Classificação de Técnicas para Análise de Desempenho de Sistemas Paralelose Distribuídos

Etore Marcari Jr.*
DEE/FEIS/Unesp
Ilha Solteira, SP
etore@dee.feis.unesp.br

Aleardo Manacero Jr.
DCCE/Ibilce/Unesp
S. J. do Rio Preto, SP
aleardo@dcce.ibilce.unesp.br

Resumo

A área de análise de desempenho de sistemas paralelos e distribuídos apresenta uma grande variedade de ferramentas e técnicas para análise. Tais ferramentas e técnicas podem ser agrupadas segundo diferentes estratégias de classificação. Isto acaba por dificultar o processo de escolha da ferramenta mais adequada para uma dada aplicação. Este artigo faz a classificação de ferramentas para análise de desempenho de sistemas paralelos a partir das estratégias existentes no meio científico, facilitando o processo de escolha da ferramenta adequada.

1 Introdução

Em qualquer sistema, computacional ou não, seu bom desempenho é um fator que deve ser buscado durante todo o seu desenvolvimento. Em sistemas computacionais um desempenho ruim implica na necessidade de mais tempo para resolver o problema ou de máquinas mais potentes para sua execução. Esses dois parâmetros representam custos associados com a manutenção de pessoal e máquinas na execução do serviço em que o sistema (hardware ou software) está aplicado. Como esses custos são proporcionais ao tempo necessário para executar a tarefa e ao custo do ambiente em que se faz o processamento, fica claro que um sistema com baixo desempenho significa alto custo. Sendo assim, a busca pelo desempenho ótimo se torna importante para sistemas que sejam muito utilizados ou que tenham carga computacional elevada.

Dada essa importância, é necessário alguma forma de se medir ou dimensionar o desempenho desses sistemas. Para realizar esse dimensionamento são utilizadas ferramentas para medição e análise do desempenho, que disponibilizam ao analista medidas e métricas distintas. Essas medidas e métricas variam também com a aplicação, existindo, por exemplo, técnicas explicitamente projetadas para serviços como web ou bancos de dados.

A escolha da ferramenta ou método para análise de desempenho adequado é um fator importante na obtenção de medidas confiáveis para cada usuário. Devido à diversidade de medidas de desempenho [Sahni, 1996] e de ferramentas existentes no mercado, encontrar a ferramenta mais adequada para uma dada aplicação torna-se uma tarefa difícil. Este processo torna-se mais complexo quando se busca ferramentas ou técnicas para análise de sistemas paralelos, quando o desempenho passa a depender também de um bom sincronismo entre os vários processos.

Neste trabalho será apresentada a classificação de diversas técnicas e ferramentas disponíveis para a análise de desempenho de sistemas paralelos e distribuídos, segundo diversas estratégias de classificação. Como descrito a seguir, as estratégias escolhidas aqui representam os modos clássicos (e portanto relativamente padronizados) de se classificar técnicas e ferramentas. Do mesmo modo, as técnicas e ferramentas que aparecem classificadas foram escolhidas primeiro por sua relevância no meio científico, e depois por similaridades de abordagem (buscando uma boa cobertura de abordagens). Finalmente, embora o objetivo seja abranger ferramentas para sistemas paralelos e distribuídos, que generalizamos como paralelos daqui por diante, parte das estratégias de classificação e de ferramentas de análise podem, também ser usadas para sistemas seqüenciais.

Com essa sistematização procura-se diminuir a dificuldade de escolha de uma ferramenta ou técnica para análise de desempenho. Como resultado dessa classificação espera-se que um usuário tenha melhores condições de fazer a escolha da ferramenta ou técnica, ao tê-las agrupadas de modo coerente.

Na seção dois são apresentadas as principais estratégias de classificação de ferramentas para análise de desempenho; na seção seguinte são apresentadas resumidamente algumas ferramentas e técnicas; na seção quatro se apresenta

*Esse trabalho tem o apoio da FAPESP - Fundação de Amparo à Pesquisa do Estado de São Paulo, processo 00/04933-1.

a classificação das ferramentas e técnicas estudadas e, depois dela, apresenta-se as conclusões tiradas desse trabalho.

2 Estratégias de classificação

Dentre as diversas formas para classificação de ferramentas de predição e análise de desempenho, algumas podem ser consideradas como mais abrangentes e outras como de grande uso. A sistematização de uma classificação de ferramentas e técnicas procura, essencialmente, determinar o enquadramento de cada uma sob um dado critério. Os critérios de classificação aqui apresentados representam situações extremas (arquitetura e tipo de sistema) e de abordagem (tipo de instrumentação, medidas e análise), sendo escolhidos por cobrirem todo o espectro de opções de escolha. São eles:

- tipo de sistema (se hardware ou software) em que o método ou técnica é empregado;
- abordagem usada na obtenção de resultados;
- modo de instrumentação [Pierce, 1994];
- tipo de medida [Reed, 1994];
- arquitetura da aplicação em que é empregado.

A seguir são apresentadas as classificações utilizadas neste trabalho, juntamente com algumas considerações sobre cada uma delas.

2.1 Quanto ao sistema aplicado

Pode-se classificar as técnicas de análise e predição de desempenho em dois grandes grupos: técnicas que estudam sistemas de hardware e técnicas que estudam sistemas de software. Esta classificação, apesar de simples e imediata, é importante na diferenciação entre os vários métodos.

Técnicas de análise de software medem o desempenho de programas computacionais, com o objetivo de identificar funções ou blocos de instruções que representem gargalos de execução no sistema.

Técnicas de análise de hardware são as técnicas que se preocupam em analisar o desempenho das máquinas em que serão executados os programas paralelos. Ferramentas nessa categoria podem medir o desempenho do sistema como um todo ou de módulos individuais, como subsistemas de memória, de entrada/saída, de comunicação, etc.

2.2 Quanto a abordagem na obtenção de resultados

Tanto para a análise de desempenho de hardware quanto para a análise de desempenho de software pode-se dividir os métodos em três grupos, segundo a abordagem utilizada na obtenção dos resultados: métodos analíticos, métodos baseados em simulação e métodos baseados em *benchmarking*. Essa é uma estratégia de classificação clássica, sendo quase um padrão na área.

Métodos analíticos são baseados em modelos matemáticos, como Redes de Petri e Cadeias de Markov. Consistem no principal meio de predição do desempenho de sistemas que ainda não estão fisicamente disponíveis.

A criação de um modelo analítico preciso para determinado sistema é muito difícil, devido à complexidade do seu processo de desenvolvimento do modelo, especialmente quando se trata de sistemas paralelos. Embora procurem fazer uma determinação analítica e provavelmente exata do ponto ótimo de operação, tais métodos têm a aplicabilidade condicionada à obtenção de um equacionamento preciso e computacionalmente factível do sistema, no qual devem ser levados em consideração uma quantidade elevada de parâmetros.

Os métodos baseados em simulação se assemelham muito com os métodos analíticos (grande parte desses métodos é baseada na simulação de eventos em grafos dirigidos e em Redes de Petri). A diferença entre métodos analíticos e de simulação está na maneira como os resultados são obtidos. Nos métodos baseados em simulação, no lugar das equações matemáticas, existem regras de comportamento que definem o comportamento dos eventos e dos estados do sistema. O grande problema no uso de simulação reside na obtenção de um modelo que seja fiel ao problema real.

Métodos baseados em *benchmarking* apresentam a menor complexidade na implementação. Por outro lado, exigem que a máquina em que serão realizadas as medições esteja disponível, o que faz com que seu custo seja elevado. A técnica de medição resume-se em executar o programa na máquina alvo e retirar todos os dados necessários para realizar a análise do desempenho do sistema.

A partir das características desses métodos, pode-se listar um conjunto de parâmetros que ajudam o usuário na decisão de qual técnica será aplicada para análise ou predição de desempenho. Tais parâmetros são listados na tabela 1, tirada de Jain [Jain, 1991], ordenados em importância decrescente.

O parâmetro chave na escolha da técnica para análise de desempenho é o estágio de desenvolvimento em que o sistema se encontra, ou seja, se um protótipo existe ou não. O segundo parâmetro é o tempo disponível para a análise, quando se examina a urgência ou não das informações. Outro parâmetro consiste na disponibilidade de ferramentas, incluindo perícia em modelagem, linguagens de simulação, e instrumentos de medição.

O nível de exatidão desejado é outro importante parâmetro. Em geral, modelagens analíticas requerem muitas simplificações, degradando a exatidão, enquanto que

simulações podem incorporar mais detalhes, tornando os métodos mais precisos.

Um quinto critério é a facilidade com que se pode avaliar alternativas ao sistema testado, procurando identificar possíveis melhorias nele. Outro parâmetro na escolha, bastante importante por sinal, é o custo do processo de análise de desempenho a ser utilizado, procurando determinar o melhor para o orçamento disponível para a tarefa. Por fim, ainda é preciso julgar se os resultados terão fácil receptividade na empresa, o que pode ser determinado pela credibilidade do método utilizado, o que as vezes depende da capacidade de compreensão dele pelos usuários.

2.3 Quanto ao modo de instrumentação

Outra forma de classificação, voltada para análise de sistemas de software, é a apresentada por Pierce e Mudge [Pierce, 1994], que divide as ferramentas entre as que fazem monitoração e as que fazem modificação de código do programa. A monitoração pode ser feita por hardware ou por chamadas controladas pelo sistema operacional (software). Já a modificação de código pode ser feita tanto no próprio código fonte como no objeto ou diretamente no executável, diferindo entre si pela etapa do processo de produção de um programa em que ocorre a modificação do código.

Monitoração é realizada através de equipamentos, como analisadores lógicos, quando feita por hardware, ou por coleta de informações a partir de primitivas do sistema operacional, quando feita por software. Em qualquer caso a precisão é boa mas obtida a um custo bastante alto, além de exigir a presença de profissionais especializados.

A modificação do código fonte consiste na inserção de comandos especiais no programa (chamadas ao relógio do sistema) para obtenção das medidas de desempenho. É considerada a forma mais fácil de modificação, apesar da necessidade do código fonte do programa (nem sempre disponível) e da imprecisão dos resultados introduzida pelo acréscimo nem sempre mensurável de código.

A modificação do código objeto é realizada inserindo os comandos para as medições no código objeto, o que pode ser realizado por um editor específico que reescreve o código na linguagem original e depois refaz a alocação do código e dos dados para torna-lo objeto novamente. Apresenta os mesmos problemas encontrados com a modificação do código fonte.

A modificação do código executável é realizada através da inserção de comandos já no código executável. Consiste na forma mais complexa de inserção de código, pois exige que o usuário possua um decompilador ou um programa que realize a adição do código no arquivo binário.

Os três tipos de instrumentação apresentam problemas (como possível aumento no tempo de execução e perda de precisão) por serem métodos invasivos. Sendo assim, pode-se concluir que monitoração é precisa porém cara, além de exigir especialistas, enquanto a modificação de código é uma tarefa simples e menos custosa e precisa, sendo, em geral, a técnica preferida em ferramentas para análise de desempenho.

2.4 Quanto ao tipo de medida

A classificação proposta por Reed [Reed, 1994] consiste na divisão das técnicas de modificação de código em grupos, de acordo com o tipo de informação resultante da medição e a forma como é realizada. Reed descreve quatro categorias: *profiling*, contagem de eventos, medição de intervalos de tempo e extração de traços de evento.

A idéia principal de ferramentas baseadas em *profiling* é a obtenção de informações sobre o quanto cada trecho do programa ocupa do tempo total de execução. *Profilers* obtêm os dados sobre o uso do processador através da amostragem do contador de programas em intervalos fixos, o que causa problemas quanto à precisão das medidas realizadas, pelo tamanho do intervalo de amostragem.

Métodos de contagem de eventos modificam o código para que se conte ocorrências de determinados eventos (como o número de vezes que um laço do programa é executado, por exemplo). Tal modificação causa acréscimo no tempo de execução do programa e também das instruções que estiverem sendo contadas.

A medição de intervalos de tempos ocorre através da inserção de chamadas para medir o relógio do sistema e posterior soma dos tempos medidos para obtenção dos resultados estatísticos. Como acontece em algumas ferramentas de *profiling*, suas medidas são imprecisas devido à baixa freqüência de atualização do relógio do sistema.

Extração de traços de eventos, ou *event tracing*, é a técnica que fornece os resultados mais detalhados sobre os eventos do sistema, pois baseia-se em registros completos da ocorrência de cada evento. Tal precisão vem de um volume demasiadamente grande de dados pela alta freqüência em que eventos ocorrem.

Das técnicas indicadas, tem-se que *profiling* e extração de traços de eventos são as mais usadas nas ferramentas de análise de desempenho existentes.

2.5 Quanto a arquitetura da aplicação

A última forma de classificação aqui utilizada divide as ferramentas pelo modelo de arquitetura da máquina em que elas são empregadas. Existem quatro modelos clássicos de arquitetura (classificação de Flynn [Flynn, 1972]), que são as máquinas SISD, SIMD, MISD, MIMD, que serão usados como referencial para esta classificação.

Embora pareça óbvio, essa classificação é necessária pois ferramentas projetadas para uma arquitetura SIMD

Critério	Métodos Analíticos	Simulação	*Benchmarking*
1. Estágio de desenvolvimento	qualquer	qualquer	pós-protótipo
2. Tempo na obtenção de resultados	pequeno	médio	varia
3. Ferramentas para análise	analistas humanos	linguagem de computadores	instrumentação
4. Exatidão	baixa	moderada	varia
5. Avaliação de alternativas	fácil	moderada	difícil
6. Custo	baixo	médio	alto
7. Credibilidade do método	baixo	médio	alto

Tabela 1. Critérios para escolha de técnicas de análise ou predição de desempenho.[Jain, 1991]

provavelmente terão uma precisão bastante baixa quando aplicadas a uma arquitetura MIMD, por exemplo. Assim, é interessante que o analista de desempenho tenha conhecimento sobre para que arquitetura a ferramenta foi projetada.

3 Técnicas de análise de desempenho

Dentre as diversas técnicas de análise de desempenho, algumas se destacam por serem constantemente referenciadas, outras por seu uso no meio empresarial. Classificar todas as ferramentas e técnicas já apresentadas na literatura da área não é necessário pois nem todas conseguem alguma repercussão. Assim, as técnicas apresentadas nesta seção foram escolhidas a partir de critérios relativamente subjetivos, como a freqüência de citações na literatura, que é um indicativo de sua importância teórica, e a abordagem de medição utilizada, que permite que sejam apresentadas técnicas em diferentes categorias segundo as várias estratégias de classificação.

As técnicas descritas resumidamente aqui, agrupadas pela abordagem utilizada para a obtenção dos resultados (analíticas, simulação e *benchmarking*). Em especial, no caso de *benchmarking* separa-se as técnicas entre as aplicadas a hardware e a software. Sempre que possível lista-se outras técnicas, geralmente mais recentes, que tenham funcionamento semelhante.

Técnicas Analíticas

– **GSPN [Gandra, 1992]** - Uma das formas de se aplicar redes de Petri em análise de desempenho é através do uso de GSPNs (Redes de Petri Estocásticas Generalizadas). Uma característica fundamental de GSPN's, que as diferenciam das redes de petri ordinárias, é o fato de serem estocásticas, ou seja, admitirem que as transições possam utilizar probabilidade de disparo (ou tempo de disparo probabilístico para as transições temporizadas).

– **Teoria de filas [Balsamo, 1998]** - Balsamo descreve em seu trabalho um modelo baseado em teoria de filas para o estudo da predição de desempenho de sistemas cliente-servidor heterogêneos, determinando o desempenho a partir dos limitantes calculados de pior e melhor caso.

Técnicas de Simulação

– **PAWS [Pease, 1991]** - PAWS (Parallel Assessment Window System) é um sistema que permite a análise de desempenho de vários tipos de máquinas, mesmo que ainda em fase de desenvolvimento. O sistema é composto por quatro ferramentas: de aplicação, de caracterização de arquitetura, de análise de desempenho e de visualização gráfica, com o usuário modelando seu objeto com as duas primeiras e obtendo resultados com as demais.

– **PDL [Vemuri, 1996]** - PDL (Performance Description Language) é uma linguagem de alto nível que possuí os mesmos objetivos que outras linguagens de descrição de hardware: fornecer um método simples para especificação de um sistema, possibilitando a análise de seu desempenho através da simulação. Originalmente desenvolvida para análise de hardware, pode também ser utilizada para análise do sistema hardware/software.

Ferramentas que também utilizam técnica de simulação são Rsim [Hughes, 2002] e Simics [Magnusson, 2002].

***Benchmarks* para hardware**

– **Linpack [Dongarra, 1992]** - Desenvolvido por Jack Dongarra, este *benchmark* consiste de um conjunto de funções que resolvem sistemas densos de equações lineares. Os programas no Linpack podem ser caracterizados como possuindo um grande número de operações de ponto flutuante (tanto adições quanto multiplicações), e fornecem as medidas de desempenho em Mflops (milhões de operações de ponto flutuante por segundo).

– **STAP [Hwang, 1999]** - O *benchmark* STAP foi originalmente desenvolvido pelo Laboratório do Lincoln MIT, implementado em código C seqüencial, para processamento de sinais de radar em estações de trabalho. Kai Hwang e outros em [Hwang, 1999] tornaram o conjunto de ferramentas aplicáveis à computação paralela. O STAP (tanto em sua versão serial quanto paralela) consiste de cinco programas de processamento de sinais provenientes de radares.

– **SPEC [Uniejewski, 1989]** - SPEC (Standard Performance Evaluation Corporation) é uma corporação formada por grandes fabricantes de computadores que buscam "desenvolver, manter e apoiar um conjunto padronizado de *benchmarks* que possam ser aplicados a mais nova geração de computadores de alto desempenho". Seu pacote é constituído por 15 programas que avaliam os sistemas de alto desempenho tanto através de operações inteiras quanto de ponto flutuante. Os resultados aparecem na forma de SPECmarks, divididos em SPECint e SPECfp com a indicação da

Ferramenta	H/S	Abordagem	Arquitetura	Pierce	Reed
AIMS [Yan, 1995]	S	*Benchmarking*	SIMD, MIMD	Instrumentação do código fonte	Extração de traços
ALPES [Kitajima, 1994]	S	*Benchmarking*, simulação	NA†	NA	NA
Asim [Emer, 2002]	H	Simulação	NA	NA	NA
Enus Total View [Etnus, 2002]	S	*Benchmarking*	SIMD, MIMD	Intrumentação do código fonte ou executável	Extração de traços
IDTrace [Pierce, 1994]	S	*Benchmarking*	NA	Instrumentação do código fonte	Extração de traços
Linpack [Dongarra, 1992]	H	*Benchmarking*	SISD	NA	NA
Medea [Calzarossa, 1996]	S	*Benchmarking*	NA	NA	Extração de traços
Pablo [Reed, 1994]	S	*Benchmarking*	SIMD, MIMD	Instrumentação do código fonte	Extração de traços
Paradyn [Miller, 1995]	S	*Benchmarking*	SIMD, MIMD	Instrumentação do código executável	Extração de traços
PAWS [Pease, 1991]	H/S	Simulação	SIMD, MIMD	NA	NA
PDL [Vemuri, 1996]	H/S	Simulação	SIMD, MIMD	NA	NA
Rsim [Hughes, 2002]	H	Simulação	NA	NA	NA
Simics [Magnusson, 2002]	H/S	Simulação	SIMD, MIMD	NA	NA
SimpleScalar [Austin, 2002]	H/S	Simulação	NA	NA	NA
SPEC benchmark suite [Uniejewski, 1989]	H	*Benchmarking*	SIMD, MIMD	NA	NA
Vampir [Browne, 1997]	S	*Benchmarking*	SIMD, MIMD	Instrumentação do código fonte	Extração de traços
Vispat [Hondroudakis, 1995]	S	*Benchmarking*	SIMD, MIMD	Instrumentação do código fonte	Extração de traços
VT [Browne, 1997]	S	*Benchmarking*	SIMD, MIMD	Intrumentação do código fonte ou executável	Extração de traços, contagem de eventos

† NA = Não se aplica

Tabela 2. Classificação das ferramentas para análise de desempenho.

versão. Al ém disso a SPEC fornece medidas para servidores web e diversas outras aplicações.

***Benchmarks* para software**

– **IDTrace [Pierce, 1994]** - IDTrace é uma ferramenta que realiza a extração de traços de eventos do programa e gera um arquivo com os dados para posterior simulação. É simples, de fácil manuseio e entendimento. A geração do arquivo para simulação é realizada em três etapas (criação do executável, execução da ferramenta, execução do programa gerado pelo IDTrace).

– **Pablo [Reed, 1994]** - Pablo é uma ferramenta desenvolvida na Universidade de Illinois que apresenta uma diversidade de medidas de avaliação bastante elevada. É composta por vários componentes para instrumentação de programas paralelos e para análise de dados produzidos por programas instrumentados. A interação entre os vários componentes da ferramenta é baseada no seu formato para dados (SDDF - Self-Describing Data Format). Esse formato acaba facilitando sua difusão por ser o padrão nativo para a ferramenta proprietária para análise de desempenho ParAide, usada nas antigas máquinas Paragon XP/S da Intel.

Outras ferramentas que realizam *benchmarking* para extração de traços de eventos são AIMS [Yan, 1995], Medea [Calzarossa, 1996], MPE e Vampir [Browne, 1997].

– **Paradyn [Miller, 1995]** - Paradyn é uma ferramenta utilizada para análise de desempenho de programas paralelos de grande escala. É uma ferramenta que realiza a extração de traços de eventos, porém com instrumentação dinâmica, sem que o usuário precise alterar o código fonte do programa se quizer outros dados. Tanto a instrumentação como a análise do desempenho do programa são realizados durante a execução do programa aplicação.

4 Classificação das ferramentas

Como já antecipado, a escolha da ferramenta mais adequada para cada caso depende de vários fatores. Tais fatores são utilizados como critérios de classificação nas estratégias apresentadas na seção 2, sendo que a escolha por um determinado critério depende, fundamentalmente, das características do problema do futuro usuário. Nessa seção apresenta-se a classificação de diversas técnicas e ferramentas, incluindo as introduzidas na seção anterior, segundo as estratégias aqui definidas. Como conclusão à essa classificação apresenta-se também um exemplo de como utilizar as tabelas descritas a seguir.

A tabela 2 apresenta a classificação de várias ferramentas para análise de desempenho, enquanto a tabela 3 apresenta a classificação de técnicas Nas duas tabelas aparecem não apenas as ferramentas e técnicas descritas na seção anterior, como também algumas outras técnicas que foram incluídas por pertecerem a categorias diferentes das demais. Adicionalmente, a tabela 4 fornece os tipos de medidas que cada ferramenta (ou técnica) apresenta ao seu usuário.

Técnica	H/S	Abordagem	Arquitetura	Pierce	Reed
Análise de subsistema de E/S [Ganger, 1998]	H	Simulação	NA†	NA	NA
Análise do tempo de execução de tarefas de comunicação [Kim, 1996]	H	*Benchmarking*	SIMD, MIMD	NA	NA
Caracterização de desempenho da memória compartilhada [Dekker, 1998]	H	*Benchmarking*	MIMD	NA	NA
Escalabilidade da arquitetura de computador vetorial com memória compartilhada	H	*Benchmarking*, analítico	NA	NA	NA
Estudo de casos de prefetching de cache [Hsu, 1998]	H	*Benchmarking*	NA	NA	NA
GSPN [Gandra, 1992]	H	Simulação	NA	NA	NA
Limitantes de desempenho em modelo de teoria de filas [Balsamo, 1998]	H	Analítico	NA	NA	NA
Modelagem de limitante de desempenho [Lui, 1998]	H/S	Analítico	NA	NA	NA
Paralelização do benchmark STAP [Hwang, 1999]	H	Analítico, *benchmarking*	SIMD	NA	Medição de intervalo de tempos

† NA = Não se aplica

Tabela 3. Classificação das técnicas para análise de desempenho.

Ferramenta / Técnica	Medidas que apresenta
AIMS	eventos de entrada/saída, troca de mensagens, tempo de execução do programa e de subrotinas
Conversão do código executável em grafo de execução	tmepo de processamento, velocidade média dos processos, speedup, tempo médio de espera do cliente
Etnus Total View	eventos relacionados a comunicação, entrada/saída e processamento
IDTrace	eventos relacionados a laços da execução, execução do programa, referência à memória
Linpack	número de operações (MFLOPS)
Modelagem de limitante de performance	speedup, valores médios dos tempos de execução
Pablo	estatísticas sobre eventos, rotinas, chamadas e pedidos de E/S
Paradyn	tempo de CPU, taxa de troca de mensagens, utilização de E/S
PAWS	tempo de execução dos programas, grau de paralelismo, speedup e número máximo de operações do sistema
PDL	tempo de execução dos programas, medidas específicas (custo, p.e.)
Redes de Petri	várias, inclusive tempo de execução de um ciclo de sistema, intervalos entre eventos, tempo médio de execução de cada tarefa
SPEC benchmark suite	medidas sobre o sistema, CPU, subsistemas de comunicação, E/S e armazenamento
STAP	tempo de execução, speedup
Vampir / Vampirtrace	estado dos processos, tempo de processamento, quantidade de chamadas a dispositivos de E/S
VT	estado do sistema e dos procesos, eventos de comunicação

Tabela 4. Medidas fornecidas pelas ferramentas

Apesar da separação entre ferramentas e técnicas, ambas tabelas têm suas colunas compostas pelo nome da ferramenta ou técnica e das classificações de acordo com o sistema aplicado, abordagem utilizada na obtenção dos resultados, arquitetura da aplicação, tipo de instrumentação (Pierce e Mudge) e de acordo com o tipo de medida (Reed).

Para ilustrar o uso dessas tabelas e, principalmente, das várias estratégias de classificação, considere, como exemplo, que um analista tem em mãos a tarefa de medir o desempenho de um algoritmo paralelo de ajuste de curvas que será executado em um *cluster Beowulf*. Qual seria então o critério de escolha por uma ferramenta de análise de desempenho?

Primeiro, o que se quer medir é o desempenho de software, o que já restringe em parte o universo de escolha.

Segundo, o *cluster Beowulf* é uma máquina MIMD, com granulação média para alta, restringindo ainda mais as ferramentas.

Terceiro, o usuário deve definir se o algoritmo já foi codificado, o que permite o uso de *benchmarking*, ou não. Supondo que o programa já exista, podemos pensar em ferramentas de *benchmarking*.

Quarto, que tipo de dados são do interesse do usuário. Embora não apareça nas tabelas por questões de espaço, cada ferramenta disponibiliza diferentes tipos de medidas de desempenho. A partir da descrição das ferramentas na seção anterior e da tabela 4, uma opção razoável caso se deseje tempos consumidos em funções do programa e o código fonte estiver disponível, seria o uso de Pablo ou do Etnus Total View (versão comercial). Já se a medida desejada não for previamente conhecida e se queira evitar a execução de muitos *benchmarks*, uma opção seria o Paradyn.

Para outros tipos de situações os passos são exatamente os mesmos, ou seja, verifica-se o tipo de sistema, incluindo sua arquitetura, depois a sua existência física e, finalmente, que tipo de dados interessam. Ao seguir esse procedi-

mento, com os devidos cuidados, pode-se assegurar que a ferramenta escolhida, será no mínimo adequada, podendo ser a ótima em boa parte das ocasiões. Observe-se que a determinação de uma ferramenta ótima depende de fatores como disponibilidade de acesso à mesma.

5 Conclusões e trabalhos futuros

Este trabalho apresentou estratégias para classificação de técnicas e ferramentas para análise de desempenho de sistemas paralelos e distribuídos. O objetivo de tal classificação é auxiliar analistas no processo de escolha da ferramenta/técnica apropriada para análise de desempenho de suas aplicações.

Embora ainda incompleta quanto ao número de ferramentas examinadas, essa classificação já permite aos usuários de sistemas computacionais uma forma mais rápida e direcionada em escolher a ferramenta para análise de desempenho que seja mais adequada ao seu problema. Essa tarefa é normalmente complexa pela inexistência de um padrão de classificação que seja significativamente abrangente em relação aos vários parâmetros de escolha possíveis.

A falta de certas ferramentas e técnicas não é prejudicial ao conteúdo do trabalho pois as aqui apresentadas e classificadas são as mais representativas na comunidade de usuários de computação de alto desempenho. A omissão de alguma técnica ou ferramenta pode ser facilmente contornada através do exame de similares que estejam aqui apresentadas.

Para a continuidade desse trabalho pode-se vislumbrar vários caminhos. Um deles seria o acréscimo de uma maior quantidade de técnicas e ferramentas para análise de desempenho de sistemas paralelos, ampliando o escopo da atual classificação. Outro possível caminho é o de aprofundar a análise das ferramentas, acrescentando comparações quanto a eficiência e variedade de informações provida por cada uma. Um terceiro caminho é encontrar uma nova forma de classificação que reuna, numa única escala, as principais estratégias aqui descritas.

Referências

[Austin, 2002] Austin, T., L. E. and Ernst, D. (2002). Simplescalar: An infrastructure for computer system modeling. In *Computer*, volume 35, n. 2, pages 59–67.

[Balsamo, 1998] Balsamo, S., D. L. and van Dijk, N. (1998). Bound performance models of heterogeneous parallel processing systems. In *IEEE Trans. on Parallel and Distributed Systems*, volume 9, n. 10, pages 1041–1056.

[Browne, 1997] Browne, S., D. J. and London, K. (1997). Review of performance analysis tools for mpi parallel programs.

[Calzarossa, 1996] Calzarossa, M., et alii (1996). Parallel performance evaluation: the medea tool. In *LNCS 1067, Intl. Conf. and Exhibition on High-Performance Computing and Networking*, pages 522–529.

[Dekker, 1998] Dekker, E. (1998). Architecture scalability of parallel vector computers with shared memory. In *IEEE Trans. on Computers*, volume 47, n. 5, pages 614–624.

[Dongarra, 1992] Dongarra, J. (1992). Performance of various computers using standard linear equations software.

[Emer, 2002] Emer, J., et alii (2002). Asim: A performance model framework. In *Computer*, volume 35, n. 2, pages 68–76.

[Etnus, 2002] Etnus (2002). Totalview 4.1 multiprocess debugger. Natick, MA, EUA.

[Flynn, 1972] Flynn, M. (1972). Some computer organizations and their effectiveness. In *IEEE Trans. on Computers*, volume 21, n. 9, pages 948–960.

[Gandra, 1992] Gandra, M., D. J. and Gregorio, J. (1992). Performance evaluation of parallel systems by using unbounded generalized stochastic petri nets. In *IEEE Trans. on Software Engineering*, volume 18, n. 1, pages 55–71.

[Ganger, 1998] Ganger, G. and Patt, Y. (1998). Using system-level models to evaluate i/o subsystems designs. volume 47, n. 1, pages 667–678.

[Hondroudakis, 1995] Hondroudakis, A., et alii (1995). Performance evaluation and visualization with vispat. In *LNCS 964, Third Intl. Conf. on Parallel Computing Technologies*, pages 180–185.

[Hsu, 1998] Hsu, W.-E. and Smith, J. (1998). A performance study of instruction cache prefetching methods. In *IEEE Trans. on Computers*, volume 47, n. 5, pages 497–508.

[Hughes, 2002] Hughes, C., et alii (2002). Rsim: Simulating shared-memory multiprocessors with ilp processors. In *Computer*, volume 35, n. 2, pages 40–49.

[Hwang, 1999] Hwang, K., W. C. W. C.-L. and Xu, Z. (1999). Resource scaling effects on mpp performance: the stap benchmark implications. In *IEEE Trans. on Parallel and Distributed Systems*, volume 10, n. 5, pages 509–527.

[Jain, 1991] Jain, R. (1991). *The Art of Computer Systems Performance Analysis: Techniques for Experimental Design, Measurement, Simulation, and Modeling*. John Wiley & Sons, 2nd edition.

[Kim, 1996] Kim, J. and Shin, K. (1996). Execution time analysis of communicating tasks in distributed systems. In *IEEE Trans. On Computers*, volume 45, n. 5, pages 572–579.

[Kitajima, 1994] Kitajima, J. and Plateau, B. (1994). Modeling parallel program behaviour in alpes. In *Information and Software Technology*, volume 36, n. 7, pages 457–464.

[Lui, 1998] Lui, J.C.S., M. R. and D., T. (1998). Computing performance bounds of fork-join parallel programs under a multiprocessing environment. In *IEEE Trans. on Parallel and Distributed Systems*, volume 9, n. 3, pages 295–311.

[Magnusson, 2002] Magnusson, P., et alii (2002). Simics: A full system simulation platform. In *IEEE Computer*, volume 35, n. 2, pages 50–58.

[Miller, 1995] Miller, B. (1995). The paradyn parallel performance measurement tool. In *IEEE Computer*, volume 28, n. 11, pages 37–46.

[Pease, 1991] Pease, D., et alii (1991). Paws: a performance evaluation tool for parallel computing systems. In *IEEE Computer*, pages 18–29.

[Pierce, 1994] Pierce, J. and Mudge, T. (1994). Idtrace - a tracing tool for i486 simulation. In University of Michigan, M., editor, *Research Report CSE-TR-203-94*.

[Reed, 1994] Reed, D. (1994). Experimental analysis of parallel systems: Techniques and open problems. In *Joint Symposium on Parallel Processing (JSPP)*, pages 239–256.

[Sahni, 1996] Sahni, S. and Thanvantri, V. (1996). Performance metrics: keeping the focus on run time. In *IEEE Parallel and Distributed Technology*, volume 4, n. 1, pages 43–56.

[Uniejewski, 1989] Uniejewski, J. (1989). Spec benchmark suite: designed for today's advanced systems. In *Techinical Report 1, SPEC Newsletter*.

[Vemuri, 1996] Vemuri, R., M. R. and Meduri, V. (1996). Performance modeling using pdl. In *IEEE Computer*, pages 44–53.

[Yan, 1995] Yan J., S. S. and P., M. (1995). Performance measurement, visualization and modeling of parallel and distributed programs using the aims toolkit. In *Software - Practice and Experience*, volume 25:4, pages 429–461.