Avaliando e Comparando Diferentes Estruturas de Programas Paralelos Através de Modelos Analíticos de Desempenho

Jean Marcos Laine, Edson Toshimi Midorikawa
LAHPC - Laboratory of Architecture and High Performance Computing
Departamento de Engenharia de Computação e Sistemas Digitais
Escola Politécnica - Universidade de São Paulo
Av. Prof. Luciano Gualberto, trav. 3, 158
São Paulo – SP – 05508-900, Brasil
jmlaine@gmail.com, edson.midorikawa@poli.usp.br

Resumo

Um programa paralelo e distribuído pode ter seu código fonte estruturado de diferentes modos. A organização da divisão e distribuição dos dados é uma atividade crítica para o desempenho final da aplicação. Assim, é importante que exista uma metodologia capaz de auxiliar estudos com o objetivo de comparar diferentes abordagens de soluções e prever qual modelo é o mais adequado para organizar a solução da aplicação. Nesse artigo, demonstramos como a metodologia PEMPIs-Het pode ser utilizada para esse propósito. Os resultados obtidos confirmam a capacidade da metodologia em avaliar e predizer corretamente o desempenho de diferentes estruturas de programas paralelos.

1 Introdução

O desenvolvimento de aplicações paralelas eficientes não é uma tarefa trivial. Explorar o paralelismo das aplicações é tão complicado quanto estruturar a solução para otimizar o uso dos recursos computacionais e alcançar o desempenho desejado. Essa atividade é ainda mais complexa se o sistema computacional for distribuído e heterogêneo. Soluções paralelas eficientes para ambientes distribuídos devem levar em consideração não só a correta computação do problema mas também a forma como o trabalho é dividido e distribuído entre os recursos computacionais, a fim de maximizar o uso da capacidade individual de cada uma das máquinas do ambiente. Portanto, vários problemas devem ser levados em consideração pelo desenvolvedor, tais como balanceamento de carga, insuficiência de paralelismo, sincronismo entre as tarefas, comunicações e contenção de recursos [8].

Nesse contexto, a avaliação de diferentes estruturas organizacionais para uma solução paralela distribuída é uma tarefa imprescindível no desenvolvimento das aplicações. Decidir qual modelo é o mais eficiente requer análises sobre o desempenho de cada abordagem bem como predições sobre o comportamento de cada solução. Em geral, uma solução paralela pode ser organizada através dos seguintes modelos: mestre-escravo, paralelismo de dados, *workpool* e divisão e conquista. No entanto, o modelo mestre-escravo é um dos mais difundidos [6, 21, 27]. Nesse modelo, o processo mestre é o responsável por organizar a divisão e distribuição do trabalho. Cabe aos escravos processar a tarefa recebida e devolver o resultado da execução para o mestre. Quando todos os escravos enviam os resultados parciais, a solução final do problema é obtida [4].

Pelo nosso conhecimento, a área da computação paralela e distribuída é carente de metodologias que auxiliem estudos sobre estruturas de programas paralelos. Alguns trabalhos têm avaliado elementos específicos de um programa paralelo, como as primitivas de comunicação [16], as estruturas de repetição [23, 29], o escalonamento das tarefas [3] ou o desempenho de aplicações [1, 8]. No entanto, desconhecemos trabalhos que avaliem especificamente estruturas de programas paralelos com o intuito de gerar soluções otimizadas para um ambiente computacional conhecido.

Assim, definimos em [12, 14] uma metodologia, denominada PEMPIs-Het (*Performance Estimation of MPI Programs in Heterogeneous Systems*), que especifica um conjunto de técnicas de modelagem, avaliação e predição de desempenho que pode ser utilizado para avaliar diferentes modelos de soluções paralelas. Nosso objetivo é avaliar qual estrutura é a mais adequada para uma aplicação no sistema alvo.

Normalmente, os trabalhos encontrados na literatura apresentam modelos para avaliar o desempenho de primi-

tivas de comunicação, de estruturas básicas de programas (laços, condicionais, etc) ou de uma aplicação em particular. Portanto, não conhecemos uma metodologia que possa ser aplicada para avaliar o que estamos propondo e da forma como pretendemos conduzir os estudos.

Assim, o objetivo principal desse artigo é apresentar como a metodologia PEMPIs-Het permite avaliar e comparar estruturas de programas paralelos. Alguns resultados obtidos em testes experimentais mostram a capacidade da metodologia em conduzir avaliações e estudos que permitam alcançar o objetivo destacado.

As seções seguintes desse artigo estão organizadas da seguinte forma: na Seção 2 comentamos sobre alguns trabalhos relacionados ao tema central desse artigo; o esquema geral da metodologia PEMPIs-Het é apresentado na Seção 3; a seção seguinte descreve as versões do modelo mestre-escravo avaliadas; a Seção 5 apresenta o ambiente computacional; a Seção 6 discute os resultados obtidos com as análises e, por último, temos as conclusões geradas no trabalho.

2 Trabalhos Relacionados

Alguns autores têm utilizado a modelagem analítica para identificar os fatores que podem influenciar no desempenho das aplicações [24, 1]. Na literatura existem vários trabalhos sobre modelagem e predição de desempenho de aplicações paralelas. No entanto, grande parte das publicações acabam restringindo as análises para alguns elementos do programa, como as estruturas de repetição ou as primitivas de comunicação.

Em [16], o autor apresenta uma modelagem das primitivas de comunicação, levando em consideração o impacto da heterogeneidade dos processadores no desempenho da transmissão dos dados. Modelos analíticos, em função do tamanho da mensagem, são gerados para caracterizar o comportamento das operações ponto-a-ponto e coletivas. Em [17] os autores caracterizam o comportamento da aplicação em uma rede não dedicada de computadores heterogêneos. Para isso, curvas de desempenho baseadas em intervalos são utilizadas. A estratégia determina que o desempenho da aplicação pode ser estimado por um limite superior e outro inferior (intervalo de predição). A avaliação e a predição realizada trata especificamente uma aplicação particular.

Em [7, 8] é proposto um sistema para modelar o desempenho de programas paralelos que utilizam o modelo de programação baseado em passagem de mensagem. Esse sistema, denominado PEVPM (*Performance Evaluating Virtual Parallel Machine*), faz uso de máquinas paralelas virtuais para realizar as atividades relacionadas à análise e avaliação de desempenho das aplicações, em uma abordagem *bottom-up* [9]. A estrutura do programa é dividida em segmentos de código seqüencial e de comunicação, sendo cada trecho modelado separadamente. Portanto, as análises são sobre as estruturas básicas dos programas.

Já em [23] e [29] os autores apresentam uma modelagem para laços de repetição de programas paralelos. Além disso, eles implementam mecanismos de balanceamento de carga do tipo *self-scheduling* para dividir o trabalho contido nessas estruturas. Toda a avaliação é sobre o desempenho computacional dos laços de repetição.

Em trabalhos anteriores formalizamos um conjunto de estratégias para a modelagem e predição de desempenho de aplicações MPI. A especificação da metodologia PEMPIs [19, 18] foi desenvolvida para permitir análises de desempenho em ambientes homogêneos. Nossas avaliações também estavam restritas a modelagem e predição de desempenho de uma aplicação particular.

Assim, ao descobrir a necessidade de avaliar diferentes estruturas de programas paralelos, também em ambientes heterogêneos, estendemos nossa metodologia e criamos o PEMPIs-Het (*Performance Estimation of MPI Programs in Heterogeneous Systems*) [13]. Essa metodologia permite, além da modelagem e predição de desempenho de aplicações paralelas, realizar estudos com o propósito destacado neste artigo.

3 A Metodologia PEMPIs-Het

A metodologia PEMPIs-Het [13] define um processo estruturado de fácil aplicação capaz de auxiliar o desenvolvimento de aplicações paralelas de alto desempenho. Esse processo compreende atividades de modelagem, avaliação e predição de desempenho de programas paralelos distribuídos tanto em ambientes homogêneos quanto heterogêneos. A organização da metodologia é ilustrada na Figura 1. Uma breve explicação sobre os principais processos da metodologia é dada a seguir:

- AME - ***A**pplication **M**odeling **E**nvironment*: especifica um conjunto de atividades relacionadas à modelagem analítica das aplicações paralelas distribuídas;
- PWD - ***P**erformance Estimation and **W**orkload **D**istribution*: especifica funções associadas a atividade de predição de desempenho e balanceamento de carga do sistema;
- MWD - ***M**iddleware for **W**orkload **D**istribution*: um plano de execução inicial é definido. Este plano busca uma distribuição adequada das cargas computacionais no ambiente;
- PEM - ***PE**rformance **M**onitor*: define um conjunto de funções para acompanhar, em tempo real, o desempenho das aplicações e a carga de trabalho individual de

cada uma das máquinas do ambiente. As informações de desempenho são coletadas por um monitor. No processo MWD, é possível avaliar se o planejamento inicial está adequado à situação observada.

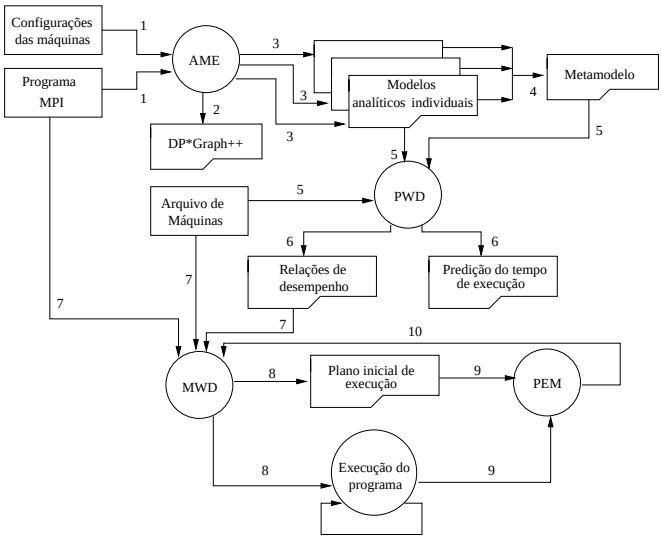

Figura 1. Metodologia PEMPIs-Het.

4 Modelos para Aplicações Paralelas

Essa seção descreve algumas maneiras de se estruturar programas paralelos para organizar a divisão e distribuição de tarefas entre processos distribuídos.

O paralelismo de dados é uma técnica de programação que divide uma grande quantidade de dados em partes menores que podem ser processados em paralelo pelo programa [2, 5, 25]. Após o término do processamento, os resultados parciais são combinados e a solução final é obtida. No modelo *workpool* [10] as tarefas são disponibilizadas em uma estrutura de dados global e os processos acessam posições de memória compartilhada para adquirir os dados que serão utilizados no processamento. Já no modelo de divisão e conquista [11] um trabalho complexo é dividido em tarefas mais simples para facilitar a solução. Basicamente, o problema é dividido recursivamente em problemas menores até que ele possa ser resolvido diretamente.

Nesse trabalho, em particular, estamos analisando e comparando variações do modelo mestre-escravo [3, 4] em relação a divisão e distribuição das cargas computacionais. Normalmente, a distribuição do trabalho entre os escravos é feita de maneira estática, como no caso da estratégia VRP (Seção 4.2). Isso significa que no início da execução do programa todo o trabalho é dividido e distribuído pelo mestre. Essa estratégia é adequada quando o ambiente de execução é homogêneo e dedicado. Se a carga computacional das máquinas sofre variação ao longo do tempo, a distribuição estática não produz bons resultados. Assim, uma abordagem dinâmica pode ser utilizada para melhorar o desempenho da solução. As versões SS, VRP-CSS e VRP-CSS-Adapt são estratégias dinâmicas para distribuição de cargas. Nessas estratégias a divisão e distribuição do trabalho ocorre em fases. A medida que os processos terminam uma execução parcial, um novo trabalho é enviado pelo mestre. Isso se repete até que não haja mais trabalho a ser processado. Dessa forma, os processos que estão sendo executados em máquinas de maior capacidade computacional ou com menor carga de trabalho, conseguem colaborar mais na execução do programa.

Para a realização de nossas avaliações implementamos quatro versões para o programa de multiplicação de matrizes, baseadas no modelo mestre-escravo: *self-scheduling*, VRP, VRP-CSS e VRP-CSS-Adaptativo. Essas estratégias são descritas nas próximas seções.

4.1 Self-Scheduling

Na estrutura *Self-Scheduling* (SS) o processo mestre divide o trabalho total em pequenas tarefas, denominadas cargas unitárias de trabalho, para, em tempo de execução, enviá-las aos processos escravos. A medida que os processos vão terminando o processamento da tarefa recebida eles enviam a resposta ao mestre e solicitam uma nova carga unitária. Portanto, é uma estratégia dinâmica de balanceamento de carga. Um dos problemas dessa abordagem é a excessiva quantidade de mensagens trocadas entre os processos no decorrer da execução do programa. Uma ilustração desse modelo é apresentada na Figura 2. Na literatura é possível encontrar algumas variações para essa estratégia, tais como a *Chunk Self-Scheduling (CSS)*, *Guided Self-Scheduling (GSS)* e a *Trapezoidal Self-Scheduling (TSS)* [20, 22, 26, 28, 30].

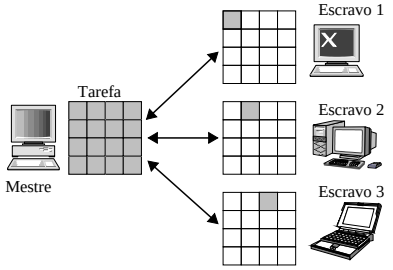

Figura 2. Modelo SS.

4.2 Vetor de Desempenho Relativo

Essa estratégia, denominada VRP, foi especificada e implementada para permitir que cada processo receba uma

tarefa adequada à capacidade computacional da máquina onde o mesmo está sendo executado [14]. Para isso, o algoritmo utiliza um vetor com um conjunto de índices associados à capacidade de processamento individual de cada máquina do ambiente. Esses valores são obtidos através dos modelos analíticos de desempenho gerados para a aplicação. Através desse vetor é possível determinar a quantidade de trabalho que deve ser enviada a cada nó de processamento, conforme ilustrado na Figura 3.

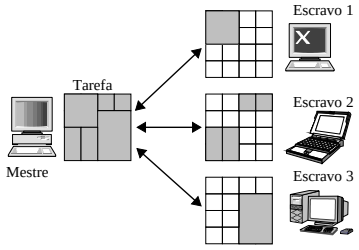

Figura 3. Modelo VRP.

O intuito desse modelo de organização e distribuição de dados é diminuir a quantidade de mensagens trocadas na estratégia *Self-Scheduling* e, conseqüentemente, melhorar o desempenho da solução. No entanto, o sucesso dessa abordagem está relacionada a precisão dos valores especificados no VRP. Além disso, a distribuição dos dados é estática. Isso significa que alterações nas relações de desempenho das máquinas, provocadas pela mudança na carga computacional dos nós, podem atrapalhar o desempenho da solução.

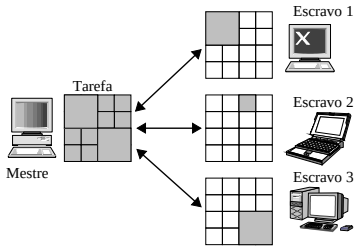

Figura 4. Modelo VRP-CSS.

4.3 VRP com Chunk-Self-Scheduling

Essa estratégia, denominada VRP-CSS, foi implementada para eliminar os problemas da versão VRP. O balanceamento para essa estratégia é dinâmico e segue a abordagem adotada no *Chunk Self-Scheduling (CSS)*. Em tempo de execução cada processo escravo recebe do mestre um trabalho adequado à capacidade de processamento local da máquina que está sendo utilizada para sua execução. O tamanho da tarefa é determinado através dos valores especificados no VRP, assim como é feito na estratégia anterior. Ao terminar o processamento os escravos enviam a resposta ao mestre e recebem uma nova fatia do processamento. A idéia é que os processos mais rápidos possam calcular uma parte maior da solução, minimizando a quantidade de mensagens trocadas e sincronizando a execução final de cada processo escravo. Uma ilustração da estratégia pode ser visualizada na Figura 4.

4.4 VRP CSS Adaptativo

Essa estratégia, denominada VRP-CSS-Adapt, é uma abordagem adaptativa da versão VRP-CSS. Em tempo de execução, o processo mestre calcula, a cada resposta parcial, a capacidade de processamento efetiva (Cpe)[1] de cada uma das máquinas onde os escravos estão sendo executados. Baseado no valor calculado para o Cpe e nos índices do VRP é que o mestre determina a quantidade de trabalho ideal a ser enviado em cada iteração do processamento. Portanto, o ajuste proporcionado pela estratégia permite a aplicação reagir às alterações de carga do sistema, o que não é possível nas outras estratégias. Esse ajuste dinâmico tende a evitar eventuais sobrecargas e/ou ociosidades das máquinas.

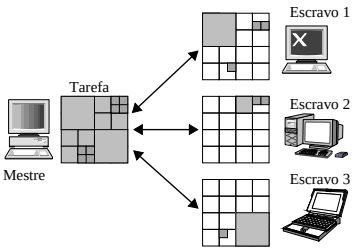

Figura 5. Modelo VRP-CSS-Adapt.

5 Ambiente Computacional

Os testes experimentais foram executados em uma rede de computadores heterogêneos formada por três tipos de máquinas, nomeadas de intel, bio e taurus. As máquinas intel possuem processadores Intel Pentium D 950, 2GB de DDR2 SDRAM, placas de rede gigabit Ethernet. As máquinas bio possuem dois processadores AMD Athlon MP 2400$^+$, com 1GB de DDR SDRAM, placas de rede Intel Ether-Express Pro Fast Ethernet. Já as máquinas taurus possuem um processador Intel Celeron 433MHz, 256 MB de SDRAM, placa de rede Fast Ethernet. O sistema operacional instalado nas máquinas é o Fedora.

[1] $Cpe = \frac{\tau}{t}$, sendo τ a quantidade de trabalho processado e t o tempo de execução gasto na computação de τ.

6 Resultados

Nessa seção discutimos a capacidade da metodologia PEMPIs-Het em avaliar diferentes modelos de soluções paralelas distribuídas e apontar qual estratégia é a mais eficiente para estruturar a aplicação avaliada no sistema alvo.

6.1 Análise das Estruturas

Para a realização das análises propostas utilizamos quatro diferentes versões para o tradicional programa de multiplicação de matrizes, denominadas Mat-SS, Mat-VRP-CSS, Mat-VRP-CSS-Adapt e Mat-VRP. A estrutura organizacional de cada solução foi apresentada e discutida na Seção 4. A influência da estrutura da solução no desempenho da aplicação pode ser facilmente observada com os dados apresentados nos gráficos das Figuras 6 e 7.

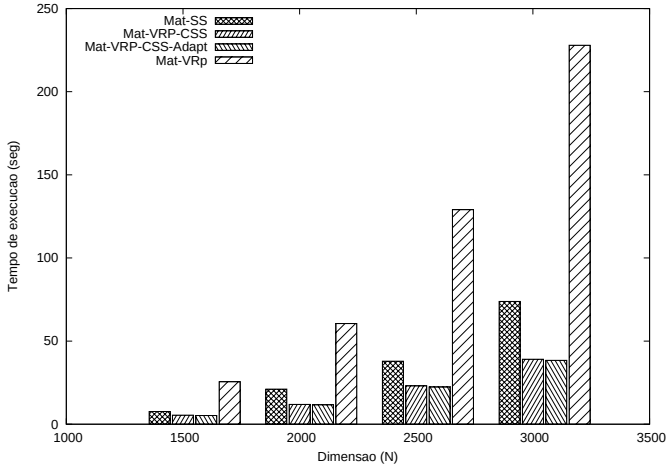

Figura 6. Desempenho medido no *cluster* dedicado.

Os resultados apresentados no gráfico da Figura 6 foram obtidos sem carga computacional extra no sistema. As versões da aplicação tiveram exclusividade no uso do ambiente (*cluster* dedicado). Já o gráfico da Figura 7 apresenta o desempenho de cada uma das versões no mesmo ambiente mas com carga artificial extra (*cluster* compartilhado). Para gerar essa carga implementamos um programa que fica em um laço infinito executando instruções de potenciação e radiciação. O consumo de CPU se mantém constante durante todo o tempo. Dessa forma, simulamos um ambiente de execução compartilhado, onde outras aplicações podem disputar os recursos computacionais com o programa de multiplicação de matrizes. Em todos os testes experimentais utilizamos 20 unidades de processamento, sendo 4 das máquinas intel, 6 das taurus e 10 das máquinas bios. A carga artificial foi executada em 1 máquina intel e 3

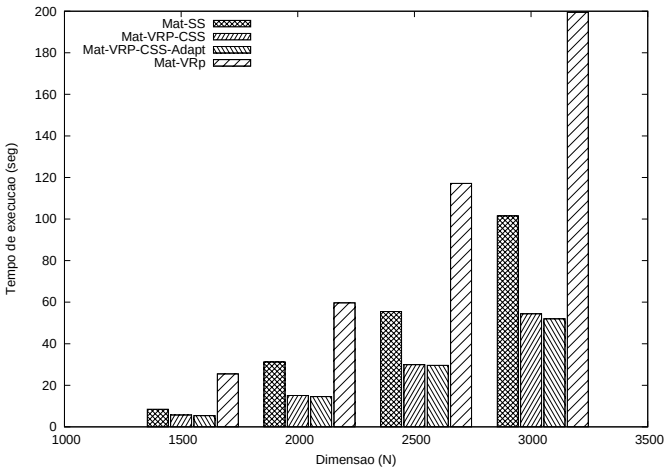

Figura 7. Desempenho medido no *cluster* compartilhado.

máquinas bio. A escolha destas máquinas foi aleatória. Nossa intenção era modificar a carga computacional em algumas máquinas do sistema. Portanto, não importa qual máquina seja escolhida. Com exceção da versão Mat-VRP, todas as demais apresentaram tempo menor no *cluster* dedicado, como devia ocorrer. O comportamento da estratégia Mat-VRP pode ser explicado se considerarmos a possibilidade dos valores do VRP estarem mais ajustados à capacidade computacional relativa das máquinas na presença da carga extra.

Em ambos os testes a versão adaptativa Mat-VRP-CSS-Adapt foi a que apresentou os melhores índices de desempenho, levemente superior a versão Mat-VRP-CSS. Isso pode ser explicado pois embora a versão adaptativa promova o ajuste das cargas computacionais, o que melhora o desempenho do sistema, o tempo gasto na estratégia pode comprometer o desempenho da solução quando o sistema apresenta pouca variação de carga, como avaliado nos testes experimentais. Nesse caso, a otimização é pequena e de pouca influência no desempenho final da aplicação.

As versões Mat-SS e Mat-VRP tiveram um desempenho consideravelmente inferior às duas primeiras e, portanto, podem ser descartadas como opções para a organização da solução do programa. Uma explicação para os resultados ruins da versão *self-scheduling* é que ela tende a perder desempenho quando a carga do sistema se mantém constante e o tamanho do problema aumenta significativamente. Isto porque a quantidade de mensagens necessárias para a distribuição das cargas individuais aumenta na mesma proporção e, portanto, o tempo de transmissão dos pacotes na rede de interconexão pode comprometer o desempenho global da aplicação. Já a versão Mat-VRP além de promover uma distribuição estática das tarefas o resultado é

extremamente dependente da precisão dos modelos de desempenho utilizados para elaborar o VRP. Como a relação de desempenho entre as máquinas se altera ao variar o tamanho do problema e os valores do VRP não são modificados e tendência é que a estratégia não acompanhe essas alterações.

No cenário de *cluster* dedicado, a diferença entre o desempenho da melhor (Mat-VRP-CSS-Adapt) e da pior estratégia (Mat-VRP), com matrizes de 3000×3000 elementos, chegou próximo a 500%. O tempo de execução da versão adaptativa foi de $38,43$ segundos e da Mat-VRP foi de $227,95$ segundos. Já no cenário com a carga artificial, a versão Mat-VRP-CSS-Adapt teve um desempenho quase 4 vezes melhor que a versão Mat-VRP. Os tempos médios medidos foram de $199,46$ para a versão Mat-VRP e $51,99$ segundos para a versão adaptativa.

6.2 Modelagem

A partir dos resultados experimentais podemos utilizar a metodologia PEMPIs-Het para modelar o comportamento das diferentes versões avaliadas e estimar o desempenho da aplicação em função da dimensão da matriz, por exemplo. Essa predição pode ajudar a definir, extrapolando os valores associados a dimensão das matrizes, qual o melhor modelo organizacional para a solução do problema, levando em consideração as características do ambiente de execução. Conforme descrito em [14], os modelos analíticos de desempenho são gerados a partir de informações sobre a complexidade computacional do programa. Uma análise sobre o código fonte permite obter o modelo teórico da aplicação e, conseqüentemente, a estrutura geral dos modelos analíticos. No caso da multiplicação de matrizes, todas as versões fizeram uso do modelo tradicional de computação, cuja complexidade algorítmica é $O(n^3)$. Portanto, o modelo teórico da aplicação, escrito em função da dimensão das matrizes (n) e da quantidade de processos (p), é dado por:

$$\delta^{mat}(n,p) = \frac{C4}{p} \times n^3 + \frac{C3}{p} \times n^2 + \frac{C2}{p} \times n + \frac{C1}{p} \qquad (1)$$

Neste artigo, em particular, reduzimos a modelagem para direcionar a análise do tempo de execução em função da dimensão das matrizes apenas. Logo, o desempenho da aplicação pode ser representado através do seguinte modelo:

$$\delta^{mat}(n) = C4 \times n^3 + C3 \times n^2 + C2 \times n + C1 \qquad (2)$$

A partir do modelo teórico podemos gerar os modelos individuais da aplicação para cada versão modelada, utilizando técnicas de ajuste de curvas e os dados obtidos nos testes experimentais [15]. Assim, é possível obter os valores para cada uma das constantes (C_i) do modelo teórico e realizar predições de desempenho em função do tamanho das matrizes. Após aplicar as técnicas contempladas na metodologia PEMPIs-Het obtivemos as constantes apresentadas nas Tabelas 1 e 2. Portanto, a instanciação do modelo de desempenho de cada uma das versões ocorre com a substituição desses valores no modelo teórico apresentado anteriormente.

Tabela 1. Modelagem para o *cluster* dedicado.

	C1	C2	C3	C4
SS	-172,18999	0,2530733	-0,0001206	$2{,}119 \times 10^{-8}$
VRP-CSS	12,119999	-0,0166867	0,0000078	$2{,}67 \times 10^{-10}$
VRP-CSS-Ad.	0,47	0,0001133	-0,0000001	$1{,}427 \times 10^{-9}$
VRP	152,93999	-0,2136767	0,0000921	$-4{,}173 \times 10^{-9}$

Tabela 2. Modelagem para o *cluster* compartilhado.

	C1	C2	C3	C4
SS	-254,62999	0,3539733	-0,0001597	$2{,}711 \times 10^{-8}$
VRP-CSS	-32,829998	0,0479167	-0,0000234	$5{,}693 \times 10^{-9}$
VRP-CSS-Ad.	-1,29	-0,00024	0,0000002	$1{,}92 \times 10^{-9}$
VRP	48,529998	-0,07239	0,0000351	$1{,}92 \times 10^{-9}$

6.3 Comparando Modelos de Estruturas

A etapa seguinte consiste em estimar o desempenho de cada versão, extrapolando valores para a dimensão das matrizes. Essa atividade permite prever, de acordo com a precisão dos modelos, o comportamento das versões modeladas no ambiente utilizado para os testes experimentais. Os resultados das projeções são apresentados nos gráficos das Figuras 8 e 9.

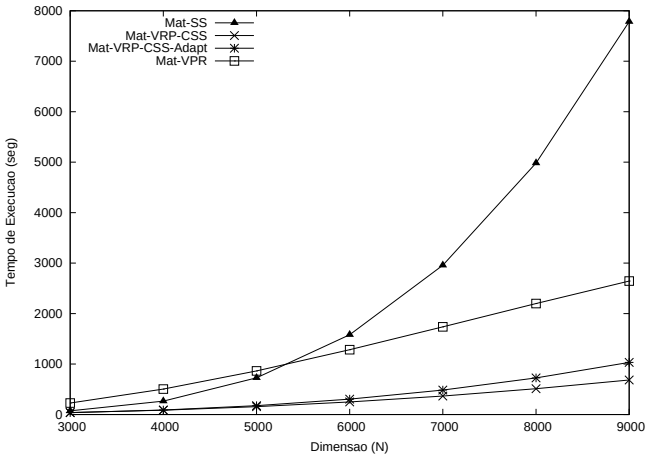

Figura 8. Predições no *cluster* dedicado.

Pelas projeções é possível notar que em ambos os cenários algumas estratégias apresentam bons resultados

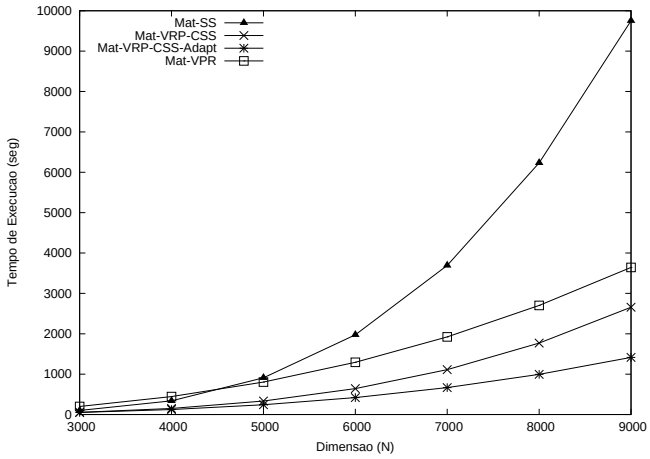

Figura 9. Predições no *cluster* compartilhado.

para menores valores de n (dimensão) mas pioram a medida que os valores aumentam. Esse é o caso, por exemplo, da versão Mat-SS. Como se esperava, pelos resultados dos testes experimentais, esse modelo não é interessante para a aplicação no ambiente avaliado, com as características que foram destacadas. No cenário de *cluster* dedicado as versões Mat-VRP-CSS-Adapt e Mat-VRP-CSS apresentam uma projeção muito semelhante para o tempo de execução. Pelos valores obtidos nos testes experimentais (Figura 6), esperávamos que a versão adaptativa tivesse melhores resultados. No entanto, as projeções mostraram que a partir de n = 4000 a estratégia Mat-VRP-CSS apresenta melhores resultados ($86,60$ seg. contra $90,63$ seg. da abordagem adaptativa), invertendo a tendência esperada. Isto pode ser explicado pois o método de ajuste de curvas empregado captura o grau de crescimento das curvas do tempo de execução e define o modelo baseado no ajuste dos maiores tempos. Uma possível causa do maior crescimento do tempo de execução medido, em relação a versão Mat-VRP-CSS, pode ser atribuída ao tempo gasto com a maior quantidade de cálculos para a definição da capacidade de processamento efetiva (Cpe) e do trabalho adequado, a medida que os valores de n aumentam.

No cenário da Figura 9 a versão adaptativa promete melhores resultados para todos os valores de n, de acordo com o modelo de desempenho obtido. Portanto, essas estimativas permitem concluir que: i) em ambos os cenários as versões Mat-SS e Mat-VRP não são boas opções para organizar a solução da aplicação no ambiente considerado; ii) no cenário com uma carga computacional extra, concorrendo pelos recursos computacionais (mais próximo de uma situação real em ambientes distribuídos), a versão adaptativa consegue melhorar o desempenho da aplicação ajustando a divisão e distribuição das tarefas.

Dessa forma, as estratégias descritas na metodologia PEMPIs-Het, aplicadas da forma como demonstramos nesse artigo, podem ser utilizadas para comparar diferentes estruturas de programas paralelos e ajudar a decidir qual modelo é o mais eficiente para a aplicação em um ambiente computacional com características conhecidas.

7 Conclusões

Nesse artigo demonstramos a capacidade da metodologia PEMPIs-Het em avaliar diferentes estruturas de programas paralelos e apontar qual estrutura e/ou, conseqüentemente, solução é capaz de oferecer o melhor desempenho computacional para um problema. Para as avaliações e predições de desempenho utilizamos quatro diferentes versões do tradicional programa de multiplicação de matrizes. Cada uma das versões implementa um modelo diferente de balanceamento de carga computacional. O reflexo dessa organização é visualizado no desempenho da aplicação, conforme ilustram os gráficos das Figuras 6 e 7.

Em geral, as projeções apresentadas nas Figuras 8 e 9 confirmam a tendência do bom ou mau desempenho das estratégias avaliadas. No entanto, algumas estimativas indicam alterações no comportamento relatado pelos testes experimentais. Pelas projeções, a versão Mat-SS passa a ser a pior estratégia a medida que a dimensão das matrizes aumenta. Nos testes realizados a estratégia com o pior desempenho foi a Mat-VRP. Esse comportamento também pode ser observado entre as versões Mat-VRP-CSS e Mat-VRP-CSS-Adapt, conforme ilustra o gráfico da Figura 8. Pelos testes experimentais a estratégia Mat-VRP-CSS-Adapt seria a mais eficiente. Contudo, as projeções mostram que a partir de $n = 4000$ a versão Mat-VRP-CSS passa a apresentar melhor desempenho, invertendo a tendência esperada.

Portanto, as informações geradas com as análises permitidas pela metodologia PEMPIs-Het pode nos ajudar a comparar diferentes estruturas de soluções e prever, para um determinado ambiente computacional, qual estratégia é capaz de otimizar o desempenho da aplicação ao variar especificações do problema ou do sistema. Embora, nesse artigo, as análises ficaram em função do tamanho do problema, é possível modificar o modelo para considerar também a quantidade de processos. Logo, as análises de desempenho poderiam ser realizadas em função desses dois parâmetros. Uma descrição de como utilizar a metodologia para elaborar modelos em função de n (tamanho do problema) e p (número de processos) pode ser obtida em [18].

Ao estimar o comportamento de alternativas de soluções é possível melhorar o desenvolvimento de programas paralelos distribuídos e ajudar em decisões de implementação para sistemas complexos, na qual o desempenho computacional é um fator importante a ser considerado.

Referências

[1] R. M. Badia, G. Rodríguez, and J. Labarta. Deriving analytical models from a limited number of runs. In *PARCO*, pages 769–776, 2003.

[2] H. E. Bal and M. Haines. Approaches for integrating task and data parallelism. *IEEE Concurrency*, 6(3):74–84, 1998.

[3] C. Banino, O. Beaumont, L. Carter, J. Ferrante, A. Legrand, and Y. Robert. Scheduling strategies for master-slave tasking on heterogeneous processor platforms. *IEEE Trans. Parallel Distributed Systems*, 15(4):319–330, 2004.

[4] O. Beaumont, A. Legrand, and Y. Robert. The master-slave paradigm with heterogeneous processors. *IEEE Transactions on Parallel and Distributed Systems*, 14(9):897–908, 2003.

[5] D. Culler, J. P. Singh, and A. Gupta. *Parallel Computer Architecture: A Hardware/Software Approach (The Morgan Kaufmann Series in Computer Architecture and Design)*. Morgan Kaufmann, August 1998.

[6] L. M. e Silva and R. Buyya. *Parallel Programming Models and Paradigms*. Prentice Hall, PTR, New Jersey, USA, 1999. In: BUYYA, R. High Performance Cluster Computing: Programming and Applications.

[7] D. A. Grove. *A Performance Modeling System for Message-Passing Parallel Programs*. PhD thesis, University of Adelaide, Department of Computer Science, Adelaide, 2003.

[8] D. A. Grove and P. D. Coddington. Communication benchmarking and performance modelling of mpi programs on cluster computers. *J. Supercomput.*, 34(2):201–217, 2005.

[9] D. A. Grove and P. D. Coddington. Modeling message-passing programs with a performance evaluating virtual parallel machine. *Perform. Eval.*, 60(1-4):165–187, 2005.

[10] J. Knopp and M. Reich. A workpool model for parallel computing. In *Proceedings of the First International Workshop on High Level Programming Models and Supportive Environments (HIPS)*, 1996.

[11] S. Kumaran and M. J. Quinn. Divide-and-conquer programming on mimd computers. In *IPPS '95: Proceedings of the 9th International Symposium on Parallel Processing*, pages 734–741, Washington, DC, USA, 1995. IEEE Computer Society.

[12] J. M. Laine. *Uma Metodologia para Desenvolvimento de Programas Paralelos Eficientes em Ambientes Homogêneos e Heterogêneos*. PhD thesis, Escola Politécnica da Universidade de São Paulo, 2008.

[13] J. M. Laine and E. T. Midorikawa. Analisando a predição de desempenho com os modelos analíticos gerados pela metodologia pempis-het. *WSCAD'07 - VIII Workshop em Sistemas Computacionais de Alto Desempenho*, 2007.

[14] J. M. Laine and E. T. Midorikawa. Using analytical models to load balancing in a heterogeneous network of computers. In V. E. Malyshkin, editor, *PaCT*, volume 4671 of *Lecture Notes in Computer Science*, pages 559–568. Springer, 2007.

[15] J. M. Laine and E. T. Midorikawa. Uso de modelos analíticos na modelagem de aplicações paralelas distribuídas. *WPerformance 2008 - Workshop em Desempenho de Sistemas Computacionais e de Comunicação*, 2008.

[16] A. Lastovetsky, I.-H. Mkwawa, and M. O'Flynn. An accurate communication model of a heterogeneous cluster based on a switch-enabled ethernet network. In *ICPADS '06: Proceedings of the 12th International Conference on Parallel and Distributed Systems*, pages 15–20, Washington, DC, USA, 2006. IEEE Computer Society.

[17] A. Lastovetsky and J. Twamley. Towards a realistic performance model for networks of heterogeneous computers. pages 39–58. Springer, 2005.

[18] E. T. Midorikawa, H. Oliveira, and J. M. Laine. Pempis: A new methodology for modeling and prediction of mpi programs performance. *International Journal of Parallel Programming*, 33(5):499–527, October 2005.

[19] E. T. Midorikawa, H. M. Oliveira, and J. M. Laine. Pempis: A new methodology for modeling and prediction of mpi programs performance. In *16th Symposium on Computer Architecture and High Performance Computing (SBAC-PAD 2004)*, pages 246–253, Foz do Iguaçu, Brazil, October 2004. IEEE Computer Society.

[20] C. D. Polychronopoulos and D. J. Kuck. Guided self-scheduling: a practical scheduling scheme for parallel supercomputers. *IEEE Transactions on Computers*, C-36(12):1425–1439, December 1987.

[21] M. J. Quinn. *Parallel Programming in C with MPI and OpenMP*. McGraw-Hill Education Group, 2003.

[22] W.-C. Shih, C.-T. Yang, and S.-S. Tseng. A performance-based approach to dynamic workload distribution for master-slave applications on grid environments. In *GPC*, pages 73–82, 2006.

[23] W.-C. Shih, C.-T. Yang, and S.-S. Tseng. A performance-based parallel loop scheduling on grid environments. *J. Supercomput.*, 41(3):247–267, 2007.

[24] A. T. C. Tam and C.-L. Wang. Realistic communication model for parallel computing on cluster. In *IWCC '99: Proceedings of the 1st IEEE Computer Society International Workshop on Cluster Computing*, page 92, Washington, DC, USA, 1999. IEEE Computer Society.

[25] V. D. Tran, L. Hluchy, and G. T. Nguyen. Parallel programming with data driven model. *pdp*, 00:205, 2000.

[26] T. T. Tzen and L. M. Ni. Trapezoidal self-scheduling: A practical scheduling scheme for parallel compilers. *IEEE Transactions on Parallel and Distributed Systems*, 4(1):87–98, January 1993.

[27] B. Wilkinson and M. Allen. *Parallel programming: techniques and applications using networked workstations and parallel computers*. Prentice-Hall, Inc., Upper Saddle River, NJ, USA, 1999.

[28] C.-T. Yang and S.-C. Chang. A parallel loop self-scheduling on extremely heterogeneous pc clusters. In *International Conference on Computational Science*, pages 1079–1088, 2003.

[29] C.-T. Yang, K.-W. Cheng, and W.-C. Shih. On development of an efficient parallel loop self-scheduling for grid computing environments. *Parallel Comput.*, 33(7-8):467–487, 2007.

[30] C.-T. Yang, W.-C. Shih, and S.-S. Tseng. A dynamic partitioning self-scheduling scheme for parallel loops on heterogeneous clusters. In *International Conference on Computational Science (1)*, pages 810–813, 2006.