Comportamento de Aplicações Paralelas em Aglomerados de Computadores Heterogêneos*

Diego Luis Kreutz[1], Lucas Mello Schnorr[2], Cleverton Marlon Possani[1], Benhur Stein[1]
[1] *Laboratório de Sistemas de Computação — LSC*
Universidade Federal de Santa Maria — UFSM
{kreutz, possani, benhur}@inf.ufsm.br
[2] *Programa de Pós-Graduação em Computação — PPGC*
Universidade Federal do Rio Grande do Sul — UFRGS
lmschnorr@inf.ufrgs.br

Resumo

A computação em aglomerados heterogêneos de computadores está cada vez mais presente na área de computação de alto desempenho. Neste contexto, o objetivo desse trabalho é apresentar e analisar alguns dados de desempenho de aplicações paralelas em aglomerados desse gênero. Além disso, é apresentado um calculador de capacidade de computação para nós de processamento de máquinas virtuais LAM/MPI. Com ele o usuário pode facilmente estabelecer um balanceamento de cargas para a sua aplicação, considerando as principais características da mesma.

1. Introdução

No contexto do processamento de alto desempenho, uma das linhas de pesquisa e desenvolvimento que vem ganhando importância é a computação em aglomerados de computadores heterogêneos. Isso se deve a vários fatores. Entre eles, a dificuldade de manter ou aumentar um aglomerado homogêneo, principalmente por causa da evolução tecnológica. Outro fator está associado ao custo. Um aglomerado heterogêneo pode ter um custo financeiro de aquisição/construção menor. Isso porque ele pode ser implementado com conjuntos de equipamentos já existentes ou através da compra gradual de equipamentos variados, de baixo custo. Essa aquisição gradual de novos equipamentos reduz a quantidade de investimento imediato.

A capacidade de memória, poder de processamento e a latência e vazão da rede são geralmente as características mais importantes das máquinas de um aglomerado heterogêneo. Essas características devem ser levadas em conta na hora de rodar aplicações paralelas nesse tipo de máquinas paralelas, pois elas poderão influenciar significativamente o desempenho dessas aplicações.

Este trabalho apresenta dados estatísticos que mostram a relação de desempenho entre processadores e adaptadores de rede. Na obtenção dos dados foram utilizados diferentes tipos de processadores, comutadores e adaptadores de rede. A análise desses dados possibilita a identificação de situações em que um balanceamento de carga será capaz de explorar melhor os recursos computacionais disponíveis.

Inicialmente foi desenvolvida uma aplicação para extrair medidas de desempenho em um ambiente heterogêneo. Os resultados obtidos mostraram que pode ser útil considerar as características de um aglomerado heterogêneo na execução de uma aplicação paralela. Optou-se então pelo desenvolvimento de um avaliador de capacidades de computação para a execução de programas paralelos reais. Este calculador fornece informações para a aplicação paralela realizar um balanceamento de carga no início da sua execução.

As próximas seções apresentam o ambiente heterogêneo utilizado e a aplicação de medição de desempenho. Na seqüência é apresentado o calculador de capacidade de carga desenvolvido. Dando continuidade, são mostradas as aplicações utilizadas e os resultados de execução. Por fim, aparecem a conclusão e os trabalhos futuros.

2. O Ambiente

Para a execução das aplicações foi utilizado um aglomerado de computadores heterogêneo construído no LSC[1]. Ele é composto por máquinas de diferentes capacidades de processamento e memória. Além disso, o aglomerado possui diversos adaptadores de rede Gigabit e Fast Ethernet, interligados por diferentes comutadores.

Na tabela 1 são mostradas as configurações das máquinas que fazem parte do aglomerado heterogêneo utilizado nos testes. Nesta tabela, cada tipo de máquina possui o seu

* Fomento: CNPq, processo 380049/03-1

1 Laboratório de Sistemas de Computação - UFSM

identificador, características de processamento e a memória total. Esse identificador será utilizado ao longo do texto para referenciar o tipo da máquina.

Identificador	Processador	Memória
A1700	Athlon XP 1700+ 1.5GHz	256MB
PIIID	Dual Pentium III 1GHz	1000MB
AMP2400	Dual Athlon MP 2400 2GHz	1000MB
PIII	Pentium III 800MHz	256MB
PIIIC	Pentium Celeron 1GHz	128MB
P4	Pentium 4 2.4GHz	512MB

Tabela 1. Máquinas do aglomerado heterogêneo

A tabela 2 mostra a identificação, que será utilizada ao longo do texto, e a descrição dos dispositivos de rede, que compreendem os comutadores e as placas de rede. As primeiras duas linhas da tabela apresentam os comutadores utilizados na análise. As demais linhas apresentam os adaptadores de rede empregados.

Ident.	Descrição
3C17700	Switch Gigabit 3COM 4900 3C17700
GS224S	Switch Fast Ethernet Hawking GS224S
Netgear	Netgear GA302T Gigabit Ethernet
DLink	DLink DGE-550T Gigabit Ethernet
3Com	3COM Gigabit Ethernet 3C996T
BCM5700	Broadcom Gigabit Ethernet BCM5700
BCM5782	Broadcom Gigabit Ethernet BCM5782
Hawking	Hawking Gigabit Ethernet GA132T
3c905c	3COM Fast Ethernet 3c905C-TX
I82557	Intel Fast Ethernet 82557/8/9
I82801BA	Intel Fast Ethernet 82801BA
Sis650	Silicon Fast Ethernet SiS650

Tabela 2. Comutadores e adaptadores de rede

A execução da análise de desempenho utilizou o aglomerado heterogêneo, composto por 14 nós. Cada nó possui o sistema operacional Linux, utilizando a distribuição Gentoo. A versão do núcleo do sistema operacional é 2.6.2 e do compilador gcc é 3.3.2. As aplicações que são mostradas neste artigo utilizam as bibliotecas de comunicação LAM/MPI (lam-7.0.4) e PVM (pvm-4.3.3).

3. A aplicação de teste

Foi criada e executada uma aplicação de *ping-pong* com o objetivo de realizar uma avaliação experimental da interferência de diferentes adaptadores de rede, comutadores e processadores no desempenho de aplicações paralelas. Esta aplicação foi implementada utilizando a biblioteca de comunicação MPI[2].

A figura 1 ilustra o funcionamento da aplicação. São criados um ou mais processos em cada nó. Os processos do nó mestre iniciam o envio de mensagens. Seus processos correspondentes, nos nós escravos, recebem as mensagens e as enviam de volta. Esse processo se caracteriza como um *ping-pong*.

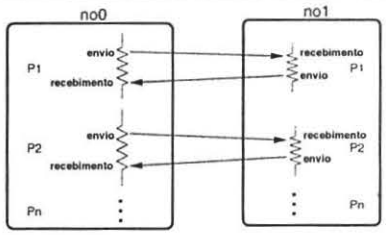

Figura 1. Ilustração da aplicação de *ping-pong*

Para executar, a aplicação recebe o tamanho da primeira e da última mensagens e o número de bytes de incremento. A partir daí ela inicia os *ping-pongs* variando o tamanho dos pacotes, através do valor de incremento, até atingir o número de bytes do último pacote.

O objetivo da execução de *ping-pongs* simultâneos foi avaliar o comportamento dos dispositivos de rede e processadores, considerando o aumento da quantidade de dados trafegando entre eles e a concorrência pelo processador.

Resultados e estatísticas

Nesta seção são apresentados alguns dados, aspectos e características comuns em aglomerados heterogêneos. É interessante levar em conta essas informações na hora de executar aplicações nesse tipo de aglomerado.

Foi realizado um grande número de execuções utilizando diferentes máquinas e diferentes dispositivos de comunicação. Os gráficos desta seção apresentam a média de 10.000 execuções para cada tamanho de pacote.

O gráfico da figura 2 mostra o tempo de transferência de pacotes de 100 bytes entre duas máquinas, executando 1, 2, 3 e 4 cópias da aplicação simultaneamente. Em todas as execuções foram utilizadas os mesmos adaptadores (**3Com**) e comutador (**3C17700**) de rede, com pares de máquinas diferentes. Esse comportamento é semelhante com pacotes de até 1500 bytes. Com pacotes maiores, a diferença entre as máquinas é menor.

A diferença entre 1 e 2 *ping-pongs* é relativamente pequena. Já com 3 e 4 *ping-pongs* simultâneos há uma distanciação maior segundo o gráfico da figura 2, ou seja, uma defasagem mais significativa no desempenho como um todo.

Pode ainda ser observado a influência da *cache* e dos processadores. O pior desempenho é apresentado pela comunicação entre dois ***PIIIC***. Neste caso, há uma grande distanciação entre a execução de um ping-pong e quatro *ping-*

2 Message Passing Interface

pongs. Essa máquina é a que possui a menor memória *cache*.

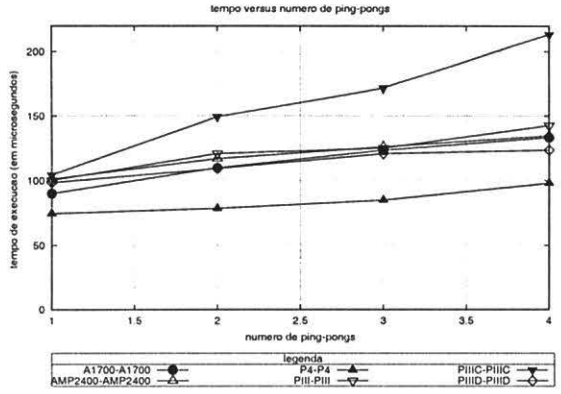

Figura 2. Número de *ping-pongs* versus tempo de execução com mensagens de 100 bytes, usando Gigabit Ethernet.

A melhor estabilidade de três para quatro *ping-pongs* é atingida na comunicação ***PIIID-PIIID***. Essas são máquinas com dois processadores e um bom sistema de memória, se comparadas aos demais computadores utilizados.

O gráfico da figura 3 apresenta duas comunicações entre máquinas de diferentes tipos, utilizando 1, 3 e 4 cópias simultâneas da aplicação. As execuções foram realizadas sempre entre as mesmas máquinas. A única mudança efetuada foi a localização dos processos mestres. Ora foram executados em uma máquina, hora em outra. Essa simples alternância causou o comportamento apresentado no gráfico.

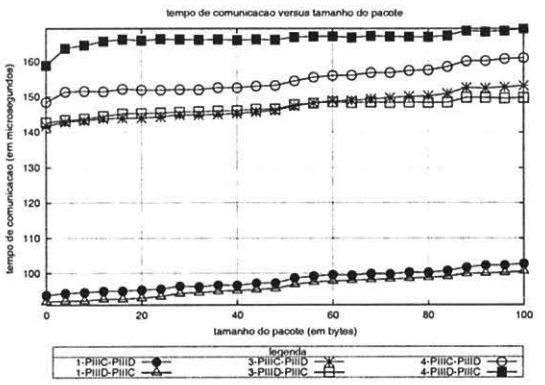

Figura 3. Execução de 1, 3 e 4 *ping-pongs*

Nas comunicações *1-PIIIC-PIIID* e *1-PIIID-PIIIC* o melhor desempenho foi quando os processos mestres foram lançados na máquina **PIIID**. A execução de três *ping-pongs* simultâneos aumenta a carga da rede e das CPUs. Isso pode ser visto nas comunicações *3-PIIIC-PIIID* e *3-PIIID-PIIIC*. Neste caso, a partir de aproximadamente 50 *bytes*a computação com os processos mestres na máquina ***PIIIC*** começa a apresentar o melhor desempenho. Entre as comunicações *4-PIIIC-PIIID* e *4-PIIID-PIIIC*, executando quatro cópias simultâneas da aplicação, ocorre uma inversão de desempenho, em relação a execução de apenas uma instância do programa. A execução ***PIIIC-PIIID***, com os processos mestres na máquina ***PIIIC***, apresenta o melhor desempenho.

A alternância de desempenho acontece, em parte, porque o tempo de resposta da máquina mais rápida (***PIIID***) é menor devido a sua maior capacidade de processamento e sistema de memória *cache*. Quando essa máquina está controlando os processos mestres dos *ping-pongs*, eles dependem da resposta dos processos escravos, que estão na outra máquina (***PIIIC***, menos potente). Os processos mestres podem ficar esperando dados durante tempos prolongados enquanto os processos escravos são escalonados e executados.

Apesar dessa diferença de desempenho (gráfico da figura 3) parecer insignificante, ela pode crescer com a disparidade de desempenho entre as máquinas de processamento e com o número de processos disputando as CPUs. Essa diferença será mais significativa quanto mais heterogêneas forem as máquina ou quanto maior o número de processos executando. Dependendo das características da aplicação que se deseja executar, é interessante analisar esses aspectos a fim de explorar ao máximo os recursos computacionais disponíveis.

4. Cálculo de capacidade de carga

Os resultados obtidos na seção 3 mostram que pode ser útil levar em consideração as características das máquinas de um aglomerado heterogêneo. Pensando em desenvolver e testar aplicações paralelas reais que façam uso dessas características, decidiu-se construir um avaliador de capacidade de carga. O objetivo desse avaliador é o de obter a capacidade de carga de cada máquina considerando determinadas características. Essas capacidades podem então ser usadas pelas aplicações paralelas, permitindo uma melhor distribuição de carga.

O avaliador tem como base a utilização de fatores como CPU, memória, latência e vazão da rede, além do peso de cada um desses fatores. O programa inicialmente obtém o valor de cada fator de todas as máquinas do aglomerado heterogêneo. Com esses valores e o peso de cada fator fornecido pelo usuário, o avaliador obtém o valor de cada característica para cada máquina do aglomerado. Este valor indica a capacidade de carga de cada máquina em relação a capacidade de carga das outras máquinas do aglomerado, de acordo com os parâmetros do usuário. A equação 1 mostra o cálculo dessa capacidade de carga para cada máquina i dentre os n fatores X.

$$capacidade_i = \sum_{X=0}^{n} \left(\frac{fatorX_i}{\sum_{j=0}^{n} fatorX_j} * peso_{fatorX} \right) \quad (1)$$

É feita uma normalização dos valores de cada fator obtido de cada máquina. Essa normalização consiste inicialmente em tomar como referência o valor máximo de cada característica (CPU, memória, rede). Esse valor máximo será igualado a 100 (representação em escala de porcentagem - isso é válido para todas as características) e os demais valores de determinada característica serão calculados a partir desse valor de referência, ou seja, serão menores ou igual a 100.

Os valores de latência e vazão obtidos pelo avaliador são em relação a máquina onde foi lançado o mestre desse programa. Não é levado em consideração no cálculo a latência e a vazão entre as outras máquinas.

Quando se considera latência no cálculo da capacidade de carga, é necessária a inversão do seu significado. Isso ocorre porque quanto menor a latência com determinado nó, melhor é a capacidade deste nó.

Um arquivo com informações é gerado após o cálculo da capacidade de cada máquina, de acordo com os parâmetros do programa. Este arquivo contém a identificação dos nós LAM/MPI e o respectivo valor da capacidade de carga. Ele pode ser utilizado por aplicações paralelas para enviar tarefas mais pesadas às máquinas de maior poder de computação, fazendo assim um balanceamento de carga de acordo com a capacidade de processamento, memória e comunicação das máquinas do aglomerado heterogêneo. Em aplicações do tipo mestre-escravo as capacidades de computação podem também ser utilizadas para definir a granularidade das tarefas. O tamanho das tarefas pode ser definido a partir da capacidade relativa de computação de cada elemento de processamento.

O poder relativo de computação de cada máquina deve ser calculado levando em consideração as características da aplicação. Isso quer dizer que se a aplicação paralela tiver muita comunicação, por exemplo, deve-se calcular a capacidade de computação com pesos maiores de latência e/ou vazão. Caso a aplicação tenha muitos dados para serem transferidos e uma grande demanda por processamento, o cálculo do poder de computação deve ser feito combinando a vazão, memória e processador, cada um com os pesos relativos às características da aplicação paralela.

Abaixo segue um exemplo de uso do balanceador. Supondo que o objetivo seja executar uma aplicação paralela que demande aproximadamente 70% de processamento, 20% de comunicação (mensagens pequenas) e 10% de memória. Neste caso, deve-se indicar ao avaliador os fatores utilizados no cálculo e seus respectivos pesos. O avaliador irá então obter os valores referentes as capacidades relativas de computação de cada nó. A seguir é mostrado o comando de execução do avaliador para o exemplo de uso acima.

```
avaliador -cpu uma -mem livre -rede vazao -rede
latencia -potenciacpu 70 -potenciaredelatencia 15
-potenciaredevazao 5 -potenciamem 10
```

No comando acima, o parâmetro `-cpu` indica se será considerada apenas uma ou todas as CPUs das máquinas. A opção `-mem` define a memória livre ou total do sistema utilizada no cálculo. O parâmetro `-rede` indica a utilização de vazão, latência ou os dois valores no cálculo. Os parâmetros `-potencia` definem os pesos de cada fator. Esses pesos devem ser definidos pelo desenvolvedor da aplicação, de acordo com as principais características e necessidades do programa paralelo. Na execução do avaliador será lançado um processo por nó da LAM/MPI. Cada processo coletará as informações e medidas referentes ao nó e as repassará ao processo mestre. Este irá computar os dados e apresentar valores como os do exemplo abaixo.

```
0 40
1 20
2 15
3 15
```

Esses valores representam as capacidades relativas de computação de cada um dos quatro nós da suposta máquina paralela em uso. A partir desses valores o desenvolvedor/usuário poderá realizar o balanceamento de cargas em sua aplicação.

5. Testes e Resultados

As próximas seções apresentam três aplicações reais e os resultados de desempenho utilizando o calculador de capacidade de computação. A primeira aplicação é da área de processamento de imagens e se refere a paralelização do filtro de mediana. Em seguida é apresentada um algoritmo para resolução de sistemas de equações lineares de grande porte. A terceira é uma implementação da meta-heurística GRASP para o problema do caixeiro viajante com demandas heterogêneas. A seção final apresenta alguns comentários e conclusões.

5.1. Filtro de Mediana Paralelo

Na área de processamento de imagens, existem filtros que melhoram a qualidade de uma imagem e/ou evidenciam suas características. O filtro de mediana é um exemplo. Ele é responsável por retirar os ruídos do tipo "sal e pimenta" de uma imagem, através da obtenção do valor mediano de cada pixel da imagem e dos pixels vizinhos [11]. Neste caso, uma máscara define a quantidade de pixels vizinhos que será levada em conta para o cálculo que obtém o valor mediano. Esta máscara é uma matriz quadrada de um determinado tamanho, sendo que este tamanho está diretamente relacionado com a quantidade de vizinhos que serão utilizados no cálculo.

O tempo de execução da aplicação do filtro de mediana aumenta conforme a máscara aumenta. Em alguns casos, quando a máscara é muito grande, o tempo de execução passa a ser proibitivo. Nessas situações, a paralelização

da aplicação pode ser uma solução viável, pois a obtenção do valor mediano entre um pixel e seus vizinhos é uma operação independente do cálculo da mediana dos outros pixels.

Neste contexto, foi construída uma aplicação paralela que aplica o filtro de mediana em uma imagem. Esta aplicação divide a imagem em frações (conjunto de linhas da imagem) que são então enviadas para cada processo da aplicação paralela. O tamanho da fração enviada é calculada de acordo com a capacidade de carga da máquina onde o processo que cuida desta fração será executado.

A implementação desta aplicação utilizou MPI e foi construída com a metodologia mestre/escravo. O mestre, no início da execução, carrega de um arquivo a imagem de entrada. Após isso, a imagem de entrada é dividida em frações de acordo com a máquina onde será executado o processo que aplicará o filtro de mediana nesta fração. Cada escravo, ao receber esta fração, aplica o filtro e envia o mesmo tamanho de fração como resultado. O mestre, depois de enviar trabalho para cada escravo, processa a sua fração e aguarda os resultados de seus escravos.

Resultados A aplicação do filtro de mediana foi executada em diferentes situações com uma imagem de entrada de tamanho 500x750 aplicando uma máscara de tamanho 3x3. Cada situação é representada por um determinado peso para a vazão e outro para a CPU de cada nó do aglomerado de computadores heterogêneos. Na tabela 3 são apresentadas quatro situações diferentes executadas em 12 máquinas do aglomerado heterogêneo descrito na seção 2. O mestre da aplicação nessas situações foi uma máquina **AMP2400**.

A primeira coluna da tabela 3 mostra os pesos do poder de processamento e vazão da rede, da segunda a sexta coluna a quantidade percentual de dados enviado para cada máquina, conforme os dados do avaliador de capacidades. A última coluna mostra o tempo de execução de cada caso.

A primeira situação mostra o tempo de execução da aplicação sem realizar o balanceamento, enviando para cada processo em cada máquina a mesma quantidade de dados. A segunda configuração mostra o balanceamento levando-se em conta apenas o poder de processamento. Pode ser observado uma melhora de 25% em relação a situação sem fazer o balanceamento. Na terceira situação, o valor de capacidade de cada máquina é composto por 50% de processamento e 50% de vazão. O tempo de execução mostra uma melhora de 27% em relação a primeira situação. A última configuração leva em consideração na capacidade de cada máquina apenas a vazão. O tempo de execução mostra uma melhora de 29% em relação a configuração sem balanceamento. Isso ocorre devido ao tamanho da imagem e da máscara serem relativamente pequenos. Neste caso, grande parte da imagem é processada na máquina local, pelo mestre da aplicação, pois a vazão local é relativamente maior se comparada a vazão dos outros nós. No entanto, para imagens e máscaras maiores, a última configuração da tabela provavelmente prejudicará o desempenho da aplicação, sobrecarregando o processo mestre.

cpu-vaz	Qtdade. dados em porcentagem					tempo
	AMP2400	A1700	PIIIC	PIIID	P4	
-	20	20	20	20	20	0.3160
100-0	25	18	13	12	30	0.2377
50-50	40	15	12	11	24	0.2281
0-100	72	6	6	6	6	0.2216

Tabela 3. Cada linha representa os pesos de processamento e vazão, a porcentagem de dados enviados para cada máquina e o tempo de execução

5.2. Método de resolução de equações lineares

Grande parte dos problemas da física e de engenharia estão apresentados matematicamente por sistemas de equações diferenciais parciais [6]. Mesmo para condições simples de contorno a maioria dos métodos numéricos como o método de diferenças finitas, o método de elementos finitos e o método de volumes finitos, transformam estas equações em sistemas de equações lineares algébricas de grande porte. A solução analítica dessas equações é muito difícil de ser obtida. Para a obtenção de soluções em tempos aceitáveis, a utilização de processamento paralelo utilizando aglomerados de computadores é indispensável quando não se tem acesso a supercomputadores.

O algoritmo seqüencial para o método de Householder é muito utilizado na área de matemática aplicada [9, 12]. O número de cálculos nesta aplicação aumenta de acordo com o tamanho da matriz.

A paralelização desse método, dividindo a matriz em conjuntos de tarefas (grupos de linhas), tornou viável a resolução de sistemas de equações algébricas com matrizes de grande ordem, com tempos de resposta satisfatórios. O algoritmo demanda grandes quantidades de comunicação e processamento. Cada nó da máquina virtual paralela irá receber um determinado número de linhas para calcular. A cada coluna calculada, por cada nó de processamento, são necessárias ao menos duas comunicações para efetuar atualização e compartilhamento de dados.

Resultados A paralelização do método Householder utiliza bastante comunicação e processamento. Quanto maior a matriz utilizada, maior o processamento, a memória necessária e a quantidade de dados a ser transferida. Em matrizes menores o principal peso estará sobre a comunicação, vazão e latência.

Na seqüência são apresentados alguns gráficos da execução do algoritmo paralelo de Householder. Os dados mostram a utilidade do avaliador de cargas e a necessidade de distribuir as tarefas e a quantidade de carga de trabalho de

acordo com as características da aplicação e o tamanho da matriz utilizada. As tarefas são representadas por conjuntos de linhas (equações) a serem processadas.

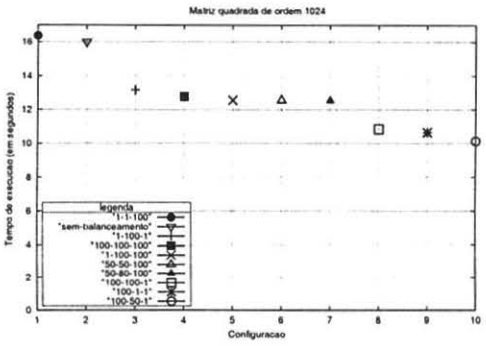

Figura 4. Tempos de execução matriz de ordem 1024

No gráfico da figura 4 é apresentada a execução do programa utilizando uma matriz de 1024x1024. Para a execução da aplicação, em todos os casos, foram utilizados 10 nós de processamento do aglomerado de computadores heterogêneos descrito na seção 2. Os resultados nesse gráfico mostram que a configuração de pesos 100-50-1 (latência-vazão-CPU) foi a que apresentou o melhor tempo de execução. Ou seja, para um tamanho de matriz dessa magnitude o que mais influencia na distribuição da carga de processamento é a latência e a vazão dos dispositivos de rede. A configuração 1-1-100, levando em consideração apenas o poder das CPUs, levou a aplicação a obter um desempenho baixo.

Com exceção da configuração 1-1-100, todas as demais configurações proporcionaram um desempenho melhor em relação a execução sem balanceamento. O melhor caso atingiu um desempenho aproximadamente 37% melhor que a execução sem balanceamento.

O gráfico da figura 5 apresenta os tempos de execução de uma matriz de ordem 2048. As execuções foram repetidas utilizando duas redes diferentes. Neste caso, executando a aplicação na rede Gigabit, a configuração 100-50-30-1 (latência-vazão-CPU-memória) apresentou o melhor desempenho. A necessidade de processamento aumentou devido ao tamanho da matriz em relação ao gráfico da figura 4. O número de operações matemáticas do algoritmo aumenta significativamente com o tamanho da matriz.

A pior configuração de capacidades de computação ocorreu com um peso alto para a memória e um peso baixo para os demais elementos. Ou seja, a memória é provavelmente um fator pouco relevante. Mesmo na máquina com a menor quantidade de memória a parte correspondente a sua tarefa (número de linhas da matriz) cabe por completo em memória.

No gráfico da figura 5, percebe-se que executando o avaliador de capacidade e a aplicação sobre a mesma configuração mas com diferentes ambientes de comunicação (Gigabit e Fast Ethernet) pode alterar o comportamento do desempenho da aplicação. No caso da rede Fast Ethernet existem diferentes adaptadores de rede e um comutador mais lento. O diferente comportamento da curva do gráfico está diretamente relacionada a esses fatores. Neste caso, as máquinas que possuíam os melhores adaptadores Gigabit provavelmente não possuem os melhores adaptadores Fast Ethernet, o que leva o avaliador de carga a atribuir novas capacidades relativas de computação aos nós de processamento. Mesmo assim, o desempenho das configurações que sofreram um maior distúrbio continua superior ao desempenho da execução sem balanceamento de cargas.

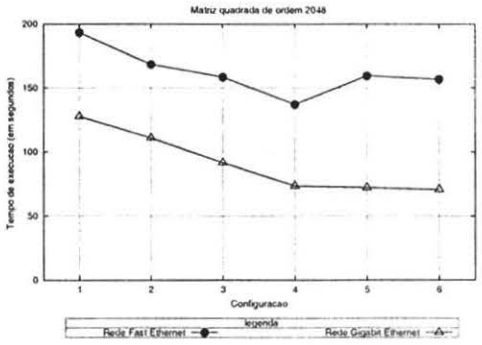

Figura 5. Tempo de execução. Configuração: Latência-Vazão-CPU-Memória. Representação do eixo horizontal: 1 - 1-1-1-100; 2 - *Sem balanceamento*; 3 - 100-100-100-100; 4 - 100-1-1-1; 5 - 100-50-1-1; 6 - 100-50-30-1.

O avaliador de cargas pode também ser útil para executar simulações e tentar identificar os limites de tamanho de matrizes para se atingir um melhor desempenho sobre um determinado conjunto de máquinas e fatores. Isso pode reduzir o tempo total de processamento da aplicação. Tendo uma boa estimativa dos pesos da memória, CPU, latência e vazão provavelmente seja possível atingir ganhos consideráveis de desempenho em aglomerados de computadores heterogêneos.

5.3. O problema do caixeiro viajante com demandas heterogêneas

O GRASP (*Greedy Randomized Adaptive Search Procedures*) [7] é uma meta-heurística baseada em heurísticas de construção de soluções iniciais gulosas e heurísticas de busca local. O método GRASP é um algoritmo iterativo. Nesse processo, cada iteração é independente e composta de uma fase construtiva e uma busca local. Na fase construtiva é gerada uma solução inicial que será melhorada na fase de busca local. A cada iteração, a melhor solução é guardada como resultado.

O GRASP pode ser aplicado para a solução de uma variedade bastante grande de problemas [14, 13]. Grande parte

dos problemas de otimização combinatória podem ser representados e resolvidos por essa técnica.

Neste trabalho o algoritmo de GRASP foi utilizado para encontrar soluções do problema do caixeiro viajante com demandas heterogêneas [15, 5]. Essa variante do problema do caixeiro viajante gera uma explosão combinatória elevada por produzir um acréscimo de variáveis ao problema original. Além das distâncias, os nós ou cidades que farão parte do grafo receberão pesos que irão variar de acordo com a rota e as demandas. O objetivo dessa nova variante do problema do caixeiro viajante é analisar o comportamento e o custo da rede com a inclusão dessas demandas diferenciadas.

Na implementação do algoritmo paralelo foi utilizado a técnica de variar o principal valor do algoritmo [2, 3] ou o número de iterações e a solução inicial de cada processo do programa. O objetivo é buscar tempos menores e soluções de melhor qualidade.

Resultados Os dados a seguir apresentam algumas estatísticas da execução da aplicação sobre a instância 15_7_2_70. Uma instância 15_7_2_70 significa que existem 15 cidades, o nó inicial é 7, o valor alfa é 2 e o percentual de conectividade é de 70%.

A instância utilizada é considerada de alta densidade, o que significa que a probabilidade é maior de ser encontrada a solução ótima ou uma boa solução. Em instâncias de baixa densidade, ao contrário, o número de possíveis caminhos (rotas) entre duas cidades é bastante reduzido, dificultando a busca por uma solução boa ou ótima.

Os dados mostram a média de tempo de 100 execuções. O objetivo principal foi apresentar o tempo computacional e a qualidade da solução com a utilização do avaliador de cargas para aglomerados heterogêneos (seção 4).

A aplicação é de processamento intensivo. A comunicação é necessária apenas para realizar algumas trocas de dados entre o processo mestre e os escravos no início e no fim da execução. Em instâncias grandes (muitas cidades) a comunicação será cada vez menos significante. A memória é apenas necessária para armazenar as soluções e as matrizes de demandas.

A tabela 4 apresenta os dados da execução da instância 15_7_2_70. Foram utilizados doze nós processadores, com um processo por nó. A primeira coluna representa os pesos indicados ao avaliador de cargas. No caso, os pesos estão representados na forma X-Y-Z-W. Onde X é o peso da CPU, Y é o peso da memória, Z é o peso da vazão da rede e W é o peso da latência da rede. A partir dessas entradas o avaliador de cargas obtém as capacidades relativas dos nós. Estes pesos são utilizados para efetuar a distribuição de cargas.

Nessa tabela pode ser observado que o melhor tempo é atingido com a configuração 80-10-0-10. O tempo dessa configuração é menor que os tempos das configurações como 97-1-1-1 e 100-0-0-0 porque a comunicação e memória tem um certo peso para a aplicação. A configuração 90-0-0-10 é melhor que a configuração 90-0-10-0 porque a aplicação utiliza apenas mensagens pequenas, logo, a vazão não terá muita influência. Todas as configurações executadas, segundo as principais características da aplicação, resultaram em um desempenho final superior a execução do programa sem a utilização de balanceamento de cargas. O melhor caso (quarta linha da tabela) teve um desempenho em torno de 20% superior a execução sem balanceamento de cargas (primeira linha da tabela). Esses resultados demonstram mais uma vez a utilidade do avaliador de cargas. Em todas as execuções a qualidade da melhor solução encontrada foi a mesma.

cpu-me-va-la	tempo (s)	cpu-me-va-la	tempo (s)
-	100.13	-	100.13
70-10-10-10	81.66	90-0-10-0	88.74
80-0-10-10	82.14	90-10-0-0	86.24
80-10-0-10	80.17	97-1-1-1	89.63
90-0-0-10	84.84	100-0-0-0	91.01

Tabela 4. Execução da instância 15_7_2_70.

5.4. Trabalhos relacionados e comentários

A área de distribuição de cargas em ambientes heterogêneos de computação continua sendo alvo de pesquisa, estudo e desenvolvimento de ferramentas. Idéias, protótipos e ferramentas continuam a ser projetados [1, 8, 10, 16, 4]. Grande parte dessas propostas e utilitários tem como objetivo conjuntos específicos de aplicações e ambientes. Alguns modelos e propostas estão ainda em fase de concepção e projeto.

Nesse contexto, o avaliador de capacidade de computação vem contribuir no desenvolvimento de ferramentas para ambientes heterogêneos de computação. Ele pode ser uma boa opção para usuários que buscam simplicidade e flexibilidade e/ou pretendem estabelecer balanceamento de carga em nível de aplicação.

O avaliador de cargas aqui apresentado tem se mostrado uma boa ferramenta para conseguir ganhos de desempenho no balanceamento de cargas em aplicações paralelas executadas em aglomerados de computadores heterogêneos. Mesmo o usuário fornecendo pesos que não representam devidamente as características principais da aplicação, o avaliador de cargas consegue estabelecer valores de capacidade de computação que provavelmente irão gerar ganhos de desempenho se comparados a uma execução sem balanceamento de cargas.

A utilização do avaliador de capacidade computacional é simples. Para o desenvolvedor utilizar os valores produzidos pelo avaliador basta lê-los da entrada padrão e realizar a distribuição de cargas da aplicação de acordo com esses valores. Os nós LAM/MPI com maiores capacidades relativas

de computação devem receber as maiores cargas de trabalho. Enquanto que os nós com capacidades menores devem receber menos carga para a aplicação obter um desempenho melhor.

O avaliador de cargas pode também ser utilizado como uma ferramenta auxiliar para encontrar gargalos de desempenho e problemas nas aplicações. Estabelecendo pesos, fatores e executando a aplicação com os valores de capacidade gerados pelo avaliador , o usuário pode tentar verificar aonde estão os pontos de maior computação do programa paralelo. Se aumentando os pesos da comunicação o desempenho da aplicação melhora, pode significar que ela está utilizando mais comunicação. Se essa não deveria ser uma característica do programa, existe uma probabilidade de alguma falha de desenvolvimento ter ocorrido. Neste caso, o usuário pode tentar melhorar os pontos de comunicação da aplicação e/ou encontrar a falha.

6. Conclusão e Trabalhos Futuros

Com os dados apresentados durante o texto é possível concluir que é interessante, e muitas vezes necessário, conhecer o ambiente de execução para se obter o máximo de desempenho com os recursos computacionais disponíveis.

Informações como e onde executar processos mestres, de determinados tipos de aplicações, e qual o nível de concorrência que melhor se adapta em uma combinação de nós heterogêneos são aspectos que dão suporte a um melhor escalonamento e distribuição de cargas.

O avaliador de capacidade de computação é uma ferramenta que pode auxiliar desenvolvedores de aplicações paralelas a encontrar valores de balanceamento de cargas para aglomerados de computadores heterogêneos. Além disso, ele pode ser útil para encontrar gargalos ou fatores de otimização para os algoritmos paralelos do usuário.

Trabalhos futuros. A intenção é continuar testando aspectos de desempenho em aglomerados heterogêneos utilizando aplicações específicas de avaliação de desempenho e aplicações reais. A idéia é dar suporte e desenvolver algoritmos de balanceamento de carga e escalonamento que tratem melhor dos aspectos e características de desempenho apresentadas neste texto. Um passo inicial foi o avaliador de capacidade relativa de computação para aplicações que busquem estabelecer balanceamento de cargas utilizando fatores como CPU, memória e rede. O objetivo é continuar desenvolvendo esta ferramenta para estendê-la a outros tipos de aplicações e ambientes. Além disso, dar suporte a execuções automáticas de aplicações pelo avaliador de capacidade de computação. Pretende-se com isso transformar essa ferramenta em um sistema que busca automaticamente configurações ótimas de pesos para diferentes fatores computacionais. Pretende-se também incluir fatores como características do disco rígido e a latência e vazão entre processadores e sistemas internos de memória.

Referências

[1] Micah Adler, Ying Gong, and Arnold L. Rosenberg. Optimal sharing of bags of tasks in heterogeneous clusters. In *Proceedings of the fifteenth annual ACM symposium on Parallel algorithms and architectures*, pages 1–10. ACM Press, 2003.

[2] A.C.F. Alvim. Estratégias de paralelização da metaheurística GRASP. Master's thesis, Departamento de Ciência da Computação, PUC-RIO, Abril 1998.

[3] A.C.F. Alvim and C.C. Ribeiro. Balanceamento de Carga na Paralelização da Meta-heurística GRASP. Technical report, Departamento de Ciência da Computação, PUC-RIO, 1998.

[4] Janez Brest, Viljem Žumer, and Milan Ojsteršek. Dynamic scheduling on a pc cluster. In *Proceedings of the 1999 ACM symposium on Applied computing*, pages 496–500. ACM Press, 1999.

[5] A. Z. Cordenonsi, F. M. Müller, H. P. L. Lima, and J. F. M. Sarubbi. Aplicação do algoritmo de savings para o problema do caixeiro viajante com demandas heterogêneas. In *XXVI Congresso Nacional de Matemática Aplicada e Computacional*, 2003.

[6] M. C. C. Cunha. *Métodos Numéricos*. Editora da UNICAMP, 2 edition, 2000.

[7] T. A. Feo and M. G. C. Resende. Greedy randomized adaptive search procedures. *Journal of global optimization*, 6:109–133, 1995.

[8] Boon Ping Gan and et.al. Load balancing for conservative simulation on shared memory multiprocessor systems. In *Proceedings of the fourteenth workshop on Parallel and distributed simulation*, pages 139–146. IEEE Computer Society, 2000.

[9] O. Khatchatourian and P. A. Borges. *Métodos Numéricos da Álgebra*. Editora Unijuí, 1995.

[10] Kaoutar El Maghraoui and et.al. Adaptive computation over dynamic and heterogeneous networks. In *Proc. Fifth International Conference on Parallel Processing and Applied Mathematics (PPAM 2003)*, volume 3019, pages 1083–1090, 2004.

[11] Marion. *An Introduction to Image Processing*. Chapman and Hall, 1991.

[12] J. M. Ortega. *Introduction to parallel and Vector Solution of Linear System*. Plenum Press, New York, 1998.

[13] P.M. Pardalos, L.S. Pitsoulis, and M.G.C. Resende. A parallel GRASP implementation for the quadratic assignment problem. In *Parallel Algorithms for Irregularly Structured Problems – Irregular'94*, pages 115–130. Kluwer Academic Publishers, 1995.

[14] P.M. Pardalos, L.S. Pitsoulis, and M.G.C. Resende. A parallel GRASP for MAX-SAT problems. *Lecture Notes in Computer Science*, 1184:575–585, 1996.

[15] João Fernando Machry Sarubbi. Um modelo linear de fluxos para o problema do caixeiro viajante com demandas heterogêneas. Master's thesis, Mestrado em Ciência da Computação, UFMG, 2003.

[16] S.Chau and A. Fu. Load balancing between heterogenous computing clusters. In *The Second International Workshop on Grid and Cooperative Computing*, December 2003.