Um Modelo Analítico para Estimativa de Desempenho de Multicomputadores

Marcelo Knörich Zuffo

Laboratório de Sistemas Integráveis
Escola Politécnica
Universidade de São Paulo
Av. Luciano Gualberto, nº158, trav.3
05508-900 São Paulo S.P. Brasil
email: *mkzuffo@lsi.usp.br*

RESUMO

Este artigo descreve um modelo analítico para a estimativa de desempenho de multicomputadores homogêneos. Aspectos intrínsecos da organização interna dos multicomputadores como a arquitetura dos nós, políticas de roteamento e protocolos de comunicação são fatores importantes no desempenho global desta classe de computadores de alto desempenho. Algumas medidas experimentais são realizadas sobre o protótipo do multicomputador TRGR_01, utilizando um algoritmo para síntese de imagens e os resultados são confrontados com os resultados levantados a partir do modelo proposto.

ABSTRACT

This paper describes an analytical model for homogeneous multicomputer performance evaluation. Instrinsic aspects of multicomputer internal organization as node organization, routing and communication protocols are important factors to the global performance of this high performance computer class. Some measurements were done over the multicomputer TRGR_01, using an algorithm for image synthesis. The results are confronted with the model previsions.

1 Introdução

Quando se utilizam multicomputadores o objetivo principal é o ganho de desempenho obtido em relação aos unicomputadores.

Percebe-se claramente a tendência dos fabricantes de lançarem no mercado multicomputadores modulares, supostamente escaláveis com um número crescente de processadores. Porém, algumas questões surgem no decorrer do desenvolvimento destes sistemas:

a) Qual é o número máximo de nós a se adicionar, obtendo ainda ganho de desempenho?

b) quais são os fatores limitantes de desempenho?

Propomos uma modelagem analítica de multicomputadores, levando em conta características específicas destes computadores, principalmente aspectos que dizem respeito à comunicação. Por fim, realizamos uma avaliação experimental do modelo adotado sobre uma aplicação gráfica sobre o protótipo TRGR_01.

O objetivo principal é oferecer uma ferramenta que, na medida do possível, possibilite a análise e previsão teórica do desempenho de multicomputadores. Outro objetivo é possibilitar um melhor entendimento de que aspectos são importantes no projeto de multicomputadores na paralelização de aplicações em máquinas desta classe.

2 Multicomputadores

Neste trabalho, utilizamos a popular taxomonia de FLYNN [JSCA91] para a classificação de arquiteturas de computadores. Uma boa descrição da taxonomia de FLYNN e outros tipos de taxonomias podem ser encontrados em [JSCA91], [HWAN89], [HWAN84] e [HOCK88].

Sistemas de processamento do tipo **MIMD** ("Multiple Instruction Multiple Data"), envolvem dois tipos bem definidos de máquinas: sistemas com memória compartilhada ("shared memory"), também designados por **Multiprocessadores** ([HWAN89], [HWAN84], [FOX88]), e sistemas com memória distribuída ("distributed memory") designados por **Multicomputadores** [BELL92]. Pode-se considerar estes sistemas como extensões alternativas à arquitetura tradicional proposta por Von-Neumman.

Os **Multicomputadores** são caracterizados pela utilização de dezenas a centenas de processadores, cada um com sua própria memória local, interligados através de um sistema de interconexão. Outra característica importante destas máquinas é que o tempo de acesso à memória é dependente da localização do processador na topologia e do segmento de memória do sistema endereçado. Basicamente, temos tempos de acesso muito distintos se a memória acessada é local ou pertence a outro processador. Por este motivo e o fato de que o acesso à memória comparado aos multiprocessadores é relativamente menor e não uniforme, os multicomputadores são denominados também de **Sistemas Fracamente Acoplados** ("loosely coupled systems").

Consideremos um multicomputador composto de N elementos interligados por um sistema de intercomexão, onde cada elemento desta rede é um computador sequencial tradicional (Von Neumann), isto é, processador mais memória. Estes elementos serão designados por **Nós**. Quando todos os Nós que compõem o multicomputador são idênticos, denominamos a máquina de **Multicomputador Homogêneo**. Quando os nós são distintos e especializados (com diferentes quantidades de memória, especializados para processamento numérico etc...), denominamos a máquina de **Multicomputador Heterogêneo**.

O **Sistema de Interconexão** de um multicomputador é caracterizado pela estrutura da mensagem, uma topologia, um conjunto de canais de comunicação e um conjunto de políticas para roteamento de mensagens. O sistema de interconexão é de vital importância, pois define o grau de acoplamento entre os processadores

e, consequentemente, o desempenho global do sistema. A **Mensagem** é o elemento de troca de informação entre processos, sendo que normalmente as mensagens não possuem tamanho fixo. Alguns autores como SEITZ e DALLY [SUAY90] propõem uma hierarquia de elementos que constituem uma mensagem. Esta hierarquia é composta pelos seguintes elementos em ordem crescente de complexidade: **Flits, Pacotes** e **Mensagens** (figura 1).

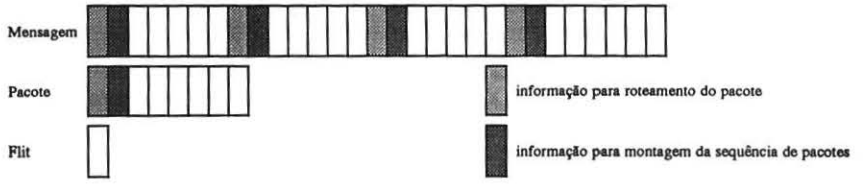
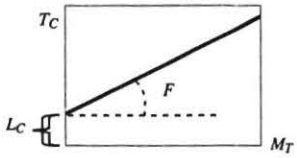


Figura 1 - Mensagens, Pacotes e Flits

A topologia define a organização e a forma de conexão dos canais de comunicação dos nós. A escolha da topologia deve ser um compromisso entre limitações tecnológicas e desempenho na comunicação das mensagens.

A **Latência de Comunicação** (L_C) expressa o tempo de transmissão de um pacote com tamanho zero, ou seja, a latência mede a sobrecarga de comunicação entre o nó origem e o nó destino independente do tamanho do pacote. É importante salientar que a latência é composta pelas sobrecargas ("overheads") tanto da circuitaria de comunicação ("hardware") como dos programas de comunicação. Denominaremos **Latência Física** quando a latência for imposta pela circuitaria de comunicação e denominaremos **Latência Lógica** quando a latência for gerada pelos protocolos de comunicação e roteamento. A **Latência de Comunicação** é composta pela soma da **Latência Física** e da **Latência Lógica**. O **Tempo de Comunicação** (T_C) é o tempo total de transmissão de uma mensagem incluindo a latência de comunicação.



$$T_C = L_C + \frac{M_T}{F} \qquad (2.1)$$

T_C = Tempo de Comunicação (segundos)
L_C = Latência de Comunicação (segundos)
F = Taxa de Comunicação (bytes/segundo)
M_T = Tamanho da Mensagem (bytes)

A equação 2.1 modela o **Tempo de Comunicação** (T_C) entre dois nós adjacentes, em função do tamanho da mensagem (M_T), a taxa de comunicação (F) e a latência de comunicação (L_C).

Definiremos como **Distância** (d_{ij}) o número de canais intermediários que uma mensagem tem que percorrer para que, partindo de seu nó origem, chegue em seu nó destino, dado um certo trajeto a ser percorrido.

Roteamento é o método utilizado para decidir o caminho que a mensagem deve percorrer entre o nó origem e o nó destino pelo sistema de interconexão. Os algoritmos de roteamento de mensagens são intimamente ligados à topologia escolhida [SUAY90].

Associados ao roteamento existem protocolos de comunicação [ZUFF93] responsáveis pela transmissão dos pacotes. Suas diferenças estão baseadas na rapidez e na segurança de transmissão dos pacotes que compõem a mensagem. A classificação dos protocolos é realizada sob dois aspectos: bloqueados versus não-bloqueados e síncronos versus assíncronos. A utilização destes protocolos tem influência na segurança e rapidez na transmissão dos vários pacotes que compõem a mensagem.

Além dos protocolos de comunicação, existem ainda, técnicas que permitem o fluxo de pacotes pelo sistema de interconexão. Chamaremos estas de **Políticas de Controle do Fluxo de Pacotes.** Algumas políticas são mais vantajosas que outras. Assim, a escolha de uma política de controle de fluxo em detrimento de outra se baseia em vários fatores como o desempenho da comunicação e a quantidade de memória necessária para ser usada como memória de comunicação de cada nó. Estudaremos as políticas de controle de fluxo **Armazene e Siga** ("store-and-forward"), **Transpasse** ("worm-hole") e **Transpasse Virtual** ("virtual-cut-through"), as quais são muito utilizadas nos multicomputadores atuais.

Na política de controle de fluxo **Armazene e Siga** ("store-and-forward"), os pacotes são indivisíveis, do ponto de vista lógico, e em cada nó um acoplador ("buffer") é alocado por mensagem (ou pacotes) enviada. Neste caso, antes de ser retransmitido para outro nó, o pacote é inteiramente armazenado no acoplador ("buffer") intermediário (figura 2).

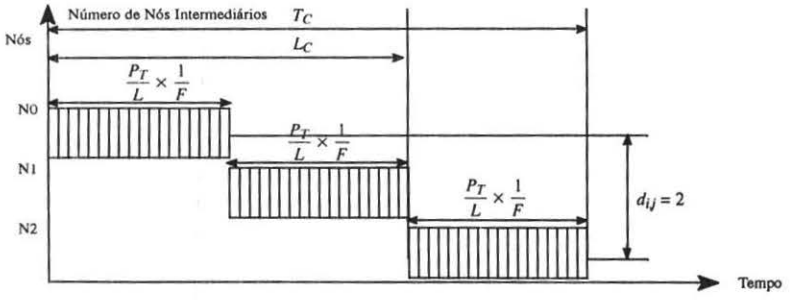


Figura 2 - Tempo de Transmissão para Política Armazene e Siga

Podemos calcular a latência de comunicação L_C, a partir da figura 2, que é expressa por:

$$L_C = \frac{1}{F} \times \frac{P_T}{L} \times d_{ij} \qquad (2.2)$$

Ou compondo as equações 2.1 e 2.2 podemos escrever T_C como:

$$T_C = \frac{1}{F} \times [\frac{P_T}{L} \times (d_{ij} + 1)] \qquad (2.3)$$

Podemos considerar que a política Armazene e Envia ("store-and-forward") é uma solução muito desvantajosa devido a latência de comunicação e tempo de comunicação serem proporcionais a distância entre os nós.

Na política de controle de fluxo de **Transpasse ("worm-hole")**, ilustrada na figura 3, os canais e acopladores ("buffers") são alocados flit a flit e a comunicação é feita diretamente entre o nó origem e o nó destino. Nesta técnica de a vantagem em relação a técnica Armazene e Siga ("store-and-forward") é dupla: os acopladores ("buffers") são de dimensão muito menor (do tamanho dos flits) e a latência de comunicação é menor.

Neste caso a latência é expressa por:

$$L_C = \frac{1}{F} \times d_{ij} \qquad (2.4)$$

O tempo de comunicação total expresso pela equação (2.4)

$$T_C = \frac{1}{F} \times (\frac{P_T}{L} + d_{ij}) \qquad (2.5)$$

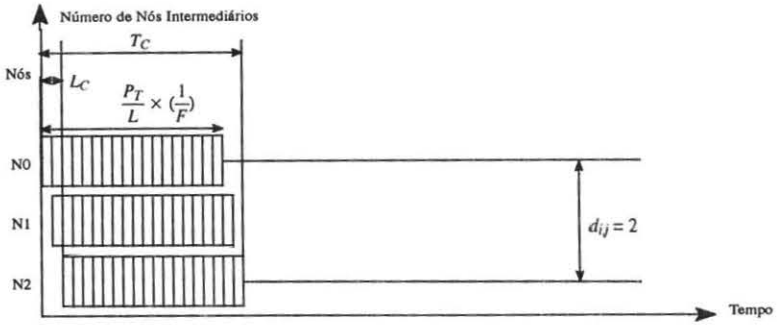


Figura 3 - Tempo de Transmissão Para a Política de Transpasse

A política de Transpasse ("worm-hole") possui várias vantagens em relação a política armazena e siga ("store-and-forward"), entre elas podemos destacar a necessidade de pouca memória de mensagem e a latência muito menor. Como desvantagem temos a maior probabilidade de colisões dada a utilização simultânea de um maior número de canais durante a transmissão do pacote do que na política armazena e siga.

3 Um Modelo Analítico Para Estimativa de Desempenho

Uma forma de para se estimar o desempenho de computadores paralelos é a utilização de modelos analíticos. Neste caso, são definidas equações matemáticas a partir de certas suposições do comportamento da arquitetura. Estes modelos, desde que bem estruturados e conceitualmente bem definidos, oferecem bons resultados.

Para a avaliação do desempenho, na literatura de forma geral, se utilizam indicadores como o **Ganho de Velocidade** e a **Eficiência**.

O Ganho de Velocidade *(S)* é definido pela relação tempo de execução em um nó (T_1) pelo tempo de execução em N nós (T_N):

$$S = \frac{T_1}{T_N} \qquad (3.1)$$

A **Eficiência** *(E)* é um indicador da utilização efetiva dos N nós utilizados na execução de um programa, a eficiência normalmente é medida em porcentagem e é definida por:

$$E = \frac{S}{N} \times 100\% = \frac{T_1}{T_N \times N} \times 100\% \qquad (3.2)$$

3.1 A Regra de AMDAHL

A regra de AMDAHL [AMDA67] vem sendo utilizada como exemplo clássico de modelo analítico para análise de desempenho de computadores paralelos pois apresenta resultados e estimativas razoáveis para algumas aplicações e computadores específicos. O modelo proposto por AMDAHL pressupõe que um programa é composto por uma secção de código intrinsicamente sequencial e uma secção de código paralelizável que pode ser distribuída pelos processadores[1].

1. Usaremos processadores aqui pois AMDAHL desenvolveu este modelo para computadores vetoriais

Neste modelo, o tempo de execução de um programa em um nó (T_1) é definido pela soma do tempo de execução de uma seção de programa intrinsecamente sequencial (T_S) mais uma seção de programa paralelizável (T_P) :

$$T_1 = T_S + T_P \tag{3.3}$$

Supondo a execução do mesmo programa num computador paralelo, o tempo de execução em cada processador (T_N) é tal que o tempo de execução serial (T_S) se mantêm e o tempo de execução paralelizável (T_P) cai proporcionalmente ao número de processadores utilizados:

$$T_N = T_S + \frac{T_P}{N} \tag{3.4}$$

Definidos T_1 e T_N, podemos calcular as equações de Ganho de Velocidade e Eficiência:

$$S = \frac{N \times (T_S + T_P)}{N \times T_S + T_P} \tag{3.5}$$

$$E = \frac{T_S + T_P}{N \times T_S + T_P} \times 100\% \tag{3.6}$$

A maior deficiência da Regra de AMDAHL é o fato de não considerar o tempo de comunicação interprocessadores, pois esta regra foi criada para a análise de computadores vetoriais convencionais, os quais têm um pequeno número de processadores partilhando uma memória principal comum. Esta regra não se adequa bem aos multicomputadores pois nestes a questão da comunicação é fundamental, principalmente em arquiteturas que envolvam um número muito grande de nós.

3.2 O modelo de STONE

Para a análise de desempenho utilizamos o modelo proposto por STONE [STON87] como ponto de partida. Este modelo foi escolhido pois considera a comunicação como fator limitante de desempenho em computadores paralelos, aderindo de forma bastante razoável à organização dos multicomputadores. A partir do modelo proposto por STONE, faremos um estudo do comportamento das aplicações, prevendo também o caso de utilização de um processador dedicado apenas à comunicação e prevendo o comportamento destas arquiteturas para um número muito grande de nós. As interpretações deste modelo modificado também foram aprofundadas levando-se em conta particularidades de sistemas multicomputadores.

3.3 Granularidade de Tarefa

A **Granularidade de Tarefa** ("task granularity") (G_M) é um conceito intimamente relacionado com o comportamento da aplicação numa certa arquitetura. A Granularidade de Tarefa é um termo consagrado na literatura e serve para definir o grau de particionamento em função da taxa de comunicação e processamento de um programa pelos processadores num computador paralelo. O desempenho de uma certa aplicação depende fortemente da Granularidade de Tarefa.

STONE quantificou esta componente. Basicamente a Granularidade de Tarefa G_M de uma certa aplicação é definida pela relação:

$$G_M = \frac{R_M}{C_M} \tag{3.7}$$

Onde R_M é o tempo de execução de uma tarefa para um particionamento M, e C_M é a sobrecarga ("overhead") de comunicação gerada pela tarefa para sincronização ou troca de variáveis com as outras tarefas.

O índice M representa o **Particionamento** da granularidade, ou seja, em quantas tarefas a aplicação foi particionada.

O conceito de granularidade está relacionado com dois aspectos: arquitetura e estratégia de paralelização. A granularidade depende de quão rápido é o processador de cada nó e seu respectivo sistema de comunicação e depende também do particionamento do problema.

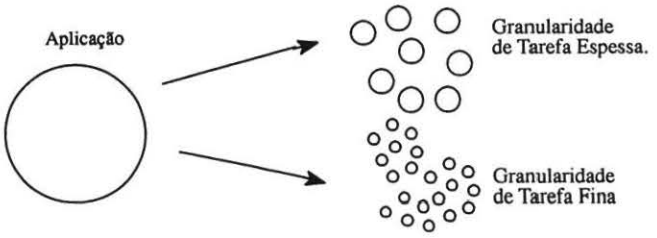


Figura 4 - Granularidade de Tarefa Fina X Granularidade Tarefa Espessa

É importante perceber também que a granularidade é função do particionamento da aplicação. A figura 4 mostra uma aplicação sendo particionada em grãos espessos e grãos finos. Uma idéia intuitiva, mas nem sempre correta, é considerar que quanto maior o particionamento menor a granularidade. No modelo de STONE, isto não corresponde totalmente à verdade, visto que a relação entre o tempo de processamento e a sobrecarga de comunicação nem sempre diminui com o aumento do particionamento.

Cabe salientar que, para um grande número de aplicações, a granularidade pode ser "controlada" ou, em outras palavras, a granularidade é fortemente dependente da forma de particionamento do problema sobre a arquitetura. Uma questão básica é, dada uma aplicação, descobrir qual a melhor granularidade e o número ótimo de nós alocados para a mesma.

Vamos supor que uma aplicação seja particionada em M tarefas, cada tarefa demanda um tempo R_M de processamento e um tempo C_M de sobrecarga ("overhead") de comunicação.

O número de tarefas executadas em um nó i é definido por k_i, Assim temos:

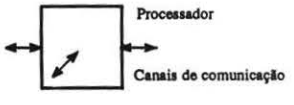
$$\sum_{i=1}^{N} k_i = M \qquad (3.8)$$

Assumiremos que a aplicação pode ser distribuída uniformemente sobre um multicomputador homogêneo. A hipósete do balanceamento uniforme será mantida visto que muitas aplicações executadas em multicomputadores possuem esta característica. Desta forma, o multicomputador está uniformemente balanceado e podemos escrever a equação (3.8) da seguinte forma:

$$k = \frac{M}{N} \qquad (3.9)$$

3.4 Nós Simples

Para uma arquitetura genérica, podemos considerar o tempo total de execução como a soma do tempo de execução das tarefas nos processadores mais o tempo de comunicação inter-tarefas. Deste modo, temos:



$$T_N = \underbrace{r(R_M, M, N, k_i)}_{\text{Tempo de Execução de Tarefas}} + \underbrace{c(C_M, M, N, k_i)}_{\text{Sobrecarga de Comunicação}} \qquad (3.10)$$

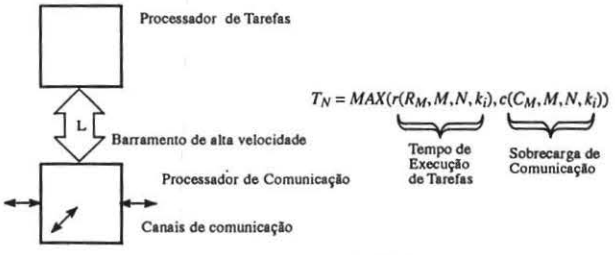
Figura 5 - Nó Simples

Na equação anterior, admite-se que para realizar uma comunicação, o processador é obrigado a interromper o seu processamento de tarefas. Neste caso, utiliza-se um processador por nó, responsável pela comunicação e processamento de tarefas. Denominaremos nós com esta organização de **Nós Simples** (figura 5).

3.5 Nós Compostos

Podemos considerar também multicomputadores em que cada nó é implementado de forma tal que são utilizados dois processadores (figura 6), um dedicado ao processamento de tarefas e outro dedicado exclusivamente a comunicação. Consideraremos ainda que são interconectados de tal forma, por exemplo utilizando canais paralelos de alta velocidade, que a comunicação entre eles possa ser considerada desprezível. Denominaremos de **Nó Composto** este tipo de nó.

Deste modo, para este tipo de nó, temos:



(3.11)

Figura 6 - Nó Composto

Considerando a situação em que apenas um nó é utilizado a sobrecarga de comunicação é nula deste modo o tempo total de execução para um nó (T_1) em ambos os casos é:

$$T_1 = \sum_{i=1}^{N} k_i \times R_M = M \times R_M \tag{3.12}$$

3.6 Estimativa da Sobrecarga de Comunicação

Já citamos que para a comunicação entre nós não adjacentes torna-se necessária a utilização de protocolos de comunicação sofisticados capazes de rotear a mensagem pela rede a fim de que ela chegue a seu destino o mais rápido possível.

A sobrecarga de comunicação (C_M) em multicomputadores é função de vários fatores como:

a) estratégia de paralelização utilizada;

b) implementação e sintonia do sistema de comunicação;

c) características topológicas;

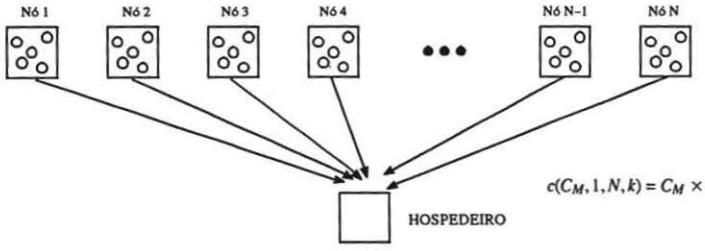
d) políticas de roteamento adotadas;

e) distribuição de tarefas.

A seguir faremos uma estimativa da comunicação em função do número de nós N e número de tarefas M.

3.7 Estimativa: Difusão De Todos Para Um

Uma situação muito comum encontrada em programas executados em multicomputadores é a difusão de uma mensagem de todas as tarefas para um nó da rede. Denominaremos esta comunicação de **Difusão de Todos Para Um** ("All-To-One Broadcast") (figura 7). Neste caso a estimativa da sobrecarga de comunicação é facilmente expressa pela equação (3.13).



$$c(C_M, 1, N, k) = C_M \times M \quad (3.13)$$

Figura 7 - Difusão de Todos Para Um

Estas situações são encontradas em muitas aplicações onde um processo gerente controla a execução de tarefas em processos escravos. Normalmente o processo gerente é executado no hospedeiro do sistema. Um aspecto interessante é que a comunicação não depende do número de nós do sistema, mas sim do particionamento da aplicação.

3.8 Análise de Desempenho Para *N* Nós Simples

Vamos fazer a análise de desempenho considerando *N* nós simples, das equações (3.13) e (3.10) extraímos:

$$T_N = \frac{R_M \times M}{N} + C_M \times M \quad (3.14)$$

a partir da equação (3.14) obtemos o ganho de velocidade e de eficiência:

$$S = \frac{(G_M + 1) \times N}{G_M + N} \quad (3.15)$$

$$E = \frac{G_M + 1}{G_M + N} \quad (3.16)$$

Nas equações (3.15) (3.16), observamos, que tanto as equações de ganho de velocidade como de eficiência não dependem diretamente do particionamento, mas sim apenas da granularidade. Neste caso granularidades grandes são interessantes pois o ganho de velocidade nestas situação se aproximam da situação ideal.

Para um número muito grande de nós temos:

$$\lim_{N \to \infty} S = G_M + 1 \quad (3.17)$$

$$\lim_{N \to \infty} E = 0 \quad (3.18)$$

Os gráficos da figura 8 apresentam as curvas de ganho de velocidade e eficiência para um número variável de nós e granularidades tanto para nós simples como para nós compostos.

escalabilidade continua limitada pelo mesmo valor quando utilizamos nós simples, ou seja, nesta classe de aplicações a utilização de nós compostos não aumenta a capacidade máxima de processamento do sistema que continua limitada pela equação (3.22).

4 Análise de Desempenho Experimental de Uma Aplicação Gráfica

Para a análise experimental de desempenho foi paralelizado um programa de síntese de imagens baseado na técnica que denominaremos de traçado de raios, ou rastreamento de raios ou ainda de lançamento de raios ("ray-tracing") ([WHIT80], [GREE91]). Esta técnica de visualização tridimensional foi escolhida pela sua intensa demanda de processamento e memória, bem como pela sua simplicidade e potencial de paralelização. Como plataforma de análise foi utilizado o protótipo do Multitransputador Gráfico TRGR_01 ([LOPE90],[LOPE93], [ZUFF93]).

Para a avaliação experimental foi escolhido o programa MTV, de domínio público, que implementa a técnica de traçado de raios. O MTV foi compilado e executado sobre o sistema TRGR_01 utilizando as ferramentas oferecidas pelo sistema de programação EXPRESS 2.0. As medidas foram realizadas para diferentes granularidades, a fim de se estudar o comportamento do sistema em diversas situações.

4.1 A Técnica de Lançamento de Raios

Apresentada em 1980 por WHITTED ([WHIT80], [GREE91]), a técnica de Lançamento de Raios ("ray-tracing") é uma das técnicas mais populares de apresentação de imagens.

Apesar da qualidade de imagem sintetizada e sua difusão pela comunidade acadêmica e industrial, o lançamento de raios é uma técnica extremamente custosa para geração de imagens. O atual estágio tecnológico não permite ainda a síntese de imagens usando esta técnica em tempo real.

A algoritmo é relativamente simples. Basicamente temos um observador, uma tela de projeção e uma cena onde estão descritos os objetos e suas cacarterísticas ópticas, bem como as fontes de luz. A partir do observador raios são lançados para cada pontel que compõe a tela de projeção. Estes raios interceptam os vários objetos que compõe a cena e em cada intersecção uma equação de modelamento de iluminação global é utilizada para descobrir a contribuição de cor daquele ponto. Recursivamente novos raios são gerados a partir da interseção e novamente, caso hajam novas intersecções, novos raios são gerados.

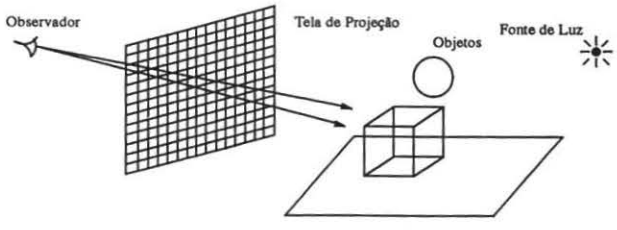


Figura 9 - Traçado de raios

4.2 Estratégia de Paralelização Adotada

Muitos trabalhos já foram desenvolvidos sobre a paralelização da técnica de traçado de raios ("ray-tracing"). Um estudo comparativo das abordagens de paralelização, bem como inúmeros exemplos podem ser encontrados em ([SANT93], [SPEE91], [ZUFF91], [ZUFF93].

A paralelização utilizada é extremamente simples e baseada no particionamento do espaço de imagem, visto que cada raio lançado é independente em relação aos outros. O espaço de tela, de resolução $X \times Y$ é dividido

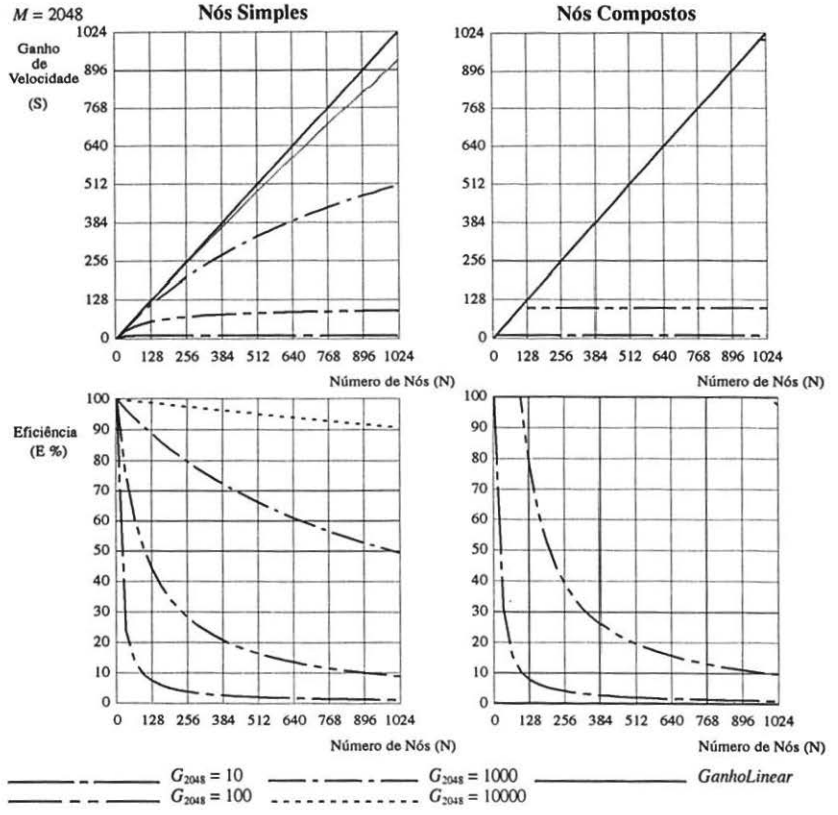


Figura 8 - Gráficos Para Difusão de Todos Para Um Para Nós Simples e Nós Compostos

3.9 Análise de Desempenho Para *N* Nós Compostos

Considerando agora a nossa análise para nós compostos temos:

$$T_N = MAX(\frac{R_M \times M}{N}, C_M \times M) \qquad (3.19)$$

Como agora o tempo total depende do valor máximo dos dois termos, a função deve ser considerada por partes e obtemos:

$$S = N \quad E = 100\% \quad \text{para} \quad N \leq G_M \qquad (3.20)$$

$$S = G_M \quad E = \frac{G_M}{N} \quad \text{para} \quad N \geq G_M \qquad (3.21)$$

$$\lim_{N\to\infty} S = G_M \quad \lim_{N\to\infty} E = 0 \qquad (3.22)$$

Comparando as equações (3.22) e (3.17), percebemos que elas são praticamente idênticas, nesta aplicação também a utilização de nós compostos melhora significativamente a eficiência dos nós porém a

em pequenos retângulos de dimensão a x b. Desta forma o particionamento M é definido pelo número de retângulos a serem processados (equação (4.1)):

$$M = \frac{X \times Y}{a \times b} \quad (4.1)$$

X = Resolução Vertical da Imagem
Y = Resolução Horizontal da Imagem
a = Resolução Vertical do Retângulo
b= Resolução Horizontal do Retângulo

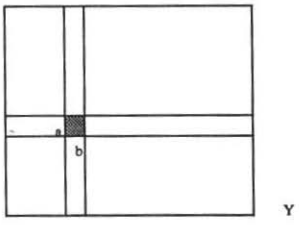


Figura 10 - Resolução da Tela e Número de Tarefas

Cada pontel da imagem é representado por um byte. Desta forma, o tamanho de cada mensagem enviada pelos nós de processamento ao nó de controle é definida pela equação:

$$M_T = a \times b \qquad (bytes) \qquad (4.2)$$

Neste caso, todos os nós executam o mesmo programa e a cada um deles é atribuído um determinado número de pacotes. O nó 0 é responsável pela gerência dos retângulos e atualização da tela. O processo executado no nó 0 é denominado gerente, pois é responsável por distribuir as tarefas realizadas pelos outros nós e compor o resultado fornecido pelos processos escravos na memória gráfica.

Os processos escravos são responsáveis pela execução do traçado de raios para os vários retângulos que compõem a cena. Eles recebem sua tarefa do processo gerente (dimensões do retângulo a ser executado), executam a tarefa e devolvem o resultado, no caso, o retângulo com os respectivos pontéis calculados para o processo gerente. Neste caso temos a **difusão de todos para um** estudada no item 3.7. Desta forma podemos fazer a análise de desempenho utilizando as ferramentas definidas e estimar o desempenho caso fossem utilizadas várias placas TRGR_01 associadas.

As hipóteses definidas são mantidas. O balanceamento de carga dos nós é uniforme, pois supondo um número grande de tarefas, na média todos executarão aproximadamente M/N tarefas.

O modelo de nó utilizado é o **nó simples**, este modelo foi escolhido dada as particularidades de implementação do microprocessador IMST800 utilizado na TRGR_01.

4.3 O Multicomputador TRGR_01

O Multitransputador para Aplicações Gráficas TRGR_01 [LOPE90] é um multicomputador baseado em microprocessadores disponíveis comercialmente. Basicamente o sistema é composto por placas modulares, cada uma composta de um a nove nós. Cada nó é composto por um microprocessador IMST800 [INMO90], e uma quantidade de memória de 1 a 4 Mbytes de memória RAM. Estas placas podem ser conectadas entre si formando um multicomputador com dezenas a centenas de nós.

Um transputador ("transputer") é um microprocessador desenvolvido especialmente para a implementação de sistemas distribuídos, possuindo canais de comunicação que permitem ligar um transputador ao outro [INMO90].

O Sistema de Interconexão é composto por um Chaveador de Elos implementado pelo circuito integrado da INMOS IMSC004 [INMO90] e uma interface para interconexão de outras placas.

Foi implementada uma versão da TRGR_01 com 9 nós. Este protótipo foi inserido numa estação de trabalho Sun386i da companhia Sun Microsystems.

Como ambiente de programação foi instalado o ambiente EXPRESS. O EXPRESS ([PARA90a], [PARA90b], [PARA90c] é um produto comercial desenvolvido pela empresa PARASOFT [PARA90a] [PARA90b]. O EXPRESS é um sistema de programação portável para multicomputadores homogêneos.

O EXPRESS é composto por rotinas de alto nível para a alocação e carga de programas em "C" e FORTRAN, rotinas para a comunicação entre processos, rotinas para o particionamento regular da estrutura de dados e o sistema de Entrada/Saída.

O EXPRESS implementa rotinas de comunicação em vários níveis. O nível de comunicação mais alto implementa o protocolo de comunicação assíncrono com bloqueio e sem bloqueio, a política de controle de fluxo de pacotes utilizada é uma variante do transpasse ("worm-hole") denominado transpasse virtual ("virtual-cut-trought") [KERM79].

4.4 Sintonia do Sistema de Comunicação

No programa paralelizado a sobrecarga de comunicação existe devido à comunicação dos nó gerente para a distribuição das tarefas, bem como, a comunicação dos escravos com o gerente para a devolução dos dados já calculados.

A granularidade G_M pode ser sintonizada variando-se o tamanho dos retângulos que compõe a imagem final. Quanto menores os retângulos maior será o número de pontéis a serem transmitidos implicando maior a sobrecarga de comunicação, diminuindo assim a granularidade.

Considerando o particionamento expresso por (4.1), o modelo de comunicação caracterizado pela difusão de todos para um e a sobrecarga de comunicação definida pelo tempo de comunicação levantado no ítem 2, podemos levantar analiticamente o tempo de execução a partir das equações (2.5) e (3.14):

A equação (4.2) nos oferece conclusões muito interessantes quanto a influência da latência no desempenho global do sistema. Percebemos claramente que quanto maiores os retângulos maior o produto $a \times b$ e consequentemente menor a influência da latência sobre o desempenho do sistema. Caso os retângulos sejam muito pequenos a latência passa a aumentar a sobrecarga de comunicação, diminuindo a Granularidade e o a Eficiência e Ganho de velocidade global do sistema.

Para a realização das medidas foi fixada uma resolução de imagens de 512x512. Os dados experimentais foram levantados para até 8 nós e a cena utilizada foi a mesma para todas as medidas.

$$T_N = \frac{R \times M}{N} + (X \times Y) \times \left(\frac{L_C}{a \times b} + \frac{1}{F}\right) \qquad (4.2)$$

A partir dos dados adquiridos experimentalmente foram construídos os gráficos das figura 11 respectivamente os gráficos de ganho de velocidade e eficiência. Analisando os resultados obtidos constatamos que o erro entre o levantamento experimental e o levantamento teórico usando as equações definidas no item 3 é da ordem de 3% a 5%. Este erro deve-se provavelmente a imprecisões na medida dos tempos de execução da aplicação.

Uma outra possível fonte de erro é uma limitação do modelo em não modelar a taxa de colisões. Este erro ocorre principalmente quando temos um número muito grande de nós. A latência tende a crescer aumentando consequentemente a sobrecarga de comunicação e modificando o valor da granularidade. Dado o pequeno número de nós, este efeito não foi notado.

5 Limitações do Modelo

O modelo apresentados neste artigo é limitado em certos aspectos devidos as várias considerações utilizadas para a formulação e desenvolvimento das equações. As principais limitações são listadas abaixo.

a) Nem sempre o balanceamento de carga pode ser considerado ideal, muitas aplicações não são homogêneas.

b) A formulação da sobrecarga de comunicação não levou em conta características do sistema de interconexão como a topologia e algoritmo de roteamento.

c) Em alguns multicomputadores a latência aumenta proporcionalmente com a distância dos nós e em função da taxa de colisão entre mensagens e da política de fluxo de mensagens, sendo um fator que muito contribui para a degradação do sistema de comunicação

Desta forma possíveis extensões do modelo proposto podem ser feitas levando-se em conta outros parâmetros como a taxa de colisões topologia, protocolos de comunicação e roteamento

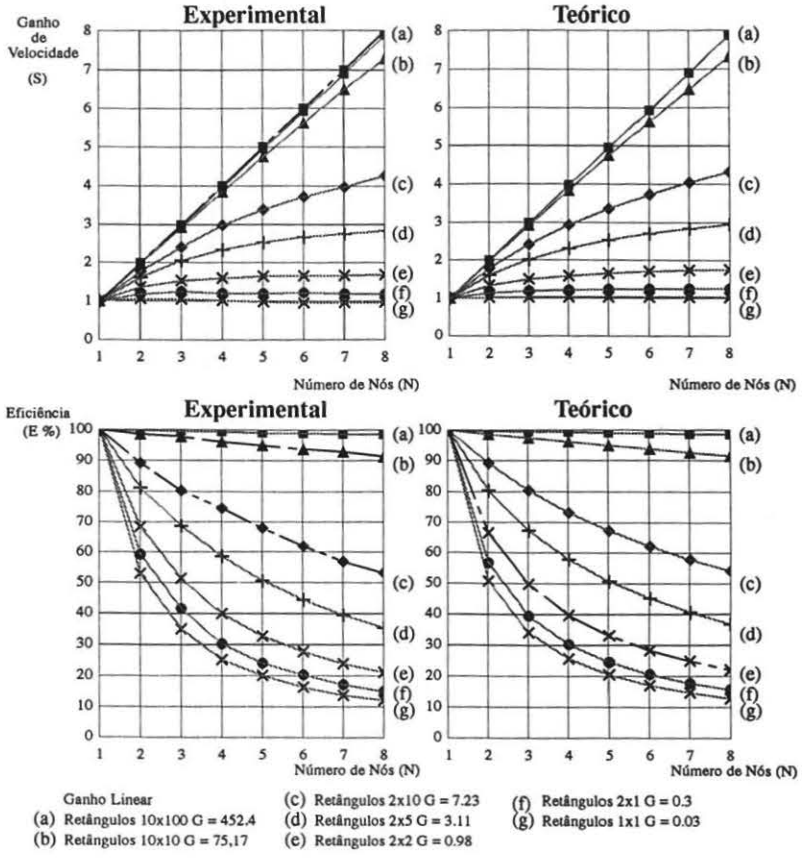


Figura 11 - Ganho de Velocidade Levantamento ExperimentalxTeórico

6 Conclusões

Neste artigo decrevemos um modelo analítico para a estimativa de desempenho de multicomputadores, levantamos experimentalmente o desempenho de uma aplicação gráfica no protótipo do multicomputador TRGR_01, e comparamos estes resultados com os resultados oferecidos pelo modelo analítico.

Como conclusões importantes comprovamos a importância da sintonia do sistema de comunicação nos multicomputadores, particularmente a escolha de pacotes com tamanho adequado, e o efeito dramático que a latência de comunicação tem sobre o desempenho do multicomputador. Cabe salientar que apesar do sistema EXPRESS 2.0 implementar uma política extremamente eficiente de controle de fluxo de pacotes, no caso o transpasse virtual, a latência ainda influi significativamente no sistema de comunicação.

Outra contribuição importante deste artigo foi a comprovação experimental de uma estimativa de desempenho apresentada no item 3.

Estas equações são muito utéis para o levantamento teórico aproximado do desempenho do sistema a partir de alguns pontos tomados de curvas experimentais ou a partir da extração de parâmetros como a granularidade e o particionamento, parâmetros estes levantados sem muita dificuldade na maioria dos sistemas multicomputadores.

Uma outra conclusão importante é que o investimento em nós compostos melhora de forma significativa a eficiência das aplicações estudadas. Esta conclusão poderá ser validada no futuro em outros multicomputadores que incorporem este tipo de organização de nós.

7 Bibliografia

[AMDA67] AMDAHL. G. M. Validity of the single processor aproach to achieving large scale computing capabilities. **Proceedings of AFIPS**. Reston Va. 1967. p.483-485

[BELL92] BELL. C. G. Ultracomputers. a teraflop before its time. **Communications of ACM**. Agosto 1992. V.35. N.8

[FOX88] FOX. G. et al. **Solving problems on concurrent processors volume I**. Prentice Hall. 1988 New Jersey.

[GREE91] GREEN. S. **Parallel processing for computer graphics**. Research Monographs in Parallel and distributed Computing. The MIT Press. Cambridge Massachussets. 1 ed. 1991.

[HOCK88] HOCKNEY. R. W.; JESSHOPE. C. R. **Parallel computers 2**. IOP Publishing Ltd. 1988. 2. ed.

[HWAN84] HWANG. K.; BRIGGS. F. A. **Computer architecture and parallel processing**. McGraw-Hill Book Company. 1984. New-York.

[HWAN89] HWANG. K.; DEGROOT. D. **Parallel processing for supercomputers**. McGraw-Hill Book Company. 1989. New- York.

[INMO90] INMOS **The transputer dataBook**. 2. ed. 1990 Inmos Corporation.

[JSCA91] JSCAD 91 **Jornada EPUSP/IEEE em sistemas de computação de alto desempenho**. J.A. Zuffo. ed.. São Paulo. março de 1991

[KERM79] KERMANI. P; KLENROCK. L. Virtual cut-trought a new computer communication switching technique. **Computer Networks** 3(4). p.267-286. sept. 1979.

[LOPE90] LOPES. R. L.; ZUFFO. M. K. **Sistema multitransputador para aplicações gráficas**. Anais do III SIBGRAPI - Simpósio Brasileiro de Computação Gráfica e Processamento de Imagens. Gramado RS. 1990. p.383-386.

[LOPE91] LOPES. R. D.; ZUFFO. M. K. **Placa gráfica multitransputadora TRGR_01**. Documento de Especificação Versão 1.0 rev. 1.0. relatório Técnico LSI. São Paulo. 5 de janeiro de 1991.

[LOPE93] LOPES. R. D. **O multicomputador TRGR e a paralelização da síntese de imagens** Dissertação de Mestrado EPUSP 1993.

[PARA90a] PARASOFT **Express C user's guide version 3.0.** Parasoft Corporation. Pasadena CA USA. 1990.

[PARA90b] PARASOFT **ExpresS introduction guide version 3.0.** Parasoft Corporation. Pasadena CA USA. 1990.

[PARA90c] PARASOFT **Express C reference guide version 3.0.** Parasoft Corporation. Pasadena CA USA. 1990.

[SANT93] SANTOS. E. T. **Avaliação do algoritmo de ray-tracing em multicomputadores.** São Paulo 1993 Dissertação (Mestrado) - Escola Politécnica, Universidade de São Paulo.

[SPEE91] SPEER. R. L. **A cross indexed guide to the ray-tracing literature.** Computer Graphics Forum. n.10 1991. p. 145-174.

[STON87] STONE. H. S. **High performance computer architecture**. 1.ed.. Addison-Wesley. 1987.

[SUAY90] SUAYA R.; BIRTWISTLE G. **Frontiers. VLSI and parallel computation.** Morgan and Kaufmann Pub. Inc. San Mateo USA 1990.

[WHIT80] WHITTED. T. An improved illumination model for shaded display, **Communications of ACM**. v. 23. n. 6. june 1980.

[ZUFF91a] ZUFFO. M. K.; JECEL M. A.; LOPES. R. D.; KOFUJI. S. T. **Processamento paralelo aplicado à computação gráfica.** Tutorial III Simpósio Brasileiro de Computação Gráfica e Processamento de Imagens São Paulo julho 1991.

[ZUFF93] ZUFFO. M. K. **Um Multicomputador Para Aplicações Gráficas**, São Paulo, 1993, dissertação (mestrado). Escola Politécnica Universidade de São Paulo.