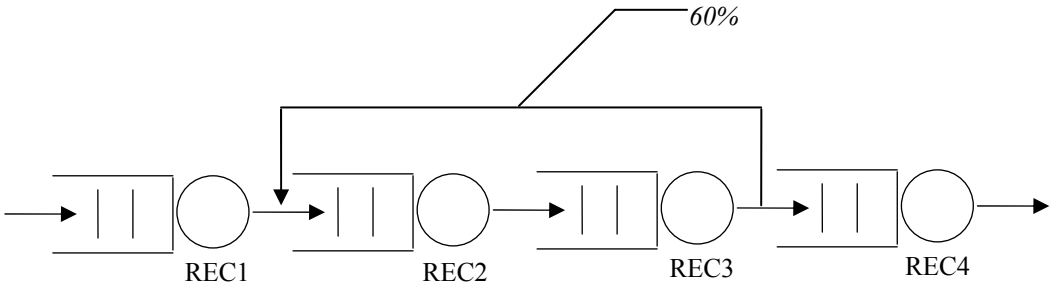


Figura 5.31: Modelo hipotético 2.

A população do sistema não é fixa, e foram analisadas três configurações distintas para os processos lógicos que constituem o modelo. Os modelos de redes de filas considerando diferentes configurações dos processos lógicos são ilustrados na Figura 5.32, na Figura 5.33 e na Figura 5.34. As diferentes configurações dos processos lógicos foram analisadas para verificar o impacto da comunicação no desempenho da simulação.

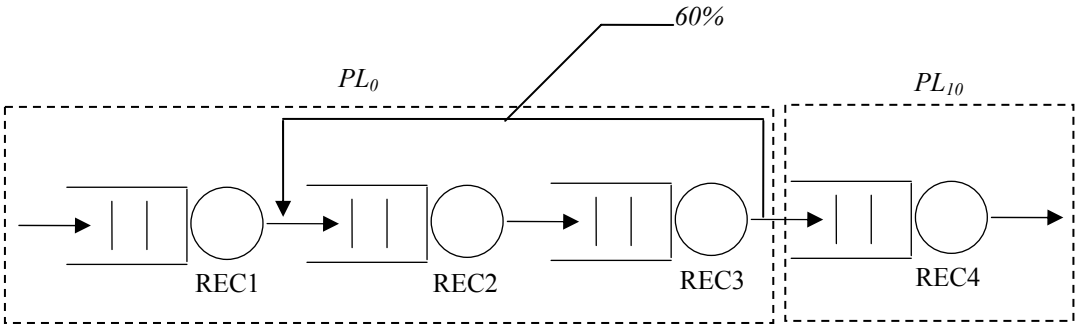


Figura 5.32: Modelo hipotético 2 na configuração 1.

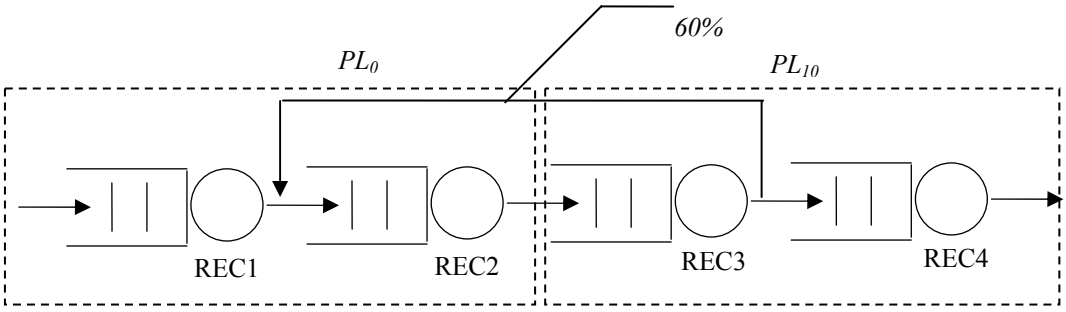


Figura 5.33: Modelo hipotético 2 na configuração 2.

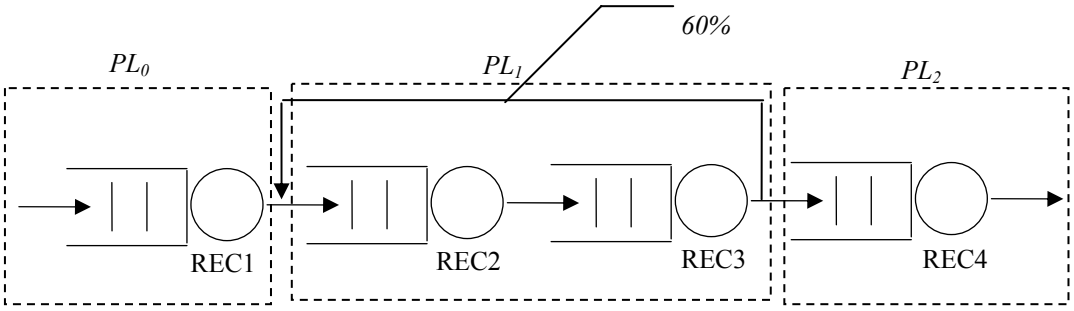


Figura 5.34: Modelo hipotético 2 na configuração 3.

Os tempos de execução obtidos com a execução da simulação dos modelos citados são apresentados na Figura 5.35. A Figura 5.36 mostra o número de mensagens, nulas e completas, geradas em cada abordagem. Na Configuração 2, devido ao fato de que os recursos envolvidos em realimentação foram agrupados em processos lógicos distintos, os tempos de execução e número de mensagens na simulação distribuída são maiores do que nas outras configurações. Na Configuração 3, o particionamento dos processos lógicos também não é o ideal, devido ao fato da realimentação estar em processos separados, por isso o tempo de execução e número de mensagens nulas geradas são maiores do que na configuração 1.

As configurações 1 e 3 permitem a redução da comunicação entre os processos, enquanto na Configuração 2 a separação dos recursos envolvidos na realimentação tende a provocar um aumento na comunicação entre os processos na simulação distribuída, o que leva à necessidade de redução do valor do *lookahead* e, conseqüentemente, a um maior tempo de execução. Percebe-se que na CMB-*Simulation,* em suas duas abordagens, a Configuração 1 é a mais indicada para o modelo Hipotético 2. Esse fato justifica-se, pois ela define um valor de *lookahead* superior, podendo proporcionar um ambiente de sincronização com um número reduzido de mensagens, e conseqüentemente um menor tempo de execução na simulação distribuída.

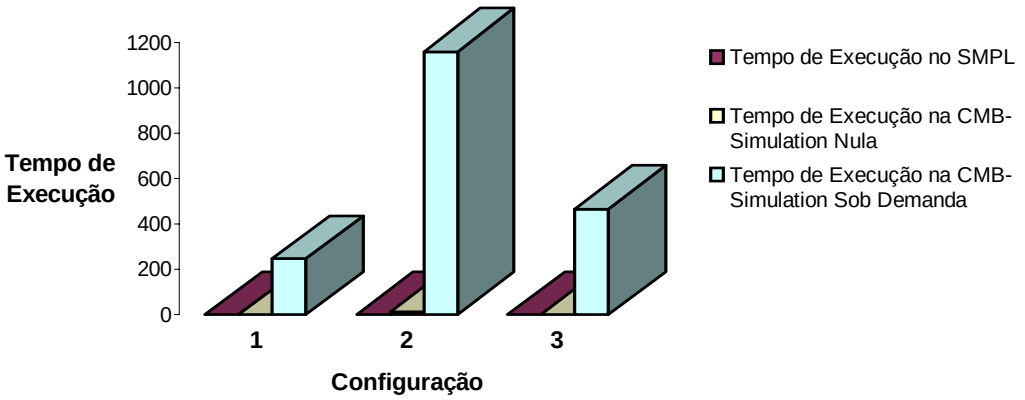


Figura 5.35: Modelo hipotético 2: configuração x tempo de execução.

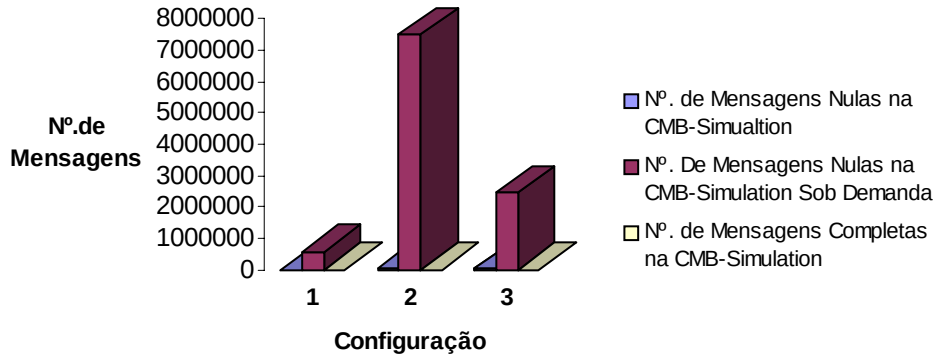


Figura 5.36: Modelo hipotético 2: configuração x nº de mensagens.

Por se mostrar mais indicada, a Configuração 1 foi analisada com diferentes granulosidades. Nos resultados obtidos, como mostram as Figura 5.37 e 5.38, o desempenho da CMB-*Simulation* aumenta de acordo com o aumento da granulosidade, mas não supera o desempenho da execução seqüencial.

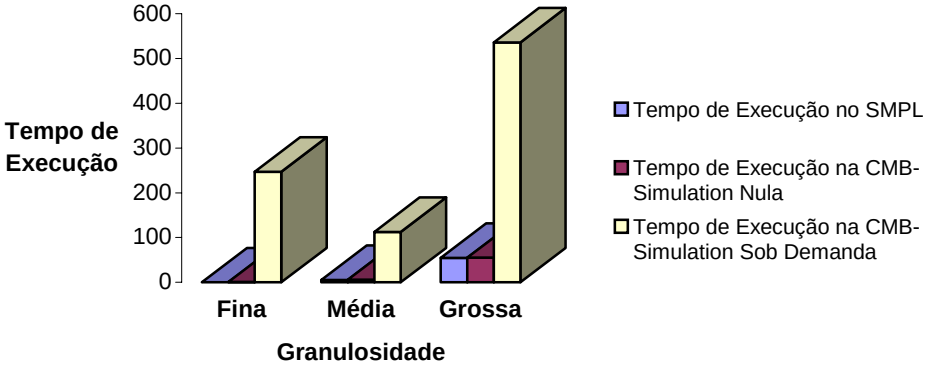


Figura 5.37: Modelo hipotético 2: granulosidade x tempo de execução.

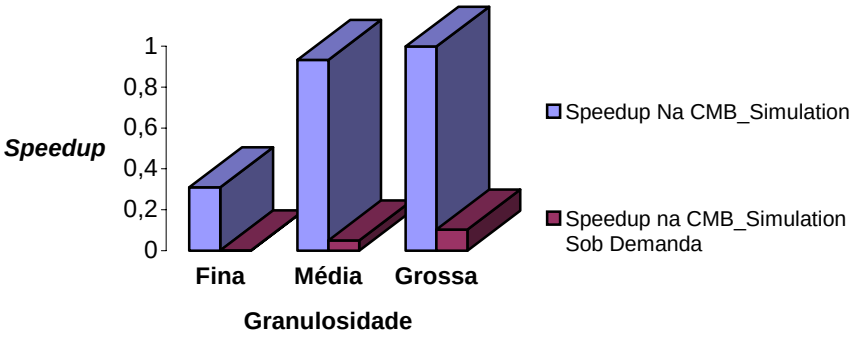


Figura 5.38: Modelo hipotético 2: granulosidade x *speedup*.

Foi efetuada uma análise sobre o impacto do *lookahead* do modelo com relação ao tempo de execução, ao número de mensagens nulas geradas e ao *speedup*, como mostra a Tabela 5.2, com granulosidade fina. É importante ressaltar que a abordagem com mensagens nulas sob demanda não faz uso do *lookahead*

Tabela 5.2: Hipotético 2 na configuração 1: análise do *lookahead.*

Tempo Seqüencial = 0,1953			
CMB-*Simulation* na abordagem sob demanda			
Lookahead	**TDist**	**Nº. de Mensagens Nulas**	***Speedup***
-	247,1292	525105,8	0,0008
CMB-*Simulation* na abordagem nula			
Lookahead	**TDist**	**Nº. de Mensagens Nulas**	***Speedup***
0,1	5,2401	32116,2	0,0373
0,5	1,4249	7957,06	0,1371
1,0	0,9217	4766,06	0,2119
1,5	0,7268	3607,46	0,2687
2,0	0,6305	2950,86	0,3097

Como o *lookahead* é uma característica do modelo, a influência do *lookahead* no número de mensagens nulas pode ser observada na Figura 5.39, onde pode ser constatado que o número de mensagens aumenta conforme o valor do *lookahead* diminui. Isso é devido ao fato de que um número maior de mensagens nulas é necessário para a sincronização, quando se têm valores muito pequenos de *lookahead*.

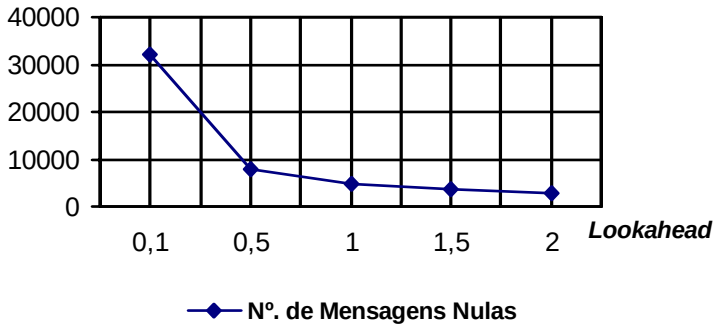


Figura 5.39: Hipotético 2: *lookahead* x n^0. de mensagens nulas.

Devido ao fato de que o número de mensagens nulas diminui quando o valor do *lookahead* aumenta, o tempo de execução da simulação distribuída também diminui, como mostra a Figura 5.40.

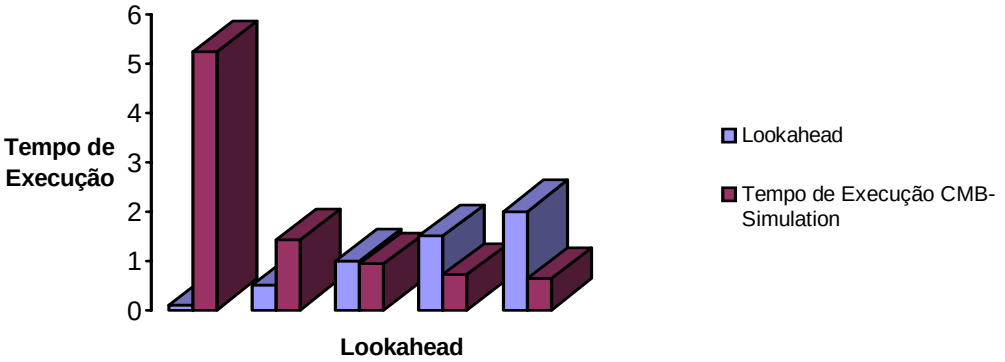


Figura 5.40: Modelo hipotético 2: *lookahead* x tempo de execução.

A Figura 5.41 mostra que o aumento do *lookahead* contribuiu positivamente na execução da simulação distribuída, proporcionando um *speedup* melhor, devido à redução no número de mensagens nulas e no tempo de execução da simulação distribuída.

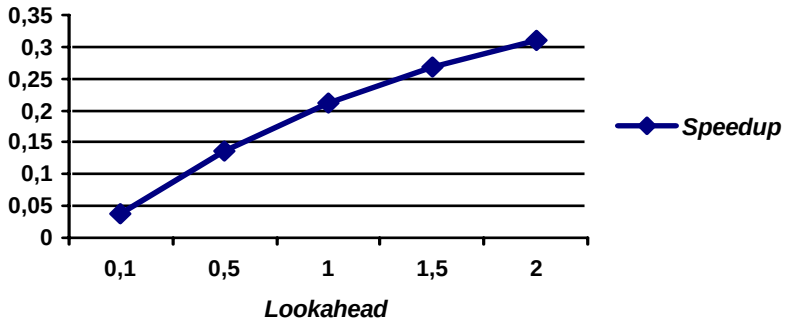


Figura 5.41: Modelo hipotético 2: *lookahead* x *speedup.*

A inferência estatística visa tirar conclusões sobre parâmetros populacionais a partir de análise dos dados amostrais. Alguns pontos importantes são o tamanho das amostras e o modo de selecionar a amostra, a natureza da inferência desejada e a precisão das conclusões. Existem dois tipos de inferências, a estimação dos parâmetros, que pode ser realizada pela determinação de intervalos de confiança, e o teste de hipóteses. O teste de hipóteses e sua aplicação nos resultados obtidos neste trabalho são apresentados na Seção 5.3.

5.3 Teste de Hipóteses

Neste trabalho utiliza-se o teste de hipóteses quando a variância é desconhecida, o qual é detalhado nesta seção. O teste de hipóteses tem como objetivo verificar se as diferenças observadas pelos tempos de execução dos modelos analisados são estaticamente significativas. Ou seja, deseja-se comprovar que os tempos não foram obtidos ao acaso, não representando uma amostra "viciada".

São definidas hipóteses para a comparação dos dados obtidos, e a partir da aceitação ou rejeição dessas hipóteses, conclui-se sobre a validade de uma determinada afirmação. H_0 é chamada hipótese de nulidade, e H_1 é a hipótese alternativa. Para provar qual hipótese é correta, tenta-se provar a hipótese de nulidade. Se não é possível provar a hipótese de nulidade, a hipótese alternativa é considerada verdadeira (para manter a uniformidade, o resultado final é dado em termos da hipótese H_0, ou seja, aceitar ou rejeitar H_0). Nesse teste de significância são considerados os seguintes parâmetros [Dot01]:

- μ_{dist} = tempo médio de execução na simulação distribuída;
- μ_{seq} = tempo médio de execução na simulação seqüencial;

- $\mathbf{S^2_{seq}}$ **e** $\mathbf{S^2_{dist}}$ representam as variâncias amostrais;
- $\mathbf{n_{seq}}$ **e** $\mathbf{n_{dist}}$ representam o tamanho das amostras;
- **t** é o valor obtido a partir dos valores amostrais;
- Hipótese Nula (**H_0**): $\mu_{dist} = \mu_{seq}$;
- Hipótese Alternativa (**H_1**): $\begin{cases} - \mu_{dist} < \mu_{seq} \text{ (A).} \\ - \mu_{dist} > \mu_{seq} \text{ (B).} \end{cases}$

Como σ^2 é desconhecido, os possíveis testes de hipóteses que podem ser utilizados são mencionados na Tabela 5.3. Neste trabalho utiliza-se o teste unilateral à direita e a esquerda, dependendo da hipótese alternativa utilizada a hipótese H_1 é seguida da identificação A ou B, de acordo com a Tabela 5.3.

Tabela 5.3: Tipos de testes de hipóteses.

Tese de Hipótese	Hipótese Nula	Hipótese Alternativa	Área de Rejeição
Bilateral	H_0: $\mu_1 - \mu_2 = 0$	H_1: $\mu_1 - \mu_2 \neq 0$	$\lvert t \rvert > t_{\sigma/2}$
Unilateral à Direita	H_0: $\mu_1 - \mu_2 = 0$	H_1: $\mu_1 - \mu_2 > 0$ (A)	$t > t_{\sigma}$
Unilateral à Esquerda	H_0: $\mu_1 - \mu_2 = 0$	H_1: $\mu_1 - \mu_2 < 0$ (B)	$t < -t_{\sigma}$

Para aplicar o teste sobre as médias amostrais (μ_{seq} e μ_{dist}), deve-se verificar se as variâncias (S^2_{seq} e S^2_{dist}) são iguais ou não. Para averiguar a igualdade das variâncias usa-se a distribuição de Fischer (F_0), a qual deve ser calculada de acordo com a Fórmula 5.1com a maior variância sobre a menor.

$$F_0 = \frac{S^2_{seq}}{S^2_{dist}} \sim F_{(nseq-1,\ ndist-1)} \tag{5.1}$$

Para exemplificar um teste de hipótese usou-se os resultados amostrais obtidos no modelo de redes de filas em série com granulosidade grossa. Dessa forma, pode-se dizer com 95% se segurança, que se $F_0 \leq F_{\sigma/2,}$ com $\sigma = 0{,}05$ de significância, onde rejeita-se a hipótese de igualdade de variâncias, caso contrário não é absurdo assumir que $S^2_{seq} = S^2_{dist}$.

Aplicando o teste (5.1) no conjunto de valores amostrais obtidos a partir de 15 amostras com a simulação seqüencial e distribuída no modelo de sistema de filas em série constata-se a diferença das variâncias:

$S^2_{seq} = 1{,}38300$

$S^2_{dist} = 24{,}5332$

$$F_0 = \frac{S^2_{dist}}{S^2_{seq}} = \frac{24{,}5332}{1{,}38300} = 17{,}74$$

O valor de $F_{\sigma/2}$ com (n_{seq} -1) 14 graus de liberdade encontrada na tabela de Fischer é 2,483, com numerador igual a 15 e denominador igual a 14. Como 17,74 é maior que 2,483, pode-se assumir a diferença das variâncias [Men92].

A hipótese em teste no modelo de filas em série é unilateral à esquerda, ou seja, a hipótese de nulidade é H_0: $\mu_{dist} - \mu_{seq} = 0$ e a hipótese alternativa é H_1: $\mu_{dist} < \mu_{seq}$, a qual implica que o tempo de execução da simulação distribuída é menor que o tempo de execução da simulação seqüencial.

O passo seguinte consiste em encontrar o valor de t_σ, com um nível de significância (σ) igual a 0,05 de acordo com a distribuição *t-student* a qual é bi-caudal e simétrica, com (15 + 15 -2) graus de liberdade, assim tem-se $t_{0,05} = 2{,}048$. Isso significa que existe a probabilidade de no máximo 5% de errar ao se rejeitar a hipótese nula. O próximo passo é obter o valor de t, com grau de liberdade 28 ($n_{seq} + n_{dist}$ -2):

$$\mathbf{t} = \frac{\mu_{dist.} - \mu_{seq}}{\sqrt{\frac{S^2_{dist}}{n_{dist}} + \frac{S^2_{seq}}{n_{seq}}}} \qquad (5.2)$$

Ao aplicar os resultados obtidos na simulação do modelo de filas em série na fórmula 5.2 obteve–se o valor de ***t*** foi igual a -20,10, como -20,10 < -2,048 a hipótese de nulidade é rejeitada, e a hipótese alternativa é aceita, ou seja, o tempo de execução da simulação distribuída é menor com relação ao tempo de execução da simulação seqüencial.

$$t = \frac{365{,}976 - 393{,}045}{\sqrt{\frac{24{,}533}{15} + \frac{1{,}38300}{15}}} = -20{,}10$$

Ao aplicar o teste de hipóteses unilateral à esquerda em abordagens especificas no modelo Filas em Série e Hipotético 1, os valores de ***t*** obtidos são apresentados na Tabela 5.4.

Tabela 5.4: Resultados do teste de hipóteses unilateral à esquerda.

	t	t_σ **= -2,048**
Modelo de Filas em Série com Granulosidade Grossa		
CMB-Simulation Nula	-20,10	Aceita H_{1A} e Rejeita H_{0A}
CMB-Simulation Demanda	-44,48	Aceita H_{1A} e Rejeita H_{0A}
Hipotético 1 na Configuração 1 com Granulosidade Grossa		
CMB-Simulation Nula	-85,79	Aceita H_{1A} e Rejeita H_{0A}
CMB-Simulation Demanda	-90,87	Aceita H_{1A} e Rejeita H_{0A}

A Tabela 5.5 mostra os valores de ***t*** obtidos com a aplicação do teste de hipóteses unilateral à direita, em modelos como Filas em Série, Servidor Central, Sistema Computacional, Hipotético 1 e Hipotético 2.

Tabela 5.5: Resultados do teste de hipóteses unilateral à direita.

Modelo de Filas em Série com Granulosidade Fina		
	t	t_σ **= 2,048**
CMB-*Simulation* Nula	69,83	Aceita H_{1B} e Rejeita H_{0B}
CMB-*Simulation* Sob Demanda	96,03	Aceita H_{1B} e Rejeita H_{0B}
Modelo Servidor Central na Configuração 1		
CMB-*Simulation* Nula	126834,38	Aceita H_{1B} e Rejeita H_{0B}
CMB-*Simulation* Sob Demanda	470852,10	Aceita H_{1B} e Rejeita H_{0B}
Modelo Servidor Central na Configuração 1		
CMB-*Simulation* Nula	126834,38	Aceita H_{1B} e Rejeita H_{0B}
CMB-*Simulation* Sob Demanda	470852,10	Aceita H_{1B} e Rejeita H_{0B}
Modelo Sistema Computacional na Configuração 1		
CMB-*Simulation* Nula	126,34	Aceita H_{1B} e Rejeita H_{0B}
CMB-*Simulation* Sob Demanda	10,90	Aceita H_{1B} e Rejeita H_{0B}
Modelo Hipotético 1 na Configuração 1		
CMB-*Simulation* Nula	224,09	Aceita H_{1B} e Rejeita H_{0B}
CMB-*Simulation* Sob Demanda	31,645	Aceita H_{1B} e Rejeita H_{0B}
Hipotético 2 na Configuração 1 com Granulosidade Grossa		
CMB-*Simulation* Nula	0,41	Aceita H_{1B} e Rejeita H_{0B}
CMB-*Simulation* Sob Demanda	218,98	Aceita H_{1B} e Rejeita H_{0B}

5.4 Considerações Finais

No estudo do comportamento da CMB-*Simulation*, constatou-se que a capacidade de *lookahead* é individual para cada modelo, e essa capacidade pode variar de acordo com a partição do modelo em processos lógicos, assim como a posição dos recursos envolvidos na realimentação, caso essa exista. As análises efetuadas com um dos modelos mostraram que fatores como por exemplo o número de mensagens nulas, o valor do *lookahead* e o tempo de execução estão interligados ao valor de *speedup*, isto é, quanto maior o valor do *lookahead* menor o número de mensagens nulas e, conseqüentemente, menor o tempo de execução.

O desempenho da simulação distribuída com o envio de mensagens nulas nos casos observados é afetado pela comunicação entre os processos. Por esse motivo, sua utilização é indicada para modelos particionados de forma a agrupar processos envolvidos em realimentação.

Os estudos apresentados nesse capítulo mostraram que a CMB-*Simulation* é uma ferramenta viável para efetuar estudos de desempenho do protocolo conservativo CMB, em diferentes situações de particionamento do modelo. Também mostraram a viabilidade e a adequabilidade de se utilizar a CMB-*Simulation* para a simulação de redes de filas.

O Capítulo 6 apresenta as conclusões gerais deste trabalho, além das contribuições e propostas para trabalhos futuros.

Capítulo 6

Conclusões, Contribuições e Propostas para Trabalhos Futuros

A superioridade no desempenho dos protocolos conservativos em relação aos otimistas tem sido muito discutida e é difícil determinar as regras de quando devem ser utilizados, uma vez que fatores como *hardware* e *software* disponíveis e as características do sistema em estudo devem ser considerados [Fuj00]. Avaliar o grau de influência das características do protocolo CMB na simulação distribuída não é uma tarefa fácil, dado que diversos outros fatores afetam o desempenho da simulação, como por exemplo, o modelo simulado, a divisão em processos lógicos e as plataformas utilizadas.

Embora conseguir um bom *speedup* com a utilização de uma simulação distribuída nem sempre seja possível, ele pode ser alcançado minimizando o custo da comunicação e adotando uma granulosidade adequada, dependendo do sistema em estudo. Nos experimentos realizados, constatou-se que o aumento da granulosidade dos modelos pode levar a um bom *speedup*. Devido à sobrecarga da comunicação, deve-se tentar minimizar a necessidade de comunicação, isto é, a partição do modelo deve ser feita considerando as características do modelo e da arquitetura de *hardwar*e a ser utilizada. No estudo realizado constatou-se que o valor do *lookahead* está diretamente ligado ao tipo do modelo em questão, assim como ao tipo de particionamento realizado. Essa foi uma das maiores dificuldades do trabalho, que consistiu em determinar qual valor de *lookahead* melhor se adapta a cada modelo.

6.1 Contribuições

O trabalho desenvolvido nesta dissertação faz uma revisão sobre simulação seqüencial e simulação distribuída conservativa, que poderá servir de base para futuros trabalhos na área.

A principal contribuição do trabalho é a ferramenta CMB-*Simulation*, que tem a vantagem de possuir características e diretrizes semelhantes às do SMPL, pois é uma extensão do mesmo. Com isso, os usuários do SMPL podem ser beneficiados, pois para utilizar a CMB-*Simulation* eles devem aprender a efetuar o particionamento da aplicação e também conhecer os princípios da simulação distribuída conservativa, não sendo necessário conhecer uma nova linguagem de programação. Exemplos de como utilizar a ferramenta em comparação como a versão seqüencial foram apresentados e podem servir de guia para novos usuários da simulação distribuída conservativa.

Além disso, a ferramenta implementa o protocolo CMB com mensagens nulas e com mensagens nulas sob demanda, o que permite ao usuário a possibilidade de escolher qual protocolo melhor se adapta ao problema em estudo.

O Capítulo 5 mostrou análises de desempenho de alguns modelos de filas, o que permitiu observar situações em que existe ou não viabilidade de se usar uma ou outra abordagem CMB para sincronização da simulação distribuída.

Os resultados obtidos neste trabalho podem servir de base para validar os resultados obtidos com a simulação do protocolo CMB efetuada por Morselli [Mor00].

6.2 Propostas para Trabalhos Futuros

Como sugestões para a continuidade do trabalho pode-se elencar:

- incluir um módulo para efetuar a análise de saída dos resultados;
- desenvolver estudos da capacidade de *lookahead* para outros modelos de redes de filas, incluindo modelos fechados;
- utilizar a CMB-*Simulation* em outros ambientes computacionais, com outras topologia de rede;

- utilizar outros ambientes de troca de mensagens e efetuar comparação com o desempenho da implementação efetuada neste trabalho;

- realizar estudos de análises comparativas entre o protocolo conservativo CMB e o otimista *Time Warp*, assim como a identificação de classes de aplicação que melhor se adaptam a cada caso;

- implementar o mecanismo de troca dinâmica proposto por Morselli [Mor00];

- desenvolver interfaces gráficas, através das quais o usuário poderá escolher, de acordo com o modelo, por qual abordagem da CMB-*Simulation* optar.

Referências Bibliográficas

[Ale01] C. Alexopoulos, A. F. Seila. Output data analysis for simulation. *Proceedings* of *the 33rd Winter Simulation Conference*, p. 155–122, 2001.

[Aya92] R. Ayani, H. Rajaei. Parallel simulation using conservative time windows: a performance study. *Concurrency Practice and Experience*, v.6, n.2, 1992.

[Bag94a] R. L. Bagrodia. The *Maise* environment for parallel simulation. *Proceedings of the 27th Annual Simulation Symposium*, p. 4-12, 1994.

[Bag94b] R.L. Bagrodia. Language support for parallel discrete-event simulations *Proceedings of the 26th Winter Simulation Conference*, p. 1324-1331, 1994.

[Bag98] R. L. Bagrodia, R. Meyer, M. Takai, Y. Chen, X. Zeng, J. Martin, H. Y.Song. Parsec: A parallel simulation environment for complex systems. *IEEE Computer*, v. 31, n. 10, p. 77–85, 1998.

[Bag99] R. Bagrodia, E. Declman, S. Docy, T. Phan. Performance: prediction of large parallel applications using parallel simulations. *Proceedings of the 7th ACM SIGPLAN Symposium on Principles and Practice of Parallel Programming*, p. 151-162, 1999.

[Ban98] J. Banks. *Handbook of Simulation*. John Wiley & Sons, Inc, p 3–30, 1998.

[Ban01] J. Banks, J. S. Carson II, B. L. Nelson, D. M. Nicol. *Discrete-event system simulation*. 3rd Edition. Prentice-Hall. International Series, 2001.

[Bru97] S. M. Bruschi. *Extensão do ASIA para simulação de arquitetura de computadores*. Dissertação de Mestrado, Instituto de Ciências Matemáticas de São Carlos - USP, 1997.

[Bru00] S. M. Bruschi. ASDA – An automatic distributed simulation environment. *Proceedings of 2000 Summer Computer Simulation Conference* (SCSC´2000). Vancouver, Canada, 2000.

[Bru03] S. M. Bruschi. *Um ambiente de simulação distribuída automático ASDA*. Tese de Doutorado, Instituto de Ciências Matemáticas e de Computação – USP, 2003.

[Bry77] R. E. Bryant. *Simulation of packet communications architecture computer systems*. Technical Report, MIT-LCS-TR-188, 1977.

[Car00] C. D. Carothers, R. Fujimoto. Efficient execution on Time Warp programs on heterogeneous, NOW platforms. *IEEE Transactions on Parallel and Distributed Systems*, v. 11, p. 299-317, 2000.

[Car02] J.S Carson II. Model verification e validation. *Proceedings of the 31st Winter Simulation*, p. 52-18, 2002.

[Cha79] K. M. Chandy, J. Misra. Distributed simulation: a case study in design and verification of distributed programs. *IEEE Transactions on Software Engineering*, v.5, n.1, p.440–452, 1979.

[Cha81] K. M. Chandy, J. Misra. Asynchronous distributed simulation via a sequence of parallel computations. *Communications of the ACM*, v. 24, n. 4, p. 198-206, 1981.

[Cub95] R. Cubert, P. Fiswick. *SIM++* Version 1.0. Departament of Computers and Information Science and Engineering, 1995. (disponível em http://www.cis.ufl.edu/~fishwick/simpack/simpp.ps), acessado em 11/10/2004.

[Cra99] R. Crain, J. O. Henriksen. Simulation using GPSS/H. *Proceedings of the 31st Winter Simulation Conference*, p. 182-187, 1999.

[Dot01] M.A. Dota. *Avaliação de desempenho de uma ferramenta para simulação distribuída baseado no protocolo Time Warp*. Dissertação de Mestrado, Instituto de Ciências Matemáticas e de Computação – USP, 2001.

[Ewi99] G. C. Ewing, k. Pawlikowski, D. Mcnickle. Akaroa-2: exploring network computing by distributing stochastic simulation. *Proceedings of European Simulation Multiconference (ESM99)*, p.175-181, 1999.

[Fer95] A. Ferscha. Adaptive model parallelism exploration in parallel simulation. *Proceedings of EuroSim*, p. 241–248, 1995.

[Fis92] P. A. Fishwick. Simpack: getting started with simulation programming in C and C++. *Proceedings of the 24th* Winter *Simulation Conference*, p. 154-162, 1992.

[Fis96] P. A. Fishwick. Extending Object –Oriented Design for Physical Modeling. *ACM Transactions on Modeling and Computer Simulation*,1996.

[Fuj90] R. M. Fujimoto. Parallel discrete event simulation. *Communications of the ACM*, v. 33, n. 10, p.31–53, 1990.

[Fuj93] R. M. Fujimoto. Parallel and distributed discrete event simulation: algorithms and applications. *Proceedings of the 25th Winter Simulation Conference*, p. 106-114, 1993.

[Fuj95] R. M. Fujimoto. Parallel and distributed simulation. *Proceedings of the 31st Winter Simulation Conference*, p. 122-131, 1995.

[Fuj00] R. M. Fujimoto. *Parallel and Distributed Simulation Systems*. John Wiley & Sons, 2000.

[Fuj02] R.Fujimoto, K. Perumalla, T. McLean, e G. Riley. Experiences applying parallel and interoperable network simulation techniques in on-line simulations of military networks. *Proceedings of the 6th Winter Simulation Conference*, p. 97-104, 2002.

[Gob91] J. Goble. Introduction to SIMFACTORY II.5 *Proceedings of the 23rd Winter Simulation Conference*, p. 77-80, 1991.

[Gho96] S. Ghosh. On the proof of correctness of yet another asynchronous distributed discrete event simulation algorithms (YADDEAS). *IEEE Transactions on Systems*, v. 26, n. 1, p. 68-80, 1996.

[Gom95] F. Gomes, J. Cleary, A. Covington, S. Franks, B. Unger, Z. Ziao. SimKit: a high performance logical process simulation class library in C++. *Proceedings of the 27th Winter Simulation Conference,* p. 706-713, 1995.

[Gro99] W. Gropp, E. Lusk, A. Skjellum. *Using MPI – Portable Parallel Programming with the Message–Passing Interface*. The MIT Press, segunda edição, 1999.

[Har03] C. R. Harrel, R. N. Price. Simulation modeling using promodel technology. *Proceedings of the 35th Winter Simulation Conference,* p. 175-181, 2003.

[Hla02] J. Hlavicka, S. Racek. C-Sim – The C language enhancement for discrete-time Simulation. *Proceedings of the International Conference on Dependable Systems and Networks.* p. 539, 2002.

[Hus04] A. Hussain, A.Kapoor, J. Heidemann. The effect of detail on Ethernet simulation. *Proceedings of the 18th Workshop on Parallel and Distributed Simulation,* p. 97-104, 2004.

[Hu01] A. Hu, Y. San, Z, Wang. Verifying and validating a simulation model. *Proceedings of the 33rd Winter Conference Simulation Conference,* p. 595-599, 2001.

[Jha96] V. Jha, R. Bagrodia. A performance evaluation methodology for parallel simulation protocols. *Proceedings of the 10th Workshop on Parallel and Distributed Simulation)*, p. 180–185, 1996.

[Jef85] D. R. Jeferson. Virtual time. *ACM Transaction on Programming Languages and Systems*, v. 7, n. 3, p. 404–425, 1985.

[Kre97] W. Kreutzer, J. Hopkins, M. van Mierlo. SimJava – A framework for modeling queueing networks in Java. *Proceedings of the 29th Winter Simulation Conference*, p. 483-488, 1997.

[Lam04] The LAM/MPI Team. LAM/MPI Users Guide, Version 7.0.4. Open *Systems Lab*. http://www.lam-mpi.org, January, 2004.

[Laz90] E. D. Lazowska. Exploiting lookahead in parallel simulation. *Computer Systems Science and Engineering*, v.1, p. 457–469, 1990.

[Le04] J. Lemiere, B. Smets, P. Cara, E. F. Dirkx. Exploiting symmetry properties for partitioning of models in parallel discrete event simulation. *Proceedings of the 18th Workshop on Parallel and Distributed Simulation*, p. 189-194, 2004.

[L´e98] P. L'Ecuyer. Random number generation. *Banks, J. Handbook of Simulation*, p. 93–137, 1998.

[Lit93] M.C. Little, D. L. McCue. Construction and use of a simulation package in C++. Computing Science Technical Report, University of Newcastle –. n. 337, 1993. (disponível em http://cxxsim.ncl.ac.uk, acessado em 22/01/2004).

[Low99] Y. Low, C. Lim, W. Cai, S. Huang, W. Hsu, S. Jain, S. J.Turner. Survey of languages and runtime libraries for parallel discrete event simulation. *Simulation*, v. 72, n. 3, p. 170–186, 1999.

[Mac87] M. H. MacDougall. *Simulation computer systems - techniques and tools*. The MIT Press, 1987.

[Mac01] M. Machacek e T. Rosgen. Development of a quantitative flow visualization tool for application in industrial wind tunnels. *ICIASF – International Congress of Instrumentation in Aerospace Simulation Facilities/IEEE*, p.125-134, 2001.

[Mas95] E. Mascarenhas, V. Rego, e J. SANG. DISplay: a system for visual – interaction in distributed simulations. *Proceedings of the 27th Winter Simulation Conference*, p. 689-705, 1995.

[Men92] W. Mendehall, T. Sincich. *Statistics for engineering and the sciences*. Dellen Publishing Company, 1992.

[Men99] X. Meng. Distributed simulation: issues and implementations in clusters of workstations environment. Computer Systems Science and Engineering, v. 14, n.1, p. 27–57, 1999.

[Mis86] J. Misra. Distributed discrete-event simulation. *ACM Computing Surveys*, v. 18, n. 1, p.39–65, 1986.

[Mor00] J. C. M. J. Morselli. *Mecanismo para troca dinâmica de protocolos de sincronização*. Tese de Doutorado. Instituto de Física de São Carlos – USP, 2000.

[Nas97] C. Nasu, M. O. Silva. Simulação e SMPL - um estudo. Projeto Final do Curso de Ciência da Computação – DCT – UFMS, 1997.

[Nic96] D. M. Nicol. Principles of conservative parallel simulation. *Proceedings of the 28th Winter Simulation Conference*, p. 128–135, 1996.

[Orl95] R. C. G. S. Orlandi. *Ferramentas para análise de desempenho de sistemas computacionais distribuídos*. Dissertação de Mestrado, Instituto de Ciências Matemáticas de São Carlos - USP, 1995.

[Paw90] K. Pawlikowsky. Steady-state simulation of queueing processes: a survey of problems and solutions. *ACM Computing Surveys*, v. 22, n. 2, p. 123–170, 1990.

[Por97] J. Porras. Improving performance of the Chandy-Misra algorithm to the distributed simulation of a cellular network. *Proceedings of the 29th Winter Simulation Conference*, 1997.

[Ric91] D. O.Rich, R. E. Michelsen An assessment of the ModSim /TWOS parallel simulation environment. *Proceedings of the 23rd Winter Simulation Conference*, p. 509-518, 1991.

[Ril03] P. F. Riley, G. F. Riley. SPaDes – A Distributed agent simulation environment with software – In – The – Loop Execution. *Proceedings of the 35th Winter Simulation Conference*. p.817-825, 2003.

[Roh02] M. W. Rohrer, I. W. McGregor. Simulation reality using AutoMod. *Proceedings of the 34th Winter Simulation Conference*, p.173-181, 2002.

[Sad99] D. Sadowski, V. Bapat. The Arena product family: enterprise modeling solutions. *Proceedings of the 31st Winter Simulation Conference*, p.159-166, 1999.

[San94] R. H. C. Santana, M. J. Santana, R. S. Orlandi, R. Spolon, N. Calônego. *Técnicas para avaliação de desempenho de sistemas computacionais*,. Notas Didáticas - ICMSC – USP, 1994.

[San97] R. H. C. Santana, M. J. Santana, P. S. L. Souza, M. A. Souza, A. E. T. Piekarzki. *Computação Paralela*, Notas de Aula - ICMC – USP, 1997.

[Sat01] B. Stanley. The AutoMod product suite tutorial. *Proceedings of the 33rd Winter Simulation Conference*, p.209-216, 2001.

[Sha88] R. Sharma, L.I. Rose. Modular design for simulation. *Software, Practice and Experience* v. 18, n.10, p. 945-966, 1988.

[Soa92] L. F. G. Soares. *Modelagem e simulação discreta de sistemas.* Editora Campus LTDA, 1992.

[Spo92] R. Spolon. *Extensão funcional de Modula 2 – para simulação de sistemas discreos.* Relatório de Iniciação Cientifica, Instituto de Ciências Matemáticas e de Computação de São Carlos – USP, 1992.

[Spo94] R. Spolon. *Um editor gráfico para um ambiente de simulação automático*. Dissertação de Mestrado, Instituto de Ciências Matemáticas e de Computação de São Carlos – USP, 1994.

[Spo00] R. Spolon, R. S. Ulson, M. J. Santana, R. H. Carlucci. *Simulação distribuída, Time Warp e variantes: Descrição e análise comparativa*, Relatório Técnico, 2000.

[Spo01] R. Spolon. *Um método para avaliação de desempenho de protocolos de sincronização otimistas para simulação distribuída.* Tese de Doutorado, Instituto de Física de São Carlos – USP, 2001.

[Swe01] R. J. Swet, G. R. Drake. The Arena Product Family: Enterprise modeling solutions. *Proceedings of the 33rd Winter Simulation Conference*, p. 9-12, 2001.

[Uls99] R. S. Ulson. *Simulação distribuída em plataformas de portabilidade: viabilidade de uso e comportamento do protocolo CMB*. Tese de Doutorado, Instituto de Física de São Carlos – USP, 1999.

[Var01] A. Varga. The OMNeT++ discrete event simulation system. *Proceedings of the European Simulation Multiconference.* ESM 2001, p. 6-9, 2001.

[Zho04] S. Zhou, S. J. Turner, W. Cai, H. Zhao, X. Pang. A Utility model for timely state update in distributed wargame simulations. *Proceedings of the 18th Workshop on Parallel and Distributed Simulation*, p. 105-11, 2004.

[Won96] P. Wonnacott, D. Bruce. The APOSTLE simulation language: granularity control and performance data. *Proceedings of 10th Workshop on Parallel and Distributed Simulation,*. p. 114-123, 1996.

Anexo A – Implementação dos Modelos Estudados

A.1 Modelo de Filas em Série

```
#include "CmbNula.h"
#include "CmbRand.h"
int main(int argc, char* argv[])
{
 double tc = 3.0,ts1 = 5.0,ts2 = 9.0,te = 10000.0;
 int   REC1,REC2,customer =1,event,token;
 CMB_INIT_SIMULATION(argc,argv,2,"Sistema M/M/1 em Série",0,te);
 REC1   = CMB_INIT_FACILITY(0,"REC1",1);
 REC2 = CMB_INIT_FACILITY(1,"REC2",1);
 CMB_CHANNEL_PROCESS(0,1);
 CMB_LOOKAHEAD(4.0);
 CMB_SCHEDULE(0,1,0.0,customer);  // Escalona um evento.
 while(CMB_TIME() <= te)
 {
    CMB_GET_EVENT(event,token);
    switch(event)
    {
      case 1:  /* Chegada de cliente */
       CMB_SCHEDULE(0,2,0.0,token);
       CMB_SCHEDULE(0,1,CMB_EXPNTL(tc,1),++customer);
       break;
      case 2:  /* Requisição do servidor  */
        if(CMB_REQUEST(0,REC1,event,token,0)==0)
           CMB_SCHEDULE(0,3,CMB_EXPNTL(ts1,2),token);
        break;
      case 3:  /* Fim de serviço */
        CMB_RELEASE(0,REC1,token);
        CMB_SCHEDULE(1,4,0.0,token);
        break;
      case 4:
        if( CMB_REQUEST(1,REC2, event,token, 0) == 0)
             CMB_SCHEDULE(1,5,CMB_EXPNTL(ts2, 3), token);
        break;
      case 5:
        CMB_RELEASE(1,REC2,token);
        break;
      default: break;
    }
  }
 CMB_REPORT();
 CMB_END_SIMULATION(0);
 return 0;
}
```

A.2 Modelo Hipotético 1

A.2.1 Modelo Hipotético 1 na Configuração 1

```
#include "CmbNula.h"
#include "CmbRand.h"
#define Taxa 60
int main(int argc, char* argv[])
```

```
{
  double tc = 1.0,ts[3] = {1.5, 2.0, 2.5},te = 3000.0;
  int Rec_1, Rec_2, Rec_3, Rec_4,prob,customer = 1,token,event;
  CMB_INIT_SIMULATION(argc, argv, 2,"Modelo Hipotético 2 (Quatro Filas
M/M/1 em Serie - Com Feedback)",0,te);
  Rec_1  = CMB_INIT_FACILITY(0,"REC_1",1);
  Rec_2  = CMB_INIT_FACILITY(0,"REC_2",1);
  Rec_3  = CMB_INIT_FACILITY(0,"REC_3",1);
  Rec_4  = CMB_INIT_FACILITY(1,"REC_4",1);
  CMB_CHANNEL_PROCESS(0,1); // estabelece os canais entre os processos
  CMB_LOOKAHEAD(1.5);//define o valor do lookahead para simulação
  CMB_SCHEDULE(0,1,0.0,customer);
  while( CMB_TIME() <= te )
  {
    CMB_GET_EVENT(event,token);
    switch(event)
    {
      case 1:  /* Chegada de cliente */
      CMB_SCHEDULE(0,2,0.0,token);
        CMB_SCHEDULE(0,1,CMB_EXPNTL(tc,2),++customer);
      break;
      case 2:  /* Requisição de servidor */
      if(CMB_REQUEST(0,Rec_1,event,token,0)==0)
           CMB_SCHEDULE(0,3,CMB_EXPNTL(ts[0],3),token);
            break;
      case 3:  /* Fim de serviço */
      CMB_RELEASE(0,Rec_1,token);
        CMB_SCHEDULE(0,4,0.0,token);
        break;
      case 4:
      if( CMB_REQUEST(0,Rec_2, event,token, 0) == 0)
          CMB_SCHEDULE(0,5,CMB_EXPNTL(ts[1],4), token);
       break;
      case 5:
      CMB_RELEASE(0,Rec_2,token);
        CMB_SCHEDULE(0,6,0.0,token);
        break;
      case 6:
      if( CMB_REQUEST(0,Rec_3, event,token, 0) == 0)
          CMB_SCHEDULE(0,7,CMB_EXPNTL(ts[2],5), token);
        break;
      case 7:
      prob = CMB_RANDOM(1, 100,6);
        CMB_RELEASE(0,Rec_3,token);
        if( prob >= Taxa )
          CMB_SCHEDULE(0,4,0.0, token);
        else
         CMB_SCHEDULE(1,8,0.0, token);
      break;
      case 8:
        if( CMB_REQUEST(1,Rec_4, event,token, 0) == 0)
          CMB_SCHEDULE(1,9,CMB_EXPNTL(ts[2],7), token);
        break;
      case 9:
        CMB_RELEASE(1,Rec_4,token);
        break;
     default:break;
    }
 }
  CMB_REPORT();
  CMB_END_SIMULATION(0);
  return 0;
}
```

A.2.2 Modelo Hipotético 1 na Configuração 2

```
#include "CmbNula.h"
#include "CmbRand.h"
#define NTASKS  10 // Número de tarefas.
#define NRES     7 // Número de unidades de discos.
int main(int argc, char* argv[])
{
  double ts[4] = {5.0, 7.0, 4.0, 8.0};
  int Rec_1,Rec_2,Rec_3,Rec_4,Rec_5,Rec_6,Rec_7, event, token, customer =
1, prob;
  CMB_INIT_SIMULATION(argc, argv,2,"Modelo Hipotético (Sete
Recursos)",12,0);
  Rec_1  = CMB_INIT_FACILITY(0,"REC_1",1);
  Rec_2  = CMB_INIT_FACILITY(0,"REC_2",1);
  Rec_3  = CMB_INIT_FACILITY(0,"REC_3",1);
  Rec_4  = CMB_INIT_FACILITY(1,"REC_4",1);
  Rec_5  = CMB_INIT_FACILITY(1,"REC_5",1);
  Rec_6  = CMB_INIT_FACILITY(1,"REC_6",1);
  Rec_7  = CMB_INIT_FACILITY(1,"REC_7",1);
  CMB_CHANNEL_PROCESS(0,1); // estabelece os canais entre os processos
  CMB_LOOKAHEAD(0.002);//define o valor do lookahead para simulação
  for(int i = 0; i < NTASKS; i++)
   CMB_SCHEDULE(0,1,0.0,i);
  while( CMB_TOUR())
  {
    CMB_GET_EVENT(event,token);

    switch(event)
    {
      case 1:  // Inicio do tour.
       CMB_SCHEDULE(0,2,0.0,token);
       break;

      case 2:  // Requisição da cpu.
        if(CMB_REQUEST(0,Rec_1,event,token,0)==0)
           CMB_SCHEDULE(0,3,CMB_EXPNTL(ts[0],1),token);
       break;

      case 3:  // Libera CPU, seleciona disco.
      CMB_RELEASE(0,Rec_1,token);
        CMB_SCHEDULE(0,4,0.0,token);
       break;

      case 4: // Requisita disco.
      if( CMB_REQUEST(0,Rec_2, event,token, 0) == 0)
          CMB_SCHEDULE(0,5,CMB_EXPNTL(ts[1],2), token);
        break;

      case 5:
      CMB_RELEASE(0,Rec_2,token);
        CMB_SCHEDULE(0,6,0.0,token);
        break;
      case 6:
      if( CMB_REQUEST(0,Rec_3, event,token, 0) == 0)
          CMB_SCHEDULE(0,7,CMB_EXPNTL(ts[2],3), token);
        break;
      case 7:
      CMB_RELEASE(0,Rec_3,token);
        CMB_SCHEDULE(1,8,0.0,token);
        break;
```

```
      case 8:
       prob = CMB_RANDOM(1, 100, 4);
        if( prob <= 50 )
        {
       if( CMB_REQUEST(1,Rec_7, event,token, 0) == 0)
            CMB_SCHEDULE(1,9,CMB_EXPNTL(ts[2],5), token);

        }
        else if( prob > 50 && prob <= 60 )
        {
       if(CMB_REQUEST(1,Rec_4, event,token, 0) == 0)
            CMB_SCHEDULE(1,10,CMB_EXPNTL(ts[3],6), token);
        }
        else if( prob > 60 && prob <= 80 )
        {
        if( CMB_REQUEST(1,Rec_5, event,token, 0) == 0)
          CMB_SCHEDULE(1,11,CMB_EXPNTL(ts[0],7), token);
        }
        else if( prob > 80 )
        {
          if( CMB_REQUEST(1,Rec_6, event,token, 0) == 0)
          CMB_SCHEDULE(1,12,CMB_EXPNTL(ts[2],8), token);
        }
        break;
      case 9:
        CMB_RELEASE(1,Rec_7,token);
        CMB_SCHEDULE(0,2,0.0,token);
        CMB_DECREASE_TOUR();
        break;
      case 10:
        CMB_RELEASE(1,Rec_4,token);
        CMB_SCHEDULE(0,6,0.0,token);
        break;
      case 11:
      CMB_RELEASE(1,Rec_5,token);
        CMB_SCHEDULE(0,6,0.0,token);
        break;
      case 12:
       CMB_RELEASE(1,Rec_6,token);
       CMB_SCHEDULE(0,6,0.0,token);
       break;
       default:break;
    }
  }
  CMB_REPORT();
  CMB_END_SIMULATION(0);
  return 0;
}
```

A.3 Modelo Hipotético 2

A.3.1 Modelo Hipotético 2 na Configuração 1

```
#include "CmbNula.h"
#include "CmbRand.h"
#define Taxa 60

int main(int argc, char* argv[])
{
 double tc = 1.0,ts[3] = {1.5, 2.0, 2.5}, te = 3000.0;
 int Rec_1, Rec_2, Rec_3, Rec_4;
```

```
 int  prob,customer = 1,token,event;
 CMB_INIT_SIMULATION(argc, argv, 2,"Modelo Hipotético 2 (Quatro Filas M/M/1
em Serie - Com Feedback)",0,te);
 Rec_1  = CMB_INIT_FACILITY(0,"REC_1",1);
 Rec_2  = CMB_INIT_FACILITY(0,"REC_2",1);
 Rec_3  = CMB_INIT_FACILITY(0,"REC_3",1);
 Rec_4  = CMB_INIT_FACILITY(1,"REC_4",1);
 CMB_CHANNEL_PROCESS(0,1); // estabelece os canais entre os processos
 CMB_LOOKAHEAD(2.0);//define o valor do lookahead para simulacao
 CMB_SCHEDULE(0,1,0.0,customer);
 while( CMB_TIME() <= te )
 {
    CMB_GET_EVENT(event,token);
    switch(event)
    {
      case 1:  /* Chegada de cliente */
      CMB_SCHEDULE(0,2,0.0,token);
        CMB_SCHEDULE(0,1,CMB_EXPNTL(tc,1),++customer);
      break;
      case 2:  /* Requisição do servidor */
      if(CMB_REQUEST(0,Rec_1,event,token,0)==0)
           CMB_SCHEDULE(0,3,CMB_EXPNTL(ts[0],2),token);
            break;
      case 3:  /* Fim de serviço */
      CMB_RELEASE(0,Rec_1,token);
        CMB_SCHEDULE(0,4,0.0,token);
        break;
      case 4:
      if( CMB_REQUEST(0,Rec_2, event,token, 0) == 0)
          CMB_SCHEDULE(0,5,CMB_EXPNTL(ts[1],3), token);
       break;

      case 5:
      CMB_RELEASE(0,Rec_2,token);
        CMB_SCHEDULE(0,6,0.0,token);
        break;
      case 6:
      if( CMB_REQUEST(0,Rec_3, event,token, 0) == 0)
          CMB_SCHEDULE(0,7,CMB_EXPNTL(ts[2],4), token);
        break;
      case 7:
      prob = CMB_RANDOM(1, 100,5);
        CMB_RELEASE(0,Rec_3,token);
        if( prob >= Taxa )
          CMB_SCHEDULE(0,4,0.0, token);
        else
         CMB_SCHEDULE(1,8,0.0, token);
      break;
      case 8:
        if( CMB_REQUEST(1,Rec_4, event,token, 0) == 0)
          CMB_SCHEDULE(1,9,CMB_EXPNTL(ts[2],6), token);
        break;
      case 9:
        CMB_RELEASE(1,Rec_4,token);
        break;
     default:break;
    }
 }
 CMB_REPORT();
 CMB_END_SIMULATION(0);
 return 0;
}
```

A.3.2 Modelo Hipotético 2 na Configuração 2

```
#include "CmbNula.h"
#include "CmbRand.h"
#define Taxa 60
int main(int argc, char* argv[])
{
 double tc = 1.0,ts[3] = {1.5, 2.0, 2.5},te = 3000.0;
 int Rec_1, Rec_2, Rec_3, Rec_4, nts = 1,prob,customer = 1,token, event;
 CMB_INIT_SIMULATION(argc, argv, 2,"Modelo Hipotético 2 (QuatroFilas M/M/1
em Serie - Com Feedback)",0,te);
 Rec_1  = CMB_INIT_FACILITY(0,"REC_1",1);
 Rec_2  = CMB_INIT_FACILITY(0,"REC_2",1);
 Rec_3  = CMB_INIT_FACILITY(1,"REC_3",1);
 Rec_4  = CMB_INIT_FACILITY(1,"REC_4",1);
 CMB_CHANNEL_PROCESS(0,1); // estabelece os canais entre os processos
 CMB_LOOKAHEAD(0.05);
 CMB_SCHEDULE(0,1,0.0,customer);
 while( CMB_TIME() <= te )
 {
   CMB_GET_EVENT(event,token);
   switch(event)
   {
     case 1:  /* Chegada de cliente */
      CMB_SCHEDULE(0,2,0.0,token);
        CMB_SCHEDULE(0,1,CMB_EXPNTL(tc,1),++customer);
      break;
      case 2:  /* Requisição do servidor */
      if(CMB_REQUEST(0,Rec_1,event,token,0)==0)
           CMB_SCHEDULE(0,3,CMB_EXPNTL(ts[0],2),token);
            break;
      case 3:  /* Fim de serviço */
      CMB_RELEASE(0,Rec_1,token);
        CMB_SCHEDULE(0,4,0.0,token);
        break;
      case 4:
      if( CMB_REQUEST(0,Rec_2, event,token, 0) == 0)
          CMB_SCHEDULE(0,5,CMB_EXPNTL(ts[1],3), token);
       break;
      case 5:
      CMB_RELEASE(0,Rec_2,token);
        CMB_SCHEDULE(1,6,0.0,token);
        break;
      case 6:
      if( CMB_REQUEST(1,Rec_3, event,token, 0) == 0)
          CMB_SCHEDULE(1,7,CMB_EXPNTL(ts[2],4), token);
        break;
      case 7:
      prob = CMB_RANDOM(1, 100,5);
        CMB_RELEASE(1,Rec_3,token);
        if( prob >= Taxa )
          CMB_SCHEDULE(0,4,0.0, token);
        else
         CMB_SCHEDULE(1,8,0.0, token);
      break;
      case 8:
        if( CMB_REQUEST(1,Rec_4, event,token, 0) == 0)
          CMB_SCHEDULE(1,9,CMB_EXPNTL(ts[2],6), token);
        break;
      case 9:
        CMB_RELEASE(1,Rec_4,token);
        break;
```

```
      default: break;
    }
  }
 CMB_REPORT();
 CMB_END_SIMULATION(0);
 return 0;
}
```

A.3.3 Modelo Hipotético 2 na Configuração 3

```
#include "CmbNula.h"
#include "CmbRand.h"
#define Taxa 60
int main(int argc, char* argv[])
{
 double tc = 1.0,
        ts[3] = {1.5, 2.0, 2.5},
        te = 3000.0;
  int Rec_1, Rec_2, Rec_3, Rec_4,prob,customer = 1,token,event;
  CMB_INIT_SIMULATION(argc, argv, 3,"Modelo Hipotético 2 (Quatro Filas
M/M/1 em Serie - Com Feedback)",0,te);
  Rec_1  = CMB_INIT_FACILITY(0,"REC_1",1);
  Rec_2  = CMB_INIT_FACILITY(1,"REC_2",1);
  Rec_3  = CMB_INIT_FACILITY(1,"REC_3",1);
  Rec_4  = CMB_INIT_FACILITY(2,"REC_4",1);
  CMB_CHANNEL_PROCESS(0,1); // estabelece os canais entre os processos
  CMB_CHANNEL_PROCESS(1,2); // estabelece os canais entre os processos
  CMB_SCHEDULE(0,1,0.0,customer);
  while( CMB_TIME() <= te )
  {
    CMB_GET_EVENT(event,token);
    switch(event)
    {
      case 1:  /* Chegada de cliente */
      CMB_SCHEDULE(0,2,0.0,token);
        CMB_SCHEDULE(0,1,CMB_EXPNTL(tc,1),++customer);
      break;
      case 2:  /* Requisição do servidor */
      if(CMB_REQUEST(0,Rec_1,event,token,0)==0)
           CMB_SCHEDULE(0,3,CMB_EXPNTL(ts[0],2),token);
            break;
      case 3:  /* Fim de serviço */
      CMB_RELEASE(0,Rec_1,token);
        CMB_SCHEDULE(1,4,0.0,token);
        break;
      case 4:
      if( CMB_REQUEST(1,Rec_2, event,token, 0) == 0)
          CMB_SCHEDULE(1,5,CMB_EXPNTL(ts[1],3), token);
       break;
      case 5:
      CMB_RELEASE(1,Rec_2,token);
        CMB_SCHEDULE(1,6,0.0,token);
        break;
      case 6:
      if( CMB_REQUEST(1,Rec_3, event,token, 0) == 0)
          CMB_SCHEDULE(1,7,CMB_EXPNTL(ts[2],4), token);
        break;
      case 7:
      prob = CMB_RANDOM(1, 100,5);
        CMB_RELEASE(1,Rec_3,token);
        if( prob >= Taxa )
          CMB_SCHEDULE(1,4,0.0, token);
```

```
        else
         CMB_SCHEDULE(2,8,0.0, token);
      break;
      case 8:
        if( CMB_REQUEST(2,Rec_4, event,token, 0) == 0)
          CMB_SCHEDULE(2,9,CMB_EXPNTL(ts[2],6), token);
        break;
      case 9:
        CMB_RELEASE(2,Rec_4,token);
        break;
      default:break;
    }
  }
  CMB_REPORT();
  CMB_END_SIMULATION(0);
  return 0;
}
```

Anexo B: Resultados Utilizados na Análise da CMB-*Simulation*

B.1 Modelo de Filas MM1

B.1.1 Resultados Obtidos com SMPL

0: Execução: Tempo de Execução: 834173

```
                              smpl SIMULATION REPORT
MODEL: Sistema MM1                                TIME:   10000.027
                                                INTERVAL:   10000.027
                    MEAN BUSY     MEAN QUEUE         OPERATION COUNTS
FACILITY    UTIL.   PERIOD         LENGTH     RELEASE   PREEMPT   QUEUE
Servidor    0.9999   1.985       2544.271        5038         0    5038
```

1: Execução : Tempo de Execução: 280065

```
                             smpl SIMULATION REPORT
MODEL: Sistema MM1                                TIME:   10000.606
                                                INTERVAL:   10000.606
                   MEAN BUSY    MEAN QUEUE         OPERATION COUNTS
FACILITY   UTIL.   PERIOD       LENGTH     RELEASE   PREEMPT   QUEUE
Servidor  0.9999   1.986       868.632      5034         0    5034
```

2: Execução - Tempo de Execução: 124269

```
                              smpl SIMULATION REPORT
MODEL: Sistema MM1                                 TIME:   10001.294
                                                 INTERVAL:   10001.294
                   MEAN BUSY  MEAN QUEUE         OPERATION COUNTS
 FACILITY  UTIL.   PERIOD    LENGTH     RELEASE   PREEMPT   QUEUE
 Servidor  0.9919   2.020     78.741      4911         0       4870
```

3: Execução - Tempo de Execução: 110060

```
                              smpl SIMULATION REPORT
MODEL: Sistema MM1                                 TIME:   10001.352
                                                 INTERVAL:   10001.352
                   MEAN BUSY  MEAN QUEUE         OPERATION COUNTS
 FACILITY  UTIL.   PERIOD     LENGTH     RELEASE   PREEMPT   QUEUE
 Servidor 0.4932   0.993      0.492       4969      0        2463
```

4: Execução - Tempo de Execução: 110024

```
                              smpl SIMULATION REPORT
MODEL: Sistema MM1                                TIME:   10000.191
                                                INTERVAL:   10000.191
                    MEAN BUSY  MEAN QUEUE         OPERATION COUNTS
FACILITY    UTIL.   PERIOD     LENGTH    RELEASE   PREEMPT   QUEUE
Servidor    0.7296   1.478      1.755      4937        0    3614
```

5: Execução - Tempo de Execução: 108720

```
                              smpl SIMULATION REPORT
MODEL: Sistema MM1                                 TIME:   10004.752
                                                 INTERVAL:   10004.752
                   MEAN BUSY  MEAN QUEUE         OPERATION COUNTS
 FACILITY  UTIL.   PERIOD     LENGTH     RELEASE   PREEMPT   QUEUE
 Servidor 0.8456   2.508      3.382       3373         0       2799
```

6: Execução - Tempo de Execução: 438922

```
                              smpl SIMULATION REPORT
 MODEL: Sistema MM1                                 TIME:   10000.926
                                                 INTERVAL:   10000.926
                      MEAN BUSY  MEAN QUEUE         OPERATION COUNTS
 FACILITY   UTIL.     PERIOD       LENGTH      RELEASE   PREEMPT   QUEUE
 Servidor  1.0000     3.000        1732.225     3334         0        3334
```

7: Execução - Tempo de Execução: 106301

```
                              smpl SIMULATION REPORT
 MODEL: Sistema MM1                                  TIME:   10002.597
                                                  INTERVAL:   10002.597
                    MEAN BUSY  MEAN QUEUE         OPERATION COUNTS
FACILITY   UTIL.    PERIOD      LENGTH      RELEASE   PREEMPT   QUEUE
Servidor   0.7470   3.010       1.735       2482        0         1840
```

8: Execução - Tempo de Execução: 150560

```
                              smpl SIMULATION REPORT
MODEL: Sistema MM1                                  TIME:   10002.090
                                                  INTERVAL:   10002.090
                    MEAN BUSY  MEAN QUEUE         OPERATION COUNTS
FACILITY   UTIL.    PERIOD      LENGTH      RELEASE   PREEMPT   QUEUE
Servidor  1.0000    4.023       457.030      2486         0       2486
```

9: Execução - Tempo de Execução: 114529

```
                              smpl SIMULATION REPORT
MODEL: Sistema MM1                                  TIME:   10001.220
                                                  INTERVAL:   10001.220
                     MEAN BUSY  MEAN QUEUE         OPERATION COUNTS
FACILITY   UTIL.     PERIOD      LENGTH      RELEASE   PREEMPT   QUEUE
Servidor  1.0000     3.977       113.577      2515         0        2515
```

10: Execução - Tempo de Execução: 373673

```
                              smpl SIMULATION REPORT
MODEL: Sistema MM1                                  TIME:   10003.494
                                                   INTERVAL:   10003.494
                    MEAN BUSY   MEAN QUEUE         OPERATION COUNTS
FACILITY  UTIL.     PERIOD       LENGTH      RELEASE   PREEMPT   QUEUE
Servidor  0.9996    2.511       1374.225     3983         0        3983
```

11: Execução - Tempo de Execução: 105794

```
                              smpl SIMULATION REPORT
 MODEL: Sistema MM1                                   TIME:   10000.994
                                                   INTERVAL:   10000.994
                   MEAN BUSY  MEAN QUEUE         OPERATION COUNTS
 FACILITY  UTIL.   PERIOD      LENGTH    RELEASE   PREEMPT   QUEUE
 Servidor 0.6442   1.960       1.130      3288        0        2125
```

12: Execução - Tempo de Execução: 105877

```
                              smpl SIMULATION REPORT
MODEL: Sistema MM1                                   TIME:   10010.918
                                                   INTERVAL:   10010.918
                    MEAN BUSY  MEAN QUEUE         OPERATION COUNTS
FACILITY   UTIL.    PERIOD      LENGTH      RELEASE   PREEMPT   QUEUE
Servidor   1.0000   7.596       48.589       1318          0    1318
```

13: Execução - Tempo de Execução: 115051

```
                              smpl SIMULATION REPORT
MODEL: Sistema MM1                                   TIME:   10000.123
                                                   INTERVAL:   10000.123
                   MEAN BUSY   MEAN QUEUE         OPERATION COUNTS
FACILITY   UTIL.   PERIOD      LENGTH      RELEASE   PREEMPT   QUEUE
Servidor  0.9998   5.962        190.213        1677          0    1677
```

14: Execução - Tempo de Execução: 104867

```
                              smpl SIMULATION REPORT
MODEL: Sistema MM1                                    TIME:   10001.818
                                                    INTERVAL:   10001.818
                   MEAN BUSY  MEAN QUEUE         OPERATION COUNTS
FACILITY  UTIL.    PERIOD     LENGTH     RELEASE   PREEMPT   QUEUE
Servidor  0.8731   4.417      5.259       1977         0    1710
```

15: Execução - Tempo de Execuçã: 160210

```
                              smpl SIMULATION REPORT
MODEL: Sistema MM1                                    TIME:   10002.346
                                                    INTERVAL:   10002.346
                    MEAN BUSY  MEAN QUEUE          OPERATION COUNTS
FACILITY   UTIL.    PERIOD     LENGTH     RELEASE   PREEMPT   QUEUE
Servidor  0.9995    4.588      555.281     2179         0       2179
```

B.1.2 Resultados Obtidos na CMB-*Simulation*

0: Execução - Tempo de Execução: 183247

```
CMB SIMULATION.REPORT(HOST: work139 - RANK:0 -- NHOSTS:1)
MODEL : Sistema MM1
                                                 LVT : 10000.027
                                               INTERVAL : 10000.027
                     MEAN BUSY   MEAN QUEUE      OPERATION COUNTS
FACILITY     UTIL.   PERIOD      LENGTH     RELEASE  PREEMPT  QUEUE
Servidor-[1] 0.9999  1.985      2544.271     5038       0      5038
======================================================================
NULL MESSAGE COUNTS       COMPLETE MESSAGE COUNTS
          0                          0
======================================================================
```

1: Execução - Tempo de Execução: 51685

```
CMB SIMULATION.REPORT(HOST: work139 - RANK:0 -- NHOSTS:1)
MODEL : Sistema MM1
                                                 LVT : 10000.606
                                               INTERVAL : 10000.606
                     MEAN BUSY   MEAN QUEUE          OPERATION COUNTS
FACILITY     UTIL.   PERIOD      LENGTH         RELEASE  PREEMPT  QUEUE
Servidor-[1] 0.9999  1.986       868.632        5034       0      5034
======================================================================
NULL MESSAGE COUNTS       COMPLETE MESSAGE COUNTS
      0                          0
======================================================================
```

2: Execução - Tempo de Execução: 13304

```
CMB SIMULATION.REPORT(HOST: work139 - RANK:0 -- NHOSTS:1)
MODEL : Sistema MM1
                                                 LVT : 10001.294
                                               INTERVAL : 10001.294
                      MEAN BUSY   MEAN QUEUE          OPERATION COUNTS
FACILITY     UTIL.    PERIOD      LENGTH         RELEASE  PREEMPT  QUEUE
Servidor-[1] 0.9919   2.020       78.741          4911       0      4870
======================================================================
NULL MESSAGE COUNTS       COMPLETE MESSAGE COUNTS
        0                          0
======================================================================
```

3: Execução - Tempo de Execução: 9494

```
CMB SIMULATION.REPORT(HOST: work139 - RANK:0 -- NHOSTS:1)
MODEL : Sistema MM1
                                                 LVT : 10001.352
```

```
                                          INTERVAL : 10001.352
                    MEAN BUSY   MEAN QUEUE       OPERATION COUNTS
FACILITY      UTIL.  PERIOD      LENGTH       RELEASE  PREEMPT  QUEUE
Servidor-[1] 0.4932  0.993       0.492        4969       0      2463
=========================================================================
NULL MESSAGE COUNTS       COMPLETE MESSAGE COUNTS
    0                        0
=========================================================================
```

4: Execução – Tempo de Execução: 10074

```
CMB SIMULATION.REPORT(HOST: work139 - RANK:0 -- NHOSTS:1)
MODEL : Sistema MM1
                                             LVT : 10000.191
                                          INTERVAL : 10000.191
                     MEAN BUSY   MEAN QUEUE         OPERATION COUNTS
FACILITY      UTIL.   PERIOD      LENGTH      RELEASE  PREEMPT  QUEUE
Servidor-[1] 0.7296   1.478       1.755       4937       0      3614
=========================================================================
NULL MESSAGE COUNTS       COMPLETE MESSAGE COUNTS
      0                        0
=========================================================================
```

5: Execução - Tempo de Execução: 7579

```
CMB SIMULATION.REPORT(HOST: work139 - RANK:0 -- NHOSTS:1)
MODEL : Sistema MM1
                                             LVT : 10004.752
                                          INTERVAL : 10004.752
                    MEAN BUSY   MEAN QUEUE         OPERATION COUNTS
FACILITY      UTIL.   PERIOD     LENGTH     RELEASE  PREEMPT  QUEUE
Servidor-[1]  0.8456  2.508      3.382       3373       0      2799
=========================================================================
NULL MESSAGE COUNTS       COMPLETE MESSAGE COUNTS
        0                        0
=========================================================================
```

6: Execução – Tempo de Execução: 86973

```
CMB SIMULATION.REPORT(HOST: work139 - RANK:0 -- NHOSTS:1)
MODEL : Sistema MM1
                                             LVT : 10000.926
                                          INTERVAL : 10000.926
                   MEAN BUSY   MEAN QUEUE         OPERATION COUNTS
FACILITY     UTIL.   PERIOD      LENGTH        RELEASE  PREEMPT  QUEUE
Servidor-[1] 1.0000   3.000      1732.225      3334       0       3334
=========================================================================
NULL MESSAGE COUNTS       COMPLETE MESSAGE COUNTS
        0                        0
=========================================================================
```

7: Execução - Tempo de Execução: 89643

```
CMB SIMULATION.REPORT(HOST: work139 - RANK:0 -- NHOSTS:1)
MODEL : Sistema MM1
                                             LVT : 10002.597
                                          INTERVAL : 10002.597
                    MEAN BUSY  MEAN QUEUE       OPERATION COUNTS
FACILITY     UTIL.   PERIOD     LENGTH       RELEASE  PREEMPT  QUEUE
Servidor-[1] 0.7470  3.010      1.735         2482       0      1840
=========================================================================
NULL MESSAGE COUNTS       COMPLETE MESSAGE COUNTS
        0                        0
=========================================================================
```

8: Execução - Tempo de Execução: 17503

```
CMB SIMULATION.REPORT(HOST: work139 - RANK:0 -- NHOSTS:1)
MODEL : Sistema MM1
                                              LVT : 10002.090
                                         INTERVAL : 10002.090
                    MEAN BUSY  MEAN QUEUE          OPERATION COUNTS
FACILITY     UTIL.   PERIOD     LENGTH         RELEASE  PREEMPT  QUEUE
Servidor-[1] 1.0000  4.023      457.030          2486       0      2486
=========================================================================
NULL MESSAGE COUNTS       COMPLETE MESSAGE COUNTS
         0                          0
=========================================================================
```

9: Execução - Tempo de Execução: 8885

```
CMB SIMULATION.REPORT(HOST: work139 - RANK:0 -- NHOSTS:1)
MODEL : Sistema MM1
                                              LVT : 10001.220
                                         INTERVAL : 10001.220
                    MEAN BUSY  MEAN QUEUE          OPERATION COUNTS
FACILITY     UTIL.   PERIOD     LENGTH         RELEASE  PREEMPT  QUEUE
Servidor-[1] 1.0000  3.977      113.577          2515       0      2515
=========================================================================
NULL MESSAGE COUNTS       COMPLETE MESSAGE COUNTS
        0                          0
=========================================================================
```

10: Execução - Tempo de Execução:72997

```
CMB SIMULATION.REPORT(HOST: work139 - RANK:0 -- NHOSTS:1)
MODEL : Sistema MM1
                                              LVT : 10003.494
                                         INTERVAL : 10003.494
                    MEAN BUSY   MEAN QUEUE          OPERATION COUNTS
FACILITY      UTIL.   PERIOD     LENGTH       RELEASE  PREEMPT  QUEUE
Servidor-[1] 0.9996   2.511      1374.225     3983       0      3983
=========================================================================
NULL MESSAGE COUNTS       COMPLETE MESSAGE COUNTS
         0                          0
=========================================================================
```

11: Execução - Tempo de Execução:16023

```
CMB SIMULATION.REPORT(HOST: work139 - RANK:0 -- NHOSTS:1)
MODEL : Sistema MM1
                                              LVT : 10000.994
                                         INTERVAL : 10000.994
                    MEAN BUSY   MEAN QUEUE          OPERATION COUNTS
FACILITY     UTIL.   PERIOD      LENGTH         RELEASE  PREEMPT  QUEUE
Servidor-[1] 0.6442  1.960       1.130          3288       0      2125
=========================================================================
NULL MESSAGE COUNTS       COMPLETE MESSAGE COUNTS
       0                          0
=========================================================================
```

12: Execução - Tempo de Execução:4945

```
CMB SIMULATION.REPORT(HOST: work139 - RANK:0 -- NHOSTS:1)
MODEL : Sistema MM1
                                              LVT : 10010.918
                                         INTERVAL : 10010.918
                    MEAN BUSY   MEAN QUEUE          OPERATION COUNTS
FACILITY     UTIL.   PERIOD     LENGTH      RELEASE  PREEMPT  QUEUE
Servidor-[1] 1.0000  7.596      48.589       1318       0      1318
```

```
=====================================================================
NULL MESSAGE COUNTS       COMPLETE MESSAGE COUNTS
          0                        0
=====================================================================

13: Execução:Tempo de Execução:7957
CMB SIMULATION.REPORT(HOST: work139 - RANK:0 -- NHOSTS:1)
MODEL : Sistema MM1
                                            LVT : 10000.123
                                       INTERVAL : 10000.123
                    MEAN BUSY   MEAN QUEUE         OPERATION COUNTS
FACILITY     UTIL.   PERIOD      LENGTH        RELEASE  PREEMPT  QUEUE
Servidor-[1] 0.9998  5.962      190.213         1677       0      1677
=====================================================================
NULL MESSAGE COUNTS       COMPLETE MESSAGE COUNTS
       0                        0
=====================================================================

14: Execução - Tempo de Execução:5481
CMB SIMULATION.REPORT(HOST: work139 - RANK:0 -- NHOSTS:1)
MODEL : Sistema MM1
                                            LVT : 10001.818
                                       INTERVAL : 10001.818
                    MEAN BUSY  MEAN QUEUE         OPERATION COUNTS
FACILITY     UTIL.   PERIOD     LENGTH        RELEASE  PREEMPT  QUEUE
Servidor-[1] 0.8731  4.417      5.259         1977       0      1710
=====================================================================
NULL MESSAGE COUNTS       COMPLETE MESSAGE COUNTS
      0                        0
=====================================================================

15: Execução - Tempo de Execução:19598
CMB SIMULATION.REPORT(HOST: work139 - RANK:0 -- NHOSTS:1)
MODEL : Sistema MM1
                                            LVT : 10002.346
                                       INTERVAL : 10002.346
                     MEAN BUSY   MEAN QUEUE         OPERATION COUNTS
FACILITY      UTIL.   PERIOD      LENGTH     RELEASE  PREEMPT  QUEUE
Servidor-[1]  0.9995  4.588       555.281     2179       0      2179
=====================================================================
NULL MESSAGE COUNTS       COMPLETE MESSAGE COUNTS
       0                        0
=====================================================================
```

B.1.3 Resultados Obtidos na CMB-*Simulation* Sob Demanda

```
0:Execução - Tempo de Execução:184277
CMB SIMULATION.REPORT(HOST: work139 - RANK:0 -- NHOSTS:1)
MODEL : Sistema MM1
                                            LVT : 10000.027
                                       INTERVAL : 10000.027

                    MEAN BUSY   MEAN QUEUE         OPERATION COUNTS
FACILITY     UTIL.   PERIOD      LENGTH        RELEASE  PREEMPT  QUEUE
Servidor-[1] 0.9999  1.985      2544.271        5038       0      5038
=====================================================================
NULL MESSAGE COUNTS       COMPLETE MESSAGE COUNTS
     0                        0
=====================================================================

1: Execução - Tempo de Execução:52804
```

```
CMB SIMULATION.REPORT(HOST: work139 - RANK:0 -- NHOSTS:1)
MODEL : Sistema MM1
                                              LVT : 10000.606
                                         INTERVAL : 10000.606
                     MEAN BUSY   MEAN QUEUE          OPERATION COUNTS
FACILITY     UTIL.    PERIOD      LENGTH      RELEASE  PREEMPT  QUEUE
Servidor-[1] 0.9999  1.986       868.632   5034       0      5034
=========================================================================
NULL MESSAGE COUNTS       COMPLETE MESSAGE COUNTS
       0                          0
=========================================================================
```

2: Execução - Tempo de Execução:13334

```
CMB SIMULATION.REPORT(HOST: work139 - RANK:0 -- NHOSTS:1)
MODEL : Sistema MM1
                                              LVT : 10001.294
                                         INTERVAL : 10001.294
                     MEAN BUSY  MEAN QUEUE          OPERATION COUNTS
FACILITY     UTIL.    PERIOD     LENGTH         RELEASE  PREEMPT  QUEUE
Servidor-[1] 0.9919  2.020      78.741          4911       0      4870
=========================================================================
NULL MESSAGE COUNTS       COMPLETE MESSAGE COUNTS
     0                          0
=========================================================================
```

3: Execução - Tempo de Execução:10080

```
CMB SIMULATION.REPORT(HOST: work139 - RANK:0 -- NHOSTS:1)
MODEL : Sistema MM1
                                              LVT : 10001.352
                                         INTERVAL : 10001.352
                      MEAN BUSY  MEAN QUEUE          OPERATION COUNTS
FACILITY      UTIL.    PERIOD     LENGTH      RELEASE  PREEMPT  QUEUE
Servidor-[1]  0.4932  0.993      0.492        4969       0      2463
=========================================================================
NULL MESSAGE COUNTS       COMPLETE MESSAGE COUNTS
       0                          0
=========================================================================
```

4: Execução - Tempo de Execução:10322

```
CMB SIMULATION.REPORT(HOST: work139 - RANK:0 -- NHOSTS:1)
MODEL : Sistema MM1
                                              LVT : 10000.191
                                         INTERVAL : 10000.191
                     MEAN BUSY  MEAN QUEUE       OPERATION COUNTS
FACILITY     UTIL.    PERIOD     LENGTH       RELEASE  PREEMPT  QUEUE
Servidor-[1] 0.7296  1.478      1.755         4937       0      3614
=========================================================================
NULL MESSAGE COUNTS       COMPLETE MESSAGE COUNTS
       0                          0
=========================================================================
```

5: Execução - Tempo de Execução:7944

```
CMB SIMULATION.REPORT(HOST: work139 - RANK:0 -- NHOSTS:1)
MODEL : Sistema MM1
                                              LVT : 10004.752
                                         INTERVAL : 10004.752

                      MEAN BUSY  MEAN QUEUE     OPERATION COUNTS
FACILITY     UTIL.     PERIOD     LENGTH     RELEASE  PREEMPT  QUEUE
Servidor-[1] 0.8456   2.508      3.382       3373       0      2799
=========================================================================
```

```
NULL MESSAGE COUNTS       COMPLETE MESSAGE COUNTS
         0                         0
=========================================================================

6: Execução - Tempo de Execução:7981
CMB SIMULATION.REPORT(HOST: work139 - RANK:0 -- NHOSTS:1)
MODEL : Sistema MM1
                                              LVT : 10004.752
                                         INTERVAL : 10004.752
                     MEAN BUSY   MEAN QUEUE      OPERATION COUNTS
FACILITY       UTIL.   PERIOD     LENGTH      RELEASE  PREEMPT  QUEUE
Servidor-[1]  0.8456   2.508      3.382         3373       0      2799
=========================================================================
NULL MESSAGE COUNTS       COMPLETE MESSAGE COUNTS
        0                         0
=========================================================================
7: Execução - Tempo de Execução:87681
CMB SIMULATION.REPORT(HOST: work139 - RANK:0 -- NHOSTS:1)
MODEL : Sistema MM1
                                              LVT : 10000.926
                                         INTERVAL : 10000.926
                     MEAN BUSY   MEAN QUEUE         OPERATION COUNTS
FACILITY      UTIL.    PERIOD      LENGTH      RELEASE  PREEMPT  QUEUE
Servidor-[1] 1.0000   3.000      1732.225      3334       0      3334
=========================================================================
NULL MESSAGE COUNTS       COMPLETE MESSAGE COUNTS
       0                         0
=========================================================================

8: Execução - Tempo de Execução:17971
CMB SIMULATION.REPORT(HOST: work139 - RANK:0 -- NHOSTS:1)
MODEL : Sistema MM1
                                              LVT : 10002.090
                                         INTERVAL : 10002.090
                    MEAN BUSY   MEAN QUEUE         OPERATION COUNTS
FACILITY     UTIL.    PERIOD      LENGTH       RELEASE  PREEMPT  QUEUE
Servidor-[1] 1.0000   4.023       457.030       2486       0      2486
=========================================================================
NULL MESSAGE COUNTS       COMPLETE MESSAGE COUNTS
         0                         0
=========================================================================

9: Execução  - Tempo de Execução:9135
CMB SIMULATION.REPORT(HOST: work139 - RANK:0 -- NHOSTS:1)
MODEL : Sistema MM1
                                              LVT : 10001.220
                                         INTERVAL : 10001.220
                   MEAN BUSY   MEAN QUEUE         OPERATION COUNTS
FACILITY    UTIL.   PERIOD      LENGTH        RELEASE  PREEMPT  QUEUE
Servidor-[1]1.0000  3.977       113.577        2515       0      2515
=========================================================================
NULL MESSAGE COUNTS       COMPLETE MESSAGE COUNTS
       0                         0
=========================================================================

10: Execução - Tempo de Execução:73865
CMB SIMULATION.REPORT(HOST: work139 - RANK:0 -- NHOSTS:1)
MODEL : Sistema MM1
                                              LVT : 10003.494
                                         INTERVAL : 10003.494
                     MEAN BUSY   MEAN QUEUE       OPERATION COUNTS
FACILITY      UTIL.    PERIOD     LENGTH      RELEASE  PREEMPT  QUEUE
Servidor-[1]  0.9996   2.511      1374.225     3983       0      3983
```

```
========================================================================
NULL MESSAGE COUNTS       COMPLETE MESSAGE COUNTS
        0                         0
========================================================================
```

11: Execução - Tempo de Execução:7537

```
CMB SIMULATION.REPORT(HOST: work139 - RANK:0 -- NHOSTS:1)
MODEL : Sistema MM1
                                              LVT : 10000.994
                                         INTERVAL : 10000.994
                    MEAN BUSY   MEAN QUEUE        OPERATION COUNTS
FACILITY     UTIL.   PERIOD      LENGTH        RELEASE  PREEMPT  QUEUE
servidor-[1] 0.6442  1.960       1.130         3288       0      2125
========================================================================
NULL MESSAGE COUNTS       COMPLETE MESSAGE COUNTS
        0                         0
========================================================================
```

12: Execução - Tempo de Execução:5102

```
CMB SIMULATION.REPORT(HOST: work139 - RANK:0 -- NHOSTS:1)
MODEL : Sistema MM1
                                              LVT : 10010.918
                                         INTERVAL : 10010.918
                    MEAN BUSY  MEAN QUEUE     OPERATION COUNTS
FACILITY     UTIL.   PERIOD     LENGTH       RELEASE  PREEMPT  QUEUE
Servidor-[1] 1.0000  7.596      48.589        1318       0       1318
========================================================================
NULL MESSAGE COUNTS       COMPLETE MESSAGE COUNTS
     0                         0
========================================================================
```

13: Execução - Tempo de Execução:8205

```
CMB SIMULATION.REPORT(HOST: work139 - RANK:0 -- NHOSTS:1)
MODEL : Sistema MM1
                                              LVT : 10000.123
                                         INTERVAL : 10000.123
                     MEAN BUSY   MEAN QUEUE     OPERATION COUNTS
FACILITY      UTIL.    PERIOD      LENGTH       RELEASE  PREEMPT  QUEUE
Servidor-[1] 0.9998    5.962      190.213       1677       0      1677
========================================================================
NULL MESSAGE COUNTS       COMPLETE MESSAGE COUNTS
        0                         0
========================================================================
```

14: Execução - Tempo de Execução:5597

```
CMB SIMULATION.REPORT(HOST: work139 - RANK:0 -- NHOSTS:1)
MODEL : Sistema MM1
                                              LVT : 10001.818
                                         INTERVAL : 10001.818
                     MEAN BUSY   MEAN QUEUE         OPERATION COUNTS
FACILITY      UTIL.    PERIOD   LENGTH      RELEASE  PREEMPT  QUEUE
Servidor-[1]  0.8731   4.417    5.259        1977       0      1710
========================================================================
NULL MESSAGE COUNTS       COMPLETE MESSAGE COUNTS
        0                         0
========================================================================
```

15: Execução - Tempo de Execução:19768

```
CMB SIMULATION.REPORT(HOST: work139 - RANK:0 -- NHOSTS:1)
MODELO: Sistema MM1
                                              LVT : 10002.346
                                         INTERVAL : 10002.346
```

```
                      MEAN BUSY   MEAN QUEUE     OPERATION COUNTS
FACILITY     UTIL.     PERIOD       LENGTH     RELEASE  PREEMPT  QUEUE
Servidor-[1] 0.9995   4.588       555.281     2179       0      2179
=====================================================================
NULL MESSAGE COUNTS       COMPLETE MESSAGE COUNTS
        0                         0
=====================================================================
```

B.2 Modelo Sistema Computacional Simplificado

B.2.1 Resultados Obtidos com *SMPL*

1: Execução - Tempo de Execução: 293715 (tc = 1.0-tcpu = 1.5-tdisco = 2.0)

```
                            smpl SIMULATION REPORT
MODEL: Sistema Computacional Simplificado        TIME:    5001.478
                                                INTERVAL:    5001.478
                   MEAN BUSY     MEAN QUEUE        OPERATION COUNTS
FACILITY   UTIL.   PERIOD        LENGTH     RELEASE   PREEMPT   QUEUE
CPU        0.9994  1.519      1222.985       3291         0    3290
Disco      0.2767  2.035         0.116        680         0     194
```

2: Execução – Tempo de Execução: 1287058 (tc = 0.5-tcpu = 1.5-tdisco = 2.0)

```
                            smpl SIMULATION REPORT
MODEL: Sistema Computacional Simplificado         TIME:    5000.002
                                                 INTERVAL:    5000.002
                    MEAN BUSY  MEAN QUEUE      OPERATION COUNTS
FACILITY   UTIL.    PERIOD     LENGTH     RELEASE   PREEMPT   QUEUE
CPU        0.9997    1.519     3729.470    3291         0    3291
Disco      0.2768    2.035        0.116     680         0     194
```

3: Execução - Tempo de Execução: 208066 (tc = 1.0-tcpu = 1.0-tdisco = 2.0)

```
                            smpl SIMULATION REPORT
MODEL: Sistema Computacional Simplificado           TIME:    5000.422
                                                   INTERVAL:    5000.422
                    MEAN BUSY  MEAN QUEUE        OPERATION COUNTS
FACILITY    UTIL.   PERIOD     LENGTH     RELEASE   PREEMPT   QUEUE
CPU        1.0000   0.996      555.199     5020         0    5019
Disco      0.4076   1.963        0.276       1038         0     408
```

4: Execução - Tempo de Execução: 109076 (tc = 1.5-tcpu = 1.0-tdisco = 2.0)

```
                            smpl SIMULATION REPORT
MODEL: Sistema Computacional Simplificado           TIME:    5000.008
                                                  INTERVAL:    5000.008
                      MEAN BUSY   MEAN QUEUE     OPERATION COUNTS
FACILITY     UTIL.     PERIOD      LENGTH    RELEASE   PREEMPT   QUEUE
CPU         0.8513      0.999      5.504       4262      0       3624
Disco       0.3451      1.974      0.161        874      0        272
```

5: Execução - Tempo de Execução: 305016 (tc = 1.0-tcpu = 1.5-tdisco = 1.5)

```
                            smpl SIMULATION REPORT
MODEL: Sistema Computacional Simplificado           TIME:    5001.478
                                                   INTERVAL:    5001.478
                     MEAN BUSY   MEAN QUEUE        OPERATION COUNTS
FACILITY     UTIL.    PERIOD       LENGTH     RELEASE   PREEMPT   QUEUE
CPU         0.9994    1.519      1223.106      3291         0    3290
Disco       0.2075    1.526         0.064       680         0     144
```

6: Execução - Tempo de Execução: 309354 (tc = 1.0-tcpu = 1.5-tdisco = 1.0)

```
                              smpl SIMULATION REPORT
MODEL: Sistema Computacional Simplificado            TIME:    5001.478
                                                  INTERVAL:    5001.478
                       MEAN BUSY  MEAN QUEUE        OPERATION COUNTS
FACILITY    UTIL.      PERIOD      LENGTH     RELEASE   PREEMPT   QUEUE
CPU        0.9994     1.519      1223.211     3291         0    3290
Disco      0.1384     1.018       0.028        680         0     102
```

7: Execução - Tempo de Execução: 208171 (tc = 1.0-tcpu = 1.0-tdisco = 1.5)

```
                              smpl SIMULATION REPORT
MODEL: Sistema Computacional Simplificado            TIME:    5000.422
                                                  INTERVAL:    5000.422
                      MEAN BUSY   MEAN QUEUE        OPERATION COUNTS
FACILITY    UTIL.      PERIOD      LENGTH     RELEASE   PREEMPT   QUEUE
CPU        1.0000      0.996       555.431    5020         0      5019
Disco      0.3057      1.473         0.145    1038         0      322
```

8: Execução - Tempo de Execução: 203069 (tc = 1.0-tcpu = 1.5-tdisco = 3.0)

```
                              smpl SIMULATION REPORT
MODEL: Sistema Computacional Simplificado            TIME:    5000.422
                                                  INTERVAL:    5000.422
                     MEAN BUSY     MEAN QUEUE        OPERATION COUNTS
FACILITY    UTIL.      PERIOD      LENGTH     RELEASE   PREEMPT   QUEUE
CPU        1.0000      0.996       554.389     5020         0      5019
Disco      0.6113      2.945        0.882      1038         0      625
```

9: Execução - Tempo de Execuçã: 304695 (tc = 1.0-tcpu = 1.5-tdisco = 2.5)

```
                              smpl SIMULATION REPORT
MODEL: Sistema Computacional Simplificado            TIME:    5001.478
                                                  INTERVAL:    5001.478
                     MEAN BUSY   MEAN QUEUE        OPERATION COUNTS
FACILITY    UTIL.     PERIOD     LENGTH     RELEASE   PREEMPT   QUEUE
CPU        0.9994     1.519      1222.846    3291       0        3290
Disco      0.3459     2.544       0.185      680        0        228
```

10:Execução - Tempo de Execução: 351520 (tc = 1.0-tcpu = 2.0-tdisco = 3.0)

```
                              smpl SIMULATION REPORT
MODEL: Sistema Computacional Simplificado            TIME:    5000.077
                                                  INTERVAL:    5000.077
                     MEAN BUSY   MEAN QUEUE        OPERATION COUNTS
FACILITY     UTIL.    PERIOD      LENGTH     RELEASE   PREEMPT   QUEUE
CPU         1.0000    2.043      1550.311    2448       0        2448
Disco       0.3085    3.025       0.145       510         0     157
```

11: Execução - Tempo de Execução: 352364 (tc = 1.0-tcpu = 2.0-tdisco = 2.0)

```
                              smpl SIMULATION REPORT
MODEL: Sistema Computacional Simplificado            TIME:    5000.077
                                                  INTERVAL:    5000.077
                   MEAN BUSY  MEAN QUEUE        OPERATION COUNTS
FACILITY   UTIL.    PERIOD     LENGTH     RELEASE   PREEMPT   QUEUE
CPU        1.0000   2.043     1550.497    2448       0        2448
Disco      0.2057   2.016      0.062       510       0         110
```

12: Execução - Tempo de Execução: 105726 (tc = 2.0-tcpu = 1.5-tdisco = 2.0)

```
                              smpl SIMULATION REPORT
MODEL: Sistema Computacional Simplificado            TIME:    5001.937
                                                  INTERVAL:    5001.937
                      MEAN BUSY   MEAN QUEUE        OPERATION COUNTS
FACILITY    UTIL.      PERIOD      LENGTH     RELEASE   PREEMPT   QUEUE
CPU        0.9596      1.520       18.922      3157        0       3026
Disco      0.2663      2.040        0.109       653        0        176
```

13: Execução - Tempo de Execução: 105861 (tc = 2.0-tcpu = 1.5-tdisco = 3.0)

```
                              smpl SIMULATION REPORT
MODEL: Sistema Computacional Simplificado           TIME:    5001.937
                                                  INTERVAL:    5001.937
                     MEAN BUSY   MEAN QUEUE        OPERATION COUNTS
FACILITY    UTIL.    PERIOD       LENGTH     RELEASE   PREEMPT   QUEUE
CPU         0.9596   1.520        19.104        3157         0    3035
Disco       0.3995   3.060         0.255         653         0     242
```

14:Execução - Tempo de Execução: 105363 (tc = 2.0-tcpu = 1.5-tdisco = 2.0)

```
                              smpl SIMULATION REPORT
MODEL: Sistema Computacional Simplificado           TIME:    5001.937
                                                  INTERVAL:    5001.937
                     MEAN BUSY   MEAN QUEUE        OPERATION COUNTS
FACILITY    UTIL.    PERIOD     LENGTH     RELEASE   PREEMPT   QUEUE
CPU        0.9596     1.520      18.922     3157         0      3026
Disco      0.2663      2.040      0.109      653         0       176
```

15: Execução - Tempo de Execução: 150622 (tc = 1.5-tcpu = 1.5-tdisco = 2.0)

```
                              smpl SIMULATION REPORT
MODEL: Sistema Computacional Simplificado           TIME:    5000.470
                                                  INTERVAL:    5000.470
                   MEAN BUSY  MEAN QUEUE        OPERATION COUNTS
FACILITY    UTIL.  PERIOD     LENGTH     RELEASE   PREEMPT   QUEUE
CPU         0.9996  1.519     394.026     3291       0       3290
Disco       0.2768  2.035     0.116       680        0        194
```

B.2.2 Resultados Obtidos na CMB-*Simulation*

1: Execução (tc =1.0 - tcpu = 1.5 - tdisco = 2.0)

P0:

```
Tempo de Execução: 786074553
CMB SIMULATION.REPORT(HOST: work139 - RANK:0 -- NHOSTS:2)
MODEL : Sistema Computacional Simplificado
                                              LVT : 5001.478
                                          INTERVAL : 5001.478
                      MEAN BUSY    MEAN QUEUE         OPERATION COUNTS
FACILITY    UTIL.      PERIOD        LENGTH        RELEASE  PREEMPT  QUEUE
CPU-[1]     0.9994     1.519         1222.985       3291       0      3290
================================================================================
NULL MESSAGE COUNTS       COMPLETE MESSAGE COUNTS
      2501871                      680
================================================================================
```

P1:

```
Tempo de Execução:785988663
CMB SIMULATION.REPORT(HOST: work138 - RANK:1 -- NHOSTS:2)
MODEL : Sistema Computacional Simplificado
                                              LVT : 5001.478
                                          INTERVAL : 5001.478
                   MEAN BUSY   MEAN QUEUE         OPERATION COUNTS
FACILITY   UTIL.    PERIOD     LENGTH        RELEASE  PREEMPT  QUEUE
DISCO-[1]  0.2767    2.035      0.116          680       0       194
================================================================================
NULL MESSAGE COUNTS       COMPLETE MESSAGE COUNTS
     2501871                      680
```

2: Execução (tc = 0.5 - tcpu = 1.5 - tdisco = 2.0)

P0:

```
Tempo de Execução: 793689859
CMB SIMULATION.REPORT(HOST: work139 - RANK:0 -- NHOSTS:2)
MODEL : Sistema Computacional Simplificado
                                              LVT : 5000.002
```

```
                                          INTERVAL : 5000.002

                       MEAN BUSY    MEAN QUEUE         OPERATION COUNTS
FACILITY    UTIL.       PERIOD       LENGTH        RELEASE  PREEMPT  QUEUE
CPU-[1]     0.9997      1.519        3729.470       3291       0      3291
================================================================================
NULL MESSAGE COUNTS       COMPLETE MESSAGE COUNTS
       2503055                     680
================================================================================
P1:
Tempo de Execução: 793678914
CMB SIMULATION.REPORT(HOST: work138 - RANK:1 -- NHOSTS:2)
MODEL : Sistema Computacional Simplificado
                                              LVT : 5000.002
                                         INTERVAL : 5000.002

                     MEAN BUSY      MEAN QUEUE         OPERATION COUNTS
FACILITY      UTIL.   PERIOD         LENGTH        RELEASE  PREEMPT  QUEUE
DISCO-[1]     0.2768   2.035          0.116          680       0       194
================================================================================
NULL MESSAGE COUNTS       COMPLETE MESSAGE COUNTS
      2503055                     680
```

3: Execução (tc = 1.0 - tcpu = 1.0 - tdisco = 2.0)

```
P0:
Tempo de Execução: 785845531
CMB SIMULATION.REPORT(HOST: work139 - RANK:0 -- NHOSTS:2)
MODEL : Sistema Computacional Simplificado
                                              LVT : 5000.422
                                         INTERVAL : 5000.422

                     MEAN BUSY    MEAN QUEUE         OPERATION COUNTS
FACILITY    UTIL.     PERIOD       LENGTH        RELEASE  PREEMPT  QUEUE
CPU-[1]     1.0000    0.996        555.199         5020       0      5019
================================================================================
NULL MESSAGE COUNTS       COMPLETE MESSAGE COUNTS
    2502575                    1039
================================================================================
P1:
Tempo de Execução: 785834657
CMB SIMULATION.REPORT(HOST: work138 - RANK:1 -- NHOSTS:2)
MODEL : Sistema Computacional Simplificado
                                              LVT : 5000.422
                                         INTERVAL : 5000.422

                      MEAN BUSY      MEAN QUEUE         OPERATION COUNTS
FACILITY    UTIL.      PERIOD          LENGTH        RELEASE  PREEMPT
QUEUE
DISCO-[1]   0.4076     1.963            0.276         1038       0
408
================================================================================
NULL MESSAGE COUNTS       COMPLETE MESSAGE COUNTS
     2502575                    1038
```

4: Execução (tc = 1.5 - tcpu = 1.0 - tdisco = 2.0)

```
P0:
Tempo de Execução: 785678344
CMB SIMULATION.REPORT(HOST: work139 - RANK:0 -- NHOSTS:2)
MODEL : Sistema Computacional Simplificado
                                              LVT : 5000.008
                                         INTERVAL : 5000.008
                      MEAN BUSY      MEAN QUEUE         OPERATION COUNTS
```

```
FACILITY   UTIL.       PERIOD          LENGTH        RELEASE  PREEMPT
QUEUE
CPU-[1]    0.8513      0.999            5.504         4262       0
3624
================================================================================
NULL MESSAGE COUNTS       COMPLETE MESSAGE COUNTS
  2503582                      878
================================================================================
```

P1:

```
Tempo de Execução: 785666969
CMB SIMULATION.REPORT(HOST: work138 - RANK:1 -- NHOSTS:2)
MODEL : Sistema Computacional Simplificado
                                              LVT : 5000.008
                                         INTERVAL : 5000.008

                      MEAN BUSY      MEAN QUEUE         OPERATION COUNTS
FACILITY    UTIL.     PERIOD          LENGTH        RELEASE  PREEMPT  QUEUE
DISCO-[1]  0.3451     1.974            0.161          874       0       272
================================================================================
NULL MESSAGE COUNTS       COMPLETE MESSAGE COUNTS
     2503583                     874
```

5: Execução (tc = 1.0 - tcpu = 1.5 - tdisco = 1.5)

P0:

```
Tempo de Execução: 786144535
CMB SIMULATION.REPORT(HOST: work139 - RANK:0 -- NHOSTS:2)
MODEL : Sistema Computacional Simplificado
                                              LVT : 5001.478
                                         INTERVAL : 5001.478
                      MEAN BUSY      MEAN QUEUE         OPERATION COUNTS
FACILITY    UTIL.      PERIOD          LENGTH        RELEASE  PREEMPT
QUEUE
CPU-[1]     0.999      1.519         1223.106         3291       0
3290
================================================================================
NULL MESSAGE COUNTS       COMPLETE MESSAGE COUNTS
     2502183                     680
================================================================================
```

P1:

```
Tempo de Execução: 786133552
CMB SIMULATION.REPORT(HOST: work138 - RANK:1 -- NHOSTS:2)
MODEL : Sistema Computacional Simplificado
                                              LVT : 5001.478
                                         INTERVAL : 5001.478

                        MEAN BUSY     MEAN QUEUE         OPERATION COUNTS
FACILITY      UTIL.      PERIOD         LENGTH        RELEASE  PREEMPT
QUEUE
DISCO-[1]     0.2075     1.526           0.064          680       0
144
================================================================================
NULL MESSAGE COUNTS       COMPLETE MESSAGE COUNTS
    2502183                     680
```

6: Execução (tc = 1.0 - tcpu = 1.5 - tdisco = 1.0)

P0:

```
Tempo de Execução: 786283894
CMB SIMULATION.REPORT(HOST: work139 - RANK:0 -- NHOSTS:2)
MODEL : Sistema Computacional Simplificado
                                              LVT : 5001.478
                                         INTERVAL : 5001.478
                      MEAN BUSY      MEAN QUEUE         OPERATION COUNTS
```

```
FACILITY    UTIL.      PERIOD          LENGTH        RELEASE  PREEMPT
QUEUE
CPU-[1]     0.9994     1.519         1223.211         3291       0
3290

================================================================================
NULL MESSAGE COUNTS       COMPLETE MESSAGE COUNTS
    2502175                     680
================================================================================
```

P1:

```
Tempo de Execução: 786272751
CMB SIMULATION.REPORT(HOST: work138 - RANK:1 -- NHOSTS:2)
MODEL : Sistema Computacional Simplificado
                                            LVT : 5001.478
                                       INTERVAL : 5001.478
                       MEAN BUSY   MEAN QUEUE         OPERATION COUNTS
FACILITY    UTIL.      PERIOD       LENGTH        RELEASE  PREEMPT  QUEUE
DISCO-[1]   0.1384      1.018       0.028          680       0       102
================================================================================
NULL MESSAGE COUNTS       COMPLETE MESSAGE COUNTS
     2502175                     680
```

7: Execução (tc = 1.0 - tcpu = 1.0 - tdisco = 1.5)

P0:

```
Tempo de Execução: 785905974
CMB SIMULATION.REPORT(HOST: work139 - RANK:0 -- NHOSTS:2)
MODEL : Sistema Computacional Simplificado
                                            LVT : 5000.422
                                       INTERVAL : 5000.422
                      MEAN BUSY      MEAN QUEUE         OPERATION COUNTS
FACILITY    UTIL.      PERIOD          LENGTH        RELEASE  PREEMPT
QUEUE
CPU-[1]     1.0000     0.996          555.431         5020       0
5019
================================================================================
NULL MESSAGE COUNTS       COMPLETE MESSAGE COUNTS
    2502580                    1039
================================================================================
```

P1:

```
Tempo de Execução: 785898674
CMB SIMULATION.REPORT(HOST: work138 - RANK:1 -- NHOSTS:2)
MODEL : Sistema Computacional Simplificado
                                            LVT : 5000.422
                                       INTERVAL : 5000.422

                      MEAN BUSY      MEAN QUEUE         OPERATION COUNTS
FACILITY      UTIL.     PERIOD          LENGTH        RELEASE  PREEMPT
QUEUE
DISCO-[1]     0.3057    1.473            0.145         1038       0
322
================================================================================
NULL MESSAGE COUNTS       COMPLETE MESSAGE COUNTS
      2502580                    1038
```

8: Execução (tc = 1.0 - tcpu = 1.5 - tdisco = 3.0)

P0:

```
Tempo de Execução: 786291861
CMB SIMULATION.REPORT(HOST: work139 - RANK:0 -- NHOSTS:2)
MODEL : Sistema Computacional Simplificado
                                            LVT : 5001.478
                                       INTERVAL : 5001.478
                      MEAN BUSY     MEAN QUEUE         OPERATION COUNTS
```

```
FACILITY    UTIL.       PERIOD        LENGTH        RELEASE  PREEMPT  QUEUE
CPU-[1]     0.9994      1.519         1222.686         3291       0
3290


================================================================================
NULL MESSAGE COUNTS       COMPLETE MESSAGE COUNTS
     2502151                     680
================================================================================
```

P1:

```
Tempo de Execução: 786280764
CMB SIMULATION.REPORT(HOST: work138 - RANK:1 -- NHOSTS:2)
MODEL : Sistema Computacional Simplificado
                                              LVT : 5001.478
                                         INTERVAL : 5001.478
                         MEAN BUSY     MEAN QUEUE          OPERATION COUNTS
FACILITY     UTIL.       PERIOD         LENGTH        RELEASE  PREEMPT
QUEUE
DISCO-[1]     0.4151     3.053          0.277           680       0       272
================================================================================
NULL MESSAGE COUNTS       COMPLETE MESSAGE COUNTS
     2502151                     680
```

9: Execução (tc = 1.0 - tcpu = 1.5 - tdisco = 2.5)

P0:

```
Tempo de Execução:786003826
CMB SIMULATION.REPORT(HOST: work139 - RANK:0 -- NHOSTS:2)
MODEL : Sistema Computacional Simplificado
                                              LVT : 5001.478
                                         INTERVAL : 5001.478
                       MEAN BUSY      MEAN QUEUE          OPERATION COUNTS
FACILITY     UTIL.       PERIOD          LENGTH        RELEASE  PREEMPT
QUEUE
CPU-[1]     0.9994      1.519         1222.846         3291       0
3290
================================================================================
NULL MESSAGE COUNTS       COMPLETE MESSAGE COUNTS
    2502166                     680
================================================================================
```

P1:

```
Tempo de Execução:785992959
CMB SIMULATION.REPORT(HOST: work138 - RANK:1 -- NHOSTS:2)
MODEL : Sistema Computacional Simplificado
                                              LVT : 5001.478
                                         INTERVAL : 5001.478
                         MEAN BUSY    MEAN QUEUE          OPERATION COUNTS
FACILITY       UTIL.      PERIOD       LENGTH        RELEASE  PREEMPT  QUEUE
DISCO-[1]     0.3459      2.544        0.185           680       0       228
================================================================================
NULL MESSAGE COUNTS       COMPLETE MESSAGE COUNTS
     2502166                    680
```

10: Execução (tc = 1.0 - tcpu = 2.0 - tdisco = 3.0)

P0:

```
Tempo de Execução:786565335
CMB SIMULATION.REPORT(HOST: work139 - RANK:0 -- NHOSTS:2)
MODEL : Sistema Computacional Simplificado
                                              LVT : 5000.077
                                         INTERVAL : 5000.077
                       MEAN BUSY    MEAN QUEUE          OPERATION COUNTS
FACILITY     UTIL.      PERIOD      LENGTH        RELEASE  PREEMPT  QUEUE
CPU-[1]     1.0000     2.043       1550.311         2448       0      2448
```

```
================================================================================
NULL MESSAGE COUNTS       COMPLETE MESSAGE COUNTS
      2502038                 511
================================================================================
```

P1:

```
Tempo de Execução:786554177
CMB SIMULATION.REPORT(HOST: work138 - RANK:1 -- NHOSTS:2)
MODEL : Sistema Computacional Simplificado
                                              LVT : 5000.077
                                         INTERVAL : 5000.077
                        MEAN BUSY    MEAN QUEUE         OPERATION COUNTS
FACILITY     UTIL.       PERIOD       LENGTH        RELEASE  PREEMPT  QUEUE
DISCO-[1]   0.3085       3.025        0.145          510       0       157
================================================================================
NULL MESSAGE COUNTS       COMPLETE MESSAGE COUNTS
    2502038                     510
```

11: Execução (tc = 1.0 - tcpu = 2.0 - tdisco = 2.0)

P0:

```
Tempo de Execução:786559817
CMB SIMULATION.REPORT(HOST: work139 - RANK:0 -- NHOSTS:2)
MODEL : Sistema Computacional Simplificado
                                              LVT : 5000.077
                                         INTERVAL : 5000.077

                       MEAN BUSY      MEAN QUEUE         OPERATION COUNTS
FACILITY    UTIL.       PERIOD        LENGTH        RELEASE  PREEMPT  QUEUE
CPU-[1]     1.0000       2.043       1550.497         2448       0
2448
================================================================================
NULL MESSAGE COUNTS       COMPLETE MESSAGE COUNTS
       2502044                  511
================================================================================
```

P1:

```
Tempo de Execução:786548826
CMB SIMULATION.REPORT(HOST: work138 - RANK:1 -- NHOSTS:2)
MODEL : Sistema Computacional Simplificado
                                              LVT : 5000.077
                                         INTERVAL : 5000.077
                        MEAN BUSY      MEAN QUEUE         OPERATION COUNTS
FACILITY    UTIL.       PERIOD          LENGTH        RELEASE  PREEMPT
QUEUE
DISCO-[1]  0.2057       2.016            0.062          510       0
110
================================================================================
NULL MESSAGE COUNTS       COMPLETE MESSAGE COUNTS
     2502044                    510
```

12: Execução (tc = 2.0 - tcpu = 1.5 - tdisco = 2.0)

P0:

```
Tempo de Execução:785572527
CMB SIMULATION.REPORT(HOST: work139 - RANK:0 -- NHOSTS:2)
MODEL : Sistema Computacional Simplificado
                                              LVT : 5001.937
                                         INTERVAL : 5001.937
                       MEAN BUSY     MEAN QUEUE         OPERATION COUNTS
FACILITY   UTIL.        PERIOD        LENGTH        RELEASE  PREEMPT  QUEUE
CPU-[1]    0.9596       1.520         18.922         3157       0      3026
================================================================================
NULL MESSAGE COUNTS       COMPLETE MESSAGE COUNTS
   2504507                     653
```

```
================================================================================
P1:
Tempo de Execução:785561455
CMB SIMULATION.REPORT(HOST: work138 - RANK:1 -- NHOSTS:2)
MODEL : Sistema Computacional Simplificado
                                              LVT : 5001.937
                                         INTERVAL : 5001.937
                        MEAN BUSY      MEAN QUEUE          OPERATION COUNTS
FACILITY     UTIL.       PERIOD        LENGTH         RELEASE  PREEMPT
QUEUE
DISCO-[1]    0.2663      2.040         0.109           653       0       176
================================================================================
NULL MESSAGE COUNTS       COMPLETE MESSAGE COUNTS
   2504507                      653
```

13: Execução (tc = 2.0 - tcpu = 1.5 - tdisco = 3.0)

```
P0:
Tempo de Execução:785939833
CMB SIMULATION.REPORT(HOST: work139 - RANK:0 -- NHOSTS:2)
MODEL : Sistema Computacional Simplificado
                                              LVT : 5001.937
                                         INTERVAL : 5001.937
                       MEAN BUSY     MEAN QUEUE          OPERATION COUNTS
FACILITY    UTIL.      PERIOD        LENGTH         RELEASE  PREEMPT  QUEUE
CPU-[1]     0.9596     1.520         19.104          3157       0      3035
================================================================================
NULL MESSAGE COUNTS       COMPLETE MESSAGE COUNTS
   2504506                     653
================================================================================
P1:
Tempo de Execução:785931854
CMB SIMULATION.REPORT(HOST: work138 - RANK:1 -- NHOSTS:2)
MODEL : Sistema Computacional Simplificado
                                              LVT : 5001.937
                                         INTERVAL : 5001.937
                      MEAN BUSY     MEAN QUEUE          OPERATION COUNTS
FACILITY       UTIL.    PERIOD       LENGTH         RELEASE  PREEMPT  QUEUE
DISCO-[1]     0.3995     3.060       0.255           653       0       242
================================================================================
NULL MESSAGE COUNTS       COMPLETE MESSAGE COUNTS
     2504506                    653
```

14: Execução (tc = 2.0 - tcpu = 1.5 - tdisco = 2.0)

```
P0:
Tempo de Execução:785584260
CMB SIMULATION.REPORT(HOST: work139 - RANK:0 -- NHOSTS:2)
MODEL : Sistema Computacional Simplificado
                                              LVT : 5001.937
                                         INTERVAL : 5001.937
                       MEAN BUS      MEAN QUEUE          OPERATION COUNTS
FACILITY    UTIL.      PERIOD        LENGTH         RELEASE  PREEMPT  QUEUE
CPU-[1]     0.9596     1.520         18.922          3157       0      3026
================================================================================
NULL MESSAGE COUNTS       COMPLETE MESSAGE COUNTS
    2504507                     653
================================================================================
P1:
Tempo de Execução:785573448
CMB SIMULATION.REPORT(HOST: work138 - RANK:1 -- NHOSTS:2)
MODEL : Sistema Computacional Simplificado
                                              LVT : 5001.937
                                         INTERVAL : 5001.937
```

```
                        MEAN BUSY      MEAN QUEUE          OPERATION COUNTS
FACILITY     UTIL.      PERIOD          LENGTH         RELEASE  PREEMPT  QUEUE
DISCO-[1]   0.2663       2.040          0.109           653        0       176
================================================================================
NULL MESSAGE COUNTS        COMPLETE MESSAGE COUNTS
   2504507                       653
```

15: Execução (tc = 1.5 - tcpu = 1.5 - tdisco = 2.0)
P0:

```
Tempo de Execução:785759692
CMB SIMULATION.REPORT(HOST: work139 - RANK:0 -- NHOSTS:2)
MODEL : Sistema Computacional Simplificado
                                              LVT : 5000.470
                                         INTERVAL : 5000.470
                        MEAN BUSY      MEAN QUEUE          OPERATION COUNTS
FACILITY    UTIL.       PERIOD           LENGTH         RELEASE  PREEMPT
QUEUE
CPU-[1]     0.9996      1.519           394.026          3291        0
3290
================================================================================
NULL MESSAGE COUNTS        COMPLETE MESSAGE COUNTS
    2503903                       680
================================================================================
```

P1:

```
Tempo de Execução:785748773
CMB SIMULATION.REPORT(HOST: work138 - RANK:1 -- NHOSTS:2)
MODEL : Sistema Computacional Simplificado
                                              LVT : 5000.470
                                         INTERVAL : 5000.470
                         MEAN BUSY      MEAN QUEUE          OPERATION COUNTS
FACILITY     UTIL.       PERIOD           LENGTH         RELEASE  PREEMPT
QUEUE
DISCO-[1]   0.2768       2.035            0.116           680        0
194
================================================================================
NULL MESSAGE COUNTS        COMPLETE MESSAGE COUNTS
   2503903                       680
```

B.2.3 Resultados Obtidos na CMB-*Simulation* Sob Demanda

1: Execução (tc =1.0 - tcpu = 1.5 - tdisco = 2.0)
P0:

```
Tempo de Execução: 184720679
CMB SIMULATION.REPORT(HOST: work139 - RANK:0 -- NHOSTS:2)
MODEL : Sistema Computacional Simplificado
                                              LVT : 5001.478
                                         INTERVAL : 5001.478
                       MEAN BUSY     MEAN QUEUE          OPERATION COUNTS
FACILITY     UTIL.     PERIOD         LENGTH         RELEASE  PREEMPT  QUEUE
CPU-[1]     0.9994     1.519         1222.985          3291        0      3290
================================================================================
NULL MESSAGE COUNTS        COMPLETE MESSAGE COUNTS
     402897                       680
================================================================================
```

P1:

```
Tempo de Execução: 184633276
CMB SIMULATION.REPORT(HOST: work138 - RANK:1 -- NHOSTS:2)
MODEL : Sistema Computacional Simplificado
                                              LVT : 5001.478
                                         INTERVAL : 5001.478
```

```
                    MEAN BUSY    MEAN QUEUE          OPERATION COUNTS
FACILITY     UTIL.   PERIOD       LENGTH         RELEASE  PREEMPT  QUEUE
DISCO-[1]    0.2767  2.035            0.116          680       0       194
================================================================================
NULL MESSAGE COUNTS       COMPLETE MESSAGE COUNTS
    402896                      680
```

2: Execução (tc = 0.5 - tcpu = 1.5 - tdisco = 2.0)

P0:

```
Tempo de Execução: 117134125
CMB SIMULATION.REPORT(HOST: work139 - RANK:0 -- NHOSTS:2)
MODEL : Sistema Computacional Simplificado
                                              LVT : 5000.002
                                         INTERVAL : 5000.002
                    MEAN BUSY    MEAN QUEUE          OPERATION COUNTS
FACILITY     UTIL.    PERIOD       LENGTH         RELEASE  PREEMPT  QUEUE
CPU-[1]     0.9997    1.519        3729.470          3291       0      3291
================================================================================
NULL MESSAGE COUNTS       COMPLETE MESSAGE COUNTS
    250001                      680
================================================================================
```

P1:

```
Tempo de Execução: 117124689
CMB SIMULATION.REPORT(HOST: work138 - RANK:1 -- NHOSTS:2)
MODEL : Sistema Computacional Simplificado
                                              LVT : 5000.002
                                         INTERVAL : 5000.002
                     MEAN BUSY    MEAN QUEUE          OPERATION COUNTS
FACILITY     UTIL.      PERIOD       LENGTH         RELEASE  PREEMPT  QUEUE
DISCO-[1]   0.2768      2.035         0.116           680       0       194
================================================================================
NULL MESSAGE COUNTS       COMPLETE MESSAGE COUNTS
     250000                     680
```

3: Execução (tc = 1.0 - tcpu = 1.0 - tdisco = 2.0)

P0:

```
Tempo de Execução: 237914876
CMB SIMULATION.REPORT(HOST: work139 - RANK:0 -- NHOSTS:2)
MODEL : Sistema Computacional Simplificado
                                              LVT : 5000.422
                                         INTERVAL : 5000.422
                    MEAN BUSY    MEAN QUEUE          OPERATION COUNTS
FACILITY    UTIL.     PERIOD      LENGTH         RELEASE  PREEMPT  QUEUE
CPU-[1]     1.0000     0.996      555.199          5020       0      5019
================================================================================
NULL MESSAGE COUNTS       COMPLETE MESSAGE COUNTS
   519685                    1039
================================================================================
```

P1:

```
Tempo de Execução: 237919142
CMB SIMULATION.REPORT(HOST: work138 - RANK:1 -- NHOSTS:2)
MODEL : Sistema Computacional Simplificado
                                              LVT : 5000.422
                                         INTERVAL : 5000.422
                     MEAN BUSY      MEAN QUEUE          OPERATION COUNTS
FACILITY      UTIL.     PERIOD        LENGTH         RELEASE  PREEMPT  QUEUE
DISCO-[1]     0.4076     1.963        0.276          1038       0       408
================================================================================
NULL MESSAGE COUNTS       COMPLETE MESSAGE COUNTS
     519684                     1038
```

4: Execução (tc = 1.5 - tcpu = 1.0 - tdisco = 2.0)

P0:

```
Tempo de Execução: 169903878
CMB SIMULATION.REPORT(HOST: work139 - RANK:0 -- NHOSTS:2)
MODEL : Sistema Computacional Simplificado
                                              LVT : 5000.008
                                         INTERVAL : 5000.008
                     MEAN BUSY     MEAN QUEUE          OPERATION COUNTS
FACILITY     UTIL.    PERIOD         LENGTH         RELEASE  PREEMPT  QUEUE
CPU-[1]     0.8513    0.999           5.513          4262       0      3630
================================================================================
NULL MESSAGE COUNTS       COMPLETE MESSAGE COUNTS
 369276                      878
================================================================================
```

P1:

```
Tempo de Execução: 169910458
CMB SIMULATION.REPORT(HOST: work138 - RANK:1 -- NHOSTS:2)
MODEL : Sistema Computacional Simplificado
                                              LVT : 5000.008
                                         INTERVAL : 5000.008
                     MEAN BUSY     MEAN QUEUE          OPERATION COUNTS
FACILITY      UTIL     PERIOD         LENGTH         RELEASE  PREEMPT  QUEUE
DISCO-[1]     0.3451   1.974           0.161           874       0       273
================================================================================
NULL MESSAGE COUNTS       COMPLETE MESSAGE COUNTS
     369276                      874
```

5: Execução (tc = 1.0 - tcpu = 1.5 - tdisco = 1.5)

P0:

```
Tempo de Execução: 169704869
CMB SIMULATION.REPORT(HOST: work139 - RANK:0 -- NHOSTS:2)
MODEL : Sistema Computacional Simplificado
                                              LVT : 5001.478
                                         INTERVAL : 5001.478

                MEAN BUSY    MEAN QUEUE          OPERATION COUNTS
FACILITY     UTIL.    PERIOD       LENGTH         RELEASE  PREEMPT  QUEUE
CPU-[1]     0.9994    1.519        1223.106          3291       0      3290
================================================================================
NULL MESSAGE COUNTS       COMPLETE MESSAGE COUNTS
    371107                     680
================================================================================
```

P1:

```
Tempo de Execução: 169695091
CMB SIMULATION.REPORT(HOST: work138 - RANK:1 -- NHOSTS:2)
MODEL : Sistema Computacional Simplificado
                                              LVT : 5001.478
                                         INTERVAL : 5001.478
                     MEAN BUSY    MEAN QUEUE          OPERATION COUNTS
FACILITY     UTIL.     PERIOD       LENGTH         RELEASE  PREEMPT  QUEUE
DISCO-[1]   0.2075     1.526        0.064           680       0       144
================================================================================
NULL MESSAGE COUNTS       COMPLETE MESSAGE COUNTS
    371106                     680
```

6: Execução (tc = 1.0 - tcpu = 1.5 - tdisco = 1.0)

P0:

```
Tempo de Execução: 177489961
CMB SIMULATION.REPORT(HOST: work139 - RANK:0 -- NHOSTS:2)
MODEL : Sistema Computacional Simplificado
                                              LVT : 5001.478
                                         INTERVAL : 5001.478
                   MEAN BUSY      MEAN QUEUE          OPERATION COUNTS
```

```
FACILITY     UTIL.    PERIOD       LENGTH        RELEASE  PREEMPT  QUEUE
CPU-[1]     0.9994    1.519        1223.211         3291        0      3290
================================================================================
NULL MESSAGE COUNTS       COMPLETE MESSAGE COUNTS
     389391                    680
================================================================================
```

P1:

```
Tempo de Execução: 177480483
CMB SIMULATION.REPORT(HOST: work138 - RANK:1 -- NHOSTS:2)
MODEL : Sistema Computacional Simplificado
                                              LVT : 5001.478
                                         INTERVAL : 5001.478
                        MEAN BUSY    MEAN QUEUE          OPERATION COUNTS
FACILITY      UTIL.      PERIOD       LENGTH         RELEASE  PREEMPT  QUEUE
DISCO-[1]    0.1384     1.018        0.028           680        0       102
================================================================================
==NULL MESSAGE COUNTS       COMPLETE MESSAGE COUNTS
     389390                      680
```

7: Execução (tc = 1.0 - tcpu = 1.0 - tdisco = 1.5)

P0:

```
Tempo de Execução: 227696840
CMB SIMULATION.REPORT(HOST: work139 - RANK:0 -- NHOSTS:2)
MODEL : Sistema Computacional Simplificado
                                              LVT : 5000.422
                                         INTERVAL : 5000.422
                      MEAN BUSY    MEAN QUEUE          OPERATION COUNTS
FACILITY   UTIL.      PERIOD       LENGTH         RELEASE  PREEMPT  QUEUE
CPU-[1]   1.0000      0.996        555.431        5020        0       5019
================================================================================
==NULL MESSAGE COUNTS       COMPLETE MESSAGE COUNTS
    498057                     1039
================================================================================
```

P1:

```
Tempo de Execução: 227699074
CMB SIMULATION.REPORT(HOST: work138 - RANK:1 -- NHOSTS:2)
MODEL : Sistema Computacional Simplificado
                                              LVT : 5000.422
                                         INTERVAL : 5000.422

                      MEAN BUSY    MEAN QUEUE          OPERATION COUNTS
FACILITY    UTIL.      PERIOD       LENGTH         RELEASE  PREEMPT  QUEUE
DISCO-[1]  0.3057       1.473        0.145          1038        0        322
================================================================================
==NULL MESSAGE COUNTS       COMPLETE MESSAGE COUNTS
     498056                    1038
```

8: Execução (tc = 1.0 - tcpu = 1.5 - tdisco = 3.0)

P0:

```
Tempo de Execução: 212816159
CMB SIMULATION.REPORT(HOST: work139 - RANK:0 -- NHOSTS:2)
MODEL : Sistema Computacional Simplificado
                                              LVT : 5001.478
                                         INTERVAL : 5001.478
                        MEAN BUSY    MEAN QUEUE          OPERATION COUNTS
FACILITY     UTIL.      PERIOD       LENGTH         RELEASE  PREEMPT  QUEUE
CPU-[1]     0.9994     1.519        1222.686         3291        0      3290
================================================================================
NULL MESSAGE COUNTS       COMPLETE MESSAGE COUNTS
      462552                    680
```

P1:

```
Tempo de Execução: 212806456
```

```
CMB SIMULATION.REPORT(HOST: work138 - RANK:1 -- NHOSTS:2)
MODEL : Sistema Computacional Simplificado
                                              LVT : 5001.478
                                         INTERVAL : 5001.478
                      MEAN BUSY      MEAN QUEUE          OPERATION COUNTS
FACILITY     UTIL.    PERIOD         LENGTH         RELEASE  PREEMPT  QUEUE
DISCO-[1]   0.4151     3.053          0.277            680       0       272
================================================================================
NULL MESSAGE COUNTS       COMPLETE MESSAGE COUNTS
      462551                      680
```

9: Execução (tc = 1.0 - tcpu = 1.5 - tdisco = 2.5)
P0:

```
Tempo de Execução: 160898871
CMB SIMULATION.REPORT(HOST: work139 - RANK:0 -- NHOSTS:2)
MODEL : Sistema Computacional Simplificado
                                              LVT : 5001.478
                                         INTERVAL : 5001.478
                    MEAN BUSY      MEAN QUEUE          OPERATION COUNTS
FACILITY    UTIL.    PERIOD          LENGTH         RELEASE  PREEMPT  QUEUE
CPU-[1]     0.9994   1.519         1222.846          3291       0      3290
================================================================================
NULL MESSAGE COUNTS       COMPLETE MESSAGE COUNTS
  351469                      680
================================================================================
```

P1:

```
Tempo de Execução: 160889634
CMB SIMULATION.REPORT(HOST: work138 - RANK:1 -- NHOSTS:2)
MODEL : Sistema Computacional Simplificado
                                              LVT : 5001.478
                                         INTERVAL : 5001.478
                       MEAN BUSY    MEAN QUEUE          OPERATION COUNTS
FACILITY      UTIL.     PERIOD       LENGTH         RELEASE  PREEMPT  QUEUE
DISCO-[1]     0.3459     2.544       0.185           680       0       228
================================================================================
NULL MESSAGE COUNTS       COMPLETE MESSAGE COUNTS
  351468                      680
```

10: Execução (tc = 1.0 - tcpu = 2.0 - tdisco = 3.0)
P0:

```
Tempo de Execução: 76087872
CMB SIMULATION.REPORT(HOST: work139 - RANK:0 -- NHOSTS:2)
MODEL : Sistema Computacional Simplificado
                                              LVT : 5000.077
                                         INTERVAL : 5000.077
                      MEAN BUSY    MEAN QUEUE          OPERATION COUNTS
FACILITY     UTIL.    PERIOD       LENGTH         RELEASE  PREEMPT  QUEUE
CPU-[1]     1.0000    2.043       1550.311        2448       0      2448
================================================================================
NULL MESSAGE COUNTS       COMPLETE MESSAGE COUNTS
   170098                     511
================================================================================
```

P1:

```
Tempo de Execução: 76078572
CMB SIMULATION.REPORT(HOST: work138 - RANK:1 -- NHOSTS:2)
MODEL : Sistema Computacional Simplificado
                                              LVT : 5000.077
                                         INTERVAL : 5000.077
                      MEAN BUSY    MEAN QUEUE          OPERATION COUNTS
FACILITY      UTIL.    PERIOD       LENGTH         RELEASE  PREEMPT  QUEUE
DISCO-[1]    0.3085    3.025        0.145           510       0       157
================================================================================
```

```
NULL MESSAGE COUNTS       COMPLETE MESSAGE COUNTS
    170097                     510
```

11: Execução (tc = 1.0 - tcpu = 2.0 - tdisco = 2.0)
P0:

```
Tempo de Execução: 110649405
CMB SIMULATION.REPORT(HOST: work139 - RANK:0 -- NHOSTS:2)
MODEL : Sistema Computacional Simplificado
                                              LVT : 5000.077
                                         INTERVAL : 5000.077
                     MEAN BUSY    MEAN QUEUE          OPERATION COUNTS
FACILITY    UTIL.     PERIOD       LENGTH         RELEASE  PREEMPT  QUEUE
CPU-[1]     1.0000    2.043      1550.497           2448       0      2448
================================================================================
NULL MESSAGE COUNTS       COMPLETE MESSAGE COUNTS
  244447                       511
================================================================================
```

P1:

```
Tempo de Execução: 110639938
CMB SIMULATION.REPORT(HOST: work138 - RANK:1 -- NHOSTS:2)
MODEL : Sistema Computacional Simplificado
                                              LVT : 5000.077
                                         INTERVAL : 5000.077
                       MEAN BUSY    MEAN QUEUE          OPERATION COUNTS
FACILITY      UTIL.     PERIOD        LENGTH         RELEASE  PREEMPT  QUEUE
DISCO-[1]     0.2057     2.016        0.062           510       0       110
================================================================================
NULL MESSAGE COUNTS       COMPLETE MESSAGE COUNTS
  244446                       510
```

12: Execução (tc = 2.0 - tcpu = 1.5 - tdisco = 2.0)
P0:

```
Tempo de Execução: 163129195
CMB SIMULATION.REPORT(HOST: work139 - RANK:0 -- NHOSTS:2)
MODEL : Sistema Computacional Simplificado
                                              LVT : 5001.937
                                         INTERVAL : 5001.937
                     MEAN BUSY    MEAN QUEUE          OPERATION COUNTS
FACILITY     UTIL.    PERIOD       LENGTH         RELEASE  PREEMPT  QUEUE
CPU-[1]     0.9596     1.520      18.925            3157       0      3033
================================================================================
NULL MESSAGE COUNTS       COMPLETE MESSAGE COUNTS
  354617                       653
================================================================================
```

P1:

```
Tempo de Execução: 163128983
CMB SIMULATION.REPORT(HOST: work138 - RANK:1 -- NHOSTS:2)
MODEL : Sistema Computacional Simplificado
                                              LVT : 5001.937
                                         INTERVAL : 5001.937
                       MEAN BUSY    MEAN QUEUE          OPERATION COUNTS
FACILITY     UTIL.      PERIOD       LENGTH         RELEASE  PREEMPT  QUEUE
DISCO-[1]    0.2663     2.040         0.110           653       0       177
================================================================================
NULL MESSAGE COUNTS       COMPLETE MESSAGE COUNTS
   354616                      653
```

13: Execução (tc = 2.0 - tcpu = 1.5 - tdisco = 3.0)
P0:

```
Tempo de Execução: 172784970
LVT = 5001.936601 - Envc.Mensa Termino:1
CMB SIMULATION.REPORT(HOST: work139 - RANK:0 -- NHOSTS:2)
```

```
MODEL : Sistema Computacional Simplificado
                                                LVT : 5001.937
                                           INTERVAL : 5001.937
                     MEAN BUSY    MEAN QUEUE          OPERATION COUNTS
FACILITY    UTIL.     PERIOD      LENGTH         RELEASE  PREEMPT  QUEUE
CPU-[1]     0.9596     1.520       19.110          3157       0      3039
================================================================================
NULL MESSAGE COUNTS       COMPLETE MESSAGE COUNTS
     375013                       653
================================================================================
P1:
Tempo de Execução: 172784914
CMB SIMULATION.REPORT(HOST: work138 - RANK:1 -- NHOSTS:2)
MODEL : Sistema Computacional Simplificado
                                                LVT : 5001.937
                                           INTERVAL : 5001.937
                     MEAN BUSY      MEAN QUEUE          OPERATION COUNTS
FACILITY     UTIL.    PERIOD         LENGTH         RELEASE  PREEMPT  QUEUE
DISCO-[1]    0.3995    3.060          0.255           653       0       242
================================================================================
NULL MESSAGE COUNTS       COMPLETE MESSAGE COUNTS
   375012                       653
```

14: Execução (tc = 2.0 - tcpu = 1.5 - tdisco = 2.0)

```
P0:
Tempo de Execução: 154290162
CMB SIMULATION.REPORT(HOST: work139 - RANK:0 -- NHOSTS:2)
MODEL : Sistema Computacional Simplificado
                                                LVT : 5001.937
                                           INTERVAL : 5001.937
                     MEAN BUSY    MEAN QUEUE          OPERATION COUNTS
FACILITY    UTIL.     PERIOD       LENGTH         RELEASE  PREEMPT  QUEUE
CPU-[1]     0.9596    1.520          18.925          3157       0      3033
================================================================================
NULL MESSAGE COUNTS       COMPLETE MESSAGE COUNTS
   336666                       653
================================================================================
P1:
Tempo de Execução: 154301399
CMB SIMULATION.REPORT(HOST: work138 - RANK:1 -- NHOSTS:2)
MODEL : Sistema Computacional Simplificado
                                                LVT : 5001.937
                                           INTERVAL : 5001.937
                     MEAN BUSY      MEAN QUEUE          OPERATION COUNTS
FACILITY     UTIL.    PERIOD         LENGTH         RELEASE  PREEMPT  QUEUE
DISCO-[1]   0.2663    2.040           0.110           653       0       177
================================================================================
NULL MESSAGE COUNTS       COMPLETE MESSAGE COUNTS
   336665                       653
```

15: Execução (tc = 1.5 - tcpu = 1.5 - tdisco = 2.0)

```
P0:
Tempo de Execução:217934525
CMB SIMULATION.REPORT(HOST: work139 - RANK:0 -- NHOSTS:2)
MODEL : Sistema Computacional Simplificado
                                                LVT : 5000.470
                                           INTERVAL : 5000.470
                     MEAN BUSY   MEAN QUEUE          OPERATION COUNTS
FACILITY     UTIL.    PERIOD      LENGTH         RELEASE  PREEMPT  QUEUE
CPU-[1]     0.9996     1.519     394.026          3291       0      3290
```

```
NULL MESSAGE COUNTS       COMPLETE MESSAGE COUNTS
    471860                      680
================================================================================
P1:
Tempo de Execução:217931180
CMB SIMULATION.REPORT(HOST: work138 - RANK:1 -- NHOSTS:2)
MODEL : Sistema Computacional Simplificado
                                                LVT : 5000.470
                                           INTERVAL : 5000.470
                      MEAN BUSY     MEAN QUEUE         OPERATION COUNTS
FACILITY      UTIL.     PERIOD      LENGTH        RELEASE  PREEMPT  QUEUE
DISCO-[1]     0.2768    2.035        0.116          680       0       194
================================================================================
NULL MESSAGE COUNTS       COMPLETE MESSAGE COUNTS
    471859                   680
```