Implementação de um Gerenciador Distribuído de Processadores para um Agregado *Myrinet**

Cristiano B. Sangoi
HP Brasil
Av. Carlos Gomes, 1340/6° andar
Porto Alegre, RS, Brasil
csangoi@terra.com.br

César A. F. De Rose
Programa de Pós-Graduação
em Ciência da Computação
Pontifícia Universidade Católica
do Rio Grande do Sul (PUCRS)
derose@inf.pucrs.br

Resumo

Este artigo descreve três implementações de um modelo de gerência distribuída de processadores para agregados de pequeno e médio porte. Estas implementações são comparadas com um gerenciamento centralizado, sob reais condições de carga. Devido ás alocações distribuídas serem executadas na máquina paralela, em cooperação com os gerentes instalados em cada nó, são analisados resultados de medições de latência e da interferência causada na execução de algumas aplicações paralelas. Os resultados mostram que a alocação distribuída é viável para este tipo de máquina com tecnologias atuais, podendo ser considerada como uma alternativa para ambientes de execução que suportem partições dinâmicas.

1 Introdução

O Processamento de Alto Desempenho é considerado uma ferramenta fundamental para as áreas da ciência e tecnologia. Sua importância estratégica é demonstrada pela quantidade de iniciativas em pesquisa e desenvolvimento nesta área, financiadas por governos de todo o mundo. Diferentes áreas da ciência (biologia molecular, química, física) têm demanda por alto desempenho. O Processamento de Alto Desempenho, contudo, depende fundamentalmente de técnicas do processamento paralelo, capazes de prover o desempenho necessário para estas aplicações.

Desta forma, Sistemas de Processamento paralelo tornaram-se mais populares em função da demanda sempre crescente por poder computacional. Infelizmente, os sistemas que oferecem a capacidade de processamento para satisfazer a demanda, ou tem custo muito elevado, ou são difíceis de programar, ou ambos.

Nexte contexto, tem-se investido, nos últimos anos, em máquinas agregadas (*clusters*), com a finalidade de explorar o processamento de alto desempenho. Essas máquinas possuem grandes vantagens em relação ao custo-benefício, usabilidade, escalabilidade e disponibilidade. Isto permite atacar problemas de elevada complexidade com configurações baratas e que podem ir crescendo à medida das necessidades.

Hoje em dia, a construção de máquinas agregadas tornou-se uma tendência, já existindo algumas com milhares de nós, como por exemplo, a máquina *Earth-Simulator*, com 640 nós e pico total de 40 TFLops, instalada em 2002 no *Earth Simulator Center*, no Japão [11]. Porém, existe um grande problema no que diz respeito à subutilzação da máquina, ou seja, nem sempre as aplicações que estiverem sendo executadas necessitam de todos os processadores da máquina, o que resulta no não aproveitamento de todo o potencial da máquina, ficando esta subutilizada. Uma possível solução para esse problema seria o particionamento eficiente da máquina entre vários usuários.

Este artigo avalia três implementações de uma gerência distribuída de processadores para uma máquina agregada interligada por uma rede rápida *Myrinet*, com o objetivo de verificar a viabilidade da utilização deste tipo de gerência nestas máquinas.

O artigo está organizado da seguinte forma: Na seção 2, será apresentado o problema da subutilização das máquinas agregadas. Na seção 3, será apresentada uma breve introdução sobre a gerência distribuída de processadores. Na seção 4, serão descritos os três modelos implementados. Na seção 5, serão apresentados os resultados referentes aos testes realizados com os algoritmos e, na seção 6, será feita uma conclusão sobre a utilização dos modelos apresentados em máquinas agregadas de alto desempenho.

*Medições feitas na máquina *Amazônia* do CPAD - PUCRS/HP

2 Subutilzação das Máquinas Agregadas

A tendência na construção de máquinas agregadas com muitos processadores é visível nos dias de hoje. Porém, existe um grande problema no que diz respeito à utilização das mesmas, ou seja, como determinadas aplicações se beneficiarão dos recursos disponibilizados pela máquina.

Cada aplicação possui um número ideal de processadores que a faz executar com um determinado desempenho. No entanto, muitas não precisam de todos os processadores da máquina para alcançar tal desempenho. Com isso, um aumento no número de processadores utilizados não leva a um aumento de desempenho da aplicação, ocasionando muitas vezes uma perda no desempenho da mesma, pela presença de complicadores (dependência entre as operações, serialização no acesso aos recursos, zona crítica, sincronização, distribuição dos dados).

Desta forma, se a máquina não for compartilhada por vários usuários, existe a possibilidade desta ficar subutilizada, ou seja, "sobram"processadores que não estão sendo utilizados.

Uma solução para o problema da subutilização seria o particionamento da máquina em máquinas menores, o que permitiria que várias aplicações pudessem ser executadas ao mesmo tempo. O objetivo desse particionamento é manter uma alta utilização da máquina paralela. A atribuição de processadores à aplicações é chamada de gerência de processadores.

É importante ressaltar que a gerência só preocupa-se com a associação de um conjunto de processadores para uma determinada aplicação e não com a forma com que estes processadores serão usados pelas aplicações (*task mapping*).

3 Gerência Distribuída

Na gerência, existem várias formas utilizadas para a alocação de processadores em máquinas agregadas [7]. Quando as partições geradas possuem a mesma topologia da máquina compartilhada, a alocação é dita *structure preserving* (preserva a estrutura), por exemplo, máquinas hipercúbicas são particionadas em hipercubos de menor dimensão. Ao contrário, quando a forma com que a partição será gerada não depende da topologia da máquina compartilhada, a alocação é dita *free-form* (forma livre).

Uma alocação é dita estática, quando as partições são geradas antes da aplicação ser executada, mantendo-se as mesmas até o final da execução desta. Uma alocação é dita dinâmica, quando o processo de alocação de processadores é executado durante a execução da aplicação, podendo, assim, serem feitas alocações locais e liberações parciais em tempo de execução, de acordo com a carga da aplicação. Sendo a alocação feita em nível de Sistema Operacional, libera-se o gerente do sistema dessa tarefa, melhorando consideravelmente a taxa de utilização os processadores compartilhados.

Ainda há outros dois tipos de alocação de processadores, a alocação disjunta, em que não existem duas tarefas mapeadas para o mesmo processador ao mesmo tempo, o que resulta em partições disjuntas, e a alocação sobreposta em que, de acordo com a carga de cada processador, podem existir mais de uma tarefa sendo executada ao mesmo tempo em um mesmo processador.

Costuma-se, hoje, utilizar a Gerência Centralizada [7] nas máquinas agregadas. Na Gerência Centralizada, o cadastro dos recursos alocados é feito com uma estrutura de dados global, que fica normalmente localizada em uma máquina hospedeira da máquina paralela. Desta forma, todas as operações de alocação e liberação têm que passar pelo hospedeiro, centralizando a gerência de recursos e aumentando de forma significativa o tráfego de mensagens entre este e a máquina paralela. Essa característica compromete tanto o desempenho de uma aplicação como a qualidade dos resultados em nível de compartilhamento de recursos. A gerência centralizada apresenta algumas deficiências, como:

- Falta de Escalabilidade: Resulta da utilização de estruturas centralizadas para o controle da ocupação dos processadores. Com o aumento do número de processadores a serem gerenciados, esta estrutura cresce juntamente com o tempo de processamento das operações e o tráfego de mensagens na rede, não forncecendo mais um tempo de resposta aceitável para um procedimento em tempo de execução, comprometendo ainda mais a gerência, agravando, assim, a condição do hospedeiro de gargalo do sistema.
- Fragmentação: Esse problema é agravado pelo fato das estratégias simplificarem a sua gerência utilizando um particionamento que se oriente na topologia da máquina alvo.
- Partições Estáticas: A utilização de estruturas centralizadas na gerência dos processadores do sistema não permite um comportamento dinâmico das aplicações paralelas alocadas, no que diz respeito à variação do número de processadores utilizados por estas durante a sua execução. As aplicações poderiam, desta maneira, reagir de forma mais flexível à variação de carga durante a execução, alocando e liberando processadores de forma dinâmica.

Na gerência distribuída [1], não existe uma estrutura de dados central que contenha as informações do estado de todos os processadores da rede, sendo as operações de gerência feitas diretamente na máquina, de forma distribuída, e não em uma estrutura de dados (figura 1).

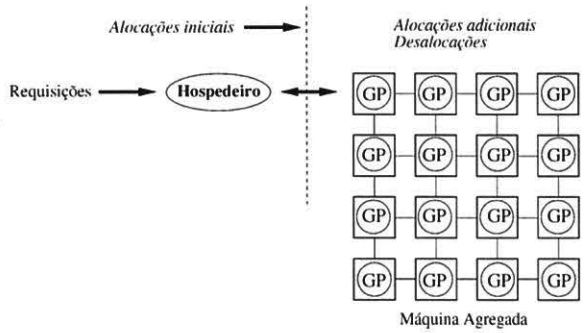

Figura 1. Gerência distribuída de Processadores

Neste modelo, o hospedeiro é responsável apenas pelo recebimento de requisições e pelo disparo da procura na rede, podendo ser visto como um ponto de entrada das requisições. Como todo o processamento ocorre de forma distribuída, não existe limitação no número de pontos de entrada para a máquina, tendo que ser apenas garantido que os resultados das requisições retornem para o ponto de entrada em que foram originadas.

Outra alteração ocorre na origem das requisições. Além das requisições em bloco, no início da execução de uma aplicação, já existentes no modelo centralizado, podem agora ocorrer requisições adicionais durante a execução de uma aplicação, visando uma adaptação do número de processadores a alterações na carga de processamento das aplicações. Isto só tornou-se possível porque alterações mais freqüentes no estado dos processadores, devido a estas alocações extras, não precisam ser repassadas para uma estrutura de dados central.

Nesta forma de gerência, cada nó da máquina agregada possui um núcleo do gerente de processadores (GP), que é responsável pelas alterações da gerência e sua interação. A gerência é feita desta forma através de estruturas que cooperam entre si, de forma não centralizada.

As principais características da gerência distribuída são:

- Uma visão global e atualizada da ocupação dos processadores não existe mais. Cada processador conhece, em princípio, o seu próprio estado. O estado de outros processadores precisa ser consultado;
- Requisições ao gerente de processadores podem originar-se em qualquer um dos nós, sem a necessidade do envolvimento da máquina hospedeira;
- Não existindo mais uma estrutura de dados que tenha que ser atualizada a cada operação de gerência, a operação de desalocação de um processador se resume a uma mensagem ao GP do nó em questão, marcando-o como livre.

A possibilidade de alterar partições pode fazer com que o mapeamento de tarefas também se altere. Em [5], é apresentado um modelo que suporta partições dinâmicas.

4 Implementação

A implementação do gerente distribuído de processadores que está sendo apresentado neste artigo foi desenvolvida para máquinas chaveadas e baseou-se nas idéias descritas por De Rose em [2], que tratam de uma gerência de processadores para máquinas com topologia em malha. A intenção desta implementação é verificar a viabilidade deste tipo de gerência nesta classe de máquinas paralelas, sendo abordados, para isso, aspectos com relação à forma de procura por nós livres na máquina e a comunicação entre estes nós e destes com a máquina hospedeira, além da forma como os nós são reservados para uma determinada aplicação.

Neste artigo são comparadas três possibilidades, utilizando TCP, UDP e TCP/UDP para a comunicação, com a finalidade de verificar a confiabilidade e o desempenho de cada uma, em termos de latência, e da interferência causada pelo gerenciador em determinadas aplicações. A gerência é feita em uma rede secundária(*Fast Ethernet*), e a alocação de processadores é feita de forma disjunta e estática. O agregado *Myrinet* utilizado foi a máquina *Amazônia* do Centro de Pesquisa em Alto Desempenho (CPAD-PUCRS/HP). Este agregado possui a seguinte configuração:

- 16 servidores HP-E60 com dois processadores Pentium III 550 MHz (2-way SMP) cada um com 128MB de memória principal e 9GB de disco (sem monitor);
- 1 servidor HP-E60 com dois processadores Pentium III 550 MHz com um monitor de 17"e duas placas de rede para função de máquina hospedeira;
- Rede primária *Myrinet* [8] e rede secundária *Fast Ethernet* [9];
- Sistema Operacional GNU Linux.

Esta máquina se enquadra na classificação de máquinas NORMA (*No-Remote Memory Access*)[3], em que um nó não pode acessar uma posição da memória de outro nó. A comunicação é feita, então, através de troca de mensagens.

O gerente distribuído de processadores (GDP) foi implementado utilizando **C** como linguagem de programação, com *sockets* [4] para a comunicação na máquina agregada. Um núcleo do gerente (GP) é disparado em todos os nós da máquina e também na máquina hospedeira, ocorrendo então, um mapeamento de um anel lógico. O GP que é disparado na máquina hospedeira é responsável pelo recebimento de requisições e pelos procedimentos de alocação e liberação de processadores na máquina agregada. Quando

uma requisição chega, é disparada uma procura na máquina por nós livres. Estes, quando encontrados, são "reservados"para a aplicação, que está esperando para executar. Os nós obtidos são acumulados em um vetor e a resposta de sucesso ou não retorna para o hospedeiro que, a partir disto, libera o usuário ou não para executar suas aplicações na máquina. Se não forem obtidos todos os processadores requisitados, é enviada uma mensagem para a máquina, informando que os nós que foram alocados parcialmente devem ser liberados. O processo de liberação consiste no simples envio de uma mensagem na máquina, marcando os nós reservados por uma requisição como livres.

Outro aspecto importante a ser salientado é que neste modelo, somente uma procura pode ser disparada na máquina, ficando as outras requisições esperando em uma fila, sendo tratadas na ordem FIFO. Com relação à estratégia utilizada, esta acomoda somente o nó para a aplicação, independente do número de processadores. Também foi estabelecido, para o algoritmo utilizado, que uma aplicação só pode fazer um pedido por um número fixo de nós, não sendo permitida a especificação de um intervalo de nós.

4.1 Implementação com TCP

A utilização de TCP nesta implementação, faz com que a comunicação seja orientada à conexão. Com isso, com o mapeamento do anel lógico após os GP's serem disparados, a procura por processadores livres começa pelo *Nó1* e passa por todo o anel, mesmo se os processadores requisitados já estiverem sido obtidos. A figura 2 ilustra o procedimento descrito acima.

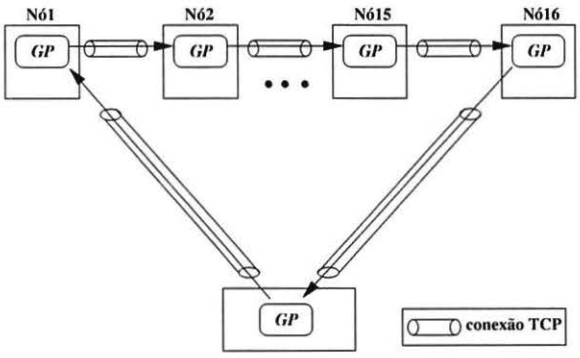

Figura 2. Implementação utilizando conexão TCP

Os cilindros representam a conexão utilizando TCP, e as setas indicam a orientação da pesquisa, começando sempre pelo *Nó1*, e terminando no último. Tanto o pedido de alocação como o de liberação de processadores começam pelo *Nó1* da máquina. Após uma procura ser disparada na máquina, esta só retorna ao hospedeiro após passar pelo último nó.

Tem-se com a utilização desta versão, uma confiabilidade maior em relação à perda de pacotes, porém, é esperada uma perda de desempenho, devido ao fato de uma pesquisa percorrer sempre toda a máquina.

4.2 Implementação com UDP

Com a utilização de UDP na implementação desta versão do GDP, não existe mais a característica de orientação à conexão. Com isso, não é mais obrigatório a pesquisa ter início no primeiro nó e percorrer toda a máquina. Os gerentes, agora, são disparados na máquina agregada e no hospedeiro, e ficam "escutando"em uma porta, por possíveis conexões. Com essa versão, pode-se fazer uma otimização do algoritmo, passando-se a utilizar a forma de alocação chamada *next-fit*[10].

Esta política funciona da seguinte forma: Quando uma pesquisa na máquina agregada tem sucesso, alguns nós foram alocados para determinada aplicação. Neste caso, procura-se evitar que estes nós já alocados sejam novamente pesquisados. Então, quando uma pesquisa retorna com sucesso, é armazenado o nó em que terminou essa pesquisa. Com isso, a próxima começa a partir do ponto em que a anterior terminou. Aqui, toma-se o cuidado de que, se necessário, uma pesquisa percorra toda a máquina por processadores livres, por exemplo, se uma pesquisa inicia no *Nó15*, esta deve terminar, se necessário no *Nó14*. A figura 3 ilustra o procedimento descrito acima. Além disso, como não existe mais a conexão entre os nós, a pesquisa pode terminar sem precisar percorrer toda a máquina, desde que todos os processadores requisitados tenham sido obtidos, podendo a estratégia de gerência utiliar-se de *atalhos*.

Percebe-se, nesta figura, que não existem mais os cilindros que representavam a conexão. As barras representam as portas e as setas os sentidos das pesquisas. Nesta versão, além das mudanças ocorridas em relação aos pedidos de alocação, as liberações são feitas analogamente à verão TCP, porém, com a possibilidade de retornar antes de percorrer toda a máquina.

A utilização desta versão pode não representar a mesma confiabilidade em relação à perda de pacotes, porém, espera-se um melhor desempenho com a utilização dos atalhos e da política *next-fit*.

4.3 Implementação com TCP/UDP

A implementação desta versão utilizando TCP com UDP foi feita com o objetivo de utilizar um algoritmo com garantia de envio e recebimento de pacotes, e com um desempenho considerável. Esta versão funciona da seguinte maneira: Para que fosse possível a utilização de TCP com UDP,

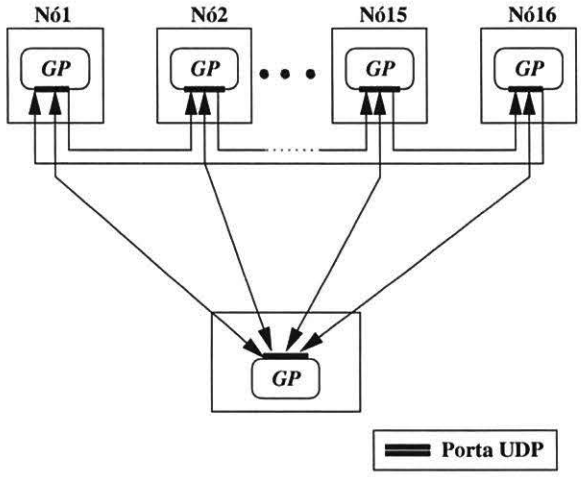

Figura 3. Implementação utilizando conexão UDP - *Next-Fit*

foi necessário o uso de *threads* nos gerentes, porque o tipo de pesquisa variava de acordo com cada requisição. Com a utilização de UDP, tornou-se possível também o retorno para o hospedeiro sem percorrer toda a máquina, além de possibilitar o início da próxima pesquisa a partir do último nó retornado da pesquisa anterior. No entanto, o que torna esta versão diferente das outras duas é o fato de que pesquisas a partir do *Nó1* da máquina se iniciam com comunicação TCP, e pesquisas a partir de qualquer outro nó se iniciam com comunicação UDP. A comunicação entre os nós é feita por TCP.

A figura 4 mostra o funcionamento desta versão. Os cilindros representam as conexões TCP, as barras representam as portas UDP, e as setas representam a orientação das pesquisas.

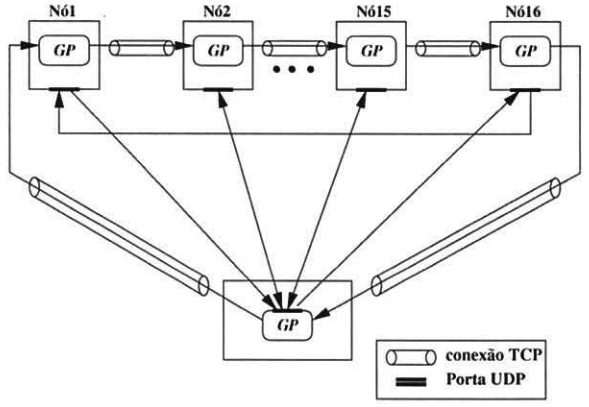

Figura 4. Implementação utilizando conexão TCP/UDP - *Next-Fit*

Com o uso dessa versão, espera-se uma maior confiabilidade na transmissão de pacotes e um melhor desempenho na comunicação entre os nós da máquina agregada e com o hospedeiro.

5 Resultados

Com o objetivo de verificar o desempenho do GDP, foram realizados os seguintes testes:

1. Medição da latência;
2. Medição da interferência.

Para ambos os testes, as requisições por processadores na máquina foram divididas em três categorias: pequenas partições (1 a 4 nós), médias partições (5 a 10 nós) e grandes partições (11 a 16 nós). O tempo de chegada das requisições foi o mesmo, ou seja, todas as requisições chegaram ao mesmo tempo na máquina hospedeira, ficando na fila, e o tempo médio de cada alocação variou entre 1 e 10 segundos. Como descrito na seção 4, somente uma requisição é feita na máquina de cada vez, não havendo suporte para múltiplas requisições.

Para o estabelecimento destes valores, foi utilizado um programa simulador de requisições, no qual se especificava a quantidade de processadores e a variação no tempo de alocação, sendo o programa responsável pela geração de um arquivo de requisições (foram usados arquivos com 200 requisições) e pela simulação da chegada das requisições.

Com o objetivo de comparação de desempenho, foi utilizada também uma versão centralizada do Gerente de Processadores. Nesta versão, não existe um gerente em cada nó da máquina, sendo todas as operações de pesquisa, alocação e liberação realizadas em um único gerente, executado na máquina hospedeira.

5.1 Medição da latência do gerenciador

As medições de latência foram realizadas com o objetivo de verificar quanto tempo uma alocação demora para ser feita, ou seja, qual a duração de uma tentativa de alocação na máquina agregada, e quanto tempo demora uma liberação na mesma.

A seguir, são apresentados os gráficos correspondentes à latência, demonstrando os tempos de alocação (figura 5) e liberação (figura 6), das versões implementadas. Esses gráficos foram feitos utilizando função logarítmica no eixo y, para que os dados fossem melhor apresentados.

Observando o gráfico de alocações representado pela figura 5, percebe-se que o tempo de alocação com a versão centralizada, para pequenas, médias e grandes partições foi inferior ao tempo das outras versões. Isto deve-se ao fato

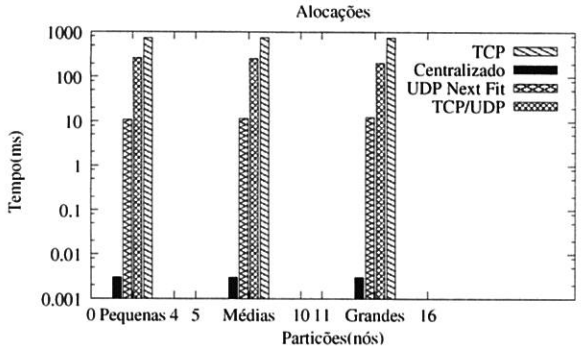

Figura 5. Gráfico Alocação de Processadores

das operações se realizarem na mesma máquina, não envolvendo tráfico da rede e tempo de comunicação entre os nós, como no caso das versões distribuídas.

Já na comparação entre as versões distribuídas, observou-se que o modelo que utiliza UDP teve um desempenho melhor em relação aos outros, por utilizar-se ddos atalhos e da política *next-fit*, permitindo qued as pesquisas retornem para o hospedeiro antes passar por todo o anel lógico. Já o pior desempenho foi o da versão TCP, devido à característica da mesma, de ser orientada à conexão, tendo as pesquisas que percorrer sempre toda a máquina. Percebe-se, com isso, que a característica de orientação à conexão (versão TCP), vem a comprometer o desempenho de uma alocação de processadores, não importando o tamanho da partição requisitada.

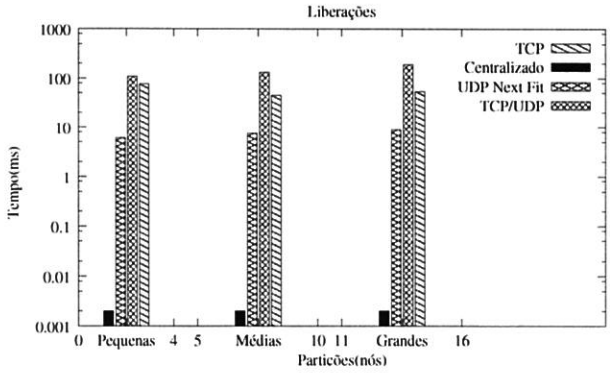

Figura 6. Gráfico Liberação de Processadores

Observando o gráfico de liberações (figura 6), verifica-se novamente que o menor tempo foi o da versão centralizada, para pequenas, médias e grandes partições. Entre as versões distribuídas, a versão com UDP, assim como na alocação, obteve desempenho melhor, pelas mesmas razões explicadas na análise do gráfico anterior. Porém, percebeu-se que a versão que utiliza TCP com UDP obteve o pior desempenho entre as três versões distribuídas do GDP. Este resultado explica-se pela complexidade um pouco maior dos cálculos executados nos GP's com esta versão, pelo uso das *threads*, e pelo processo de liberação não utilizar os atalhos e a política *next-fit* abordados no processo de alocação.

A partir dos resultados demonstrados anteriormente, pode-se concluir que a latência na gerência centralizada é muito baixa em relação à gerência distribuída. Esses tempos justificam-se pelo fato desta gerência não envolver a máquina agregada nos procedimentos de alocação e liberação de processadores, não tendo, assim, o tempo para o envio e recebimento de mensagens entre os nós da máquina. Entre as versões distribuídas, destaca-se o melhor desempenho da versão com UDP, pelo aproveitamento das características descritas anteriormente, utililzadas na implementação desta versão.

5.2 Medição da interferência do gerenciador

Como as alocações ocorrem na máquina agregada, em cooperação com os GP's instalados em cada nó, esses, quando são "acordados"durante uma pesquisa, acabam interferindo no desempenho das aplicações paralelas que estejam sendo executadas, pois competem pelo uso da CPU de cada nó.

Além disso, ocorre um tráfego maior de mensagens na rede, chamado roubo de banda passante. Esse tráfego torna-se maior, se existir apenas uma rede na máquina agregada.

Neste ambiente utilizado, a interferência não seria considerada significativa, já que a gerência é feita na rede secundária, enquanto que as aplicações executam na rede primária, além de os GP's não possuirem uma complexidade muito grande nos cálculos. Então, para que fosse causada a maior interferência possível do GDP sobre as aplicações, essts foram executadas na mesma rede (*Fast Ethernet*) em que o GDP está instalado.

Para medir a interferência, foram utilizados os programas PovRay (*Persistence of RayTracer*), Fractal de Mandelbrot e *Integer Sort (IS)*. O teste foi feito de maneira que o gerenciador estivesse recebendo constantemente requisições e disparando várias procuras na máquina, em um intervalo de tempo pequeno, no mesmo tempo de execução dos programas. Foram utilizados os três tipos de partições, e as três versões do algoritmo descritas neste artigo. Para a medição da interferência do gerenciador sobre os programas selecionados, primeiramente estes foram executados sem o GDP e após, com o GDP executando, para que os tempos pudessem ser comparados. Foram feitas 25 medições, para o PovRay e o Fractal, a fim de que se obtenha uma média mais exata do tempo de execução destes programas, e 10 medições para o IS. Foi utilizado, também, os mesmos arquivos de requisições e os mesmos critérios adotados na medição da

latência do GDP.

O programa PovRay consiste em um instrumento de renderização tri-dimensional. Este programa pega as informações de um arquivo texto externo e simula iterações com os objetos em um cenário, obtendo, então, uma imagem tri-dimensional. O tempo de execução é o tempo que o programa leva para renderizar essa imagem. Funciona com padrão de comunicação **mestre/escravo** [12], sendo que cada escravo recebe uma fatia da imagem a ser renderizada e devolve para o mestre o resultado dos cálculos, e este apresenta a imagem renderizada na tela. A comunicação entre os nós é feita por troca de mensagens e a ferramenta de programação utilizada é o MPI [6].

A tabela 1 apresenta os dados referentes à interferência sofrida pelo PovRay com o GDP, em termos de percentagens.

Tabela 1. Interferência utilizando PovRay

Versão/Partições	Pequenas	Médias	Grandes
TCP	1.31%	1.23%	2.74%
UDP	1,02%	1,43%	2.01%
TCP/UDP	1.16%	1.29%	2.15%

Analisando a tabela 1, pode-se notar que houve uma interferência de 2% em média no tempo de execução do PovRay, com o uso do gerenciador, tanto com requisições por partições pequenas, como médias e grandes, nas 3 versões distribuídas. Houve uma interferência relativamente maior nas pesquisas por partições grandes, tanto com TCP, UDP e TCP/UDP, devido à quantidade maior de alocações sem sucesso na máquina agregada, justificada pelo tamanho das partições requisitadas, comparado ao número de nós disponibilizados pela máquina agregada, gerando, assim, um tráfego maior de mensagens na rede.

O programa Fractal de Mandelbrot ilustra o conjunto de Mandelbrot. A definição se um ponto na tela pertence ou não ao conjunto é feita por cálculos sucessivos utilizando a fórmula de Mandelbrot. Se, após um número máximo de iterações, o resultado não ultrapassar um valor definido na fórmula, o ponto pertence ao conjunto, recebendo a cor preta. Caso contrário, recebe uma cor em função do número de iterações que o ponto realizou para ultrapassar o valor definido.

O padrão de comunicação é **mestre/escravo** [12], em que o mestre é responsável por dividir a tela e mandar para os escravos uma determinada fatia, e os escravos são responsáveis pelos cálculos das fatias e do retorno desta ao mestre, com a cor correspondente. A comunicação entre os nós é feita por troca de mensagens e a ferramenta de programação utilizada é o MPI [6].

A tabela 2 apresenta os dados referentes à interferência sofrida pelo Fractal com o GDP, em termos de percentagens.

Tabela 2. Interferência utilizando Fractal

Versão/Partições	Pequenas	Médias	Grandes
TCP	0.18%	0.25%	0.14%
UDP	0.15%	0.03%	0.03%
TCP/UDP	0.19%	0.17%	0.08%

Analisando a tabela 2 obervou-se que o Fractal, sofreu uma interferência menor, de 0.15% em média, do gerenciador em relação ao PovRay. Pode-se dizer então, que o desempenho desta aplicação não foi prejudicado pelo GDP, mesmo com um tráfego de mensagens muito grande pela rede.

O programa *Integer Sort (IS)*, parte integrante de um pacote de benchmarks, faz um teste de ordenação de chaves entre os nós da máquina agregada, baseado em classes de testes que variam quanto ao número de chaves. Possui uma função de geração de números randômicos, que usa um processo Gaussiano para gerar as chaves nas máquinas.

Possui padrão de comunicação **mestre/escravo** [12], na qual os escravos fazem a soma das chaves e retornam o somatório para o mestre, e *broadcast*, onde mensagens são enviadas de todos para todos (*all to all*), e mensagens específicas são enviadas de todos para todos (*all to all-v*). Por utilizar *broadcast* na comunicação e pela complexidade dos cálculos realizados, este programa pode sofrer interferência maior do GDP.

A ordenação das chaves é feita usando comunicação por troca de mensagens e segmentação do vetor de chaves, ordenando todos ao mesmo tempo em todos os nós. A ferramenta de programação utilizada é o Mpi [6].

A tabela 3 apresenta os dados referentes à interferência sofrida pelo IS com o GDP, em termos de percentagens.

Tabela 3. Interferência utilizando IS

Versão/Partições	Pequenas	Médias	Grandes
TCP	5.33%	5.57%	5.46%
UDP	0.83%	1.54%	1.27%
TCP/UDP	2.23%	4.29%	4.66%

Analisando a tabela 3, percebe-se que a interferência sofrida pelo IS foi, em média, maior que a sofrida pelo Fractal e pelo PovRay. Essa interferência foi causada pois essa aplicação possui uma complexidade maior nos cálculos, o que resulta em um processamento relativamente maior, e um tráfego elevado de mensagens, com o uso de *broadcast*.

A versão do GDP que menos atrapalhou este programa foi a que utiliza UDP para a comunicação, com interferência em torno de 1%. As outras versões, que utilizam TCP e TCP/UDP, ocasionaram uma interferência parecida no de-

sempenho do IS, em torno de 5%, devido às características destas versões descritas neste artigo.

Esta interferência pequena sobre os programas utilizados (embora as aplicações paralelas tenham sido executadas na mesma rede do GDP - rede secundária) pode ser justificada pela simplicidade dos cálculos realizados pelos GP's, que possuem poucas linhas de código, não ocupando assim, um tempo significativo de CPU, e pelo tamanho médio das mensagens transmitidas entre a máquina hospedeira e a máquina agregada, em torno de 90 *bytes*. Percebe-se ainda, que a interferência pode ser maior ou menor, dependendo da aplicação que está sendo executada.

6 Conclusão

Este artigo apresentou a implementação de um gerente distribuído de processadores para agregados *myrinet*.

Após ser feita uma breve introdução sobre o problema da subutilização das máquinas agregadas e sobre a gerência distribuída de processadores, foram apresentadas três versões de uma implemmentação do gerente, utilizando TCP, UDP e TCP/UDP, respectivamente. Em seguida, foram demonstrados os testes de latência e interferência realizados com as três versões, na máquina *Amazônia*, do CPAD (PUCRS/HP).

Verificou-se, a partir dos testes, que o melhor desempenho, em termos de latência, foi da versão centralizada, pelas próprias características da mesma descritas na seção 3. Porém, esta versão foi utilizada apenas para comparação, já que a máquina utilizada possui um número pequeno de nós, o que resultou em uma estrutura de dados pequena na máquina hospedeira.

Entre as versões distribuídas, foco principal deste artigo, o melhor desempenho foi o da versão utilizando comunicação por UDP, por não ser orientada à conexão e utilizar os conceitos da política *next-fit*, além de poder se aproveitar dos atalhos na comunicação entre os nós e o hospedeiro.

Nas medições de interferência, verificou-se que, dependendo da aplicação paralela que está sendo executada, esta será mais atrapalhada pelo gerenciador, como observou-se na utilização da aplicação IS, esta que gera um número maior de mensagens na rede.

Com um tempo de alocação em torno de um décimo de segundo e uma interferência não muito significativa, o gerente distribuído de processadores mostrou-se viável neste tipo de máquina agregada.

É, sem dúvida uma alternativa a ser considerada, se o ambiente de execução de aplicações paralelas e as próprias aplicações puderem se aproveitar de partições dinâmicas, possibilitando às aplicações utilizarem somente os nós que necessitam para sua execução, permitindo, assim, que todo o potencial disponiblizado pela máquina agregada seja aproveitado.

Referências

[1] C. A. F. DeRose and H. Heiss. Dynamic processor allocation in large mesh-connected multicomputers. In *Euro-Par 2001, 2001, Manchester, Inglaterra. Published in Lecture Notes in Computer Science (LNCS)*, pages 783–792, 2001.

[2] C. A. F. DeRose, P. O. A. Navaux, and C. R. Geyer. Distributed processor allocation in mesh-connected multicomputers. In *Ninth IEEE International Symposium On High-Performance Distributed Computing (HPDC'2000)*, aug 2000.

[3] C. A. F. DeRose and Marcelo Pasin. Fundamentos de processamento de alto desempenho. In *II Escola Regional de Alto Desempenho (ERAD'2002)*, 2002.

[4] M. J. Donahoo and K. L. Calvert. *The Pocket Guide to TCP/IP Sockets*. Morgan Kaufmann Publishers, Usa, 2001.

[5] J. C. Jacob and S. Lee. Task spreading and shrinking on multiprocessor systems and networks of workstations. *IEEE Transactions on Parallel and Distributed Systems*, pages 1082–1101, 1999.

[6] B. Mahafzah and W. Cohen. A message passing interface (mpi) tutorial. www, jan 2000.

[7] C. B. Sangoi. Um estudo sobre a gerência de processadores em máquinas paralelas. Trabalho Individual I 2000/9, Pontifícia Universidade Católica do Rio Grande do Sul, Porto Alegre, Porto Alegre, RS, Brasil, 2000.

[8] C. L. Seitz. Myrinet - a gigabit-per-second local-area-network. *IEEE Micro*, 15(1), feb 1995.

[9] A. S. Tanenbaum. *Redes de Computadores*. Editora Campus, Brasil, 1997.

[10] A. S. Tanenbaum and A. S. Woodhull. *Operating Syutems: Design and Implementation*. Prentice Hall, Usa, 1997.

[11] TOP500. Top500 supercomputer sites. http://www.top500.org, ago 2002.

[12] B. Wilkinson and M. Allen. *Parallel Programing: Techniques and Applications Using Networked Workstations and Parallel Computers*. Prentice Hall, Usa, 1999.