Uma Ferramenta para Predição Analítica de Desempenho de Aplicações Java Paralelas

Roberto Hirochi Herai Marco Aurélio Amaral Henriques
Faculdade de Engenharia Elétrica e de Computação - UNICAMP
CP. 6101, Campinas, São Paulo, Brasil, 13083-970
[rherai|marco] @dca.fee.unicamp.br

Resumo

A predição de desempenho é um importante mecanismo para avaliar a utilização de recursos e estimar o tempo de execução de aplicações em sistemas paralelos. Este trabalho apresenta uma ferramenta, chamada APET, que permite criar um modelo analítico de desempenho a partir de modelos que representam características da aplicação e dos computadores utilizados para processá-la. O modelo de desempenho permite contemplar em suas estimativas fatores de atraso que influenciam no tempo de execução de uma aplicação, tais como operacões aritméticas sobre diferentes tipos de dados e efeitos de contenção, tanto interna quanto externa. As estimativas podem ser geradas em poucos segundos e permitem analisar o impacto causado na aplicação pela utilização de diferentes configurações do sistema de processamento paralelo.

1. Introdução e trabalhos relacionados

Ferramentas de predição de desempenho permitem estimar o tempo de execução de uma aplicação antes mesmo de executá-la nos computadores de um sistema paralelo [1]. Para realizar estimativas de tempo uma ferramenta de predição deve adotar alguma técnica que forneça bons resultados e de forma rápida [2]. Além disso, é necessária alguma técnica que permita analisar efeitos de contenção no uso de recursos compartilhados quando estes são limitados. Se a aplicação estiver sendo executada em um ambiente não dedicado, ela poderá ter que compartilhar a utilização do processador com outros processos que estão em execução. Este tipo de contenção, chamada de contenção externa por Yero [3], é muito difícil de ser modelada. Outro tipo de contenção é a interna, que pode ser modelada de forma mais eficiente e simplificada, pois quem induz a contenção é o próprio sistema de processamento paralelo ou a aplicação executada por ele.

Técnicas de predição analíticas são uma boa alternativa para realizar estimativas, as quais consideram os efeitos de atraso causados pela contenção (interna e externa) na utilização de recursos. Baseadas em informações estáticas do sistema paralelo e da aplicação, elas armazenam tais informações na forma de modelos manipuláveis que permitem determinar a sensibilidade de um algoritmo às características de um conjunto de computadores que farão seu processamento. A análise de tais modelos permite estimar o tempo de execução de uma aplicação com boa precisão, de forma rápida e a um custo baixo, pois não é necessária a utilização dos computadores ou de outros recursos do sistema paralelo para a geração das estimativas de tempo. Um fator limitante é que o desenvolvimento de ferramentas que utilizam tais técnicas é bastante complexo, pois requer uma análise minuciosa do código da aplicação e das características do sistema de processamento paralelo.

Algumas propostas de ferramentas de predição baseadas em técnicas analíticas abordam o problema da criação dos modelos dividindo-o em duas partes: modelo da aplicação e modelo das máquinas. Um dos problemas envolvidos na geração do modelo da aplicação é a necessidade da inserção de marcadores explícitos no código-fonte da mesma. Ferramentas como PACE [4] e PAMELA [5] exigem que isso seja feito. As aplicações para PACE podem ser escritas em Fortran (77 ou 90) ou C. PAMELA propõe a criação de uma nova linguagem de programação, chamada Spar [6], que procura modelar efeitos de atraso causados pela contenção interna no uso de recursos. A ferramenta PerPreT [7] também propõe o uso de uma nova linguagem de programação, chamada LOOP, que limita as aplicações àquelas descritas pelo modelo SPMD (*Single Program Multiple Data*). Porém, PerPreT não exige marcação explícita no código.

O processo de geração do modelo das máquinas para cada ferramenta mencionada é baseado na execução de algoritmos de *benchmark* (algoritmos de testes) no sistema paralelo, com o intuito de avaliar o desempenho da rede e dos computadores que o compõem.

Em função do custo envolvido na geração de predições e da necessidade de ferramentas automatizadas que minimizem os esforços para realizar predições, este trabalho propõe uma nova ferramenta de predição utilizando uma abordagem de predição analítica. A ferramenta proposta, APET (*Analytical Performance Estimation Tool*), utiliza a técnica de predição analítica PAMELA Estendido [3] para a geração de estimativas de tempo. Ela complementa o trabalho apresentado por Herai et al. [8], no qual foram propostas ferramentas para a construção dos modelos das máquinas e da aplicação, para um sistema de processamento paralelo com características similares ao sistema JoiN [9].

Este sistema de predição visa orientar os usuários sobre o provável custo de executarem suas aplicações em uma dada configuração do sistema paralelo a ser utilizado.

A próxima seção faz uma revisão simplificada da técnica PAMELA Estendido utilizada, e das ferramentas que foram apresentadas por Herai et al. [8]. As demais seções descrevem a ferramenta proposta em detalhes e apresentam os resultados iniciais obtidos com a mesma.

2. Conceitos básicos

2.1. Técnica de modelagem e predição de desempenho

PAMELA Estendido, proposto por Yero [3], é uma técnica que pode ser utilizada para a construção de modelos e realizar predições. Ela permite criar um modelo analítico de desempenho a partir dos modelos das máquinas e da aplicação, para extrair estimativas de tempo.

Para modelar efeitos de contenção externa, é feito o uso da técnica de intervalos [10], na qual os parâmetros do modelo das máquinas são representados na forma de intervalos de valores do tipo $[t_{min}, t_{max}]$, os quais representam os tempos mínimo e máximo esperados para a execução de uma operação.

O modelo das máquinas é definido pelos parâmetros:
$op\_data_p = t\_op\_data_p$
e
$send_{p,q}(dataSize) = t\_send_{p,q}(dataSize)$.
O parâmetro $t\_op\_data_p$ é um intervalo de valores que representa o tempo gasto na execução de operações *op* sobre determinados tipos de dados *data* no computador p, $0 \leq p \leq P$, sendo $P+1$ o número total de computadores. A operação $t\_send_{p,q}(dataSize)$ indica o tempo gasto na execução da operação $send_{p,q}$ para o envio de dados de tamanho $dataSize$ do computador p para o q.

Para a obtenção de estimativas de tempo, o modelo analítico de desempenho pode ser utilizado com diferentes configurações de computadores e de rede. Isso é possível porque o modelo é composto por diversas expressões matemáticas com vários parâmetros. Sua construção é baseada na análise do modelo da aplicação considerando ausência e presença de contenção interna.

Na ausência de contenção, é produzida uma expressão parcial, chamada T_0, que equivale à parte do bloco paralelo que utiliza um recurso por mais tempo. A utilização de um recurso de forma paralela é representada no modelo pelas instruções paralelas *par* ou ||. Por exemplo, instruções paralelas do tipo $par(j = 1, n)Op_j$ são substituídas pela operação $max_{j=1,n}(tempo\_Op_j)$ e as instruções do tipo $a||b$ são substituídas por $max(tempo\_a, tempo\_b)$.

Na presença de contenção, o tempo gasto na utilização de um recurso de forma paralela equivale ao somatório dos tempos gastos em cada um dos blocos que utilizam o recurso de forma paralela, representado com o uso do operador de somatório *sum*. Para este caso, PAMELA considera a carga imposta em cada recurso separadamente, produzindo uma expressão parcial T_r para cada recurso r ($r = 1..R$, onde R é a quantidade de recursos). Por exemplo, instruções paralelas do tipo $par(j = 1, n)Op_j$ são substituídas pela operação $sum_{j=1,n}(tempo\_Op_j)$ e as instruções do tipo $a||b$ são substituídas por $tempo\_a + tempo\_b$.

Em ambos os casos, a utilização de um recurso de forma sequencial, representado no modelo da aplicação com o uso das instruções sequenciais *seq* e ";", implica em que o tempo gasto na sua utilização equivale à soma dos tempos dos blocos que utilizam o recurso de forma sequencial. Por exemplo, instruções sequenciais do tipo $seq(j = 1, n)Op_j$ são substituídas pela operação de somatório $sum_{j=1,n}(tempo\_Op_j)$ e as instruções do tipo $a; b$ são substituídas por $tempo\_a + tempo\_b$.

Para a geração das expressões do modelo analítico de desempenho, os parâmetros op_data_p e $send_{p,q}(dataSize)$ presentes no modelo da aplicação são convertidos em suas representações de atraso de tempo na forma simbólica, equivalentes aos parâmetros $t\_op\_data_p$ e $t\_send_{p,q}(dataSize)$ que compõem o modelo das máquinas.

Após a geração das expressões, elas são representadas em função da expressão final $T = max_{r=0,R}(T_r)$, que fornece uma estimativa de tempo para a aplicação considerada. Para avaliar a expressão T e obter uma estimativa de tempo, os parâmetros do tipo $t\_op\_data_p$ e $t\_send_{p,q}(dataSize)$ das expressões parciais são substituídos pelos seus respectivos intervalos numéricos de valores $[t_{min}, t_{max}]$ presentes no modelo das máquinas. O resultado da avaliação representa o intervalo de tempo esperado para a execução da aplicação nos computadores do sistema paralelo considerado.

2.2. O sistema JoiN

O sistema paralelo JoiN é um sistema composto por tipos distintos de componentes, no qual os nós coordenadores e

trabalhadores são responsáveis pelo controle e execução das aplicações submetidas ao sistema, respectivamente [9].

As aplicações desenvolvidas para tal sistema paralelo não se restringem ao modelo SPMD e sua construção é dividida em duas fases. Na primeira define-se com o uso da linguagem PASL (*Parallel Application and Specification Language*) a estrutura do fluxo de execução da aplicação paralela indicando os lotes, a quantidade de tarefas de cada lote e o fluxo de dados entre eles. Na segunda, define-se o algoritmo que será executado pelas tarefas da aplicação, o que é feito com o uso da linguagem Java.

A alocação das tarefas que compõem uma aplicação é feita de forma mista entre os trabalhadores, na qual a metade é alocada de forma estática e as demais de forma dinâmica. A alocação estática é feita de forma proporcional ao fator de desempenho relativo de cada trabalhador, ou seja, os trabalhadores de maior desempenho recebem mais tarefas que os demais [9]. A alocação dinâmica é feita de forma que, para cada tarefa concluída, é alocada uma nova tarefa (tarefas que não foram enviadas para processamento) ou uma réplica (tarefas enviadas para processamento que ainda não retornaram o resultado) no computador trabalhador.

Para calcular os fatores de desempenho relativo entre os computadores trabalhadores, são executados programas de *benchmark* em cada um deles e divide-se o gasto pelo computador trabalhador mais rápido pelo tempo do trabalhador sendo avaliado.

2.3. Ferramentas auxiliares para predição

Em função da utilidade de um sistema de predição de desempenho para os usuários de um sistema de processamento paralelo, foram propostas por Herai et al. [8] as ferramentas CRES (*Computational Resource Evaluation Service*) e ARC (*Analytical Representation Compiler*) para a geração dos modelos das máquinas e da aplicação, respectivamente, para o sistema JoiN, usando a notação PAMELA Estendido.

Para a geração do modelo das máquinas, a ferramenta CRES baseia-se em *benchmarks*, que resultam em intervalos de valores para os parâmetros $t\_op\_data_p$ e $t\_send_{p,q}$. Tais intervalos fazem parte do modelo das máquinas.

Os *benchmarks* podem ser obtidos de todas as máquinas que compõem o sistema paralelo ou de apenas um subconjunto representativo das mesmas, o que reduz a precisão das estimativas mas também o custo para se fazer uma predição. Eles não precisam (e nem devem) concorrer com as aplicações, pois podem ser feitos de forma separada e independente do sistema paralelo. Portanto, dependendo da freqüência com que os dados de *benchmark* alimentam a ferramenta proposta, podem não ser capturadas as variações de carga do sistema paralelo, o que também pode contribuir para uma queda na precisão das predições.

Para a geração do modelo da aplicação, a ferramenta ARC se baseia em 3 modelos e em informações sobre a estrutura e a forma com que as tarefas são escalonadas de forma real no sistema paralelo JoiN. A estrutura da ferramenta ARC é representada na Fig. 1.

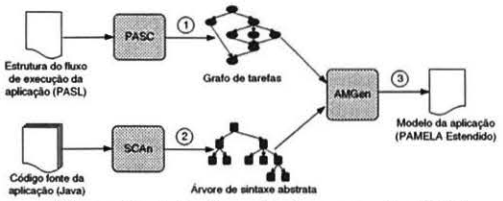

Figura 1. Estrutura da ferramenta ARC

A etapa 1 da Fig. 1 é feita por um compilador, chamado PASC (*Parallel Application and Specification Compiler*), responsável por analisar, léxica e sintaticamente, a especificação do fluxo de execução da aplicação paralela, escrita na linguagem PASL, e criar uma representação intermediária equivalente na forma de um grafo de tarefas, acíclico e direcionado, que possui as informações sobre a estrutura do fluxo de execução da aplicação paralela.

A etapa 2 da Fig. 1 é feita por um compilador, chamado SCAn (*Source Code Analyser*), que permite analisar uma aplicação escrita na linguagem Java de acordo com a gramática definida por Gosling et al. [11]. A análise produz uma representação intermediária do código-fonte Java, na forma de uma árvore de sintaxe abstrata.

A geração do modelo da aplicação (etapa 3 da Fig. 1) é feita pelo módulo AMGen (*Application Model Generator*), o qual analisa cada nó do grafo, partindo do lote inicial até atingir o final, e cria uma representação analítica das tarefas que serão enviadas através da rede para serem processadas nos computadores participantes do sistema de processamento paralelo.

A partir da árvore, AMGen analisa a execução de operações aritméticas do tipo soma, subtração, etc., sobre diferentes tipos de dados, como inteiro, ponto flutuante de precisão simples e dupla, etc. As operações aritméticas sobre os diferentes tipos de dados são convertidas no modelo para operações do tipo $t\_op\_data_p$.

A Fig. 2 ilustra de forma simplificada um trecho de código executado por uma tarefa que chama um método da mesma classe a partir de uma estrutura de condição e, ao lado, sua forma equivalente no modelo da aplicação, gerada com o uso da ferramenta ARC.

Na figura é ilustrada a forma com que o código de uma aplicação Java é convertido em uma representação equivalente ao modelo da aplicação, na linguagem PAMELA Estendido.

```
{                                   | {
  if(condicao){                     |   custo_condicao_p
                                    |   if{
    metodo_T(100000);               |     metodo_T_p;
  }                                 |   }
  else{                             |   else{
    bloco1                          |     custo_bloco1_p
  }                                 |   }
}                                   | }
...                                 | ...
                                    |
void  metodo_T(double  y){          | metodo_T={
                                    |   //custo parâmetro y
                                    |   assign_double_p;
                                    |
                                    |   //custo inicialização
    double  i;                      |   assign_double_p;
    for(i=0;i<y;i++){               |   //custo 1ª comparação
                                    |   access_p;
                                    |   compare_p;
                                    |   seq(i=1,100000){
                                    |     //custo incremento
                                    |     assign_double_p;
                                    |     access_double_p;
                                    |     sum_double_p;
                                    |     //custo comparações
                                    |     compare_p;
                                    |
                                    |     custo_bloco2_p
                                    |   }
      bloco2                        | }
    }
}
```

(a) Código-fonte (b) Modelo da aplicação

Figura 2. Geração de um Modelo de aplicação

3. APET: ferramenta de predição analítica de desempenho

Nesta seção é apresentada a proposta deste trabalho, que é o desenvolvimento de uma ferramenta de predição analítica de desempenho. A ferramenta, batizada de APET (*Analytical Performance Estimation Tool*), utiliza os modelos produzidos pelas ferramentas CRES e ARC para geração de estimativas de tempo. Assim como estas ferramentas, APET está restrita a aplicações de característica numérica e as estimativas são feitas baseando-se no escalonamento do JoiN, mas considerando que todas as tarefas são alocadas de forma estática no sistema.

A estrutura da ferramenta APET é apresentada na Fig. 3. Ela é composta por duas partes. A primeira é responsável pela combinação simbólica dos modelos das máquinas e da aplicação produzidos pelas ferramentas CRES e ARC, respectivamente, e pela produção de um modelo analítico de desempenho. A segunda avalia matematicamente o modelo analítico de desempenho e gera, a partir da escolha de uma determinada quantidade de computadores e de um modelo das máquinas, uma estimativa de tempo na forma de um intervalo de valores do tipo $[t_{min}, t_{max}]$.

A seguir é descrita cada uma das partes que compõem a ferramenta APET.

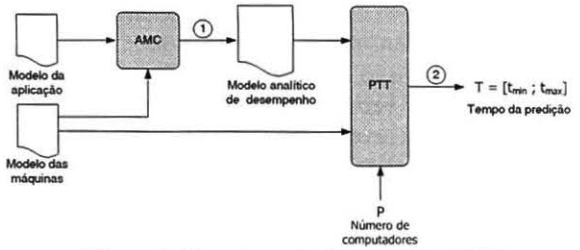

Figura 3. Estrutura da ferramenta APET

3.1. Geração do modelo analítico de desempenho

A geração do modelo analítico de desempenho, baseada na técnica apresentada por PAMELA Estendido, é realizada por um novo compilador chamado AMC (*Application Model Compiler*), que transforma o modelo da aplicação em expressões matemáticas. As expressões são representadas em forma analítica com parâmetros que podem ser utilizados com diferentes valores para avaliar o tempo da predição sob diferentes quantidades e configurações de computadores.

O compilador AMC gera uma representação intermediária do modelo da aplicação na forma de uma árvore de sintaxe abstrata, a qual é analisada e convertida em expressões matemáticas. A construção do AMC é dividida em 3 fases.

A primeira é a de criação de um analisador léxico. Para isso é definida a estrutura léxica da linguagem PAMELA Estendido com o uso da ferramenta JLex [12] para reconhecer os tokens da linguagem.

A segunda fase é a de criação de um analisador sintático para a linguagem utilizada no modelo da aplicação. Nesta fase é definida a gramática da linguagem PAMELA Estendido para reconhecer a estrutura organizacional dos tokens da linguagem. A partir da gramática, é utilizada a ferramenta CUP [12] para produzir um analisador sintático.

A terceira fase é a geração de uma árvore de sintaxe abstrata. A árvore é gerada pelo analisador sintático que cria os nós da árvore de acordo com a representação de cada token reconhecido pelo analisador léxico.

A partir da árvore, o compilador AMC faz uma combinação simbólica dos modelos das máquinas e da aplicação, produzindo um conjunto de expressões, em que os parâmetros do tipo op_data_p e $send_{p,q}$ representados na árvore são convertidos para os parâmetros de atraso de tempo $t\_op\_data_p$ e $t\_send_{p,q}$, respectivamente. As expressões representam o tempo de execução do modelo em duas situações: quando não há contenção e quando há contenção no uso de recursos de forma compartilhada.

Para a geração da expressão parcial T_0 (seção 2.1), a ferramenta analisa a árvore, nó a nó, fazendo as transformações necessárias, nas quais nós que representam instruções seqüenciais transformam-se em soma com o uso dos

operadores sum ou $+$ e nós que indicam instruções paralelas são representados pelo operador max.

Para a geração de cada uma das expressões parciais T_r, o compilador verifica, para cada um dos recursos analisados, os nós da árvore que utilizam o recurso considerado. A análise faz com que os nós que representam o uso de um recurso, por meio de instruções seqüenciais ou paralelas, sejam convertidos em operações de soma, com o uso do operador de somatório sum ou de adição $+$.

Após a criação das expressões matemáticas, é feita a geração da expressão final T, a qual receberá o maior valor entre as expressões parciais T_0 e T_r ($r = 1, ..., R$). Os parâmetros que compõem as expressões criadas mapeiam a execução de cada uma das operações de computação e de comunicação para cada um dos computadores que serão utilizados para o processamento da aplicação.

Em função disso, pode-se utilizar, na geração das estimativas, uma determinada quantidade de computadores e modelos de máquinas com valores distintos para os parâmetros $t\_op\_data_p$ e $t\_send_{p,q}$. Desta forma, o modelo analítico de desempenho pode ser utilizado por programadores para estimar o tempo de execução de uma aplicação, variando a quantidade de computadores e utilizando modelos de máquinas reais ou fictícios gerados a partir de sistemas de processamento paralelos formados por redes e computadores heterogêneos.

A Fig. 4 ilustra de forma simplificada a geração das expressões de um modelo analítico de desempenho, com o uso de AMC, a partir de um modelo reduzido da aplicação. Nele, considera-se que há contenção apenas no computador p.

```
L=...;Lj;...;
Lj =
     sum_double_p;
     mpy_int_p;
     par(i=1,10){
       use(p,metodo_Mp);
     }
metodo_Mp={ ...
     seq(i=1,1000000){
          sum_double_p;
          ...
     }
}
...
```

(a) Modelo da aplicação

$$
\begin{aligned}
T_0 &= t\_sum\_double_p + \\
&\quad t\_mpy\_int_p + \\
&\quad max_{i=1,10}(1000000 \times \\
&\quad (t\_sum\_double_p + \\
&\quad ...)) \\
T_1 &= t\_sum\_double_p + \\
&\quad t\_mpy\_int_p + \\
&\quad sum_{i=1,10}(1000000 \times \\
&\quad (t\_sum\_double_p + \\
&\quad ...)) \\
T &= max(T_0, T_1)
\end{aligned}
$$

(b) Modelo analítico de desempenho

Figura 4. Modelo Analítico de Desempenho

3.2. Geração de tempos de predição

A geração das estimativas é feita pela ferramenta PTT (*Prediction Time Tool*). Nela, é necessário informar a quantidade de computadores e o modelo das máquinas que serão considerados na predição.

A ferramenta utiliza tais informações para substituir os parâmetros do tipo $t\_op\_data_p$ e $t\_send_{p,q}$ na expressão final T do modelo analítico de desempenho pelos respectivos valores do modelo das máquinas. Em seguida a expressão final T é calculada por um avaliador de expressões matemáticas, o qual fornece como resultado um intervalo de valores do tipo $[t_{min}, t_{max}]$, que representa o intervalo de tempo esperado para que a aplicação seja executada.

A seguir são apresentados alguns resultados obtidos com o uso de APET para avaliar a qualidade das estimativas fornecidas por ela.

4. Resultados experimentais

As aplicações utilizadas nos testes, para avaliar a qualidade e utilidade da ferramenta APET, foram implementadas e executadas no sistema paralelo JoiN. A seguir, são descritos a configuração dos computadores e das aplicações consideradas e os resultados obtidos em cada um dos testes.

4.1. Configuração dos computadores

Para os testes, foi utilizado um conjunto de 32 computadores com diferentes configurações, tanto de hardware quanto de software, conforme apresentado abaixo:

- 9 computadores Pentium 4, 850 MHz, 128 Mb RAM, rede 10 Mbps, SO Linux Kurumin 4.0;
- 12 computadores Pentium 4, 1.5 GHz, 512 Mb RAM, rede 100 Mbps, SO Linux Kurumin 4.0;
- 11 computadores Pentium 4, 1.8 GHz, 256 Mb RAM, rede 100 Mbps, dos quais 5 possuem SO Linux Kurumin 4.0 e 6 possuem SO Linux Conectiva 8.0.

Apesar da diferença de configuração entre os computadores, as estimativas de tempo realizadas por APET consideram o fator de desempenho relativo de cada computador para a alocação de tarefas no sistema paralelo.

4.2. Configuração das aplicações

Vários tipos de aplicações científicas (ou de *benchmark* como o Linpack) poderiam ser utilizadas para testar a ferramenta APET. Entretanto para melhor depurar a ferramenta e facilitar a análise dos resultados foram escolhidas duas aplicações simples: cálculo do número π e busca de números primos, as quais seguem o modelo SPMD. Porém, não há restrições em APET que impeçam a predição de aplicações que não sejam SPMD.

O algoritmo utilizado para o cálculo do número π é baseado no método de Monte Carlo [13], no qual é verificado um conjunto de N pontos, representados por coordenadas x e y geradas aleatoriamente. Quanto maior for N,

maior será a precisão do número π encontrado. Esta aplicação foi testada com 4 versões, nas quais variou-se a quantidade de tarefas e de pontos. A primeira, chamada π_A, é composta por 100 tarefas e verifica 1 bilhão de pontos. A segunda, chamada π_B, é composta por 100 tarefas e verifica 2 bilhões de pontos. A terceira, chamada π_C, é composta por 200 tarefas e verifica 1 bilhão de pontos. A quarta, chamada π_D, é composta por 200 tarefas e verifica 2 bilhões de pontos. Cada tarefa recebeu um mesmo número N, indicando a quantidade de pontos sorteados por ela.

Para a aplicação de busca de números primos, o algoritmo utilizado é baseado em um método conhecido como Crivo de Eratóstenes [14], no qual são identificados todos os números primos de um determinado intervalo numérico. Esta aplicação também foi testada com 4 versões. A primeira versão, chamada pri_A, é composta por 100 tarefas e verifica o intervalo de 0 a 500 mil. A segunda, chamada pri_B, é composta por 100 tarefas e verifica o intervalo de 0 a 1 milhão. A terceira, chamada pri_C, é composta por 200 tarefas e verifica o intervalo de 0 a 500 mil. A quarta, chamada pri_D, é composta por 200 tarefas e verifica o intervalo de 0 a 1 milhão. Cada tarefa recebeu uma parcela do intervalo, que foi dividido em partes iguais.

4.3. Resultados dos testes

As 4 versões de cada aplicação foram testadas em diferentes horários do dia e da noite utilizando um mesmo conjunto de computadores compartilhados com outros usuários.

Para uma mesma configuração da aplicação foram realizadas quatro execuções, e variou-se a quantidade de computadores, iniciando com os 32 computadores disponíveis e reduzindo, em cada teste, a quantidade pela metade. Removeu-se sempre os mais lentos primeiro, até que restassem apenas os dois mais velozes.

Para obtenção das estimativas com o uso de APET, foi utilizado um modelo das máquinas atualizado no momento da realização de cada teste. Os resultados de cada estimativa foram comparados com tempos de várias execuções da aplicação, para efeito de avaliação da precisão das predições de desempenho.

Uma característica importante das aplicações testadas é que ambas possuem estruturas condicionais do tipo $if(cond)\{blc\}$, em que blc representa um bloco de computação que será executado somente se a condição $cond$ for verdadeira. Para a predição, é importante saber a quantidade de vezes que o bloco blc será executado, pois o cálculo das estimativas é feito sobre a quantidade de operações executadas pelo código.

Se a quantidade de operações consideradas pela estimativa for muito diferente da quantidade de operações realizadas quando a aplicação for executada, a comparação dos tempos de execução com as estimativas não tendem a ser boas, e é provável que fiquem fora dos limites do intervalo estimado. Em função disso, APET considera, para o cálculo das estimativas, o caso médio, que é a média aritmética entre as estimativas de tempo considerando o pior caso (bloco blc será sempre executado) e o melhor caso (bloco blc jamais será executado). Considerar apenas o pior ou o melhor caso para o cálculo das estimativas pode ser pouco eficiente, pois dependendo do tipo de aplicação ela poderá apresentar estimativas muito distantes dos tempos de execução reais.

4.3.1. Cálculo do número π

As 4 versões testadas apresentaram bons resultados, pois os tempos de execução obtidos, em todos os casos, estiveram dentro ou bem próximos do intervalo de tempo estimado por APET.

O gráfico da Fig. 5 apresenta os resultados obtidos com a execução das versões π_A e π_B. Os tempos de execução real com relação às estimativas de tempo, quando a quantidade de computadores variou, permaneceram em todos os casos sobre o intervalo estimado ou muito próximos dele. Pela

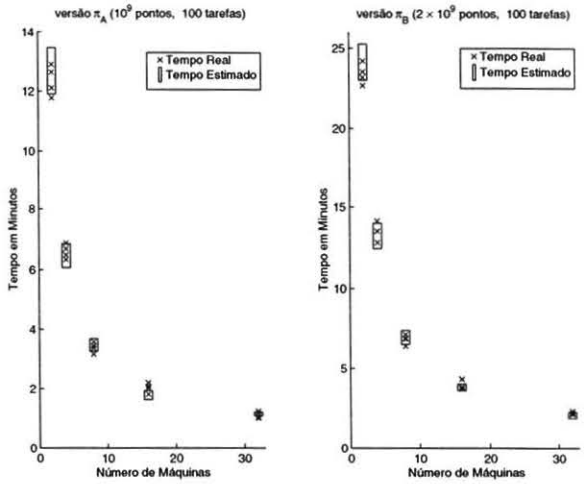

Figura 5. Tempos para cálculo do número π: versões π_A e π_B

análise do gráfico, pode-se também notar que, em alguns casos, os intervalos estimados para diferentes quantidades de computadores apresentaram alguns dos tempos de execução um pouco acima ou abaixo dos limites superior e inferior, respectivamente, do intervalo calculado. Isto provavelmente ocorreu porque o desempenho dos computadores utilizados nos testes variou durante a execução da aplicação. As outras versões testadas, π_C e π_D, apresentaram comportamento muito semelhante ao das versões π_A e π_B.

Nota-se que mesmo considerando o caso médio em desvios condicionais, as estimativas de tempo foram satisfatórias. Isso aconteceu porque o bloco da condição que verifica o número gerado aleatoriamente executa uma operação de incremento muito simples, e desta forma não causa influência considerável no tempo de processamento da aplicação e na geração das estimativas.

Desta forma, uma particularidade deste tipo de aplicação é o seu aparente determinismo, pois é possível determinar, mesmo com a presença de uma estrutura condicional no código, a quantidade aproximada de operações que serão executadas. Outra característica é o volume de dados enviados para a aplicação. Para que ela funcione, basta enviar para cada tarefa um único número que representa a quantidade de pontos avaliados pelo método. Um outro tipo de aplicação, multiplicação de matrizes, apresenta características semelhantes à aplicação testada no que diz respeito à quantidade de operações executadas pelo algoritmo. Porém, o volume de dados enviados dependeria diretamente das dimensões das matrizes a serem multiplicadas e seria bem superior ao existente neste exemplo. Neste caso o desempenho da predição seria mais dependente da avaliação do desempenho da rede.

4.3.2. Busca de números primos

As quatro versões da busca de números primos testadas, apresentaram em praticamente todos os casos tempos de execução inferiores aos intervalos calculados.

O gráfico da Fig. 6 ilustra os tempos de execução e os respectivos intervalos calculados para as versões pri_A e pri_B. Através do gráfico, é fácil visualizar a diferença entre os tempos de execução real e estimado. Mesmo variando a quantidade de computadores, ou dobrando a carga computacional da aplicação, os tempos de execução permaneceram em quase todos os casos abaixo do intervalo estimado. Nos testes das outras duas versões, pri_C e pri_D, verificou-se o mesmo comportamento das versões apresentadas. O erro nas estimativas ocorreu porque elas são calculadas considerando o caso médio nos desvios condicionais. O problema dessa abordagem nesta predição é que o bloco que faz parte da estrutura condicional e que verifica se um número é primo é executado poucas vezes. Tal bloco é composto por diversas operações que são responsáveis pela manipulação do número primo encontrado. A predição considera que tal bloco é executado uma quantidade muito maior de vezes do que na execução real da aplicação, motivo pelo qual as estimativas de tempo encontram-se sempre maiores do que os tempos reais de execução da aplicação.

Uma alternativa para melhorar a qualidade das estimativas fornecidas por APET, seria configurá-la para que, na presença de estruturas condicionais no código, fosse informada uma taxa indicando o percentual Q de vezes

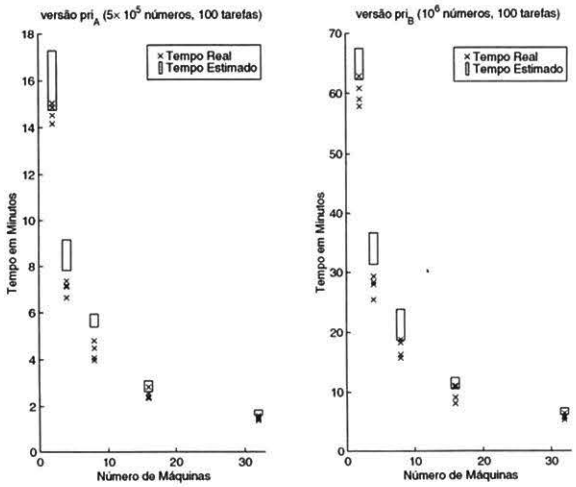

Figura 6. Tempos para busca de números primos: versões pri_A e pri_B

que o bloco da estrutura condicional seria executado. Para a aplicação de busca de números primos, sabe-se que a quantidade $prim([a,b])$ de números primos que podem ser encontrados em um intervalo do tipo $[a,b]$ é dado por $prim([a,b]) = prim(b) - prim(a)$, em que $prim(x) \approx x/(\log x - 1)$. Em função disso, é possível determinar uma quantidade aproximada de vezes em que é verdadeira a condição que verifica se um número é primo. Para determinar isso, considera-se que a quantidade M de números verificados em um intervalo $[a,b]$ é $M = b - a$, e que o percentual Q de números que podem ser primos no intervalo é dado pela relação $Q = prim([a,b])/M$. Este percentual pode então ser utilizado para calcular o intervalo de tempo da predição da aplicação de forma mais precisa, pois sabe-se a quantidade aproximada de vezes que o bloco da estrutura condicional será executado.

Desta forma, considerando uma estrutura condicional do tipo $if(cond)\{blc\}$ que verifica, através da condição $cond$, se um número é primo e que o bloco blc realiza algumas operações para tratamento de um número primo, é possível calcular um intervalo de tempo esperado, chamado T_e, da aplicação. Para calcular T_e, basta somar o atraso de tempo τ_{cond} da condição $cond$ com o resultado da multiplicação de Q com o atraso de tempo τ_{blc} do bloco blc. Assim, T_e é expresso por $T_e = \tau_{cond} + Q \times \tau_{blc}$.

No gráfico da Fig. 7 são ilustrados os tempos obtidos utilizando Q para calcular as estimativas. Pode-se verificar claramento que o intervalo calculado apresentou bons resultados, pois desta vez a estimativa e os tempos de execução coincidem ou estão bastante próximos. Apesar desta forma de se calcular as estimativas de tempo fornecer melhores resultados para aplicações que possuem estruturas condicionais do tipo if, é necessário possuir informações deta-

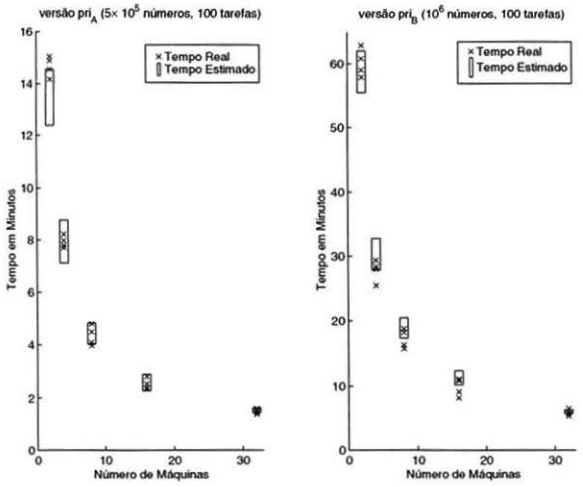

Figura 7. Tempos de pri_A e pri_B baseados em probabilidade

lhadas do algoritmo que será executado para que se possa estimar o percentual Q de vezes em que um bloco de uma estrutura condicional poderá ser executado.

Para se obter cada estimativa de desempenho para estas aplicações, foram gastos em média 3 segundos, quando a ferramenta APET foi executada em um computador Pentium III, 800 MHz, 384 Mb RAM, Windows 2000.

Presume-se que seria possível obter estimativas de tempo mais precisas, se fosse feita uma análise abstrata das expressões condicionais. Esta abordagem será tratada em um trabalho futuro.

5. Conclusões

Este artigo apresentou uma ferramenta para a geração de um modelo analítico de desempenho, fundamental para para a predição de tempo de aplicações paralelas. A ferramenta *APET* analisa modelos de máquinas e da aplicação para produzir um outro modelo contendo expressões matemáticas que podem ser analisadas, substituindo-se os parâmetros que compõem cada uma das expressões. Isso permite aos usuários do sistema paralelo avaliar o impacto causado na execução de uma aplicação quando a quantidade ou o desempenho dos computadores varia.

A ferramenta gera as estimativas de tempo em poucos segundos e produz resultados bastante satisfatórios. Estes resultados podem ser melhorados se for possível determinar a probabilidade de execução dos blocos condicionais presentes na aplicação.

A ferramenta proposta tem como principais vantagens em relação àquelas existentes na literatura a possibilidade de realizar predições considerando efeitos de atrasos causados por contenção de recursos e o uso de intervalos para representar os tempos de execução previstos.

Apesar dos testes feitos para validação da proposta terem sido focados no sistema de processamento paralelo JoiN, os conceitos que fundamentam a ferramenta APET são genéricos o suficiente para permitir a adaptação da mesma para outros ambientes de processamento paralelo.

Agradecimentos

À CAPES pelo apoio no desenvolvimento de parte deste trabalho.

Referências

[1] S. Prakash. *Performance Prediction of Parallel Programs.* Tese de doutorado, Computer Science Department, University of California, California, USA, 1997.

[2] H. Gautama. *A Probabilistic Approach to Symbolic Performance Modeling of Parallel Systems.* Tese de doutorado, DITS, Delft University of Technology, Sweden, 2004.

[3] E. J. H. Yero and M. A. A. Henriques. Contention-sensitive static performance prediction for parallel distributed applications. In *Performance Evaluation: an International Journal*, aceito para publicação em Janeiro de 2005.

[4] S. A. Jarvis, D. P. Spooner, L. C. Keung, and G. R. Nudd. Performance prediction and its use in parallel and distributed computing systems. In *17th IEEE International Parallel and Distributed Processing Symposium*, Nice, France, 2003.

[5] A. J. C. van Gemund. Automatic cost estimation of data parallel programs. Relatório Técnico 1-68340-44(2001)09, DITS, Delft University of Technology, Sweden, 2001.

[6] C. van Reeuwijk. Spar language specifi cation. Relatório Técnico PDS-2001-003, DITS, Delft University of Technology, Sweden, 2001.

[7] J. Brém, M. Madhukar, E. Smirni, and L. Dowdy. Perpret - a performance prediction tool for massively parallel systems. In *Proceedings of the Joint Conference Performance Tools / MMB*, Heidelberg, Germany, 1995.

[8] R. H. Herai and M. A. A. Henriques. Ferramentas de modelagem para a predição de performance analítica em uma plataforma de processamento paralelo. In *Quinto Workshop em Sistemas Computacionais de Alto Desempenho (WSCAD2004)*, Foz do Iguaçu, Paraná, Brasil, 2004.

[9] E. J. H. Yero, F. O. Lucchese, F. S. Sambatti, M. von Zuben, and M. A. A. Henriques. Join: The implementation of a java-based massively parallel grid. In *Future Generation Computer Systems: Parallel computing technologies*, volume 21, Issue 5, pages 791–810, Maio 2005.

[10] J. M. Schopf and F. Berman. Performance prediction using intervals. Relatório Técnico CS-97-541, University of California, San Diego, USA, 1997.

[11] J. Gosling, B. Joy, G. Steele, and G. Bracha. *The Java Language Specifi cation.* Addison-Wesley, 3rd edition, 2005.

[12] A. W. Appel. *Modern Compiler Implementation in Java.* Cambridge University Press, New York, Cambridge, 1998.

[13] P. Beckmann. *A History of Pi.* Golem Press, 1977.

[14] R. Crandall and C. Pomerance. *Prime numbers: a computational perspective.* Springer-Verlag, New York, 2001.