DIVISÃO DINÂMICA EM ANEL DO CÁLCULO DO "RAY TRACING"

DANIEL CÂMARA
ORIENTADOR PROF. ANDRÉ L. PIRES GUEDES

Universidade Federal do Paraná
Setor de Ciências Exatas
Departamento de Informática - Dinf
Centro Politécnico - Jardim das Américas
Cep 81531-990 Curitiba - Paraná
daniel@cce.ufpr.br
andre@inf.ufpr.br

Abstract: Ray tracing is an algorithm to generate realistic images from three-dimensional scenes. The main goal of this paper is to give a parallel alternative to calculate the ray tracing, improving the traditional form, which is inefficient. We're going to show some ways to paralelize the algorithm and our solution, which consists of the dynamic division of the image plane among process organized in a structure similar to a ring network.

Keywords: Ray-Tracing, Parallelism, atom.

1 Introdução

"Rendering" é a área da computação gráfica que estuda o processo de produção de imagens ou figuras realistas [Comba (1990)] . O *"Ray Tracing"* é um algoritmo para a produção deste tipo de imagens, introduzido em sua forma revisada por Whitted [Whitted (1980)] e permite a obtenção de imagens geradas com alto grau de realismo. A idéia do método é principalmente simular a interação da luz com uma cena utilizando conceitos de Ótica Geométrica [Comba (1990)] .

Os resultados do método são muito bons, mas este apresenta algumas desvantagens, como o alto custo computacional, necessitando de uma grande quantidade de memória. Para se acelerar o processo várias propostas foram feitas, como a utilização de técnicas de coerência [Martino (1991)] [Baudouel (1994)], octrees [Comba (1990)], uso de arquiteturas paralelas [Corrêa (1993)] e processamento distribuído [Silva (1994)]. Temos basicamente quatro técnicas de aceleração de "ray tracing" [Corrêa (1993)]: realizar intersecções mais rapidamente, lançar menos raios, utilizar entidades mais gerais e executar operações concorrentes.

Abordaremos a paralelização do processo de cálculo da imagem como forma de aceleração do mesmo, para cenas descritas pelo modelo CSG ("*Constructive Solid Geometry"*) de representação de cenas. Primeiramente faremos uma introdução mostrando a forma como é realizado o cálculo do Ray Tracing e posteriormente falaremos do paralelismo em si e como este pode ser aplicado ao "ray tracing", introduzindo nossa solução, com suas vantagens e desvantagens.

2 Algoritmo canônico de "ray tracing" de sólidos CSG

O modelo CSG para representação de sólidos por volume, consiste de uma estrutura composta por sólidos primitivos[1] e combinados por operações booleanas, (*união, intersecção e diferença* (Figura - 1)) . A melhor forma de se armazenar a estrutura de um sólido é uma árvore binária, onde as folhas são os sólidos primitivos e os nodos intermediários são as operações realizadas sobre os sólidos primitivos, gerando assim o novo sólido que é representado pela raiz. A esta estrutura dá-se o nome de *"Árvore CSG"*.

1. A idéia de sólidos primitivos pode ser bastante ampla, mas normalmente são sólidos com os quais o cálculo de intersecção do sólido com uma linha seja simples (*esfera, plano, cone, cilindro).* Com eles, através das operações booleanas, podemos criar sólidos mais complexos.

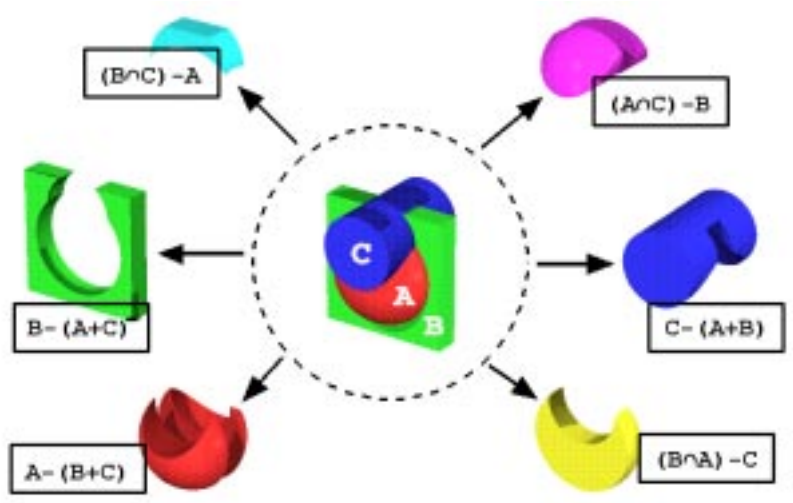

Figura 1 - Mostramos aqui como é a composição de uma figura no modelo CSG. A, B, C seriam as primitivas, estariam nas folhas da árvore, enquanto a composição final seria a raiz da árvore.

O cálculo de intersecção dos raios com cada nodo da árvore é feito conforme a operação booleana aplicada sobre os nodos filhos desta árvore. Em uma operação de união, por exemplo (Figura - 2), todos os pontos do segmento de reta (raio) que pertencerem a qualquer um dos objetos devem ser considerados como intersecção com o sólido.

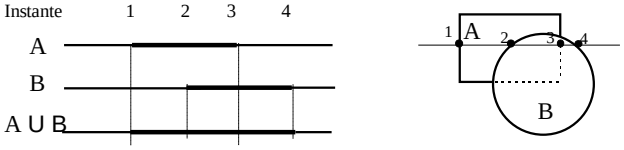

Figura 2 - Mostramos aqui como se faz a operação de intersecção de um raio com um objeto composto pela operação de união, parte mais escura da linha A U B, é a parte do segmento de reta que intercepta com o sólido composto.

Somente os sólidos primitivos, suas posições e tamanhos originais, não são suficientes para a representação das cenas. É necessário então que tenhamos operações que instanciem estes objetos de forma que possamos deformá-los e posicioná-los de acordo com as exigências da cena que queremos representar. As transformações geométricas a que poderemos sujeitar as primitivas são:

Translação: Movimentação da primitiva no espaço

Rotação: Rotacionar a primitiva com relação a seus três eixos (*x, y, z*)

Escala: Modificar a escala da primitiva com relação a seus três eixos (*x, y, z*)

Poderemos assim criar sólidos mais complexos apenas utilizando as transformações sobre as primitivas, podemos por exemplo, conseguir uma elipsóide alterando a escala de uma esfera.

3 Cálculo do "Ray Tracing"

Calculamos o segmento de reta que passa pelo observador e pelo ponto do plano de observação (pixel). Descobrimos então quais os objetos que esta reta intercepta e qual está mais próximo, se o valor for negativo o objeto é ignorado pois está atrás do observador.

A cor do pixel em questão é calculada de acordo com as características ópticas do objeto interceptado. Sua equação é [Comba (1990)] :

$$Cp = K_a I_a + K_d \sum_{J=1}^{NrpFL} Ij(N \bullet L_j) + K_s \sum_{j=1}^{NroFL} I_{lj}(S \bullet R_j)^n + K_s I_s + K_t I_t$$

Onde :

Cp - Cor do ponto.

K_a, Kd, Ks, Kt- Coeficientes de Reflexão Ambiente, Difusa, Especular, Refração

I_a- Intensidade Ambiente

I_{lj}- Intensidade da j-ésima fonte de luz

S- Vetor do raio que vai de encontro ao observador

Rj- Vetor do raio Refletido

I_s- Intensidade proveniente do acompanhamento de reflexões

I_t- Intensidade proveniente do acompanhamento de refrações

N - Normal à Superfície

L_j- j-ésima fonte de luz

NroFL- Número de fontes de luz

n- Constante especular de Phong

A partir do ponto calcula-se a quantidade de luz que este ponto recebe da fonte de luz, e quanto ele recebe de luz refletida e refratada de outros objetos. Tem-se então dois novos raios que devem ser seguidos, gerando uma árvore de raios a calcular (Figura - 3) até que a quantidade de luz que o raio possa contribuir para a quantidade total do pixel seja irrelevante, ou que a árvore do raio chegue em um nível que consideraremos como limite (por exemplo 10 níveis), ou que o raio não intercepte nenhum outro objeto da cena. Teremos então o valor do raio neste nível, somamos com o valor do nível anterior da recursão, isto para os coeficientes de luz, que é o que estamos formando neste momento, chegando ao primeiro nível com os coeficientes completos.

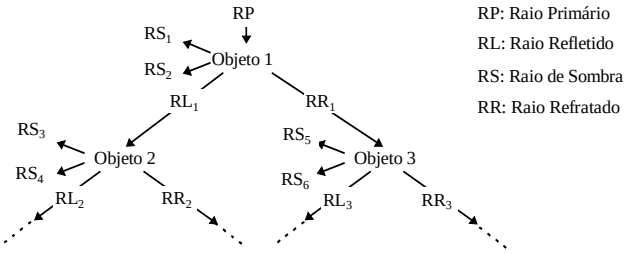

Figura 3 - Esta é a árvore de derivação de raios gerada por um ray tracing.

4 Paralelização do "ray tracing"

O "Ray tracing" é altamente paralelizável, cada ponto é completamente independente de seu vizinho (isto se não estivermos levando em consideração coerência de imagens). Uma paralelização ideal seria um processador para cada ponto da imagem, fazendo acesso a uma memória comum. Isto seria possível em uma máquina SIMD, mas mesmo assim seria inviável pelo tamanho das imagens.

É importante que neste ponto façamos uma distinção entre a cena, que são os objetos do mundo real (tridimensional), e a imagem, que é a representação desta cena em um plano bidimensional. Com isto em mente, uma das classificações possíveis do Ray Tracing é em termos de divisão tridimensional (ou espacial como classificado em [Priol (1989)]), e divisão bidimensional.

5 Divisão Tridimensional

Há várias formas de se subdividir tridimensionalmente a cena [Priol (1989)]. Uma forma simples de se paralelizar o algoritmo é dividir a cena entre os processadores, por exemplo em uma "octree" [Comba (1990)] . Cada processador fica responsável pelo cálculo das intersecções com uma das partes, e de enviar, quando necessário, o raio para outro nodo, para que seja terminado o cálculo do raio [Silva (1994)]. Esta solução se implementada em uma máquina multi-processada tem suas vantagens. A velocidade dos processadores é idêntica, ou similar, o mesmo é válido para a velocidade de acesso à cena na memória e o custo de comunicação entre os processadores é relativamente baixo. Porém se aplicada a uma rede de computadores, estas vantagens somem pelo alto custo da comunicação entre as máquinas da rede.

Outras formas são possíveis neste tipo de divisão, mas ela pode ser muito perigosa, se considerarmos um ambiente de rede, pois exige muita comunicação entre os nós exigindo assim muito da rede.

6 Divisão Bidimensional

É quando fazemos a paralelização a partir da divisão da imagem. Podemos dividi-la em dois casos distintos: distribuição estática e distribuição dinâmica.

6.1 Distribuição Estática

Tem como preocupação principal uma estratégia para a divisão da imagem, baseando-se em uma estimativa da quantidade de cálculo gerada pela mesma. O objetivo é que o trabalho de cálculo seja distribuído equilibradamente entre os processadores disponíveis. O processo só é eficiente se a estimativa o for. O grande problema neste caso é a qualidade da estimativa versus o custo computacional necessário para gerá-la. Muitos fatores podem ser levados em consideração nesta estimativa, desde a aproximação do cálculo necessário por área da figura baseado no número de objetos, até a velocidade dos processadores disponíveis, custo de rede, carga de trabalho que o processador disponível já tem (em uma rede de computadores a máquina mais rápida, por exemplo, pode estar com sobrecarga de trabalho, tornando-se "lenta". Isto deve ser levado em conta na hora da divisão do cálculo), entre outros. Estes fatores nem sempre são facilmente obtidos.

O exemplo mais simples deste tipo de distribuição é a divisão do cálculo da imagem pelo número exato de processadores disponíveis. Esta forma se implementada em uma máquina SIMD é razoável, pois não temos replicação de dados, e os processadores, teoricamente, têm a mesma velocidade. Mas mesmo assim algum processador pode ficar sobrecarregado, já que as partes da imagem não têm necessariamente a mesma complexidade, alguns processadores terminariam antes que outros ficando ociosos. Se pensarmos em uma rede de computadores (caso que consideramos mais interessante, devido aos problemas que podem ocorrer) fica um pouco pior, pois temos que distribuir entre os computadores além da imagem, também a cena. Temos ainda o problema de uma máquina mais lenta na rede (caso de uma rede heterogênea: o tempo de execução do algoritmo é o tempo da máquina mais lenta terminar sua parte do cálculo), ou num local onde o custo de rede seja alto, subtilizando os outros computadores, o que não é uma coisa desejável.

6.2 Distribuição Dinâmica

Divide-se a imagem em mais partes que o número de processadores disponíveis. Através de uma estratégia de gerenciamento de pacotes, aloca-se seus cálculos segundo a disponibilidade dos processadores. Consegue-se assim uma distribuição do cálculo da imagem de forma razoavelmente equilibrada e sem um alto custo computacional para obtê-la. Na distribuição dinâmica ainda há uma preocupação com o tamanho e quantidade de pedaços da imagem que devem ser enviadas aos processadores, mas bem menor.

Como exemplo deste tipo de divisão, podemos dividir a imagem em N partes, este N maior que os processadores disponíveis. O processo pai (chamador), divide a imagem e envia um pedaço desta juntamente com a cena para os processos filhos (fazem o cálculo). Estes ao acabarem de calcular a parte que lhe foi dada, enviam os resultados ao pai, requisitando também outra parte. Recebem uma nova ou uma mensagem avisando que não são mais necessários.

Esta forma já melhora bastante a questão de aproveitamento das máquinas. Podemos ainda fazer um cálculo sobre as requisições de novas áreas, detectando máquinas *provavelmente* mais lentas da rede. Conseguimos assim fazer um balanceamento dinâmico das cargas das máquinas. Porém a melhora do aproveitamento das máquinas disponíveis tem um custo, o pedido de partes para o cálculo. O método utiliza-se bem mais da rede, mas esse aumento de comunicação não é uma coisa desejável, e se a rede estiver lenta o algoritmo pode ter um desempenho pior que o anterior. Temos também um gargalo que é o processo pai controlando tudo, desde a distribuição e redistribuição da imagem até a organização depois de cada parte calculada. Continuamos com o problema da granularidade: *qual é o tamanho ideal para os pedaços da imagem?* Este tamanho ideal varia de acordo com o número de máquinas disponíveis, suas velocidades e a velocidade da rede, ou seja, dificilmente implementado dinamicamente.

7 Modelo Bidimensional em anel

Nossa proposta é uma melhoria sobre as duas anteriores por divisão bidimensional (Distribuição Estática e Dinâmica). Dividimos a imagem exatamente entre o número de máquinas disponíveis, cada um dos filhos subdivide sua parte em n partes iguais que chamaremos de átomos (átomos porque não mais poderão ser divididos). Cada filho cria um novo processo para o cálculo dos átomos, um neto. Este novo processo neto recebe através de um pipe o número do átomo que deve calcular. Ao término deste cálculo, envia os resultados ao processo que o criou (filho), estes vão sendo armazenados para serem repassados ao pai no final. Após o cálculo de cada átomo incrementa-se um contador de átomos e enfileira-se mais um no pipe.

Quando um processo termina a faixa da imagem que lhe foi dada calcular, envia para o pai os resultados e uma oferta de ajuda ao seu irmão da esquerda, pois já terminou o que estava fazendo e pode ajudá-lo se for necessário. Informa também o número de átomos já calculados. Se o irmão da esquerda não precisar de ajuda (já terminou seu cálculo, ou não é mais vantajosa a ajuda), envia o pacote de oferta de ajuda para o irmão que está a sua esquerda. Este pega a mensagem, compara o número de átomos que o irmão calculou com o que ele próprio já calculou, podendo desta forma fazer uma média da velocidade do irmão baseado na sua própria velocidade. Decide assim quantos átomos deve pedir ao irmão que calcule (Figura - 4) fazendo assim um balanceamento dinâmico das cargas de cálculo.

Os filhos ficam organizados como estando em uma rede em anel, e a mensagem de oferta de ajuda, funciona como um token [Baudouel (1994)]. Quando algum computador precisar da ajuda, consome o token e envia diretamente ao irmão que fez a oferta, quais átomos este deve calcular.

A função do filho acaba quando, ao enviar um pacote de oferta de ajuda para o irmão da esquerda, receber o **mesmo pacote** de seu irmão à direita, ou seja, o pacote

passou por todos os seus irmãos e nenhum precisou de ajuda. Neste momento a imagem já deve estar completa, ou quase completamente calculada, ficando aquele filho à espera de uma mensagem do pai avisando que não é mais necessário. Isto inviabiliza a possibilidade de um deadlock entre os irmãos.

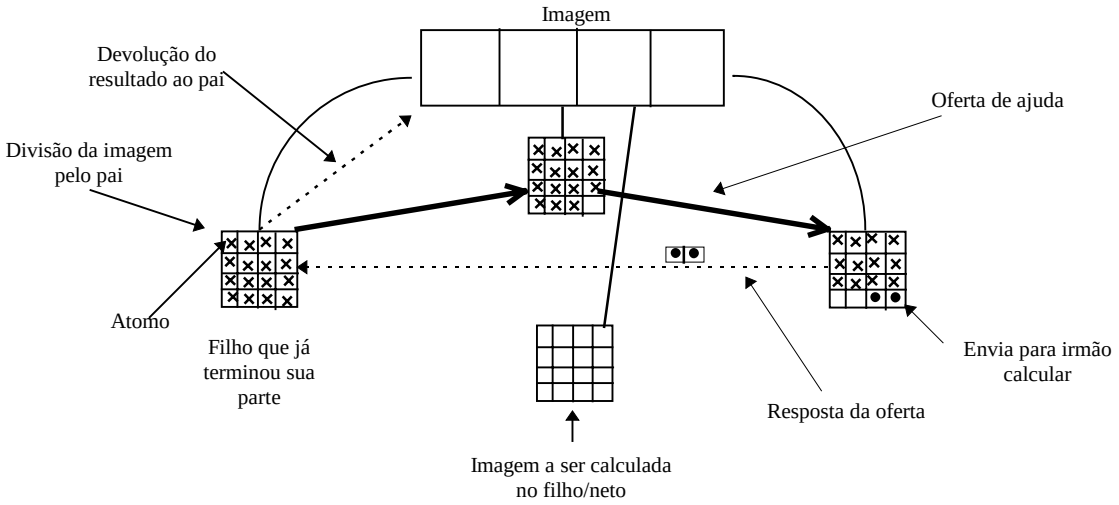

Figura 4 - Descrição de como funciona o cálculo de uma cena paralelizada pela nossa solução.

O algoritmo ocupa mais rede que o primeiro, e menos que o segundo. Seu desempenho é melhor que o segundo em termos de recursos computacionais, pois o computador não fica parado enquanto espera as partes que deve calcular (os átomos ficam enfileirados no pipe) . A perda com ociosidade é mínima se for bem implementada a granularidade dos átomos e a comunicação dentro do anel. É mais fácil encontrarmos um valor ideal à granularidade entre os átomos do que entre as partes da imagem, estas bem maiores. Com a responsabilidade de redistribuição dos átomos para os filhos, acabamos com o gargalo do pai, que agora só é responsável pela distribuição inicial e a organização das partes da imagem já calculadas, o que não representa um gargalo.

Mesmo se ocorrer de uma máquina da rede ser muito lenta e outras rápidas não é problema, esta máquina não atrapalha muito, já que seus irmãos virão em sua ajuda. Se a máquina lenta consumir todas as ofertas de ajuda ainda assim não é problema, pois todas as máquinas estarão ocupadas, e quando terminarem o cálculo destinado à máquina mais lenta, estarão disponíveis a ajudar os outros irmãos. Mesmo a máquina mais lenta pode ajudar ainda. Os irmãos sabem de sua baixa capacidade (quantidade de átomos já calculados) alocando então para o que fez a oferta uma quantidade de átomos compatível à sua velocidade. O cálculo é feito visando que todos os processos terminem ao "mesmo tempo", ou pelo menos com uma diferença mínima.

8 Algoritmo

Mostraremos agora um algoritmo em um nível genérico, do funcionamento das três partes da nossa solução.

Pai

Responsável por :

ver quantas máquinas estão disponíveis

criar um filho em cada uma delas

enviar a cena aos filhos e as partes destinadas ao cálculo no filho

aguardar as respostas dos filhos com os átomos calculados.

enviar msg. de suicídio aos filhos

Início{

Gerar filhos

Dividir imagem

Enviar partes aos filhos

Enquanto imagem não completa{

Esperar resposta dos filhos
Se imagem completa{
Mensagem de suicídio aos filhos
Aloca parte que chegou
Salva/Mostra imagem
}/* se imagem completa*/
Se não{
Aloca parte que chegou
}/* Se não */
}/* Enquanto */
}Fim

Filho

Responsável por :

comunicação com o pai: recepção de cena (parte a calcular), envio do resultado do cálculo, recebimento da mensagem de suicídio

comunicação com os irmãos: pedidos e ofertas de ajuda

comunicação com o neto: átomos a calcular e os calculados

_____

Início{
Espera parte que tem que calcular
Estabelece os pipes
Cria neto
Coloca alguns átomos no pipe para o neto calcular (2min para não deixar filho parado)
Enquanto não recebe msg. de suicídio{
Se neto terminou cálculo de átomo {
Se acabou último átomo{
Envia ao irmão da esquerda a oferta de ajuda
Aloca na posição correta
Envia ao pai a parte que calculou
}/* Se acabou último átomo */
Se não{
Coloca mais um átomo no pipe
Arruma átomo calculado na posição correta
}/*Se não/
}/* Se neto terminou */

Se mensagem do irmão {
Se oferta não é minha{
Calcula média ponderada (entre o que o irmão que faz a oferta já calculou com o que ele mesmo já calculou.)
Se valer a pena {
Envia número de átomos de acordo com resultado da média
Retira os átomos enviados do que tem a calcular
}/*Se valer a pena */
Se não {
Envia mensagem recebida do irmão da direita para irmão da esquerda.
}/* Se não */
}/*Se oferta não é minha */
Se sim /*Mensagem for a que enviei */{
Não sou mais necessário
}/*Se sim */
}/*Se mensagem do irmão */
}/*Enquanto não recebe msg. de suicídio */
Envia msg de suicídio ao neto
Fecha pipes
Suicida-se.
}Fim

Neto

Responsável por :

receber átomo do filho
calcular átomo
retornar ao filho o resultado

_____

Início{
Estabelece pipes com filho
Enquanto não mensagem de suicídio{
Lê do pipe
Se átomo a calcular{
Calcula Ray Tracing do átomo
Envia átomo calculado para filho
}/* Se átomo a calcular */
}/*Enquanto não mensagem de suicídio */
Fecha pipes
Suicida-se

}Fim

9 Projeto

A solução está em fase de implementação final, mas os testes preliminares se mostraram bem animadores, principalmente se tratando de redes heterogêneas (máquinas diferentes). A maior quantidade de cálculo recai, invariavelmente, sobre as máquinas mais rápidas ou menos utilizadas. Para a parte de paralelização está sendo utilizado a biblioteca PVM [Geist (1994)] e o projeto está sendo desenvolvido em linguagem C.

A Figura 5 contém um gráfico com o resultados dos testes preliminares realizados com uma imagem de dez objetos, com átomos de cinqüenta pixeis em uma rede de micros IBM pentium 75 com 16 Mb de memória e rodando Linux 2.0.0.

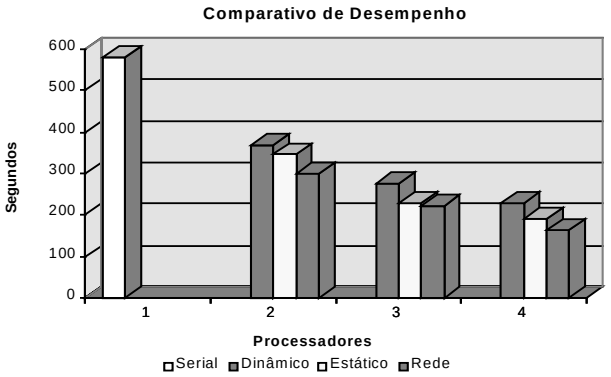

Figura 5 - Gráfico comparativo entre o desempenho do programa serial, a proposta do modelo bidimensional de divisão dinâmica, modelo bidimensional de divisão estática da imagem e o modelo em anel proposto pelo artigo.

Processadores	Serial	Dinâm.	Estát.	Rede
1	580	__	__	__
2	__	369	346	300
3	__	275	230	220
4	__	229	191	164

Figura 6- Tabela de desempenho (em segundos) dos três algoritmos de paralelização bidimensional do Ray Tracing apresentados no artigo.

Deve ser levado em consideração na análise destes resultados que as máquinas onde se realizaram os testes têm praticamente a mesma velocidade. Nosso algoritmo se mostra ainda mais robusto para situações em que as máquinas são de velocidades diferentes. Adicionamos aos testes uma quinta máquina muito mais lenta que as anteriores (observamos os resultados na figura 7). A máquina adicionada foi uma HP 9000/710 rodando HP-UX. (Esta é uma das máquinas principais do laboratório do curso de informática da UFPR. É servidora de NFS, ou seja, é bem sobrecarregada).

Processadores	Serial	Dinâm.	Estát.	Rede
4	__	229	191	164
4 + Lenta	6099	212	1220	157

Figura 7 - Quadro comparativo do impacto de uma máquina mais lenta ajudando nos cálculos.

Podemos verificar que nosso algoritmo conseguiu tirar proveito mesmo da máquina mais lenta, o mesmo não aconteceu com o algoritmo estático, já que este fica atrelado ao tempo da máquina mais lenta.

10 Análises

Podemos notar na Figura 8 que quanto menores os pacotes trocados entre os irmãos (menos átomos por pacote), maior tende a ser o tempo do cálculo. Os átomos tendem a caminhar mais pelo anel antes de serem efetivamente calculados e isto gera uma perda no desempenho geral do algoritmo. Pretendemos futuramente trabalhar a relação ideal de tamanho dos átomos tentando minimizar a comunicação entre os irmãos. Cabe ressaltar aqui que o algoritmo é dinâmico e não determinístico, não se pode prever com precisão o tempo do cálculo da imagem, já que o algoritmo é muito sensível às condições da rede e dos processadores disponíveis.

Tempo (s*10)	Mov. de Pact.	Média Tam
16	10	35
16,3	8	42,5
16,4	12	30,666
16,8	13	27,615
18,3	9	13,333

Figura 8 - Tabela comparativa de tempo de cálculo de uma imagem relacionando o movimento e o tamanho médio dos pacotes. Foi tomado como base para os dados : quatro processadores pentium 75 e átomos com tamanho de cinqüenta pixeis.

Devemos levar em consideração também que a implementação do algoritmo deve sofrer otimizações. Atualmente enfrentamos o problema da disputa pelo processador entre o processo filho e o neto, isto também acarreta num atraso no tempo global do cálculo, mas não está

relacionado com o algoritmo em si, é um problema de implementação.

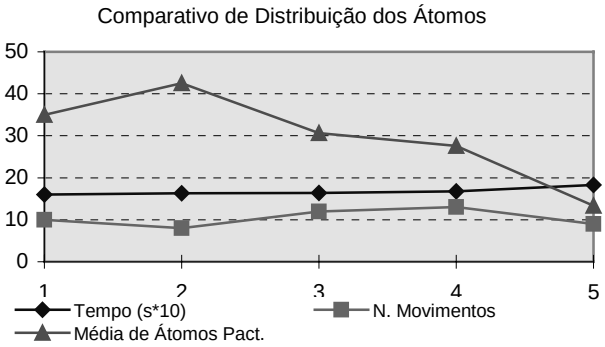

Figura 9 - Podemos notar aqui a tendência do aumento no tempo de cálculo relacionado com a diminuição do número médio de átomos por pacote.

11 Conclusões

Cremos que a solução que encontramos chega próximo ao ideal tratando-se do paralelismo do "Ray Tracing" em redes de computadores. As máquinas praticamente não ficam ociosas, mesmo com máquinas mais lentas na rede. Se algum filho tiver que calcular uma parte mais complexa da imagem, os irmãos terminando seus cálculos, o ajudam.

Ganhamos também com os computadores terminando seus cálculos praticamente juntos, o que é uma coisa desejável na maioria dos casos. A velocidade do cálculo é limitada pelo número de máquinas, pela velocidade da rede em si e pelo custo de administração da imagem, que pode variar de acordo com a movimentação dos átomos pelo anel. Não temos o "gargalo" representado pelo pai na distribuição bidimensional dinâmica implementada normalmente.

Esta técnica pode ainda ser aliada a várias outras técnicas de melhoria do "Ray Tracing", esta também é uma de suas melhores características. Não é uma melhoria que proíbe que outras sejam aplicadas, podendo assim melhorar ainda mais seu desempenho.

12 Referências

J. L. D. Comba, R. C. M. Persiano, "Ray Tracing Otimizado de Sólidos CSG utilizando "Octree". SIBIGRAPI'90 - III Simpósio Brasileiro de Computação Gráfica e Processamento de Imagens. pags. 21-30, maio/1990

T., Whitted "An improved illumination model for shaded display", Comunications of the ACM Vol 23(6) pags 343-349, 1980

J. M. de Martino, "Faster Pixel Mask Generation Exploiting Point-to-Point Coherence of Scanlines" SIBIGRAPI'91 - IV Simpósio Brasileiro de Computação Gráfica e Processamento de Imagens. pags. 67-73, julho/1991

W. T. Corrêa, M. A. de C. Lima, W. M. Junior, M. L. B. De Carvalho, " Aceleração de Ray Tracing via Heterogeneidade" SIBIGRAPI'93 - VI Simpósio Brasileiro de Computação Gráfica e Processamento de Imagens. pags. 1-8, 1993

Fabiano Silva, Leonardo Shiguemi Dinnouti,,"Otimização de um Ray-Tracing com Processamento Distribuído", Trabalho de Graduação II UFPR, Segundo semestre de 1994, não publicado

Didier Baudouel, Bouatouch, Kadi Bouatouch, Thierry Priol, "Distributing Data and Control for Ray Tracing in Parallel", IEEE Computer Graphics and Aplications, July 1994, pags. 69-77

Al Geist, Adam Beguelin, Jack Dongarra, Weicheng Jiang, Robert Manchek, Vaidy Sunderam, "PVM: Paralell Virtual Machine A User's and Tutorial for Networked Paralell Computing" Massachusetts Institute of tecnology, 1994

N.S. Williams, B.F. Buxton, H. Buxton, "Distributed ray tracing using a SIMD processor array"

Stuart A. Green, Derek J. Paddon, "A highly flexible multiprocessor solution for ray tracing" Visual Computer 1990(6), pags 62-73.

T. Priol, K. Bouatouch, "Static load balancing for a parallel ray tracing on a MIMD hypercube" The Visual Computer 1989(5), pags. 109-119.

David J. Plunkett, Michael J. Bailey, "The vectorization of a ray tracing algorithm fo improved execution speed", CG&Application, August 1985, pags. 52-60.