Analisando a Predição de Desempenho com os Modelos Analíticos Gerados pela Metodologia PEMPIs-Het

Jean Marcos Laine[1,2], Edson Toshimi Midorikawa[1]
[1]Departamento de Engenharia de Computação e Sistemas Digitais
Escola Politécnica - Universidade de São Paulo
[2]Centro Universitário FIEO
Departamento de Informática
{jean.laine, edson.midorikawa}@poli.usp.br

Resumo

Com o crescente uso da computação distribuída em ambientes heterogêneos, principalmente para processamento de alto desempenho, fica cada vez mais evidente a necessidade do desenvolvimento de metodologias e ferramentas específicas para análise, avaliação e predição de desempenho nestes ambientes. Embora existam vários trabalhos que propõem metodologias para este propósito, pouco ainda tem sido feito em ambientes heterogêneos. Motivados pela importância desta linha de pesquisa na computação de alto desempenho, desenvolvemos uma metodologia chamada PEMPIs-Het que faz uso de meta-modelos analíticos para modelar e estimar o desempenho de aplicações paralelas MPI em sistemas como clusters e grids computacionais. Neste artigo, avaliamos a aplicabilidade das técnicas que compõem a metodologia e a precisão das estratégias, tanto na predição de desempenho quanto na distribuição de cargas computacionais. Os resultados experimentais obtidos comprovaram a eficácia da modelagem e de sua aplicação na distribuição de carga em um estudo de caso realizado.

1. Introdução

Nos últimos anos, os sistemas distribuídos heterogêneos, como *clusters* ou *grids* computacionais, têm se destacado no cenário da computação de alto desempenho, como uma alternativa viável para a execução de programas paralelos. O custo-benefício, a escalabilidade e a capacidade de processamento oferecida por estes sistemas são fundamentais para sua adoção frente às convencionais máquinas multiprocessadas. No entanto, pelo fato de cada máquina possuir sua própria memória local e, conseqüentemente, seu próprio espaço de endereçamento, não é possível para os processos trocarem informações através de variáveis de memória compartilhada. Assim, alguns paradigmas baseados em trocas de mensagens foram criados para permitir a comunicação distribuída. Atualmente, o MPI (*Message Passing Interface)* [10] é o padrão de comunicação mais utilizado no desenvolvimento de aplicações paralelas distribuídas. Outros aspectos devem ser considerados ao desenvolver aplicações paralelas para estes ambientes, dentre os quais citamos: balanceamento de carga, níveis de paralelismo, sincronização dos processos, latência da comunicação e taxa de utilização dos recursos.

Neste contexto, fica cada vez mais evidente a necessidade de metodologias e ferramentas específicas para análise, avaliação e predição de desempenho. A idéia foi criar uma metodologia capaz de auxiliar os programadores a decidirem qual paradigma ou modelo de programação é mais adequado, no seu sistema alvo, para a implementação de soluções paralelas e distribuídas. Direcionar as tomadas de decisão do desenvolvedor visando a otimização de problema é uma atividade que pode ser realizada através das técnicas de modelagem e avaliação de desempenho que a metodologia oferece.

Motivados pela importância desta linha de pesquisa [2, 4, 5, 13, 15, 17], formalizamos uma metodologia, chamada PEMPIs-Het [8], que permite não só modelar e estimar o desempenho das aplicações paralelas MPI mas também planejar a distribuição das tarefas e cargas computacionais em sistemas distribuídos heterogêneos.

O objetivo deste artigo é estender o trabalho anterior [8] e realizar uma análise detalhada sobre os modelos analíticos na atividade de predição de desempenho. Além disso, desejamos identificar os possíveis fatores que podem influenciar na precisão dos modelos e comprometer atividades como a distribuição de carga.

Para alcançarmos nosso objetivo, realizamos um estudo de caso com um programa que simula a interação de

partículas em um campo gravitacional. Os resultados obtidos com as atividades de predição de desempenho e com a distribuição da carga computacional são apresentados e discutidos detalhadamente.

As próximas seções deste artigo estão organizadas da seguinte forma. Na Seção 2, apresentamos alguns trabalhos relacionados, bem como as similaridades e diferenças em relação ao nosso. Uma descrição da metodologia PEMPIs-Het é dada na Seção 3. Na Seção 4, discutimos as estratégias adotadas pela metodologia. Os resultados experimentais são apresentados na Seção 5. Por fim, concluímos nosso trabalho e apresentamos alguns trabalhos futuros.

2. Trabalhos Relacionados

Uma metodologia para análise e predição de desempenho deve ser capaz de identificar os prováveis fatores que podem influenciar o desempenho das aplicações e, além disso, mostrar como estes fatores interagem. Neste intuito, alguns autores têm utilizado a modelagem analítica para alcançar este objetivo [3, 16, 1]. Com este enfoque, elaboramos modelos matemáticos capazes de representar o comportamento das aplicações em sistemas distribuídos heterogêneos. A idéia destes modelos é estimar o tempo de execução das aplicações não só em função do tamanho do problema mas da quantidade de processos também.

Em [9] os autores caracterizam o comportamento da aplicação em uma rede não dedicada de computadores heterogêneos. Para isso, são usadas curvas de desempenho baseadas em intervalos. A estratégia determina que o desempenho da aplicação pode ser estimado por um limite superior e outro inferior (intervalo de predição). No entanto, nenhum comentário é feito sobre a possibilidade de usar estes modelos para escalonamento de tarefas ou distribuição de cargas computacionais.

Em [15] os autores usam o conceito de desempenho relativo e taxa de desempenho para realizar uma distribuição dinâmica de cargas computacionais em aplicações que utilizam o modelo mestre-escravo e que são executadas em *grids* computacionais.

Nossa metodologia propõe uma forma de caracterizar aplicações MPI em ambientes distribuídos heterogêneos, como *clusters* heterogêneos ou *grids* computacionais. Com os modelos de predição, é possível analisar, avaliar e predizer o desempenho do programa e distribuir as cargas computacionais adequadamente.

3. PEMPIs-Het

O PEMPIs-Het (*Performance Estimation of MPI Programs in Heterogeneous Systems*) é uma extensão da metodologia PEMPIs [11], elaborada para analisar, avaliar e predizer o desempenho de programas paralelos MPI em sistemas homogêneos. Alguns artefatos foram criados para auxiliar a modelagem dos programas e melhorar a precisão dos modelos analíticos de desempenho, como por exemplo o DP*Graph$^+$ [11]. O DP*Graph$^+$ é uma representação gráfica das possíveis estruturas de um programa MPI que foi criada com o intuito de ilustrar a organização estática do código fonte do programa e facilitar o entendimento da aplicação no momento da análise de complexidade do algoritmo.

A nova metodologia, ilustrada na Figura 1, não só estende o PEMPIs para sistemas distribuídos heterogêneos mas adiciona e formaliza um conjunto de atividades inicialmente não suportadas pela metodologia original. Uma das atividades implementadas no PEMPIs-Het é o planejamento da distribuição de cargas computacionais. Para isso, é utilizado os modelos analíticos de desempenho. As estratégias definidas para esta atividade permitem duas abordagens distintas: estática e dinâmica.

A versão estática, baseada no algoritmo *Round-Robin*, é indicada para ambientes distribuídos homogêneos e dedicados. Já as técnicas definidas para a versão dinâmica permitem a aplicação da estratégia em sistemas distribuídos heterogêneos e dinâmicos, como os *grids* computacionais, *clusters* heterogêneos ou até mesmo redes de computadores heterogêneos.

Para a distribuição das cargas computacionais foi implementado um componente, chamado PWD (***P**erformance Estimation and **W**orkload **D**istribution*), que utiliza os modelos analíticos, gerados pelo módulo AME (***A**pplication **M**odeling **E**nvironment*), para elaborar um conjunto de relações de desempenho aplicáveis no planejamento da distribuição das cargas computacionais. Estas relações caracterizam o desempenho relativo das máquinas que compõem o sistema.

Uma outra contribuição da metodologia PEMPIs-Het é a definição de técnicas para o desenvolvimento de meta-modelos de predição de desempenho. Inicialmente, a aplicação MPI é modelada em cada tipo de máquina que compõe o sistema. Os modelos gerados são armazenados em um repositório de dados e catalogados juntos com as características da máquina alvo, como por exemplo: quantidade de processadores ou núcleos, quantidade de memória cache e principal, interface de rede, etc. A partir dos modelos individuais, é possível gerar um meta-modelo que estima o comportamento da aplicação em uma combinação de máquinas do sistema.

Outro elemento adicionado à metodologia foi o MWD (***M**iddleware for **W**orkload **D**istribution*). Este *middleware* facilita a aplicação dos índices de desempenho gerados pelo PWD na elaboração de um plano inicial de distribuição das cargas computacionais, que tem como objetivo otimizar os recursos disponíveis no sistema e melhorar o desempenho

das aplicações. Uma possível aplicação para este componente será a sua integração com um escalonador de tarefas para *grids* computacionais que vem sendo desenvolvido em nosso grupo de pesquisa[1].

Um componente ainda não implementado, mas previsto na metodologia é o MON (***MON****itor)*. Este componente será responsável por monitorar, em tempo de execução, o desempenho da aplicação em cada uma das máquinas e avaliar se o tempo estimado no planejamento inicial está ou não de acordo com a situação real de execução. Estas informações voltarão para o MWD que poderá, dinamicamente, modificar a distribuição das cargas computacionais e ajustar o plano inicial às possíveis alterações do sistema.

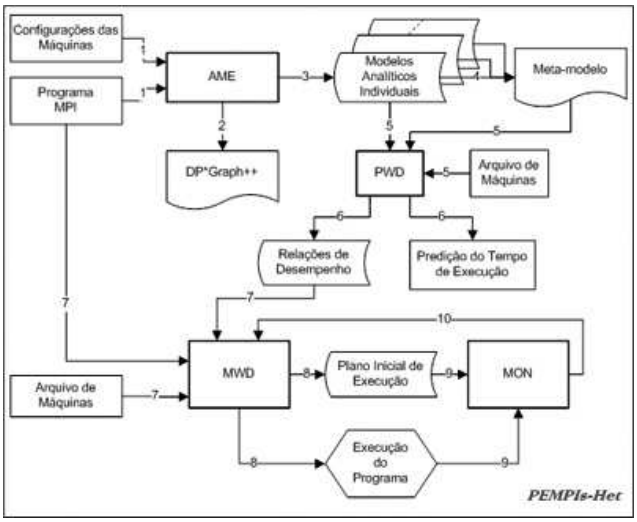

Figura 1. Metodologia PEMPIs-Het.

3.1 DP*Graph^{++}

Até o trabalho apresentado em [11], representávamos a estrutura estática das aplicações MPI através da notação do DP*Graph^{+}. Com este modelo gráfico é possível analisar não só a organização do programa mas os pontos de paralelismo e eventuais sincronismos entre os processos que executam a tarefa. No entanto, ao modelar as aplicações em ambientes heterogêneos, sentimos a necessidade de representar as estruturas cujo comportamento e/ou desempenho pode ser influenciado pela heterogeneidade do sistema. Portanto, o antigo conjunto de símbolos do DP*Graph^{+} teve que ser estendido a fim de suprir esta necessidade. Assim, criamos o DP*Graph^{++}, cujos símbolos principais são apresentados na Figura 2.

Basicamente, diferenciamos os antigos símbolos dos atuais mudando as cores de representação dos elementos. Para

[1]LAHPC - Laboratory of Architecture and High Performance Computing (POLI-USP)

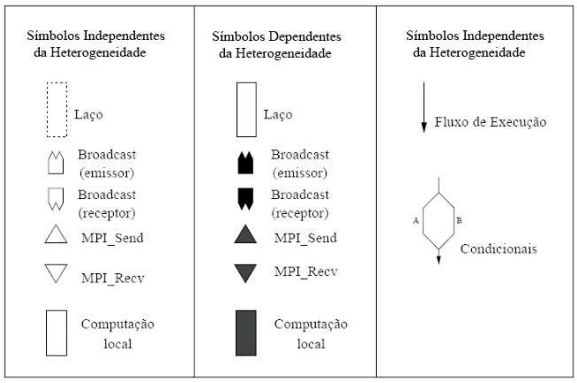

Figura 2. DP*Graph^{++}: principais símbolos.

alguns símbolos, como fluxo de execução e condicionais, não existe a necessidade de dupla representação. Um exemplo simplificado do DP*Graph^{++} é dado na Figura 3. Nesta ilustração temos um processo mestre e dois escravos colaborando na solução do problema.

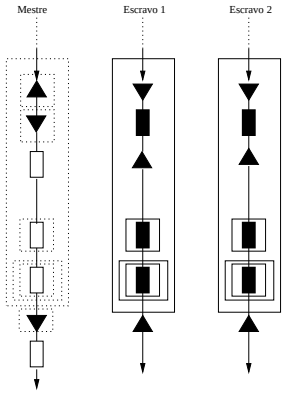

Figura 3. Exemplo do DP*Graph^{++}.

3.2 Modelo das Aplicações

Um modelo analítico é uma equação matemática capaz de representar o comportamento do programa em um determinado sistema. Em nosso trabalho, os modelos permitem representar o tempo de execução de programas MPI em sistemas distribuídos homogêneos e heterogêneos. Contudo, neste artigo estamos interessados em apresentar uma análise da aplicabilidade das estratégias em sistemas heterogêneos.

Antes de elaborar os modelos de predição, analisamos a complexidade do programa e determinamos o modelo teórico de desempenho. Por exemplo, se uma aplicação apresenta uma complexidade algorítmica $O(\frac{n^3}{p})$, o modelo analítico poderá ser expresso através do seguinte polinômio:

$$\delta(p,n) = \frac{an^3}{p} + \frac{bn^2}{p} + \frac{cn}{p} + d \qquad (1)$$

onde p representa a quantidade de processos e n o tamanho total do problema.

O modelo de predição de uma aplicação MPI deve ser capaz de representar as seguintes estruturas:

$$T_{exe} = \Sigma t_{cpu} + \Sigma t_{com} + \alpha \quad (2)$$

onde Σt_{cpu} determina o tempo total gasto no processamento de instruções, Σt_{com} representa o tempo total gasto em comunicações e α caracteriza a influência de fatores não modelados. Dentre estes fatores podemos citar eventuais sincronismos, contenções e atrasos gerados na execução dos processos e possíveis retransmissões de mensagens de comunicação. Considerar estes aspectos podem aumentar muito a complexidade dos modelos de predição e prejudicar sua aplicabilidade.

4. Apresentando as Estratégias

Na metodologia PEMPIs-Het, especificamos algumas técnicas que permitem não só estimar o desempenho das aplicações MPI mas distribuir a carga computacional do sistema.

4.1 Distribuindo Cargas

Em sistemas distribuídos, principalmente os heterogêneos, a distribuição adequada da carga computacional é fundamental para que o sistema apresente um bom desempenho. A existência de máquinas com grande capacidade de processamento é insuficiente para garantir o desempenho dos programas paralelos. Portanto, a capacidade de processamento das máquinas é tão importante quanto o planejamento adequado da divisão e distribuição da carga computacional.

Antes de apresentarmos como os modelos de predição são aplicados na tarefa de distribuição das cargas computacionais é necessário definirmos algumas relações que fundamentam esta atividade.

Definição 1: Seja $P = \{p_1, p_2, ..., p_n\}$ o conjunto que caracteriza o número de máquinas diferentes em um sistema heterogêneo S e f_i uma porcentagem do trabalho total que será executado pela máquina do tipo i. A relação que define a distribuição de carga entre as n máquinas pode ser dada pela seguinte expressão:

$$\sum_{i=1}^{n} p_i \times f_i = 1 \quad (3)$$

Esta relação mostra que a somatória total dos trabalhos realizados por cada uma das máquinas do conjunto P equivale a 100% do trabalho total. A partir desta relação é possível derivar as seguintes expressões para cada f_i:

$$f_1 = \frac{1 - p_2 \times f_2 - p_3 \times f_3 - ... - p_i \times f_i}{p_1} \quad (4)$$

$$f_2 = \frac{1 - p_1 \times f_1 - p_3 \times f_3 - ... - p_i \times f_i}{p_2} \quad (5)$$

$$\vdots$$

$$f_i = \frac{1 - p_1 \times f_1 - p_2 \times f_2 - ... - p_{i-1} \times f_{i-1}}{p_i} \quad (6)$$

Uma consideração importante, neste momento, diz respeito ao *speedup* teórico, considerado no desenvolvimento das equações. Para simplificarmos as análises e formulações realizadas consideramos um *speedup* é linear para nossas aproximações. Embora algumas aplicações possam não apresentar este comportamento, avaliaremos sempre o caso ideal. Dessa forma, o tempo de execução de uma aplicação A, manipulando um problema de tamanho n e sendo executada em um sistema S com p máquinas idênticas, pode ser expresso através da seguinte equação:

$$\delta^A(p, n) = \frac{1}{p} \times \delta^A(1, n) \quad (7)$$

Esta análise é feita para cada grupo de máquinas idênticas que constituem o sistema. Logo, o tempo de execução da aplicação A, em cada máquina do tipo i, pode ser determinado como:

$$T_i^A(f_i, n) = f_i \times \delta_i^A(1, n) \quad (8)$$

Com isso, podemos determinar a fração de trabalho f_i que cada processo deverá receber no processamento da aplicação. Nosso objetivo é equilibrar a carga dos nós computacionais de modo que a seguinte relação seja satisfeita:

$$f_i \times \delta_i^A(p, n) = f_j \times \delta_j^A(p, n), \quad \forall \quad i, j \in P \quad (9)$$

Teoricamente, esta igualdade representa o caso ideal. Na prática, o objetivo é minimizar ao máximo as diferenças entre os tempos de execução de cada um dos processos e, com isso, sincronizar a execução das tarefas eliminando eventuais tempos de espera.

Definição 2: Seja $\delta_i^A(p, n)$ o modelo de desempenho que estima o tempo de execução de uma aplicação A em p máquinas do tipo i. Considere também, $\delta_j^A(p, n)$ como sendo a estimativa do tempo de execução, da mesma aplicação A, em p máquinas do tipo j. A relação de desempenho g, que caracteriza a capacidade relativa das máquinas em função do tamanho do problema n, pode ser definida como:

$$g_{i,j}(n) \triangleq \frac{\delta_i^A(p, n)}{\delta_j^A(p, n)} \quad (10)$$

A relação $g_{i,j}(n)$ determina quantas vezes a máquina i é mais rápida em relação a máquina j ao executar uma mesma aplicação A com uma carga computacional n.

Usando as equações de 5 a 6, 9 e 10 é possível determinar os respectivos valores para cada f_i, como apresentado a seguir:

$$f_1 = \frac{1}{p_1 + p_2 g_{1,2}(n) + ... + p_m g_{1,m}(n)} \quad (11)$$

$$f_2 = \frac{g_{1,2}(n)}{p_1 + p_2 g_{1,2}(n) + ... + p_m g_{1,m}(n)} \quad (12)$$

$$\vdots$$

$$f_i = \frac{g_{1,i}(n)}{p_1 + p_2 g_{1,2}(n) + ... + p_i g_{1,i}(n)} \quad (13)$$

Com todas estas informações é possível utilizar as equações e estimar o tempo de execução da aplicação MPI.

4.2 Estimando Desempenho

O desempenho de uma aplicação pode ser medido através de diferentes métricas, como por exemplo: o tempo de execução, o tempo de resposta, a quantidade de requisições tratadas, etc. Com os modelos analíticos elaborados pelo PEMPIs-Het, a métrica avaliada nos estudos de desempenho é o tempo de execução do programa.

Para obter uma estimativa do tempo de execução individual, de cada um dos processos MPI, é necessário aplicar uma das equações identificadas de 11 à 13 na equação 8. Após a substituição, o tempo de execução de um processo p, ao executar uma aplicação A em uma máquina i do ambiente, é dado pela seguinte equação:

$$T_i^A(f_i, n) = \frac{g_{1,i}(n)}{p_1 + p_2 g_{1,2}(n) + ... + p_m g_{1,m}(n)} \times \delta_i^A(1, n) \quad (14)$$

Esta equação leva em consideração não só o modelo de desempenho elaborado para a aplicação A na máquina do tipo i mas também a carga computacional que será processada neste nó. É possível generalizar, com esta equação, a estimativa dos tempos individuais de cada processo, já que a relação de desempenho expressa em 9 foi assumida no desenvolvimento do formalismo.

Em geral, os programas MPI usam a abordagem mestre-escravo na organização da solução. Nesta abordagem, o tempo total de execução do programa é fortemente dependente do processo mais lento. Isto reforça a importância de uma boa estratégia de distribuição de carga.

4.3 Descrição do Ambiente de Teste

Os testes experimentais foram realizados em um ambiente computacional composto por três tipos de máquinas: intel, bio e taurus. As máquinas intel possuem um processador dual-core Intel Pentium D 950, 2GB de DDR2 SDRAM, duas interface de rede gigabit Ethernet. As máquinas bio possuem dois processadores AMD Athlon MP 2400$^+$, com 1GB de DDR SDRAM, duas interfaces de rede Intel Ether-Express Pro Fast Ethernet. As máquinas taurus possuem um Intel Celeron 433MHz, 256 MB de SDRAM, interface de rede Fast Ethernet e roda RedHat Linux. A distribuição MPI utilizada foi o LAM-MPI. O sistema operacional instalado em todas as máquinas é o Fedora Core 6. Durante os testes experimentais tentamos manter a mesma carga computacional nas máquinas do sistema.

5. Resultados Experimentais

Para avaliar a aplicabilidade das estratégias contempladas pela metodologia PEMPIs-Het, utilizamos como caso de uso uma aplicação similar ao bem conhecido problema dos *n-corpos*. A aplicação (PGF) simula a interação de N partículas em um campo de forças gravitacionais. Cada uma das partículas é caracterizada por sua massa, sua posição e seu momento. O algoritmo adotado na solução utiliza o método denominado partícula-partícula (PP), com complexidade O(n^2) para calcular as forças de interação entre as partículas [12].

A paralelização da aplicação é baseada no particionamento da computação entre os nós de processamento através da distribuição dos dados: cada nó recebe um conjunto de partículas para processar. Uma estrutura de dados mantendo o estado atual das partículas é mantido pelo processo mestre e enviado via *broadcast* para os escravos no começo de cada etapa da computação. Assim, cada escravo deve se comunicar com o mestre para enviar o estado de suas partículas antes de cada nova iteração.

Para gerar os modelos de predição elaboramos um conjunto de casos de testes, variando a quantidade de partículas e o número de processos escravos. Para cada configuração de teste, repetimos 40 vezes a execução do programa no ambiente, a fim de proporcionar maior confiabilidade aos valores medidos. Com os resultados, selecionamos os dados e descartamos as anomalias. Essas anomalias, conforme explicado em [6], pode resultar em decorrência de uma saturação momentânea da rede de interconexão ou de uma instabilidade do sistema operacional. Observações incompatíveis com a maioria dos resultados podem ser caracterizadas como anomalias [6].

Com os dados selecionados, aplicamos as técnicas descritas e aproximamos o comportamento do programa para o modelo teórico da aplicação. Assim, o tempo de execução da aplicação pode ser estimado em função do número de processos (p) e da quantidade de partículas (n). Durante os testes experimentais utilizamos 18 máquinas, sendo 4 do tipo intel, 6 bios e 8 taurus.

Após gerar os modelos para cada tipo de máquina, calculamos para cada quantidade de partícula os valores para as relações $g_{t,b}$ e $g_{t,i}$, instanciando $p = 1$. Dessa forma, de-

Tabela 1. Resultados completos.

	Número de Partículas			
	20.000	**40.000**	**50.000**	
Tempo mínimo medido	31,44	153,85	196,03	**Estratégia Estática**
Tempo máximo medido	34,21	174,75	232,14	
$\delta(p,n)$ **(mín. estimado)**	33,90	129,57	200,88	
$\delta + \Delta$ **(máx. estimado)**	34,21	174,75	232,14	
Δ (erro)	0,31	45,18	31,26	
$g_{t,b}$	4,2			
$g_{t,i}$	8,1			
f	0,123	0,123	0,123	
Tempo total	41,27	191,89	251,65	
Tempo mínimo medido	32,53	127,46	198,06	**Estratégia Dinâmica**
Tempo máximo medido	33,17	129,95	201,33	
$\delta(p,n)$ **(mín. estimado)**	32,54	127,87	198,25	
$\delta + \Delta$ **(máx. estimado)**	33,17	129,54	201,14	
Δ (erro)	0,63	2,08	3,08	
$g_{t,b}$	4,25	4,24	4,24	
$g_{t,i}$	7,55	7,95	7,95	
f	0,119	0,122	0,122	
Tempo total	40,15	143,35	219,52	

terminamos a capacidade relativa de processamento de cada uma das máquinas utilizadas nos testes experimentais. Os valores calculados para estas relações e as frações de trabalho f, utilizada nas predições de desempenho, são apresentados na Tabela 1.

5.1 Distribuindo a Carga Computacional

A estratégia para a distribuição da carga computacional, definida pela metodologia PEMPIs-Het, possibilita duas diferentes abordagens de aplicação: estática e dinâmica. Na abordagem estática, é elaborado um plano de execução inicial e os valores determinados para cada f_i permanecem os mesmos, independente do tamanho do problema, como mostrado na Tabela 1. Todo o trabalho é distribuído aos processos no início da execução, seguindo o modelo *Round-Robin*. Já a estratégia de distribuição de carga dinâmica calcula os valores para os f_i em função do tamanho do problema n. Esta segunda abordagem é mais precisa, pois o desempenho relativo das máquinas é alterado à medida que modificamos a carga de trabalho de cada nó (Tabela 1). Se este desempenho relativo não se alterasse, a estratégia estática seria tão precisa quanto a dinâmica.

A figura 4 apresenta o tempo de execução estimado pelo modelo de predição e os valores medidos para os processos que demoraram mais e menos tempo para processar sua carga, utilizando a versão estática de distribuição. Podemos observar que as relações de desempenho entre as máquinas se modificam ao alterar a carga de processamento, comprometendo a estratégia estática.

Conforme podemos ver no gráfico da Figura 5, a versão dinâmica gerou melhores resultados no balanceamento das cargas. Este equilíbrio também permitiu reduzir o tempo total de execução da aplicação, como apresentado na Tabela 1. A redução no tempo de execução, em até 14% no caso de 50.000 partículas, é uma conseqüência do melhor aproveitamento dos recursos computacionais. Como a aplicação faz uso do modelo mestre-escravo para organizar o processamento, o tempo de execução da aplicação é dependente do tempo do escravo mais lento. Minimizando as diferenças entre os tempos, todos os processos finalizam suas tarefas quase ao mesmo tempo e pouco tempo se gasta em esperas.

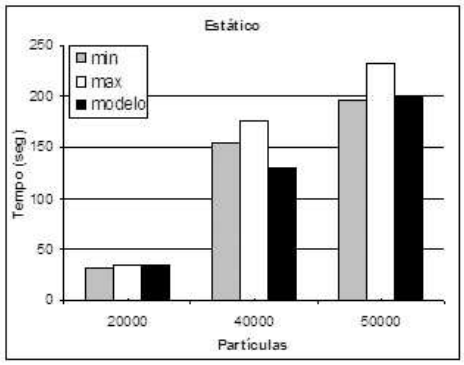

Figura 4. Distribuição estática.

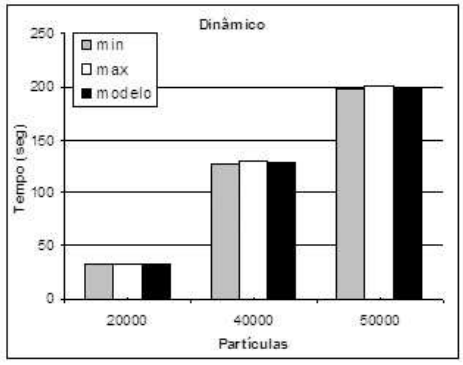

Figura 5. Distribuição dinâmica.

5.2 Estimando o Desempenho

A forma como os modelos analíticos são elaborados permite caracterizar o comportamento da aplicação MPI em cada tipo de máquina do sistema. Portanto, os aspectos modelados não estão associados somente às características da aplicação (*software*). O modelo também contempla a influência dos elementos de *hardware* no desempenho do programa. Para gerar os modelos de predição de desempenho é necessário cumprir as seguintes etapas [11]:

- elaborar o modelo teórico de desempenho, avaliando a complexidade do código fonte do programa que será modelado;

- projetar os casos de testes e inserir os monitores de tempo no código do programa;
- repetir os testes experimentais, variando o tamanho do problema (n) e a quantidade de processos ou nós (p);
- selecionar os tempos medidos e descartar as anomalias;
- calcular um tempo médio para cada trecho do programa avaliado;
- aplicar um método de ajuste de curvas sobre os dados experimentais. Na modelagem realizada pelo PEMPIs-Het é utilizado o método dos mínimos quadrados. Inicialmente, a modelagem é feita em função do n e depois em função do p.

Para representar o comportamento dos processos que participam da simulação do problema foi elaborado o seguinte modelo de predição do desempenho:

$$\delta^{PGF}(p,n) = (7,57 \times 10^{-9} + \frac{6,26 \times 10^{-7}}{p})n^2 + (-5,78 \times 10^{-4} + \frac{1.3 \times 10^{-3}}{p})n + (-1,99 + \frac{8,74}{p}) \quad (15)$$

A precisão da estimativa de desempenho da aplicação está diretamente relacionada a forma como os modelos analíticos são elaborados. Pelo fato da metodologia PEMPIs-Het aplicar estes modelos nas atividades de distribuição de cargas computacionais e predição de desempenho, a consistência e exatidão dos modelos é fundamental para o êxito das tarefas que nos propomos a desenvolver.

Como podemos observar na Tabela 1, o valor estimado pelo modelo de predição pode ser aproximado para o valor mínimo do tempo de execução. Este fato pode ser explicado através da seguinte análise. Primeiro, o tempo é estimado com base em uma distribuição de carga ideal para o sistema. No entanto, nem sempre é possível associar a cada processo exatamente o que determina o plano de distribuição elaborado pelo componente MWD (Figura 1). Por exemplo, na aplicação PGF devemos sempre atribuir a cada processo escravo um número inteiro de partículas e, dependendo da quantidade de processos e da quantidade de partículas, o cálculo ideal pode não resultar um número inteiro. Assim, por questões de arredondamento, os processos escravos podem receber uma quantidade de trabalho que não produz o balanceamento ideal previsto. Segundo, a aplicação pode sofrer interferências do ambiente devido a algumas situações, como mudanças na carga computacional do sistema, tráfego excessivo na rede, problemas no *hardware* que modifica a capacidade de processamento da máquina.

Tabela 2. Estimativas de desempenho.

Partículas	80.000	100.000	150.000
Mínimo (δ)	507,42	791,46	1776,84
Máximo ($\delta + \Delta$)	517,42	807,00	1811,53

Utilizando Intervalos

Além das razões comentadas, imprecisões no modelo de predição também pode contribuir para as diferenças relatadas entre o tempo estimado e o medido durante os testes. Uma outra estratégia para estimar desempenho das aplicações consiste em utilizar intervalos de predição [14, 9].

Nesta abordagem, é estimado um valor mínimo e outro máximo para o tempo de execução do programa. Assim, a predição é feita para um intervalo mínimo e máximo, do tipo: $[\delta - \Delta, \delta + \Delta]$, sendo δ o tempo estimado pelo modelo de desempenho e Δ um valor aceitável para o erro máximo das estimativas. Com essa abordagem, o que se pretende é garantir que o tempo de execução da grande maioria dos processos se mantenham no intervalo estipulado.

Para adaptar nossa estratégia para esta abordagem, decidimos estudar o comportamento dos valores encontrados para o DFPT - *Distance From Prediction Time* (figura 6) e modelá-los. O procedimento seguido para encontrar os coeficientes do modelo foi idêntico ao utilizado para elaborar o modelo de desempenho dos processos. A partir das análises realizadas, a seguinte expressão para Δ foi encontrada:

$$\Delta(n) = 1,514 \times 10^{-9}n^2 + 4,5 \times 10^{-6}n - 4,486 \times 10^{-2} \quad (16)$$

Esta equação representa o erro máximo para as estimativas. Como o tempo de execução previsto pelo modelo analítico (δ) é uma estimativa para a distribuição ideal das cargas computacionais, o valor calculado pelo modelo pode ser definido como o valor mínimo. Desse modo, o intervalo de predição para a aplicação PGF pode ser representado da seguinte forma: $\left[\delta^{PGF}(p,n),\ \delta^{PGF}(p,n) + \Delta(n)\right]$.

Assim, os tempos de execução de cada processo podem ser estimados como mostrado na tabela 2. Os valores calculados para o Δ são apresentados na tabela 1.

O gráfico da Figura 6 apresenta a distância dos tempos de execução de cada um dos processos escravos em relação ao tempo estimado pelo modelo (DFPT). Neste gráfico é possível verificar a precisão do modelo de predição elaborado, uma vez que a maior diferença percentual entre o valor estimado e medido não passou de 2%. Outro fato interessante para ser comentado diz respeito ao comportamento dos tempos de execução dos processos escravos. Comentamos anteriormente que o modelo prevê uma distribuição ideal mas que os processos podem sofrer interferências do ambiente capazes de prejudicar o desempenho do mesmo. Isto é confirmado nos resultados apresentados, pois quase todos os escravos demoraram mais tempo para finalizar do

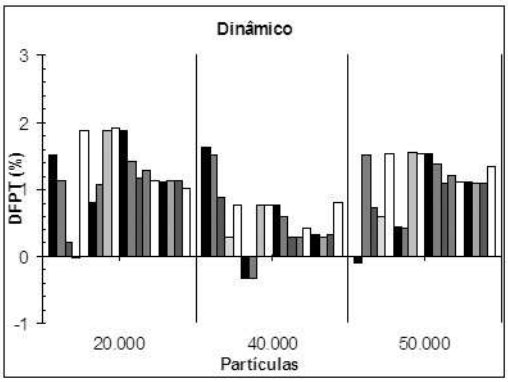

Figura 6. Distância entre os tempos medidos e o estimado pelo modelo.

que o previsto pelo modelo. Os casos onde o tempo de execução foi menor que o estimado também pode ser justificado. Pelo fato da estratégia utilizar a função teto [8] no arredondamento da divisão das cargas, é possível que alguns processos de um mesmo grupo de máquinas recebam uma quantidade de trabalho menor que os demais.

6 Conclusão

Neste artigo apresentamos um estudo sobre a aplicabilidade de algumas estratégias que a metodologia PEMPIs-Het contempla. Mostramos que os modelos analíticos de desempenho podem ser utilizados tanto em tarefas de distribuição de cargas computacionais quanto em atividades de predição de desempenho.

Duas estratégias de predição de desempenho foram apresentadas. A primeira abordagem produz uma estimativa do tempo de execução e a segunda considera um intervalo aceitável para valores mínimos e máximos da estimativa. O uso de uma ou outra abordagem pode estar relacionado aos interesses da predição. Por exemplo, se queremos utilizar estas predições em escalonadores de processos, a estratégia baseada em intervalos pode ser mais adequada, pois uma estimativa exata pode não ser necessária. Os bons resultados apresentados pelos modelos de predição e pelas estratégias de distribuição de carga confirmam a eficiência das estratégias que a metodologia descreve. Estamos trabalhando na implementação do componente MON e testando nossas estratégias em ambientes dinâmicos.

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] J. Cao, D. P. Spooner, S. A. Jarvis, S. Saini, and G. R. Nudd. Application performance prediction for the management and scheduling of clusters and grids. Technical Report Project Report, University of Warwick, United Kingdom, 2002.

[3] M. E. Crovella. *Performance prediction and tuning of parallel programs*. PhD thesis, Rochester, NY, USA, 1994.

[4] N. Drosinos and N. Koziris. The effect of process topology and load balancing on parallel programming models for smp clusters and iterative algorithms. *J. Supercomput.*, 35(1):65–91, 2006.

[5] 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.

[6] R. Jain. *The Art of Computer Systems Performance Analysis*. Wiley & sons, 1991.

[7] J. M. Laine. *Uma Metodologia para Desenvolvimento de Programas Paralelos Heterogêneos Eficientes*. PhD thesis, Universidade de São Paulo, 2007. Em preparação.

[8] J. M. Laine and E. T. Midorikawa. Using analytical models to load balancing in a heterogeneous network of computers. In *MTPP'07 - Methods and Tools of Parallel Programming of Multicomputers*, Pereslav'-Zalessky, Russia, 2007.

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

[10] Message Passing Interface Forum. MPI: A Message Passing Interface. In *Proceedings of Supercomputing '93*, pages 878–883. IEEE Computer Society Press, 1993.

[11] 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.

[12] G. M. Petrov and J. Davis. Modeling of clusters by a molecular dynamics model using a fast tree method. *The European Physical Journal D - Atomic, Molecular, Optical and Plasma Physics*, 41(3):629–639, 2006.

[13] D. Sánchez, E. M. Macías, and Á. Suárez. An application level load balancing mechanism for heterogeneous clusters programming. In *Proceedings of the International Conference on Parallel and Distributed Processing Techniques and Applications (PDPTA'02)*, pages 872–878, june 2002.

[14] J. M. Schopf and F. Berman. Performance prediction using intervals. Technical Report CS97-541, Department of Computer Science and Engineering, University of California San Diego, May 1997.

[15] 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.

[16] 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.

[17] 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.