Boletim Técnico da Escola Politécnica da USP
Departamento de Engenharia de Computação e
Sistemas Digitais

ISSN 1413-215X
BT/PCS/0302

Análise e Predição de Desempenho de Programa MPI em Redes de Estações de Trabalho

Jean Marcos Laine
Edson T. Midorikawa

São Paulo - 2003

O presente trabalho é parte da dissertação de mestrado apresentada por Jean Marcos Laine, sob a orientação do Prof. Dr. Edson T. Midorikawa.: "Análise e Predição de Desempenho de Programas MPI em Redes de Estações de Trabalho", defendida em 27/01/03, na EPUSP.

A íntegra da dissertação encontra-se à disposição com o autor e na Biblioteca de Engenharia Elétrica da Escola Politécnica da USP.

FICHA CATALOGRÁFICA

Laine, Jean Marcos
Análise e predição de desempenho de programa MPI em redes de estações de trabalho / J.M. Laine, E.T. Midorikawa. – São Paulo : EPUSP, 2003.
p. – (Boletim Técnico da Escola Politécnica da USP, Departamento de Engenharia de Computação e Sistemas Digitais, BT/PCS/0302)

1. Programação paralela I. Midorikawa, Edson Toshimi
II. Universidade de São Paulo. Escola Politécnica. Departamento de Engenharia de Computação e Sistemas Digitais III. Título IV. Série
ISSN 1413-215X CDD 004.35

Análise e Predição de Desempenho de Programas MPI em Redes de Estações de Trabalho*

Jean Marcos Laine[1], Hélio Marci de Oliveira[1], Edson T. Midorikawa[1], Liria Matsumoto Sato[1], Li Kuan Ching[2]

[1]Departamento de Engenharia de Computação e Sistemas Digitais
Escola Politécnica da Universidade de São Paulo (PCS-EPUSP)
Av. Prof. Luciano Gualberto, travessa 3, 158 – 05508-900 – São Paulo – SP – Brasil

[2]Department of Electrical and Computer Engineering
University of California, Irvine – USA

{jean.laine, helio.marci, edson.midorikawa, liria.sato}@poli.usp.br,
likuan@aardvark.ece.uci.edu

Abstract. *Writing a parallel program is quite a difficult task, specially when the programmer wants to write efficient parallel algorithms. The knowledge of the program execution time and the factors that influences this time is very important to improve program performance. In this paper we applied a new methodology to analyze and predict the execution time of parallel programs implemented with MPI communication primitives. For this, we developed analytical models to represent the behavior of a simple program for matrix multiplication. The results showed that the methodology produced very accurate models, due to the fact that most of results presented errors down to 3%.*

Resumo. *Escrever programas paralelos é uma tarefa difícil, principalmente quando o programador deseja escrever algoritmos eficientes. O conhecimento do tempo de execução do programa e dos fatores que influenciam este tempo é muito importante para melhorarmos o desempenho de uma aplicação. Neste artigo, aplicamos uma nova metodologia para análise e predição de desempenho de programas paralelos implementados com primitivas de comunicação MPI. Para isso, desenvolvemos modelos analíticos para representar o comportamento de um programa simples de multiplicação de matrizes. Os resultados mostraram que a metodologia produziu modelos muito precisos, pois a maioria dos resultados apresentou erros abaixo de 3%.*

1. Introdução

Dentre as várias alternativas para arquiteturas paralelas, as redes de estações de trabalho (NOWs) ou *clusters* têm se destacado nos últimos anos como a mais eficiente em termos de custo e desempenho. A facilidade na construção de sistemas de alto desempenho através da interligação de estações de trabalho com a utilização de redes de comunicação de alta velocidade, aliado aos recentes avanços da tecnologia de circuitos

* Trabalho financiado parcialmente pelo CNPq e CAPES.

integrados e o baixo custo destes componentes, fez com que a maioria dos fabricantes, inclusive brasileiros, optasse pelos *clusters* para a execução de aplicações paralelas [Pfister 1998].

Normalmente os programas paralelos apresentam um comportamento não esperado, devido a diversos fatores como a estruturação interna das aplicações paralelas, o sistema paralelo no qual é executado, o número de processadores/nós de processamento usados, os dados de entrada do programa e a medida de desempenho adotada [Crovella 1994]. Além disto, um escalonamento efetivo das aplicações e a adoção de uma estratégia para o balanceamento de carga são outros aspectos cruciais para alcançarmos um bom desempenho em uma rede de estações de trabalho.

Diversos pesquisadores têm se dedicado ao desenvolvimento de modelos e metodologias para a predição de desempenho de programas paralelos em ambientes homogêneos e heterogêneos. De uma maneira geral é adotado um enfoque em dois níveis lógicos: o nível de sistema e o nível de aplicação [Adve 1993]. Os aspectos considerados no nível de sistema envolvem, por exemplo, o mecanismo de comunicação usado e a capacidade de processamento dos nós da rede de estações. O nível de aplicação leva em consideração a estrutura interna do programa paralelo, os padrões de comunicação e de processamento, entre outros.

Este trabalho apresenta uma metodologia para análise e predição de desempenho de programas paralelos executados em redes de estações de trabalho [Li 2001][Li et al 2001][Oliveira et al 2002] e a sua utilização em uma pequena aplicação de multiplicação de matrizes em ambiente MPI. Este pequeno estudo ilustrativo mostra de uma maneira geral os principais pontos da metodologia, comprovando a sua efetividade.

O restante do trabalho é organizado da seguinte forma. A seção 2 apresenta a metodologia de análise e predição. Na seção seguinte é apresentado uma análise dos fatores no nível de sistema, com uma caracterização de primitivas de comunicação MPI. A seção 4 mostra como a aplicação é representada e analisada com DP*Graphs. As conclusões e propostas para trabalhos futuros são apresentadas na seção 5.

2. Metodologia de Análise e Predição de Desempenho

Escrever programas paralelos é uma tarefa difícil, especialmente quando se pretende descrever um algoritmo de forma eficiente. Além disto, a execução de um programa paralelo é dependente de diversos fatores, que interagem entre si de forma complexa. Para identificar e compreender qual seria a melhor estrutura para um programa paralelo, é importante entender quais são os fatores que podem interferir no seu desempenho.

A metodologia de análise e predição de desempenho proposta em [Li 2001] utiliza uma abordagem com grafos de tempos estendidos e um conjunto de técnicas experimentais para coleta de dados. A Figura 1 mostra a estrutura completa da metodologia proposta. O grafo de tempo T-Graph* é uma forma de abstração da aplicação num modelo de alto nível, enquanto o grafo de baixo nível DP*Graph é responsável pela representação do mesmo programa paralelo de forma detalhada, correspondendo a cada passo da seqüência de execução do código deste programa.

Um programa paralelo representado nos grafos de tempo é composto de um conjunto de trechos de código, cada qual com suas características: tempo de execução,

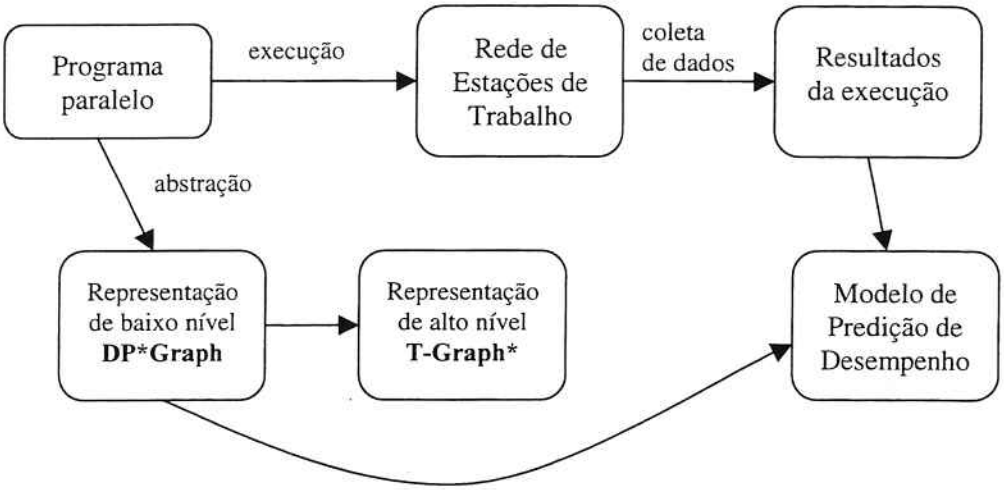

Figura 1. Metodologia de Análise e Predição proposta

quantidade de processamento, quantidade de comunicação, etc. Uma vez obtido o grafo de representação de baixo nível, o programa é submetido a várias execuções na rede de estações de trabalho, com diferentes números de processadores e tamanhos do problema. Após terminar o conjunto de testes, é obtido um modelo analítico do programa paralelo que é usado para efetuar predições de desempenho sobre ele[1].

Para isolar os efeitos não determinísticos da contenção na rede de comunicação, os pontos dentro de um programa onde ocorrem comunicações são identificados, destacados e computados isoladamente em relação às computações locais de uma tarefa.

3. Caracterização de Primitivas de Comunicação MPI

O padrão MPI (*Message Passing Interface*) é voltado para a implementação de aplicações baseadas em trocas de mensagens, provendo um ambiente para o desenvolvimento de aplicações paralelas de alto desempenho e uma biblioteca de programação bastante eficiente e portável [Al-Tawil and Moritz 2001]. Existem diversas implementações do padrão, contudo, se comparadas em uma mesma plataforma suas implementações apresentam desempenhos semelhantes [Nupairoj and Ni 1994].

As formas de comunicação previstas pelo padrão podem ser bloqueantes e não bloqueantes. O termo bloqueante significa que a rotina somente é finalizada quando o buffer de dados utilizado na comunicação puder ser reusado pela aplicação [The Ohio State University 1996]. Além disso, uma comunicação pode ser ponto-a-ponto ou coletiva, dependendo do número de processos envolvidos.

O modo de comunicação padrão do MPI permite que um processo (*sender*) envie uma mensagem a outro (*receiver*) de forma bloqueante. A mensagem é enviada para um determinado *receiver* e, embora o sistema possa armazenar algumas mensagens antes de enviá-las, espera-se que a rotina não seja finalizada corretamente até que o *receiver* correspondente esteja pronto [The Ohio State University 1996]. Esta comunicação é dita

[1] Na realidade existem outros aspectos da aplicação da metodologia que não foram mencionados aqui devido a falta de espaço. Para maiores informações consulte a referência [Li 2001].

não local, pois é necessária uma troca de mensagens com um processo remoto antes de seu encerramento.

Neste trabalho, aplicamos a metodologia de predição de desempenho descrita na seção 2 em um programa paralelo, implementado com primitivas MPI, projetado para realizar a multiplicação de matrizes de elementos do tipo *double*. É adotada a implementação LAM-MPI [The Ohio State University 1996] [University of Tennessee 1995] e o programa analisado utiliza apenas o modo de comunicação padrão para realizar as trocas de mensagens.

Para a realização dos testes, foi utilizado um *cluster* com 16 nós conectados por um *switch* 3COM SuperStac3300 e uma rede Fast-Ethernet. Cada nó de processamento possui um processador Intel Celeron 433MHz, 128MB de SDRAM (66MHz) e uma placa de rede Intel Ether-Express Pro de 100Mb/s. Além disso, as máquinas executavam o Linux Red Hat 6.2 e o LAM-MPI 6.4.

3.1 Análise do tempo de comunicação

Considerando-se programas projetados para serem executados em redes de estações de trabalho, o tempo despendido com a comunicação entre nós é um importante fator a ser considerado na análise e predição de desempenho. Segundo a metodologia proposta por [Li 2001], este tempo pode ser decomposto nos seguintes componentes:

- $t_e(n)$: tempo para transferir n elementos de dados da memória para a interface de rede;
- $t_t(n)$: tempo gasto durante a transmissão de n elementos de dados pela rede de interconexão;
- $t_r(n)$: tempo para receber a mensagem enviada, transferindo-a da interface de rede para a memória.

A Figura 2 permite visualizar melhor cada um estes componentes.

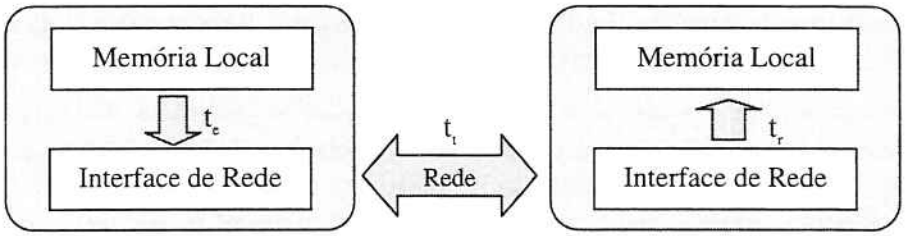

Figura 2. Componentes do tempo de comunicação

Para caracterizar apropriadamente o tempo gasto nas comunicações de uma aplicação, um modelo precisa considerar os principais fatores que podem causar variações nesse tempo. Segundo [Hu and Gorton 1997], um bom modelo deve, na medida do possível, descrever correta e precisamente o comportamento de um sistema e ser de fácil utilização. Desta forma, é importante na construção de nosso modelo identificar os principais fatores que influenciam o tempo de comunicação e descartar outros, buscando-se reduzir sua complexidade sem, contudo, comprometer sua eficiência.

Neste trabalho, consideramos dois fatores principais, representados pelas constantes k_1 (latência da rede) e k_b (largura de banda) [Li 2001].

Considerando a transmissão de uma mensagem com n elementos, representamos os componentes do tempo de comunicação, citados anteriormente, através das seguintes equações:

$$t_e(n) = n \times c_1$$
$$t_t(n) = k_b \times n + k_1$$
$$t_r(n) = n \times c_2$$

onde c_1 e c_2 são constantes.

Assim, nosso modelo analítico expressa o tempo de comunicação de uma mensagem com n elementos ($t_c(n)$) através de uma equação linear, conforme segue abaixo:

$$\begin{aligned} t_c(n) &= t_e(n) + t_t(n) + t_r(n) \\ &= n \times c_1 + k_b \times n + k_1 + n \times c_2 \\ &= (c_1 + k_b + c_2) \times n + k_1 \\ t_c(n) &= c \times n + k_1, \end{aligned}$$

onde c é uma constante.

3.2 Modelagem do modo de comunicação padrão

Os modelos analíticos gerados neste trabalho são baseados em tempos de execução coletados através de testes experimentais. Para a modelagem do tempo do *send* e do *receive*, utilizamos o benchmark MPBench [Mucci et al 1998], realizando um conjunto de execuções para cada tamanho de mensagem.

Analisando os tempos obtidos com o uso do MPBench, podemos construir equações para caracterizar os tempos das operações de *send* e *receive*. Como discutido na seção 3.1, esses tempos podem ser expressos através de funções lineares do tamanho da mensagem transmitida. Desta forma, algumas funções foram elaboradas, buscando-se aquelas que expressassem melhor o comportamento dos tempos de comunicação. Observações destas equações induziram-nos a concluir que a melhor solução seria adotarmos mais de uma função, variando conforme o tamanho da mensagem enviada.

Assim, obtivemos o seguinte modelo analítico para caracterizar os tempos gastos pelos processos *send* e *receive* na transmissão de uma mensagem no modo de comunicação padrão:

$$6824 \leq n \leq 65536 \begin{cases} T_{send}(n) = 0,0879182*n - 82,0523 \\ T_{receive}(n) = 0,0852*n + 9,706 \end{cases}$$

$$65536 < n \leq 5592404 \begin{cases} T_{send}(n) = 0,0849931*n + 2065,49 \\ T_{receive}(n) = 0,0849968*n + 2146,19 \end{cases}$$

As equações acima apresentadas foram obtidas aplicando-se uma interpolação polinomial dos pares ordenados (n, média dos tempos coletados), tanto para o *send* como para o *receive*. A Tabela 1 apresenta as médias dos tempos coletados, bem como os valores do modelo analítico para T_{send} e $T_{receive}$ e os erros percentuais observados.

Podemos constatar que os valores coletados e do modelo se aproximam bastante, sendo o maior erro encontrado no tempo de *send* com n igual a 6824, onde o valor calculado pela função diferiu do medido em apenas 4,142%.

Tabela 1. Tempos de *send* e *receive* e erros percentuais encontrados

n (bytes)	Send (µs)	T_{send}	erro (%)	Receive (µs)	$T_{receive}$	erro (%)
6.824	540,280	517,901	4,1420	589,330	591,111	0,3022
13.652	1.130,440	1.118,207	1,0821	1.173,100	1.172,856	0,0208
21.844	1.830,230	1.838,433	0,4482	1.873,660	1.870,815	0,1519
43.688	3.686,130	3.758,918	1,9746	3.733,970	3.731,924	0,0548
65.536	5.726,130	5.679,755	0,8099	5.593,890	5.593,373	0,0092
131.072	13.198,730	13.205,706	0,0528	13.246,380	13.286,891	0,3058
218.452	20.629,029	20.632,403	0,0164	20.704,529	20.713,911	0,0453
349.524	31.766,000	31.772,618	0,0208	31.851,359	31.854,612	0,0102
524.288	46.636,770	46.626,352	0,0223	46.744,352	46.708,992	0,0756
1.048.576	91.211,102	91.187,215	0,0262	91.303,109	91.271,795	0,0343
1.747.624	150.591,703	150.601,471	0,0065	150.686,453	150.688,638	0,0014
3.495.252	299.122,563	299.137,793	0,0051	299.219,250	299.231,425	0,0041
5.592.404	477.389,344	477.381,242	0,0017	477.482,906	477.482,634	0,0001

4. Representação e Modelagem de Aplicações

Na seção anterior, apresentamos os modelos analíticos elaborados para representar as primitivas de comunicação *send* e *receive* do padrão MPI, considerando o modo de comunicação padrão. Nesta seção, estaremos apresentando uma classe de grafos denominada *DP*Graph*, citada na seção 2, e a representação do programa de multiplicação de matrizes segundo esta notação. Além disso, mostramos algumas técnicas encontradas na literatura para modelagens de programas paralelos e, por fim, apresentamos o modelo analítico definido para a aplicação escolhida.

4.1. Representando aplicações com DP*Graph

O grafo de tempo denominado *DP*Graph* foi definido em [Li 2001] como sendo uma forma de abstrair a aplicação num modelo de grafo detalhado, capaz de representar tanto os trechos de computações seqüenciais quanto as primitivas de comunicações MPI. A representação de baixo nível reflete com fidelidade a seqüência de execução do código em cada um dos nós de processamento.

Através do *DP*Graph* é possível identificar com clareza os pontos de sincronismo entre os processadores/processos que cooperam na solução do problema e também os potenciais trechos de código paralelizáveis. Com isso podemos saber se o código elaborado foi ou não bem escrito ou se há paralelismo ainda não explorado na aplicação.

A seguir apresentamos a simbologia utilizada na elaboração do *DP*Graph* e um breve comentário sobre seu significado:

representa trechos de códigos seqüenciais;

representa a existência de uma primitiva de comunicação de envio de mensagem, independente de ser ponto-a-ponto, coletiva, bloqueante ou não bloqueante;

representa a existência de uma primitiva de comunicação de recebimento de mensagem;

representa uma aresta no *DP*Graph* para conectar os símbolos anteriores e indicar a seqüência da execução do programa representado.

O *DP*Graph* correspondente ao programa de multiplicação de matrizes foi elaborado segundo esta simbologia. Na Figura 3, representamos apenas os principais trechos de código seqüenciais, que correspondem a inicialização das matrizes A e B e ao cálculo da matriz produto C. Por outro lado, todas as primitivas de comunicações existentes na aplicação foram representadas.

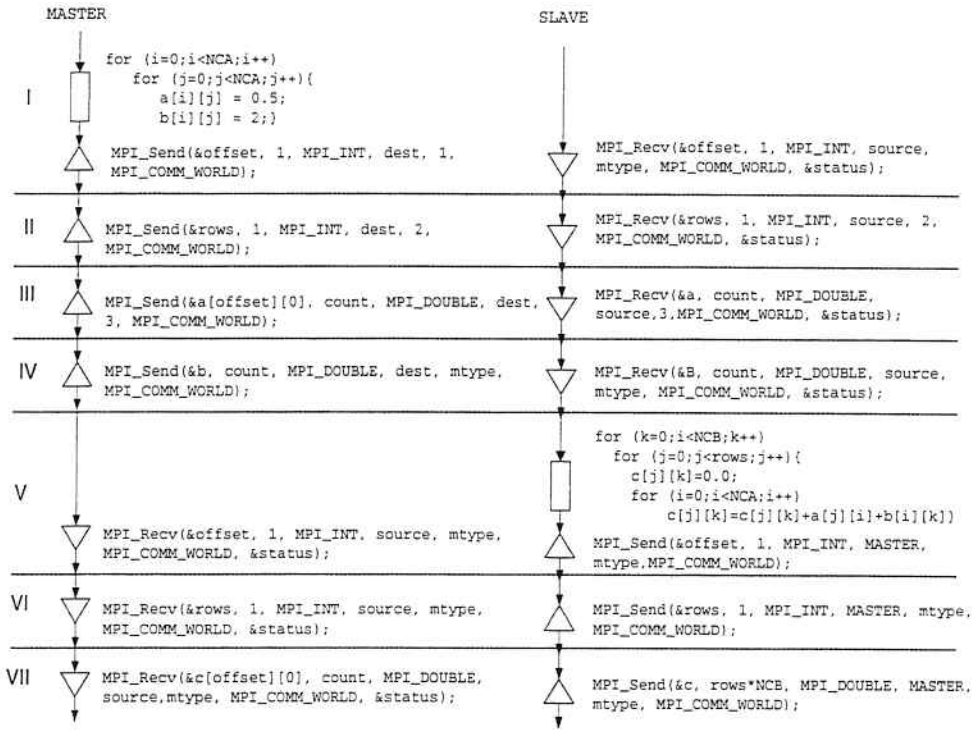

Figura 3. Representação em *DP*Graph* do programa de multiplicação de matrizes

Como já comentamos, e pode ser observado na Figura 3, um programa representado pelo *DP*Graph* mostra tanto os trechos de computações seqüenciais quanto os pontos de comunicação, interligados por arestas que direcionam o sentido da execução do código.

No *DP*Graph* apresentado acima, podemos observar os 7 trechos (I, II, ..., VII), identificados no programa de multiplicação de matrizes, com o código do programa e sua respectiva representação na simbologia do *DP*Graph*. No trecho I da Figura 3, o processo *master* realiza a inicialização das matrizes A e B e depois envia uma

mensagem para o processo *slave*. Enquanto isso, o processo *slave* fica esperando até que a mensagem enviada pelo *master* esteja disponível para que ele a receba.

No trecho V, o processo *slave* calcula o produto das matrizes A e B. Esta é a parte do programa que demanda maior poder computacional e que poderia ser distribuída entre vários processos *slaves* espalhados entre os nós participantes do processamento. Em nossos testes optamos por utilizar apenas um *slave*, pois estávamos interessados em montar um modelo mestre-escravo para medir o tempo de comunicação entre o *sender* e o *receiver*.

Segundo a metodologia desenvolvida em [Li 2001], após ter elaborado o *DP*Graph* devemos modelar analiticamente o comportamento do programa para que se possa realizar estudos de análises e predições de desempenho sobre o mesmo. Assim, na próxima seção daremos uma breve introdução a este tipo de modelagem e em seguida apresentaremos o modelo construído.

4.2. Modelos analíticos

Em sistemas multiprocessadores, a avaliação e predição de desempenho tem sido realizada com o auxílio de modelos, construção de protótipos e técnicas de simulação [Harrison and Patel 1993][Li and Sato 1999][Li and Malik 1995][Reed 1993]. Estes modelos elaborados podem ser classificados em: analíticos, estatísticos, empíricos ou híbridos [Li 2001].

A modelagem analítica utiliza equações e fórmulas matemáticas para descrever o comportamento da aplicação que será analisada. O objetivo principal é desenvolver modelos matemáticos que expressem com certa precisão o comportamento da aplicação, permitindo assim a realização de predições de desempenho. Contudo, a modelagem de sistemas complexos pode requerer a utilização e o tratamento de um número muito grande de funções e parâmetros, o que pode tornar o modelo intratável.

Portanto, a construção de um modelo analítico deve levar em consideração o nível de detalhe e a tratabilidade do mesmo. É interessante que o modelo desenvolvido seja preciso e, ao mesmo tempo, simplificado o suficiente para torná-lo viável.

4.2.1. Modelagem da aplicação

Conforme apresentado na seção 3.2, os modelos analíticos para as primitivas de comunicação *send* e *receive* foram elaborados com o auxílio de um método de interpolação polinomial aplicado sobre os resultados obtidos nos testes do *benchmark MPBench*. Nesta seção seguiremos o mesmo tipo de análise para modelar alguns trechos seqüenciais do código do programa de multiplicação de matrizes.

Durante a elaboração do modelo analítico construímos várias equações tentando encontrar uma que melhor representasse o comportamento do código e, consequentemente, tivesse mais precisão ao calcular o tempo gasto na inicialização das matrizes A e B. A seguir apresentamos a equação que apresentou os menores erros percentuais em relação aos valores de tempos experimentais:

$$\text{(I)} \quad T_{inic}(n) = 2{,}12E\text{-}7 \times n^2 - 1{,}27E\text{-}6 \times n - 0{,}00043$$

sendo n a dimensão das matrizes.

Tabela 2. Comparação entre os tempos experimentais e os calculados por T_{inic}

	Inicialização das matrizes				
n	100	200	300	400	500
Tempos experimentais	0,001558	0,007773	0,018397	0,032864	0,052005
T_{inic}	0,001565	0,0078	0,018277	0,032997	0,051958
Erro %	-0,42	-0,34	0,65	-0,40	0,09

Conforme observado na Tabela 2, o erro em relação ao tempo medido ficou abaixo de 0,65%, o que mostra a precisão e valida o modelo elaborado. Para o trecho VII do *DP*Graph*, que representa a multiplicação das matrizes, seguimos o mesmo processo e obtivemos a seguinte função:

(II) $T_{mult}(n) = 3,41E\text{-}7 \times n^3 + 2,63E\text{-}5 \times n^2 - 9,38E\text{-}3 \times n + 0,64$

Tabela 3. Comparação entre os tempos experimentais e os calculados por T_{mult}

	Multiplicação das matrizes				
n	100	200	300	400	500
Tempos experimentais	0,275609	2,662435	9,220848	23,04117	45,12641
T_{mult}	0,305428	2,543162	9,399772	22,92192	45,15628
Erro %	-10,82	4,48	-1,94	0,52	-0,07

Os resultados obtidos e mostrados na Tabela 3 são bastante satisfatórios e permite dizer que o modelo elaborado para a multiplicação das matrizes também se mostrou muito adequado e eficiente. A maioria dos erros percentuais ficou abaixo de 5%, o que é extremamente aceitável e tolerável em análise e predição de desempenho.

A modelagem dos outros trechos do programa correspondem a operações de envio e recebimento de dados entre os processos *master* e *slave*. Desta forma, para cada operação de comunicação foi obtido a quantidade de bytes transferidos em função da ordem n das matrizes. Em seguida, usando-se a caracterização das primitivas de comunicação MPI realizada na seção 3, obtivemos as funções do tempo de execução para cada um dos trechos.

Depois de ter modelado cada um dos trechos identificados no *DP*Graph*, podemos elaborar o modelo analítico que represente o programa como um todo. Este modelo será, basicamente, uma função matemática dada pela somatória de cada uma das funções encontradas anteriormente. Como o tempo total de execução do programa é dado pelo tempo que o processo *master* gasta na realização de suas tarefas, o modelo analítico é formulado com base nos trechos identificados para este processo no grafo de baixo nível.

Assim, o tempo total de execução é dado por:

onde $f_i(n)$ são as funções encontradas para representar os trechos de I a VII na Figura

$$t_{exec}(n) = \sum_{i=I}^{VII} f_i(n) + k$$

3. O termo k representa algumas constantes não consideradas durante a elaboração das funções, como o *overhead* do sistema operacional e da pilha de protocolos TCP/IP, por questões de simplificação do modelo.

Dessa forma, o tempo de execução do programa de multiplicação de matrizes obtido pela metodologia é dado por:

(III) $T_{exec}(n) = 3,41E\text{-}7 \times n^3 + 2,65E\text{-}5 \times n^2 - 9,39E\text{-}3 \times n + 0,64$

Foram realizados execuções do programa no *cluster* experimental para diversos tamanhos de matrizes. A Tabela 4 mostra os resultados para o tempo total de execução do programa e os tempos obtidos pelo modelo analítico para T_{exec}.

Tabela 4. Predição do tempo total de execução do programa

Programa de Multiplicação de Matrizes							
N	100	200	250	300	400	450	500
Tempos experimentais	0,316625	2,785438	5,326959	9,47386	23,52449	32,80233	45,87755
T_{exec}	0,307599	2,551355	5,279105	9,418468	22,95536	32,86457	45,20871
Erro %	-2,85	-8,40	-0,90	-0,58	-2,42	0,19	-1,46

Conforme pode ser observado na tabela, podemos verificar que o modelo analítico obtido pela metodologia de análise e predição de desempenho apresentou resultados bem próximos do tempo de execução experimental. Excetuando-se o caso para as matrizes de ordem 200, os erros percentuais ficaram abaixo de 3%. Apesar de sua simplicidade, o modelo obtido se mostrou bastante preciso para a predição do tempo de execução total do programa de multiplicação de matrizes.

5. Conclusões e Trabalhos Futuros

Neste trabalho apresentamos uma aplicação de uma nova metodologia para análise e predição de desempenho de programas paralelos, implementados com primitivas de comunicação MPI. A partir de modelos analíticos gerados, realizamos predições do tempo de execução de um programa de multiplicação de matrizes.

Para a construção do modelo, foi preciso dividir o programa em trechos de código, separados por pontos de sincronização. Utilizando interpolação polinomial, geramos funções para estimar o tempo de execução de cada um destes trechos de código, podendo predizer o tempo aproximado do programa através da somatória destas funções. O modelo construído é bastante simples, porém os resultados obtidos comprovam sua validade. Nos testes realizados com o programa de multiplicação de matrizes os erros mantiveram-se abaixo de 9%, sendo a maior parte deles menores que 3%.

Atualmente, estamos realizando testes com matrizes maiores, procurando verificar a eficácia do modelo quando o tamanho das mensagens trocadas e a quantidade de computação envolvida são maiores. Além disso, outras primitivas de comunicação MPI, como por exemplo operações de comunicação coletiva e outros modos de comunicação ponto a ponto bloqueante, estão sendo modeladas.

Como trabalhos futuros pretendemos aplicar a metodologia em programas de *benchmark* paralelos, como o NPB (*NAS Parallel Benchmarks*). Outros fatores que podem influenciar no desempenho dos programas, como o tempo gasto com a alocação de processadores/processos para a realização de tarefas paralelas, ainda precisam ser melhor analisados. Com isso, esperamos reduzir os erros percentuais e estender a

aplicabilidade do modelo a situações em que outros fatores, pouco enfatizados neste trabalho, podem exercer maior influência no tempo total de execução dos programas.

Referências Bibliográficas

Adve, V. S. (1993) "Analyzing the behavior and performance of Parallel Programs", PhD thesis, University of Wisconsin-Madison, Computer Science Deparment.

Al-Tawil, K. and Moritz, C. A. (2001) "Performance Modeling and Evaluation of MPI". Journal of Parallel and Distributed Computing, 61, p. 202-223.

Crovella, M. E. (1994) "Performance Prediction and Tuning of Parallel Programs", PhD thesis, University of Rochester, Computer Science Department, August.

Harrison, P. G. and Patel, N. M., Performance modeling of communication networks and computer architectures, Addison-Wesley, New York, 1993.

Hu, L. and Gorton, I (1997). "Performance Evaluation for Parallel Systems: A Survey". Technical Report UNSW_CSE_TR_9707, School of Computer Science and Engineering, The University of New South Sales, Sidney, Australia.

Li, K. C. (2001) "Análise e Predição de Desempenho de Programas Paralelos em Redes de Estações de Trabalho", Tese de doutoramento, Universidade de São Paulo, Escola Politécnica, Departamento de Engenharia de Computação e Sistemas Digitais, outubro.

Li, K. C., Sato, L. M. and Gaudiot, J.-L. (2001) "A Tool for Performance Analysis and Prediction of Parallel Computing on NOW", In: Proceedings of the Int'l Conference on Parallel and Distributed Processing Techniques and Applications (PDPTA'2001), Las Vegas, USA, p.180-185, June.

Li, K. C.; Sato, L. M. (1999) "Using Parasys for parallel algorithm performance analysis". In: Proceedings of the International Congress on Informatic Engineering (ICIE'99), Buenos Aires, Argentina.

Li, Y.S.; Malik, S. (1995) "Performance Analysis on embedded software using implicit path enumeration". In: Proceedings of the ACM SIGPLAN Workshop on Languages, Compilers, and Tools for Real-Time systems, p.95-105, La Jolla, EUA.

Martin, R. P. et al (1997). "Effects of Communication Latency, Overhead, and Bandwidth in a Cluster Architecture". In: Proceedings of the 24th Annual International Symposium on Computer Architecture, Denver, Colorado, p. 85-97.

MPI Fórum (1994), "MPI: A Message Passing Interface Standard", Technical Report UT-CS-94-230, Department of Computer Science, Michigan State University, September.

Mucci, P. J., London, K., and Thurman, J. (1998) "The MPBench report". Technical report, Department of Computer Science, University of Tenessee, November.

Nupairoj, N. and Ni, L. (1994) "Performance Evaluation of Some MPI Implementations". Technical Report MSU-CPS-ACS-94, Department of Computer Science, Michigan State University, September.

Oliveira, H. M., Laine, J. M., Midorikawa, E. T., Li, K. C., Gaudiot, J.-L. (2002) "Performance Analysis and Prediction of Some MPI Communication Primitives". Accepted in: Int'l Conference on Parallel and Distributed Processing Techniques and Applications (PDPTA'2002), Las Vegas, USA.

Pfister, G. F., In Search of Clusters, Prentice-Hall, 1998.

Reed, D. A. (1993) "Performance Instrumentation Techniques for Parallel Systems", Performance Evaluation, p. 463-490.

The Ohio State University (1996). "MPI Primer: Developing with LAM". Columbus, Ohio. http://www.lam-mpi.org/download/files/lam61.nol.doc.pdf, March.

University of Tennessee (1995). "MPI: A Message-Passing Interface Standard". Knoxville, Tennessee. http://www.mpi-forum.org/docs/mpi-11-html/mpi-report.html#Node0, March.

BOLETINS TÉCNICOS - TEXTOS PUBLICADOS

BT/PCS/9301 - Interligação de Processadores através de Chaves Ômicron - GERALDO LINO DE CAMPOS, DEMI GETSCHKO

BT/PCS/9302 - Implementação de Transparência em Sistema Distribuído - LUÍSA YUMIKO AKAO, JOÃO JOSÉ NETO

BT/PCS/9303 - Desenvolvimento de Sistemas Especificados em SDL - SIDNEI H. TANO, SELMA S. S. MELNIKOFF

BT/PCS/9304 - Um Modelo Formal para Sistemas Digitais à Nível de Transferência de Registradores - JOSÉ EDUARDO MOREIRA, WILSON VICENTE RUGGIERO

BT/PCS/9305 - Uma Ferramenta para o Desenvolvimento de Protótipos de Programas Concorrentes - JORGE KINOSHITA, JOÃO JOSÉ NETO

BT/PCS/9306 - Uma Ferramenta de Monitoração para um Núcleo de Resolução Distribuída de Problemas Orientado a Objetos - JAIME SIMÃO SICHMAN, ELERI CARDOSO

BT/PCS/9307 - Uma Análise das Técnicas Reversíveis de Compressão de Dados - MÁRIO CESAR GOMES SEGURA, EDIT GRASSIANI LINO DE CAMPOS

BT/PCS/9308 - Proposta de Rede Digital de Sistemas Integrados para Navio - CESAR DE ALVARENGA JACOBY, MOACYR MARTUCCI JR.

BT/PCS/9309 - Sistemas UNIX para Tempo Real - PAULO CESAR CORIGLIANO, JOÃO JOSÉ NETO

BT/PCS/9310 - Projeto de uma Unidade de Matching Store baseada em Memória Paginada para uma Máquina Fluxo de Dados Distribuído - EDUARDO MARQUES, CLAUDIO KIRNER

BT/PCS/9401 - Implementação de Arquiteturas Abertas: Uma Aplicação na Automação da Manufatura - JORGE LUIS RISCO BECERRA, MOACYR MARTUCCI JR.

BT/PCS/9402 - Modelamento Geométrico usando do Operadores Topológicos de Euler - GERALDO MACIEL DA FONSECA, MARIA ALICE GRIGAS VARELLA FERREIRA

BT/PCS/9403 - Segmentação de Imagens aplicada a Reconhecimento Automático de Alvos - LEONCIO CLARO DE BARROS NETO, ANTONIO MARCOS DE AGUIRRA MASSOLA

BT/PCS/9404 - Metodologia e Ambiente para Reutilização de Software Baseado em Composição - LEONARDO PUJATTI, MARIA ALICE GRIGAS VARELLA FERREIRA

BT/PCS/9405 - Desenvolvimento de uma Solução para a Supervisão e Integração de Células de Manufatura Discreta - JOSÉ BENEDITO DE ALMEIDA, JOSÉ SIDNEI COLOMBO MARTINI

BT/PCS/9406 - Método de Teste de Sincronização para Programas em ADA - EDUARDO T. MATSUDA, SELMA SHIN SHIMIZU MELNIKOFF

BT/PCS/9407 - Um Compilador Paralelizante com Detecção de Paralelismo na Linguagem Intermediária - HSUEH TSUNG HSIANG, LÍRIA MATSUMOTO SAITO

BT/PCS/9408 - Modelamento de Sistemas com Redes de Petri Interpretadas - CARLOS ALBERTO SANGIORGIO, WILSON V. RUGGIERO

BT/PCS/9501 - Sintese de Voz com Qualidade - EVANDRO BACCI GOUVÊA, GERALDO LINO DE CAMPOS

BT/PCS/9502 - Um Simulador de Arquiteturas de Computadores "A Computer Architecture Simulator" - CLAUDIO A. PRADO, WILSON V. RUGGIERO

BT/PCS/9503 - Simulador para Avaliação da Confiabilidade de Sistemas Redundantes com Reparo - ANDRÉA LUCIA BRAGA, FRANCISCO JOSÉ DE OLIVEIRA DIAS

BT/PCS/9504 - Projeto Conceitual e Projeto Básico do Nível de Coordenação de um Sistema Aberto de Automação, Utilizando Conceitos de Orientação a Objetos - NELSON TANOMARU, MOACYR MARTUCCI JUNIOR

BT/PCS/9505 - Uma Experiência no Gerenciamento da Produção de Software - RICARDO LUIS DE AZEVEDO DA ROCHA, JOÃO JOSÉ NETO

BT/PCS/9506 - MétodOO - Método de Desenvolvimento de Sistemas Orientado a Objetos: Uma Abordagem Integrada à Análise Estruturada e Redes de Petri - KECHI HIRAMA, SELMA SHIN SHIMIZU MELNIKOFF

BT/PCS/9601 - MOOPP: Uma Metodologia Orientada a Objetos para Desenvolvimento de Software para Processamento Paralelo - ELISA HATSUE MORIYA HUZITA, LÍRIA MATSUMOTO SATO

BT/PCS/9602 - Estudo do Espalhamento Brillouin Estimulado em Fibras Ópticas Monomodo - LUIS MEREGE SANCHES, CHARLES ARTUR SANTOS DE OLIVEIRA

BT/PCS/9603 - Programação Paralela com Variáveis Compartilhadas para Sistemas Distribuídos - LUCIANA BEZERRA ARANTES, LIRIA MATSUMOTO SATO

BT/PCS/9604 - Uma Metodologia de Projeto de Redes Locais - TEREZA CRISTINA MELO DE BRITO CARVALHO, WILSON VICENTE RUGGIERO

BT/PCS/9605 - Desenvolvimento de Sistema para Conversão de Textos em Fonemas no Idioma Português - DIMAS TREVIZAN CHBANE, GERALDO LINO DE CAMPOS

BT/PCS/9606 - Sincronização de Fluxos Multimídia em um Sistema de Videoconferência - EDUARDO S. C. TAKAHASHI, STEFANIA STIUBIENER

BT/PCS/9607 - A importância da Completeza na Especificação de Sistemas de Segurança - JOÃO BATISTA CAMARGO JÚNIOR, BENÍCIO JOSÉ DE SOUZA

BT/PCS/9608 - Uma Abordagem Paraconsistente Baseada em Lógica Evidencial para Tratar Exceções em Sistemas de Frames com Múltipla Herança - BRÁULIO COELHO ÁVILA, MÁRCIO RILLO

BT/PCS/9609 - Implementação de Engenharia Simultânea - MARCIO MOREIRA DA SILVA, MOACYR MARTUCCI JÚNIOR

BT/PCS/9610 - Statecharts Adaptativos - Um Exemplo de Aplicação do STAD - JORGE RADY DE ALMEIDA JUNIOR, JOÃO JOSÉ NETO

BT/PCS/9611 - Um Meta-Editor Dirigido por Sintaxe - MARGARETE KEIKO IWAI, JOÃO JOSÉ NETO

BT/PCS/9612 - Reutilização em Software Orientado a Objetos: Um Estudo Empírico para Analisar a Dificuldade de Localização e Entendimento de Classes - SELMA SHIN SHIMIZU MELNIKOFF, PEDRO ALEXANDRE DE OLIVEIRA GIOVANI

BT/PCS/9613 - Representação de Estruturas de Conhecimento em Sistemas de Banco de Dados - JUDITH PAVÓN MENDONZA, EDIT GRASSIANI LINO DE CAMPOS

BT/PCS/9701 - Uma Experiência na Construção de um Tradutor Inglês - Português - JORGE KINOSHITA, JOÃO JOSÉ NETO

BT/PCS/9702 - Combinando Análise de "Wavelet" e Análise Entrópica para Avaliar os Fenômenos de Difusão e Correlação - RUI CHUO HUEI CHIOU, MARIA ALICE G. V. FERREIRA

BT/PCS/9703 - Um Método para Desenvolvimento de Sistemas de Computacionais de Apoio a Projetos de Engenharia - JOSÉ EDUARDO ZINDEL DEBONI, JOSÉ SIDNEI COLOMBO MARTINI

BT/PCS/9704 - O Sistema de Posicionamento Global (GPS) e suas Aplicações - SÉRGIO MIRANDA PAZ, CARLOS EDUARDO CUGNASCA

BT/PCS/9705 - METAMBI-OO - Um Ambiente de Apoio ao Aprendizado da Técnica Orientada a Objetos - JOÃO UMBERTO FURQUIM DE SOUZA, SELMA S. S. MELNIKOFF

BT/PCS/9706 - Um Ambiente Interativo para Visualização do Comportamento Dinâmico de Algoritmos - IZAURA CRISTINA ARAÚJO, JOÃO JOSÉ NETO

BT/PCS/9707 - Metodologia Orientada a Objetos e sua Aplicação em Sistemas de CAD Baseado em "Features" - CARLOS CÉSAR TANAKA, MARIA ALICE GRIGAS VARELLA FERREIRA

BT/PCS/9708 - Um Tutor Inteligente para Análise Orientada a Objetos - MARIA EMÍLIA GOMES SOBRAL, MARIA ALICE GRIGAS VARELLA FERREIRA

BT/PCS/9709 - Metodologia para Seleção de Solução de Sistema de Aquisição de Dados para Aplicações de Pequeno Porte - MARCELO FINGUERMAN, JOSÉ SIDNEI COLOMBO MARTINI

BT/PCS/9801 - Conexões Virtuais em Redes ATM e Escalabilidade de Sistemas de Transmissão de Dados sem Conexão - WAGNER LUIZ ZUCCHI, WILSON VICENTE RUGGIERO

BT/PCS/9802 - Estudo Comparativo dos Sistemas da Qualidade - EDISON SPINA, MOACYR MARTUCCI JR.

BT/PCS/9803 – The VIBRA Multi-Agent Architecture: Integrating Purposive Vision With Deliberative and Reactive Planning – REINALDO A. C. BIANCHI , ANNA H. REALI C. RILLO, LELIANE N. BARROS

BT/PCS/9901 – Metodologia ODP para o Desenvolvimento de Sistemas Abertos de Automação – JORGE LUIS RISCO BECCERRA, MOACYR MARTUCCI JUNIOR

BT/PCS/9902 – Especificação de Um Modelo de Dados Bitemporal Orientado a Objetos – SOLANGE NICE ALVES DE SOUZA, EDIT GRASSIANI LINO DE CAMPOS

BT/PCS/9903 – Implementação Paralela Distribuída da Dissecação Cartesiana Aninhada – HILTON GARCIA FERNANDES, LIRIA MATSUMOTO SATO

BT/PCS/9904 – Metodologia para Especificação e Implementação de Solução de Gerenciamento – SERGIO CLEMENTE, TEREZA CRISTINA MELO DE BRITO CARVALHO

BT/PCS/9905 – Modelagem de Ferramenta Hipermídia Aberta para a Produção de Tutoriais Interativos – LEILA HYODO, ROMERO TORI

BT/PCS/9906 – Métodos de Aplicações da Lógica Paraconsistente Anotada de Anotação com Dois Valores-LPA2v com Construção de Algoritmo e Implementação de Circuitos Eletrônicos – JOÃO I. DA SILVA FILHO, JAIR MINORO ABE

BT/PCS/9907 – Modelo Nebuloso de Confiabilidade Baseado no Modelo de Markov – PAULO SÉRGIO CUGNASCA, MARCO TÚLIO CARVALHO DE ANDRADE

BT/PCS/9908 – Uma Análise Comparativa do Fluxo de Mensagens entre os Modelos da Rede Contractual (RC) e Colisões Baseada em Dependências (CBD) – MÁRCIA ITO, JAIME SIMÃO SICHMAN

BT/PCS/9909 – Otimização de Processo de Inserção Automática de Componentes Eletrônicos Empregando a Técnica de Times Assíncronos – CESAR SCARPINI RABAK, JAIME SIMÃO SICHMAN

BT/PCS/9910 – MIISA – Uma Metodologia para Integração da Informação em Sistemas Abertos – HILDA CARVALHO DE OLIVEIRA, SELMA S. S. MELNICOFF

BT/PCS/9911 – Metodologia para Utilização de Componentes de Software: um estudo de Caso – KAZUTOSI TAKATA, SELMA S. S. MELNIKOFF

BT/PCS/0001 – Método para Engenharia de Requisitos Norteado por Necessidades de Informação – ARISTIDES NOVELLI FILHO, MARIA ALICE GRIGAS VARELLA FERREIRA

BT/PCS/0002 – Um Método de Escolha Automática de Soluções Usando Tecnologia Adaptativa – RICARDO LUIS DE AZEVEDO DA ROCHA, JOÃO JOSÉ NETO

BT/PCS/0101 – Gerenciamento Hierárquico de Falhas – JAMIL KALIL NAUFAL JR., JOÃO BATISTA CAMARGO JR.

BT/PCS/0102 – Um Método para a Construção de Analisadores Morfológicos, Aplicado à Língua Portuguesa, Baseado em Autômatos Adaptativos – CARLOS EDUARDO DANTAS DE MENEZES, JOÃO JOSÉ NETO

BT/PCS/0103 – Educação pela Web: Metodologia e Ferramenta de Elaboração de Cursos com Navegação Dinâmica – LUISA ALEYDA GARCIA GONZÁLEZ, WILSON VICENTE RUGGIERO

BT/PCS/0104 – O Desenvolvimento de Sistemas Baseados em Componentes a Partir da Visão de Objetos – RENATA EVANGELISTA ROMARIZ RECCO, JOÃO BATISTA CAMARGO JÚNIOR

BT/PCS/0105 – Introdução às Gramáticas Adaptativas – MARGARETE KEIKO IWAI, JOÃO JOSÉ NETO

BT/PCS/0106 – Automação dos Processos de Controle de Qualidade da Água e Esgoto em Laboratório de Controle Sanitário – JOSÉ BENEDITO DE ALMEIDA, JOSÉ SIDNEI COLOMBO MARTINI

BT/PCS/01/07 – Um Mecanismo para Distribuição Segura de Vídeo MPEG – CÍNTIA BORGES MARGI, GRAÇA BESSAN, WILSON VICENTE RUGGIERO

BT/PCS/0108 – A Dependence-Based Model for Social Reasoning in Multi-Agent Systems – JAIME SIMÃO SICHMAN

BT/PCS/0109 – Ambiente Multilinguagem de Programação – Aspectos do Projeto e Implementação – APARECIDO VALDEMIR DE FREITAS, JOÃO JOSÉ NETO

BT/PCS/0110 – LETAC: Técnica para Análise de Tarefas e Especificação de Fluxo de Trabalho Cooperativo – MARCOS ROBERTO GREINER, LUCIA VILELA LEITE FILGUEIRAS

BT/PCS/0111 – Modelagem ODP para o Planejamento de Sistemas de Potência – ANIRIO SALLES FILHO, JOSÉ SIDNEI COLOMBO MARTINI

BT/PCS/0112 – Técnica para Ajuste dos Coeficientes de Quantização do Padrão MPEG em Tempo Real – REGINA M. SILVEIRA, WILSON V. RUGGIERO

BT/PCS/0113 – Segmentação de Imagens por Classificação de Cores: Uma Abordagem Neural – ALEXANDRE S. SIMÕES, ANNA REALI COSTA

BT/PCS/0114 - Uma Avaliação do Sistema DSM Nautilus –MARIO DONATO MARINO, GERALDO LINO DE CAMPOS

BT/PCS/0115 – Utilização de Redes Neurais Artificiais para Construção de Imagem em Câmara de Cintilação – LUIZ SÉRGIO DE SOUZA, EDITH RANZINI

BT/PCS/0116 – Simulação de Redes ATM – HSU CHIH WANG CHANG, WILSON VICENTE RUGGIERO

BT/PCS/0117 – Application of Monoprocessed Architecture for Safety Critical Control Systems – JOSÉ ANTONIO FONSECA, JORGE RADY DE ALMEIDA JR.

BT/PCS/0118 – WebBee – Um Sistema de Informação via WEB para Pesquisa de Abelhas sem Ferrão – RENATO SOUSA DA CUNHA, ANTONIO MOURA SARAIVA

BT/PCS/0119 – Parallel Processing Applied to Robot Manipulator Trajectory Planning – DENIS HAMILTON NOMIYAMA, LÍRIA MATSUMOTO SATO, ANDRÉ RIYUITI HIRAKAWA

BT/PCS/0120 – Utilização de Padrão de Arquitetura de Software para a Fase de Projeto Orientado a Objetos – CRISITINA MARIA FERREIRA DA SILVA, SELMA SHIN SHIMIZU MELNIKOFF

BT/PCS/0121 – Agilizando Aprendizagem por Reforço Através do uso de Conhecimento sobre o Domínio – RENÊ PEGORARO, ANNA H. REALI COSTA

BT/PCS/0122 – Modelo de Segurança da Linguagem Java Problemas e Soluções – CLAUDIO MASSANORI MATAYOSHI, WILSON VICENTE RUGGIERO

BT/PCS/0123 – Proposta de um Agente CNM para o Gerenciamento Web de um Backbone ATM – FERNANDO FROTA REDÍGOLO, TEREZA CRISTINA MELO DE BRITO CARVALHO

BT/PCS/0124 – Um Método de Teste de software Baseado em Casos Teste – SÉRGIO RICARDO ROTTA, KECHI HIRAMA

BT/PCS/0201 – A Teoria Nebulosa Aplicada a uma Bicicleta Ergométrica para Fisioterapia – MARCO ANTONIO GARMS, MARCO TÚLIO CARVALHO DE ANDRADE

BT/PCS/0202 – Synchronization Constraints in a Concurrent Object Oriented Programming Model – LAÍS DO NASCIMENTO SALVADOR, LIRIA MATSUMOTO SATO