Universidade Estadual Paulista Júlio de Mesquita Filho

Faculdade de Engenharia de Ilha Solteira

Departamento de Engenharia Elétrica

Classificação e Comparação de Ferramentas para Análise de Desempenho de Sistemas Paralelos

candidato : Etore Marcari Junior

orientador: Prof. Dr. Aleardo Manacero Jr.

Dissertação submetida à Faculdade de Engenharia de Ilha Solteira da Universidade Estadual Paulista Júlio de Mesquita Filho, para preenchimento dos pré-requisitos parciais para obtenção do Título de Mestre em Engenharia Elétrica.

Dezembro de 2002

Resumo

A área de análise de desempenho de sistemas paralelos e distribuídos apresenta uma grande variedade de ferramentas e técnicas para avaliação. Este fator, aliado à grande diversidade de medidas de desempenho, acaba por dificultar o processo de escolha da ferramenta ou técnica mais adequada para avaliação de uma dada aplicação. Com o objetivo de minimizar tal dificuldade, diversas estratégias foram criadas dividindo as técnicas e ferramentas em grupos de acordo com suas características.

Este trabalho apresenta um estudo sobre as várias formas de classificação de técnicas e ferramentas disponíveis para a análise de desempenho de sistemas paralelos e distribuídos. Ao fazer a classificação dessas técnicas e ferramentas segundo várias abordagens, espera-se fornecer aos usuários e desenvolvedores de sistemas paralelos uma maior facilidade no momento de escolher a ferramenta que deverá aplicar no estudo de desempenho do sistema em desenvolvimento.

Adicionalmente realiza-se a comparação de algumas das ferramentas e técnicas disponíveis, inclusive com testes sobre a funcionalidade e facilidade de uso das mesmas. Esses testes, embora em pequeno volume, podem ajudar decisões finais no processo de escolha pela ferramenta mais adequada a cada problema, servindo portanto como um pequeno repositório de informações sobre análise de desempenho.

Abstract

The field of parallel and distributed systems' performance analysis presents a wide variety of tools and techniques available for use. This variety, along with a large range of performance metrics, makes difficult the process of choosing a tool or technique that is the best match for the evaluation of a given application. Trying to overcome such difficulty, a great number of classification strategies has been created, dividing techniques and tools in groups following certain characteristics.

This work presents a study about several classification strategies that are applied over the available performance evaluation techniques and tools used on parallel and distributed systems. By performing this classification, following several approachs, it is expected that the users and developers of parallel systems may have a more comfortable situation at the moment of choosing a tool to apply in the study of performance of a system under development.

Additionally, the comparison between some available tools and techniques is performed, providing some tests about their functionality and easiness of use. The tests, although in small volume, may help final decisions in the choice process for the most appropriate tool for each problem, acting like a small repository of informations on performance analysis.

Agradecimentos

À Deus, por ter me trazido até aqui e dado forças nas horas em que mais precisava.

Ao prof. Aleardo Manacero Jr., pela orientação, pela paciência e principalmente pelo apoio nos momentos em que os problemas pareciam intransponíveis.

À minha mãe, Wanda, com quem posso partilhar toda minha vida, aflições e anseios, pelo valiosíssimo apoio que tem me dado.

À minha família, Andressa, Ludovico, Heliana e Silvia. Por terem suportado a agonia nos momentos difíceis e fazerem parte da minha vida.

À Lilian, por ter sofrido comigo quando necessário e por ter entendido a necessidade da ausência.

Ao pessoal do PUR - Projeto Universidades Renovadas, pelo apoio nas horas em que mais precisei.

Aos meus colegas de departamento, Carminha, Cristiano Martins, Delberis, Edni, Elizete, Gustavo, Josivaldo, Maira, Mara, Marcelo, Marco, Marilene, Quevedo, Rodrigo, Tadao, Tony, Uender, Wagner e prof. Sérgio, pelo convívio e companheirismo durante esses anos.

Aos meus amigos prof. Dr. Jean Lauand, Angela, Ariana, Jorge Dalvi, Marcelo, Márcia e Thiago, por me acompanharem nas empreitadas noturnas.

Aos meus amigos Henrique Ferrari Marchesi e Osvaldo Nobuo, pela amizade cultivada nestes anos e pelo auxílio na utilização e configuração do Linux e Latex.

Aos funcionários do Departamento de Engenharia Elétrica, Deoclécio, Carlos, Cristina, Sueli; e ao pessoal da SPG - Seção de Pós Graduação, pela simpatia e dedicação ao me atender.

Às pessoas que, mesmo não estando próximas, puderam contribuir: Castejon, Dalton, Leugi, Naninha e tantos outros que não me recordo.

Ao meu amigo Marcio Tadeu, de grande importância no meu processo de adaptação na cidade de Ilha Solteira.

Às pessoas que contribuiram direta ou indiretamente para realização deste trabalho.

À FAPESP - Fundação de Amparo à Pesquisa do Estado de São Paulo, pelo apoio financeiro para realização deste projeto (Processo 00/04933-1).

Dedico esse trabalho aos meus familiares.

Conteúdo

Lista de Figuras

Lista de Tabelas

Capítulo 1

Introdução

Nas últimas décadas, mais intensamente a partir da década de noventa, a avaliação de desempenho de sistemas paralelos vem se tornando uma área da ciência da computação amplamente explorada não só por especialistas e desenvolvedores de sistemas computacionais, mas também por usuários dos mesmos. O aumento da pesquisa nesta área deve-se principalmente ao contínuo avanço tecnológico e à popularização dos sistemas paralelos e distribuídos, como redes locais, a rede mundial de computadores e os chamados supercomputadores.

A avaliação de desempenho é muito importante quando se tem por objetivo minimizar ou eliminar alguns fatores ligados principalmente ao tempo gasto na resolução dos problemas e à necessidade de maior capacidade de processamento das máquinas. Esses dois parâmetros representam custos associados com a manutenção de pessoal e máquinas na execução do serviço em que o sistema está aplicado.

Juntamente com o avanço tecnológico e o crescimento de pesquisas na área crescem também o número e a variedade de técnicas e ferramentas para avaliação de desempenho. Estes fatores, aliados à grande diversidade de medidas de desempenho, dificultam a escolha da ferramenta ou técnica adequada para avaliação de uma dada aplicação.

1.1 Objetivos do trabalho

As muitas técnicas e ferramentas utilizadas na análise e predição de desempenho, apesar de possuirem características diferentes, têm o mesmo objetivo: apresentar ao analista ou usuário formas de estudar o comportamento dos sistemas.

Neste trabalho apresenta-se um estudo sobre a classificação de diversas técnicas e ferramentas disponíveis para a análise de desempenho de sistemas paralelos e distribuídos, segundo diferentes estratégias de classificação. As estratégias escolhidas representam as formas clássicas (e portanto relativamente padronizadas) 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). Deve-se observar que, embora o objetivo seja apresentar ferramentas para sistemas paralelos e distribuídos, que são generalizados 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.

1.2 Descrição do texto

No capítulo dois são apresentados conceitos necessários para a análise de desempenho de sistemas, tanto paralelos quanto seqüenciais. Apresenta-se também as principais estratégias de classificação de ferramentas para análise de desempenho.

No capítulo três são apresentadas algumas ferramentas e técnicas, dispostas de acordo com o sistema aplicado. Embora cubram um bom leque de opções, tais técnicas e ferramentas não representam a totalidade de visões, sendo escolhidas

simplesmente por sua representatividade e pela diversidade das mesmas.

No capítulo quatro se apresenta a classificação das ferramentas e técnicas estudadas, aplicando as estratégias descritas no capítulo anterior. Nele aparece também considerações sobre uma abordagem para escolha da técnica ou ferramenta mais apropriada para um dado problema ou situação.

No capítulo cinco descrevem-se alguns testes realizados com ferramentas que estavam disponíveis para avaliação prática. Tais testes permitem verificar a menor ou maior versatilidade de tais ferramentas em relação ao indicado por seus autores.

No último capítulo apresenta-se as conclusões tiradas desse trabalho, as dificuldades encontradas e as perspectivas para trabalhos futuros.

Capítulo 2

Análise e predição de desempenho de sistemas

As técnicas e ferramentas utilizadas na análise e predição de desempenho podem ser agrupadas segundo diferentes estratégias de classificação. Sua diversidade e a não existência de um padrão único de classificação tornam a escolha de uma delas para realizar análise de desempenho (tanto de sistemas de hardware quanto software) uma tarefa difícil e não trivial. No processo de seleção, um dos itens que merece cuidado especial é o conjunto de medidas obtidas como resultado da análise ou predição de desempenho; a partir dessas medidas o usuário, ou analista, pode obter informações que mostrem de forma mais ou menos precisa o comportamento e o desempenho do sistema analisado.

Na primeira parte deste capítulo são apresentadas algumas métricas utilizadas para avaliação de desempenho de sistemas. Em seguida são apresentadas as estratégias para realização da classificação e comparação das técnicas e ferramentas, tanto as de caráter mais global quanto as de caráter específico.

2.1 Avaliação de desempenho

A avaliação de desempenho consiste na execução de procedimentos, computacionais ou não, visando quantificar o comportamento de um determinado sistema, seja ele harware ou software. A expressão *avaliação de desempenho* pode ser substituída por outros dois termos mais específicos: *análise* e *predição* de desempenho.

A *análise* de desempenho tem por objetivo estudar o desempenho de sistemas existentes (que se encontram disponíveis), compreender seu comportamento e determinar um dimensionamento de seus componentes de modo a identificar pontos para melhoria no desempenho.

Já a *predição* estuda o desempenho de sistemas não disponíveis para análise e aferição[1], que podem não estar disponíveis por estarem sendo projetados ou não poderem ser perturbados. A predição se preocupa em predizer o comportamento dos sistemas, e apresentar ao usuário se o desempenho do sistema é adequado ou não em relação ao esperado.

Para melhor entendimento do texto, deste ponto em diante, o termo *avaliação de desempenho* será utilizado quando o objetivo for referenciar *análise* e *predição* em conjunto.

2.2 Medidas de desempenho

Juntamente com a seleção da ferramenta ou técnica utilizada, a escolha da medida ou do conjunto de medidas de desempenho representa um fator chave durante o processo de avaliação. Uma escolha equivocada de medidas pode fazer com que os resultados sejam espúrios, e não demonstrem o comportamento real do sistema.

Dentro do conjunto de métricas para avaliação de desempenho, algumas se destacam como padrões por serem utilizadas por grande parte dos analistas de desempenho ou por estarem inseridas em ferramentas de avaliação, enquanto que outras são úteis por serem aplicáveis a problemas específicos. Neste sentido, existe

[1] Verificação da exatidão, avaliação.

um consenso sobre a não existência de uma medida universal (que pode ser aplicável a todos os problemas), mas de medidas que melhor ou pior se adequam aos problemas de avaliação.

Quando o sistema alvo da avaliação de desempenho é um sistema serial[2], um determinado conjunto de medidas pode ser utilizado na avaliação. São exemplos de medidas aplicáveis a sistemas seriais: carga no sistema, porcentagem de falha de paginação, tempo de espera em operações de entrada e saída. Já quando o alvo são sistemas paralelos, além das medidas utilizadas na avaliação de sistemas seriais (devidamente modificadas) outras medidas devem ser empregadas, para atender as características específicas de sistemas paralelos (como memória compartilhada, por exemplo). Exemplos dessas medidas são escalabilidade e aceleração ("speedup").

As medidas de desempenho podem ser divididas de várias maneiras. Neste trabalho decidiu-se tomar como parâmetro de classificação a dependência (ou não) das medidas em relação à velocidade de execução dos sistemas. Tal parâmetro foi escolhido pois o tempo de execução de um sistema é considerado um fator de grande importância [Sahni, 1996]. A partir desse parâmetro, pode-se classificar as medidas de desempenho em dependentes de velocidade, e independentes de velocidade.

2.2.1 Medidas dependentes de velocidade

Medidas dependentes de velocidade são aquelas de alguma forma ligadas à *velocidade* (e conseqüentemente ao *tempo*) de execução dos sistemas, sejam eles paralelos ou não. Os dois critérios mais importantes, quando relacionados a sistemas paralelos, são o "speedup" e a eficiência.

Aceleração (speedup)

O fator que mais justifica a necessidade da utilização de sistemas paralelos é o possível ganho de velocidade (e consequënte diminuição no tempo) de execução dos sistemas. Sendo assim, o "speedup", que representa o ganho de velocidade (ou

[2]Entende-se por sistemas seriais sistemas de hardware baseados na máquina de Von Neumann e sistemas de software que possuam linha de execução seqüencial.

aceleração) no processamento, torna-se a métrica de maior importância quando analisados sistemas desse tipo.

A partir do conceito de ganho de velocidade, várias abordagens foram criadas para o cálculo do "speedup". A primeira delas, apresentada por Amdahl [Amdahl, 1967], mostrava que o ganho de velocidade não era infinito, mesmo quando a porção serial fosse quase inexistente e o número de processadores em paralelo tendesse ao infinito. A fórmula apresentada por Amdahl é descrita pela equação 2.1.

$$speedup = \frac{(s+p)}{(s+p/N)} = \frac{1}{(s+p/N)} \tag{2.1}$$

em que s representa a parte serial do programa, p a parte paralela e N o número de processadores.

O valor resultante dessa formulação foi considerado um desestímulo ao paralelismo, pois o ganho com a paralelização de sistemas sempre apresentava um valor muito baixo e inexpressivo.

Em um de seus trabalhos, Gustafson [Gustafson, 1988] coloca que a formulação de Amdahl só pode ser considerada válida se o tamanho do problema não variar com o número de processadores (o que não acontece em problemas reais); e por este motivo classifica a definição de "speedup" apresentada por Amdahl como "speedup" de carga fixa (ou tamanho fixo).

No mesmo trabalho, Gustafson apresenta o "speedup" de tempo fixo, em que considera que o tempo de execução do sistema permanece inalterado, aumentando-se o tamanho do problema quando o tamanho (número de processadores) da máquina paralela aumenta. Com essas características, o "speedup" é calculado em relação ao tempo de execução seqëncial do novo problema, tornando a paralelização de sistemas uma técnica vantajosa. A fórmula para esse "speedup" aparece na equação 2.2, com N correspondendo ao número de processadores e s à parte serial do sistema em questão.

$$speedup\ tempo\ fixo = N + (1 - N) \cdot s \tag{2.2}$$

O "speedup" apresentado por Gustafson pode ser também chamado de

"speedup" escalonado (baseado em uma máquina escalável). Uma máquina é considerada escalável se permitir que o tamanho do problema aumente à medida em que o número de processadores que compõem a máquina paralela aumenta. Ambos os tipos de "speedup" possuem um problema em comum: a definição da referência de execução seqüencial (ambas as definições levam em consideração o tempo de execução do sistema paralelo em relação ao tempo de execução seqüencial).

Um outro problema presente no processo de seleção de métricas para avaliação de desempenho é o referencial a ser tomado para a definição das medidas. Em relação ao "speedup", por exemplo, alguns autores defendem que deve ser calculado em relação ao tempo de execução do algoritmo serial mais rápido e outros que o fator deve ser calculado tomando-se em conta valores que tornam o fator mais significativo.

Em busca de soluções para esse problema, vários autores tratam o assunto [Sahni, 1996][Hwang, 1993]. Sahni, um dos defensores do cálculo do "speedup" tendo como referência apenas o tempo de processamento, apresenta várias formulações a partir dessa característica; algumas das formulações descritas são apresentadas na tabela a seguir, onde *PP* significa programa paralelo e *MPS* melhor programa serial.

Tipo de "speedup"	Fórmula
Relativo	$\frac{T(PP\ em\ 1\ processador)}{T(PP\ em\ n\ processadores)}$ (2.3)
Real	$\frac{T(MPS\ em\ 1\ processador)}{T(PP\ em\ n\ processadores)}$ (2.4)
Absoluto	$\frac{T(MPS\ no\ processador\ mais\ rápido)}{T(PP\ em\ n\ processadores)}$ (2.5)

Tabela 2.1: Diferentes definições de "speedup".

A argumentação apresentada por Sahni (tomar em consideração apenas o tempo de processamento) é válida do ponto de vista do usuário do programa paralelo, o qual prefere resultados mais rápidos. Do ponto de vista do administrador do sistema nem sempre tempo de execução menor significa um melhor desempenho do sistema.

A partir das diversas formas de "speedup", outras medidas são formuladas, com o intuito de melhor traduzir o desempenho de sistemas paralelos. São elas

eficiência, redundância, utilização e qualidade de paralelismo. Deve-se apontar aqui que cada tipo de "speedup" gera um conjunto de medidas. A seguir são apresentadas medidas oriundas da definição de "speedup" relativo (equação 2.3) [Hwang, 1993], onde *T(1) = T(PP em 1 processador)*, *O(n)* o número de operações realizadas em n processadores, *T(n)* o tempo em pulsos de relógio e, por convenção, *T(1) = O(1)*.

Eficiência

A eficiência é uma medida que busca apresentar o quanto o ganho de velocidade (ou aceleração) no processamento está próximo ou não do ótimo. É definida pela fórmula 2.6:

$$E(n) = \frac{Speedup(n)}{n} = \frac{T(1)}{n \cdot T(n)} \tag{2.6}$$

A eficiência varia entre 0 (0%) e 1 (100%). Dificilmente o valor máximo da eficiência é obtido, pois sempre ocorrem perdas que impedem que tal valor seja obtido (sobrecarga na comunicação e de sincronismo, p.e.).

Redundância e utilização

A redundância busca averiguar se o programa em questão foi paralelizado de forma adequada em relação ao hardware disponível. É definida da seguinte forma:

$$R(n) = \frac{O(n)}{O(1)} \tag{2.7}$$

A utilização de um sistema indica a porcentagem de recursos (processadores, memória, etc.) que permanecem ocupados durante a execução de um determinado programa paralelo. pode ser escrita como apresentado na equação 2.8:

$$U(n) = R(n)E(n) = \frac{O(n)}{nT(n)} \tag{2.8}$$

Qualidade de paralelismo

A qualidade de paralelismo apresenta ao analista de desempenho uma medida mais completa sobre as interações do conjunto programa/máquina, ao estabelecer relações entre o quanto se podia aumentar a velocidade de processamento e o quanto o programa está bem (ou mal) estruturado. É definida pela fórmula a seguir:

$$Q(n) = \frac{S(n) \cdot E(n)}{R(n)} = \frac{T^3(1)}{n \cdot T^2(n) \cdot O(n)} \tag{2.9}$$

Sendo uma relação entre "speedup", eficiência e redundância, a qualidade procura mostrar ao analista se o conjunto está de acordo com o que se deveria esperar do ponto de vista de aumento de velocidade.

2.2.2 Medidas independentes de velocidade

Um outro conjunto de medidas, diferente do apresentado anteriormente, é também utilizado na avaliação de desempenho de sistemas: o conjunto das medidas independentes de velocidade. As medidas que o compõem, normalmente, são utilizadas para quantificar o desempenho de subsistemas específicos ou de sistemas de software.

Quando o alvo da avaliação são sistemas de hardware, são utilizadas métricas que mostram a ocupação de registradores, falta de dados em cache, movimentação na memória (medidas que demonstram a quantidade de memória livre, p.e.), além de algumas específicas, como TPS (transações por segundo) e PPS (pacotes por segundo), medidas utilizadas na avaliação de comunicação entre computadores; entre outras.

Na avaliação de sistemas de software, são utilizadas métricas que fornecem informações sobre a execução dos programas (quanto determinado trecho do programa ocupa do tempo total de execução, número de ocorrências de determinadas funções, tempo gasto em comunicação, etc.).

Esse conjunto se torna vasto a medida que o número de aplicações aumenta. Por exemplo, quando o sistema analisado é um servidor de disco, o número de operações simultâneas suportadas pelo servidor é importante, bem como tráfego de dados gerado pelo sistema, ou a carga empregada ao subsistema de entrada e saída, entre outras medidas.

Neste conjunto algumas medidas podem assumir diferentes definições, por serem utilizadas na avaliação do desempenho de diferentes sistemas. Um exemplo de medida que pode assumir diferentes definições é a *eficiência*.

A eficiência pode ser utilizada na avaliação de diferentes sistemas, como em [Hsu, 1998], em que é utilizada para mostrar a relação entre acertos e erros da técnica de "prefetching", ou quando é utilizada para demonstrar a porcentagem de acerto de um sistema de banco de dados ao tentar buscar dados em determinado arquivo, etc.

Existem também medidas criadas por pesquisadores com a finalidade de melhor demonstrar o desempenho de sistemas particulares, como acontece em [Miller, 1992], em que uma nova medida é criada e comparada com as fornecidas por outras técnicas já existentes.

2.3 Estratégias de classificação

Dentre as diversas formas de classificação de ferramentas de avaliação de desempenho, algumas podem ser consideradas como mais abrangentes e outras como de grande uso. As formas de classificação escolhidas para este trabalho não constituem um padrão na comunidade acadêmica e empresarial, mas conseguem abranger se não todas, grande parte das ferramentas, métodos e técnicas existentes atualmente.

As estratégias, segundo critérios de abrangência e uso são: tipo de sistema (se hardware ou software) em que a ferramenta ou técnica é empregada; a arquitetura a que é empregada a ferramenta ou técnica; abordagem usada na obtenção de resultados; forma de instrumentação e medição; modelos dos dados gerados para avaliação. Essas duas últimas, por caracterizarem mais a forma de instrumentação, são tratadas

em uma seção separada.

Para melhor compreensão do texto, o termo técnicas será empregado para designar técnicas, métodos e ferramentas, pois todas as estratégias apresentadas nessa seção são aplicadas a essas entidades.

2.3.1 Quanto ao sistema aplicado

Pode-se classificar as técnicas de avaliação de desempenho em dois grandes grupos: técnicas que estudam sistemas de hardware e técnicas que estudam sistemas de software. Os próprios nomes já resumem a classificação que, apesar de simples, é importante na diferenciação entre várias técnicas.

Técnicas de avaliação 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. Podem ser divididas em técnicas de monitoração ou de modificação de código, segundo a estratégia usada para medição.

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

2.3.2 Quanto a arquitetura da aplicação

Uma outra estratégia de classificação 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[3], usadas como referencial para esta classificação.

[3]SISD – Single Instruction Stream Single Data Stream, SIMD – Single Instruction Stream Multiple Data Streams, MISD – Multiple Instruction Streams Single Data Stream, MIMD - Multiple Instruction Streams Multiple Data Streams.

Embora pareça óbvio, essa classificação é necessária pois ferramentas projetadas para uma arquitetura SIMD 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. Uma classificação de arquiteturas mais recente é apresentada em [Duncan, 1990] mas, como é baseada na apresentada por Flynn, preferiu-se tomar a primeira como padrão.

2.3.3 Segundo a abordagem utilizada para obtenção dos resultados

Tanto para a avaliação de desempenho de hardware quanto para avaliação de desempenho de software, em relação à abordagem utilizada na obtenção dos resultados, pode-se dividir as técnicas em três grupos: técnicas baseadas em modelos analíticos, técnicas baseadas em simulação e técnicas baseadas em medição. Um fato relevante para escolha dessa estratégia é que ela é considerada por grande parte da comunidade científica e comercial como um padrão de classificação de técnicas que avaliam sistemas, tanto seqüênciais quanto paralelos.

Técnicas baseadas em modelagem analítica

Técnicas baseadas em modelagem analítica são aquelas, como o próprio nome define, baseadas em modelos analíticos. Isso tanto para modelos de redes de Petri e cadeias de Markov [Granda, 1992], em que o sistema é definido como uma seqüência de estados com transições atreladas a equações matemáticas, como também as baseadas em modelos de filas [Chang, 2000].

Esse tipo de técnica é amplamente utilizada nos períodos de elaboração e desenvolvimento de sistemas (juntamente com técnicas baseadas em simulação) e quando o alvo da avaliação não se encontra disponível. Possui baixo custo e velocidade de resposta variável, pois a velocidade está intimamente ligada à complexidade do modelo (quanto maior a complexidade, maior o tempo necessário para otimização e resolução das equações): pode ser rápida quando o modelo é simplificado e reduzido

e lenta quando o modelo é complexo (por exemplo, um modelo preciso para um subsistema específico de um sistema paralelo).

A precisão dos resultados fornecidos por esse tipo de técnica depende do processo matemático envolvido na criação do modelo. Caso o equacionamento matemático e a otimização estejam corretos, os resultados fornecidos serão precisos. Deve-se salientar aqui que quanto maior a complexidade do sistema maior a dificuldade da resolução analítica e maior a possibilidade da ocorrerência de erros. Um outro fator que interfere na precisão dos resultados é a quantidade de simplificações e suposições; quanto maior o número de simplificações e suposições, menor é a precisão dos resultados. Fazem parte desse conjunto a GSPN (rede de Petri estocástica generalizada, métodos baseados em cadeias de Markov, teoria de filas e os modelos determinísticos).

Técnicas baseadas em simulação

As técnicas baseadas em simulação consistem no processo de modelagem matemática ou lógica de sistemas na tentativa de representar fielmente a realidade. Essas técnicas assemelham-se muito com técnicas analíticas, em que a diferença básica entre essas duas técnicas reside na maneira como os resultados são obtidos: nas técnicas baseadas em simulação, substituem-se os sistemas de equações que definem o andamento do sistema por regras que definem seu comportamento.

Um grande problema no uso de simulação é a dificuldade na obtenção de um modelo que seja fiel ao sistema real, e mesmo que o modelo criado esteja correto, existe ainda a possibilidade da imprecisão dos resultados (a precisão dos resultados está ligada às condições em que a simulação é realizada). Entretanto, essa imprecisão pode ser corrigida através do posterior refinamento do modelo a ser simulado (através de dados de "benchmarks", por exemplo).

Em contrapartida, a simplicidade na elaboração do modelo de simulação, a flexibilidade (possibilidade de adequação a outros sistemas) e a possibilidade de tratamento de parâmetros difíceis de dimensionar analiticamente são fatores a favor da utilização desse tipo de técnica.

Fazem parte desse conjunto ferramentas como PAWS [Pease, 1991], PDL [Vemuri, 1998], Simics [Magnusson, 2002], a técnica apresentada por Manacero Jr. em [Manacero Jr., 1997], entre outras.

Técnicas baseadas em "benchmarking"

As técnicas baseadas em "benchmarking" são técnicas baseadas em medições reais dos sistemas em avaliação. Os dados são obtidos através da execução dos programas (tanto na avaliação de sistemas de hardware quanto na avaliação de sistemas de software).

Essas técnicas apresentam, dentre todas, a menor complexidade na implementação e grande flexibilidade (desde que se tenha sistemas e programas disponíveis para teste). Por outro lado, dependem da disponibilidade da máquina onde serão realizadas as medições, da sua velocidade e do número de repetições necessárias para obtenção de dados confiáveis, o que pode elevar seu custo operacional. Em determinadas situações o uso de "benchmarking" torna-se proibitivo, pois a realização dos testes implica multiplicação do hardware de teste (máquinas reais).

A criação de um programa ("benchmark") esbarra em problemas que vão desde a quantificação da carga aplicada ao sistema até a escolha dos procedimentos de medição. Fazem parte desse conjunto os chamados "microbenchmarks" e os "benchmarks" proprietários (desenvolvidos por analistas para sistemas pré-determinados).

As técnicas baseadas em "benchmarking" são amplamente utilizadas devido à precisão dos programas e ao fato que muitas das ocasiões em que analistas desejam obter o desempenho de determinadas máquinas, eles já as possuem para realização das medições. Existem também ferramentas que realizam medições para análise de software. Tais ferramentas extraem, durante a execução dos programas, os dados necessários para a avaliação do desempenho.

Critérios de escolha

A partir das características dessas técnicas, pode-se listar um conjunto de parâmetros que ajudam o usuário na decisão de qual técnica será aplicada para a avaliação de desempenho. Tais parâmetros são listados na tabela 2.2, em que a maioria dos parâmetros são apresentados por Jain em [Jain, 1991].

O parâmetro chave na escolha da técnica para avaliação de desempenho é o estágio de desenvolvimento em que o sistema se encontra, ou seja, se o sistema já se econtra disponível ou não. As técnicas baseadas em "benchmarking" são utilizadas quando o sistema já está pronto, ou quando uma versão similar (próxima da desejada) se encontra disponível. Se o sistema em questão ainda não existir a modelagem analítica e a simulação são as únicas técnicas que poderão ser utilizadas. Essas duas últimas podem ser utilizadas em qualquer situação.

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	programa para instrumentação
4. Exatidão/precisão	baixa	moderada	varia
5. Avaliação de alternativas	fácil	moderada	difícil
6. Custo	baixo	médio	alto
7. Validação/receptividade	baixo	baixo/médio	alto
8. Flexibilidade	baixa	alta	alta/varia

Tabela 2.2: Critérios para escolha de técnicas de avaliação de desempenho.

O segundo parâmetro é o tempo disponível para análise (o quão urgente ou não são as informações a serem geradas pela avaliação). Na maioria das situações a necessidade dos resultados é urgente, situação em que a modelagem analítica pode ser a melhor escolha, por ser a técnica que teoricamente "gasta" menos tempo para avaliação. Simulações tomam muito tempo, enquanto que técnicas que utilizam medições ("benchmarking") tomam mais tempo que a modelagem analítica, e podem ou não ser mais rápidas que simulações, devido à alta suscetibilidade a problemas. Na tabela, o termo *médio* foi atribuído às técnicas baseadas em simulação por esse conjunto geralmente gastar menos tempo que medições e mais tempo que técnicas

analíticas.

Outro parâmetro consiste no conjunto de requisitos necessários (ferramentas) para avaliação. Na modelagem analítica é necessário apenas o analista com conhecimento matemático a ser aplicado na criação do modelo. Já na simulação, além do conhecimento do analista, é necessário alguma linguagem de computador ou ambiente de simulação. Nas técnicas baseadas em "benchmarking" é necessário o programa que realizará a instrumentação do sistema a ser avaliado, no caso de sistemas de software, e do programa que extrairá as medidas do sistema de hardware.

A exatidão ou precisão da técnica é outro importante parâmetro. Em geral, a precisão de técnicas baseadas em modelos analíticos é baixa devido ao grande número de simplificações, que degradam a exatidão. As simulações possuem resultados mais precisos, por poderem incorporar mais detalhes. As técnicas baseadas em medição podem não fornecer resultados precisos devido à má escolha dos parâmetros de ambiente como, por exemplo, configuração do sistema, tipo de carga de trabalho e número de execuções para cálculo das métricas. Além disso, as variáveis de ambiente podem não representar as encontradas no mundo real, o que acaba por tornar exatidão dos resultados variável.

Um quinto critério é a facilidade com que se pode avaliar alternativas ao sistema testado, procurando identificar possíveis melhorias nele (através da análise da interação entre os parâmetros do sistema). Modelagens analíticas geralmente provêm melhor visão dos efeitos de vários parâmetros e suas interações (através de análise detalhada das equações). Nas simulações, pode ser possível encontrar o espaço de valores dos parâmetros para combinações ótimas, mas, muitas vezes não é claro quais são as alternativas possíveis entre diferentes parâmetros (devido, às vezes, ao grande número de estados do sistema). Já nas medições, a análise de alternativas não é precisa, pois não se pode afirmar que a melhora ou não no desempenho do sistema é resultado de alguma mudança aleatória no ambiente computacional ou devido a alguma mudança de um parâmetro particular; um meio de certificar a melhoria ou degradação no desempenho devido a mudança nos parâmetros é através do aumento na quantidade de execuções para obtenção dos dados.

Outro parâmetro na escolha, bastante importante por sinal, é o custo do

processo de avaliação de desempenho a ser utilizado, procurando determinar o melhor para o orçamento disponível para a tarefa. Medições requerem equipamento real, instrumentos e tempo, sendo a mais custosa das três técnicas. O custo, juntamente com a necessidade de mudança de configuração dos sistemas, são os parâmetros que tornam necessárias simulações de sistemas de custo elevado. A modelagem analítica requer o tempo do analista e de conhecimento para criação do modelo, sendo a alternativa mais barata.

A validação dos resultados obtidos e a receptividade no meio acadêmico e empresarial são áreas em que as técnicas baseadas em modelagem analítica e as baseadas em simulação encontram dificuldades. A validação e a receptividade das técnicas baseadas em medições são facilmente aceitas nos dois meios, enquanto que os analistas que criam modelos analíticos e de simulação necessitam validar os resultados através de comparações com modelos já existentes.

Por fim, um critério não citado por Jain, mas também importante, é a *flexibilidade* da técnica ou modelo criado. Técnicas baseadas em modelos analíticos são consideradas de baixa flexibilidade pois, para a alteração do modelo é necessário modificar todo o conjunto de equações. Já as baseadas em simulação possuem flexibilidade alta, devido a possibilidade de alteração do modelo sem grandes modificações. As técnicas baseadas em "benchmarking" possuem flexibilidade variável, pois esta é ligada à disponibilidade do código do programa de medição. Se acaso o programa estiver disponível, ele poderá ser alterado para adequação (flexível, neste caso) e, caso não esteja, não existe flexibilidade nenhuma.

2.4 Estratégias de instrumentação

Além das classificações já apresentadas, ainda é possível dividir as técnicas em grupos mais específicos, de acordo com a forma de instrumentação e medição e com os modelos dos dados gerados para análise, podendo assim diminuir o espaço de busca do usuário por ferramentas de determinada característica. Duas estratégias de classificação fazem parte desse conjunto, e são apresentadas a seguir.

2.4.1 Classificação segundo a forma de medição e instrumentação

Pierece e Mudge apresentam uma classificação partindo do princípio que é possível obter dados sobre o desempenho de sistemas de duas formas: através de monitoração (por hardware ou software) ou através de instrumentação do código (fonte, objeto ou executável) do programa, como descrito a seguir.

Monitoração por hardware

Existem diversas maneiras que podem ser usadas na obtenção do comportamento dos sistemas de software, mas o mais direto deles consiste no uso de analisadores lógicos acoplados diretamente ao processador ou ao barramento do sistema para obtenção dos dados desejados.

Embora possa obter medidas precisas, a monitoração não é adequada para a análise de desempenho devido ao alto custo envolvido em sua implementação, à necessidade de um dispositivo especial para armazenamento dos dados e pela exigência de profissional especializado no hardware de monitoramento.

Monitoração por software

A forma mais simples de realizar monitoração por software consiste no uso das interrupções para gravação das informações sobre a execução dos programas. Este tipo de monitoração pode ser desaconselhável por ser extremamente lento quando o número de interrupções e de referências à memória durante a execução é elevado.

As técnicas pertencentes às classes apresentadas a seguir modificam, em alguma instância, o código do programa analisado. Tal fato as torna técnicas invasivas.

Modificação do código fonte

Consiste na forma mais fácil de instrumentação. Comandos especiais são inseridos no código fonte do programa, os quais farão as medições desejadas. Este tipo de modificação não é aconselhado quando não há disponibilidade do código fonte ou este não pode ser obtido, quando o tamanho do código a ser modificado é expressivo e, em alguns casos, quando existe um grande número de referências à bibliotecas e rotinas de sistema.

Modificação do código objeto

A modificação é realizada inserindo os comandos para as medições no código objeto. Pode ser realizada por um editor específico que reescreve o código na liguagem original (ou alguma linguagem legível) e depois refaz a alocação do código e dos dados para torná-lo objeto.

Modificação do código executável

É realizada através da inserção de comandos já no código executável. Consiste na forma mais complicada de inserção de código, pois exige que o usuário possua um descompilador ou um programa que realize a adição do código no programa.

2.4.2 Classificação segundo os modelos dos dados gerados

A classificação proposta por Daniel Reed 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. O autor descreve quatro categorias de técnicas de medição por modificação do código: "profiling", contagem de eventos, medição de intervalos de tempo e extração de traços de eventos.

"Profiling"

Esse tipo de técnica busca obter informações sobre o tempo gasto por cada trecho do programa em relação ao tempo total de execução. Essas informações são obtidas através da amostragem do contador de programas em intervalos fixos (o número total de amostragens corresponde aproximadamente ao tempo gasto pelos trechos), fato que gera problemas quanto à precisão das medidas, pois o intervalo de amostragem é muito grande. Os dados sobre o uso do processador e sobre os trechos do programa são normalmente apresentados na forma de tabelas de funções, com estatísticas como tempo de execução de cada trecho.

Técnicas baseadas nesse tipo de instrumentação apresentam problemas relativos a precisão, pois são dependentes de mecanismos externos de coleta de dados (normalmente, sistema operacional) e também costumam apresentar problemas no tratamento de compartilhamento de CPU.

Contagem de eventos

Este tipo de técnica consiste em modificar o código para que sejam contadas ocorrências de determinados eventos (como o número de vezes que um laço do programa é executado, por exemplo). Em sistemas paralelos, além de aumentar o tempo de execução do programa instrumentado, pode alterar a ordem natural dos eventos, mudando o número de instruções contabilizadas a cada processador.

O excessivo volume de dados e a carga computacional adicionada fazem com que a técnica não seja muito usada.

Medição de intervalo de tempos

Aqui se faz inserção de chamadas para medir o relógio do sistema e posterior soma dos tempos medidos para obtenção dos resultados estatísticos. Esses métodos são ineficazes devido à baixa freqüência do relógio do sistema. Outro problema muito comum nesse tipo de técnica é obtenção de tempos irreais devido ao não tratamento de compartilhamento de CPU.

Extração de traços de eventos

"Event tracing", como é chamada, é a técnica mais invasiva e a que fornece resultados mais detalhados sobre eventos do sistema dentre as quatro apresentadas nessa seção. É baseada em registros datados da ocorrência de cada evento do sistema. Tal precisão gera um volume excessivo de dados pela alta freqüência em que os eventos ocorrem.

Das técnicas de instrumentação de código indicadas, tem-se que "profilers" e extração de traços de eventos são as mais usadas nas ferramentas de avaliação de desempenho existentes. A adição nos tempos de execução devido à instrumentação realizada pelas técnicas citadas acima pode ser corrigida através da adequação do tempo final de execução dos programas. Sendo assim, tal fato não torna proibitivo o uso das mesmas.

2.5 Trabalhos correlatos

Existem diversas estratégias de classificação de técnicas para avaliação de sistemas paralelos. Grande parte delas assemelha-se às apresentadas neste capítulo, e as que não se assemelham buscam dividir as técnicas utilizando os mesmos critérios de classificação.

Dentre todas as classificações propostas, uma merece destaque por ser uma iniciativa de pesquisadores nacionais, e estar sendo utilizada como padrão em algumas universidades. É a classificação apresentada em [Santana, 1994], em que os autores dividem as técnicas em dois grandes conjuntos: técnicas de aferição e técnicas de modelagem.

Como apresentado em [Santana, 1994] e em [Tatsumi, 2002], a coleta de dados envolve uma monitoração do sistema enquanto ele está sendo submetido a uma carga em particular, através de software ou por meio de hardware específico. A técnica denominada "benchmark" utiliza carga sintética tendo como principal objetivo comparar o comportamento de vários sistemas sob a mesma carga enquanto que a construção de protótipos baseia-se na construção de uma aproximação simplificada

do sistema real.

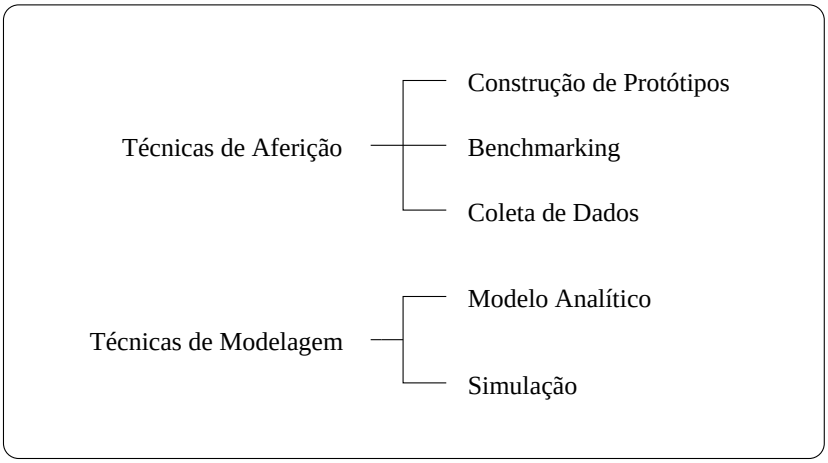

Figura 2.1: Classificação de técnicas para análise de desempenho proposta em [Santana, 1994].

As técnicas de modelagem compreendem a modelagem analítica e a simulação. A modelagem analítica utiliza métodos matemáticos para obter a informação sobre os tópicos de interesse enquanto que a simulação consiste em um paradigma computacional usado para permitir que um programa de computador emule o comportamento de certos tipos de modelos.

Como apresentado anteriormente, a classificação aqui proposta busca englobar, se não todas, grande parte das técnicas para análise de desempenho de sistemas paralelos, dando destaque a técnicas de análise de software através do refinamento da classificação.

Capítulo 3

Métodos, técnicas e ferramentas para análise de desempenho

Como visto anteriormente, a grande diversidade de métodos, técnicas e ferramentas para análise de desempenho torna a escolha de uma delas uma tarefa difícil, até para os analistas experientes. Por conseguinte a comparação também torna-se complexa.

Neste capítulo apresenta-se diversos métodos, técnicas e ferramentas para análise de desempenho de sistemas paralelos e distribuídos. Aqui se faz apenas a indicação das características e funcionalidades de cada item, deixando para o próximo capítulo a classificação dos mesmos segundo as estratégias apresentadas.

Para melhor apresentação do texto dois termos associativos são utilizados: *ferramenta* para designar ferramentas e implementações de modelos disponíveis e *técnica* designando técnicas, modelos e métodos.

As técnicas e ferramentas aqui apresentadas foram retiradas de um amplo conjunto, sendo escolhidas ou por se destacarem dentro de suas classes ou por apresentarem características comuns a outras técnicas e ferramentas que não são apresentadas no texto. Mesmo sendo o capítulo seguinte dedicado à classificação das técnicas e ferramentas, para uma maior continuidade do texto este capítulo as divide de acordo com o sistema em que a técnica ou ferramenta é aplicável e em relação

a abordagem utilizada na obtenção dos resultados, começando com técnicas para hardware e software, concluindo com ferramentas para hardware depois software.

3.1 Técnicas para hardware

Um grupo de técnicas têm por objetivo analisar ou predizer o desempenho de sistemas de hardware, quer como sistemas completos, quer como subsistemas. Embora algumas das técnicas apresentem indicações sobre a paralelização de programas, tem como alvo os sistemas de hardware, e não os programas paralelizados, pertencendo então a essa categoria. Nesta seção são apresentadas algumas das técnicas aplicáveis a sistemas de hardware.

3.1.1 Técnicas baseadas em simulação

Análise de subsistemas de Entrada/Saída [Ganger, 1998]

Com esta técnica Ganger busca resolver os problemas da análise de desempenho do subsistema de entrada e saída que, de acordo com seu estudo, residem no tempo excessivo para simulação, na necessidade de grandes dispositivos de armazenamento para os dados oriundos da simulação e no não tratamento de tempos de espera falsos[1]. O autor apresenta uma técnica baseada na modelagem e simulação do subsistema de entrada e saída isoladamente e, depois disso, em contato com todo o sistema computacional. O autor modela e simula algumas unidades reais de entrada e saída, comparando os dados obtidos na simulação com dados reais fornecidos pelos fabricantes.

Por julgar que as atuais metodologias no estudo do desempenho de subsistemas de entrada e saída não relatam o que realmente acontece, o autor apresenta uma nova metodologia, incluindo o conceito de "*request criticality*" (urgência de requisição), que consiste na divisão das requisições de entrada e saída em três níveis

[1] Tempo de espera falso é o tempo em que o processador fica executando ciclos vazios porque todos os processos em atividade estão bloqueados aguardando que os pedidos de E/S sejam atendidos e concluídos.

(requisições com tempo crítico, requisições com tempo limitado e requisições sem tempo crítico).

Estudo de casos de "prefetching" de instruções em "cache" [Hsu, 1998]

Com esta técnica avalia-se o desempenho de alguns métodos de "prefetching", técnica que consiste na utilização de previsão para alocação de dados na memória antes de serem requisitados pelo sistema operacional (neste caso a memória em questão é a memória "cache" e os dados são as instruções). Um conjunto de dez "benchmarks" seriais é utilizado para extração dos dados (arquivos de traços) a serem usados na simulação, para a obtenção das medidas de desempenho. Por esse motivo, essa técnica, apesar de estar sendo classificada como uma técnica baseada em simulação, pode ser considerada uma técnica mista[2].

A técnica é originalmente aplicável a supercomputadores que possuam "pipelines" e a programas executados nesses sistemas, mas podem ser utilizadas na análise de desempenho de computadores com arquitetura RISC que possuam "pipelines". Os parâmetros que variam durante a aplicação dos testes são o tamanho da memória "cache" disponível, o tamanho das linhas de instrução a serem transferidas para memória, os algoritmos de substituição de dados e as estratégias de predição (previsão dos conjuntos de dados a serem levados à memória).

3.1.2 Técnicas baseadas em modelos analíticos

Teoria de filas aplicada no cálculo de limitantes [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, calculando o desempenho a partir dos limitantes de pior e melhor caso. Os limitantes, como também os índices de desempenho, são calculados através de equacionamento matemático e de diagramas de transição de estados.

[2] É apresentada como uma técnica baseada em simulação pois a maior parte dos resultados apresentados é oriunda das simulações.

O modelo criado utiliza um sistema de filas "fork-join" com $N \geq 2$, como é apresentado na figura 3.1.

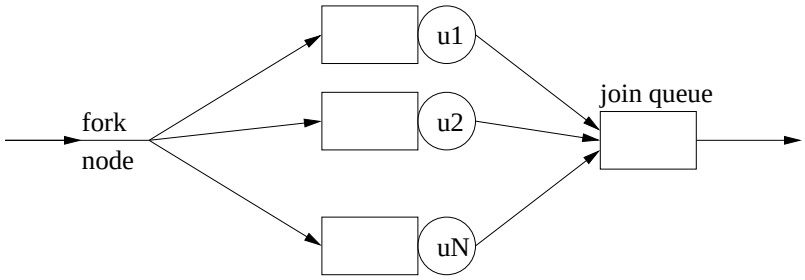

Figura 3.1: Sistema de filas fork-join com N elementos.

Algumas das medidas de desempenho fornecidas são tempo de resposta do processo, tempos das tarefas, distribuição do tamanho da fila de junção ("join queue"), atraso gerado por sincronismo e o "speedup".

Caracterização do desempenho de um computador paralelo vetorial com memória compartilhada [Dekker, 1998]

Neste trabalho, partindo do conceito de supercomputador como um computador vetorial paralelo com memória compartilhada, Dekker apresenta uma formulação que avalia a escalabilidade e o desempenho de supercomputadores com arquitetura ideal (sem considerar fatores que degradam seu desempenho). A técnica exibe o comportamento teórico das arquiteturas quando submetidas ao procedimento para resolução de equações lineares chamado Saxpy[3].

Os dados para análise são oriundos das equações que modelam as arquiteturas e o programa sintético. Como forma de validação do modelo proposto, é apresentada uma comparação com os dados de computadores vetoriais reais fornecidos pelo "benchmark" Linpack, adequados às características estudadas.

[3]Saxpy é uma forma de resolução de equações lineares de formato $z = \alpha x + y$

3.1.3 Técnicas baseadas em "*benchmarking*"

Paralelização do benchmark STAP [Hwang, 1999]

O "benchmark" STAP foi originalmente desenvolvido no MIT, sendo implementado em código C seqüencial, para processamento de sinais de radar em estações de trabalho. Kai Hwang [Hwang, 1999], em conjunto com outros pesquisadores, tornou o conjunto de ferramentas aplicável à computação paralela. O STAP (tanto em sua versão serial quanto paralela) consiste de cinco programas de processamento de sinais provenientes de radares. O paralelismo é estudado pela exploração do paralelismo massivo de dados.

Devido à rápida incidência de sinais de entrada de dados, o "benchmark" necessita realizar entre 10^{10} e 10^{14} operações de ponto flutuante por segundo, entre a decodificação de sinais, processamento e cálculo do alvo. Em seu funcionamento, após particionamento dos dados da entrada pelo nó mestre, os dados são divididos igualmente para todos os nós, que executam o mesmo programa, e após processamento retornam os dados tratados ao nó mestre, onde são combinados e o resultado é emitido.

Na técnica apresentada pelos autores, foram consideradas duas configurações de radar: uma simples (gerando baixa carga de trabalho) e outra complexa (gerando alta carga de trabalho). A escalabilidade dos sistemas foi analisada em várias dimensões: tamanho da máquina, tamanho do problema, velocidade do processador, atraso de comunicação, e tamanho agregado da banda de passagem de mensagens. Os resultados obtidos foram comprovados através de resultados oriundos da resolução das equações do modelo analítico, previamente desenvolvido pelos autores.

Análise do tempo de comunicação e determinação do caminho baseada em desempenho [Kim, 1999]

Neste trabalho os autores apresentam um conjunto de técnicas denominado PBPD ("*performance-based path determination*" - determinação do caminho baseado no desempenho). Tal conjunto tem por objetivo avaliar o desempenho dos diferentes tipos de sistemas de comunicação (Ethernet, ATM, FDDI, HiPPI e fibra

óptica) disponíveis para uso e organizá-los de forma a obter o maior ganho na velocidade de comunicação entre processadores[4].

Os dados sobre o desempenho dos sistemas são fornecidos por "benchmarks" (CG, FT, GAUSS, MG, IS, TRFD), que diferem no tipo de comunicação (ponto-a-ponto, broadcast, etc.) e na carga de comunicação necessária. Após a coleta e análise dos dados gerados na etapa de avaliação de desempenho e adequação das bibliotecas de comunicação (PVM e MPI), o programa aplicação executa dinamicamente a escolha da melhor opção (um único sistema ou sistemas agregados) a ser utilizada na comunicação.

Como o principal fator de comparação na técnica apresentada é o tamanho das mensagens, as medidas de desempenho tomadas são o "speedup" e o atraso na comunicação em diferentes conjuntos, onde o número de eventos (mensagens enviadas) na comunicação e o tamanho das mensagens variam. Para validação dos resultados são realizadas comparações com alguns modelos analíticos de alguns casos de teste pesquisados.

Análise do desempenho da memória em sistemas DSM - de memória distribuída e compartilhada: estudo de um caso [Abandah, 1998]

Abandah e Davidson apresentam uma técnica que utiliza "microbenchmarks" para avaliação de desempenho de sistemas DSM. Apesar de analisar todo o sistema computacional, seu foco é o subsistema de memória. A técnica é aplicada em um computador Convex SPP1000.

Os "microbenchmarks" são aplicados para caracterizar o desempenho do sistema e dos diversos níveis de memória (memórias local e compartilhada, memórias "cache" local e "cache" interconectado[5]). Outros aspectos, como sobrecarga no escalonamento dos procesos e o tempo gasto no sincronismo das tarefas também são analisados.

Os programas são aplicados em situações normais de execução e em situa-

[4]O termo processadores é utilizado devido à possibilidade da técnica ser aplicada a sistemas multiprocessadores e multicomputadores.

[5]Memória cache do "hypernode", compartilhada entre os processadores.

ções consideradas críticas pelos autores, como quando dois ou mais processadores competem pela utilização da memória compartilhada e pela rede de conexão ("crossbars" e anéis de conexão). As configurações do ambiente alvo são alteradas para melhor compreensão de seu comportamento; alguns aspectos alterados são a distância entre os nós de processamento e a forma de execução do programa baseado no algoritmo produtor-consumidor.

3.2 Técnicas para software

De acordo com a classificação previamente apresentada, nesta seção são apresentadas técnicas que têm por objetivo a análise de sistemas de software.

3.2.1 Técnicas baseadas em simulação

Modelagem de limitantes de desempenho [Lui, 1998]

Assim como Balsamo [Balsamo, 1998] fez para sistemas de hardware, Lui desenvolveu um trabalho que utiliza sistemas de filas e "fork-join", para modelar um programa paralelo.

Em seu trabalho, Lui modela um programa paralelo escrito com paradigma "fork-join" como um sistema de filas, avaliando seu desempenho. A análise é realizada através da alteração da ordem de chegada de tarefas e a distribuição do serviço entre os diversos processadores no decorrer do tempo de execução das tarefas. Baseado no sistema de filas, o autor constrói um modelo Markoviano e apresenta também algoritmos para obtenção de limitantes de desempenho superior e inferior em um tempo esperado de resposta, para determinados modelos de distribuição de probabilidades.

Conversão do código executável em grafo de execução [Manacero Jr., 1997]

Manacero Jr. apresenta em sua tese de doutorado uma metodologia para realizar medidas de desempenho em programas paralelos sem a necessidade da adição de código extra no programa analisado.

Tal metodologia consiste em reescrever o código executável para um grafo de execução, que é um grafo dirigido que armazena informações sobre o tempo de processamento do programa. Nesse grafo os vértices agrupam blocos seqüenciais, enquanto os arcos entre qualquer par de vértices indicam a relação de precedência entre eles. A informação contida no grafo é usada por um simulador que o percorre para fazer as medições necessárias para a predição ou análise do desempenho.

São exibidas medidas como velocidade de processamento global, velocidade de processamento de cada cliente, "speedup" e tempo de comunicação em diferentes configurações (tamanho do grau de paralelismo).

3.2.2 Técnicas baseadas em *benchmarking*

A avaliação do desempenho de programas através de "benchmarking" é, na realidade, a técnica mais usada para se medir desempenho de programas. Na realidade sempre que se executa um programa e se mede seu tempo de execução ocorre uma medida de desempenho. Se essas medições passam a ser instrumentadas pelo código, obtendo medidas de tempo internas ao programa, então ocorre um "benchmark" de fato.

Na literatura é possível identificar centenas de aplicações em que se fez "benchmark" de aplicações específicas. Em particular é interessante relatar os trabalhos de McCracken [Cracken, 1998] e Marcari [Marcari Jr., 1999], que avaliam o desempenho de programas aplicados à física de estrutura molecular de polímeros, identificando gargalos e blocos do programa e nas alternativas de paralelização.

3.2.3 Técnicas baseadas em modelos analíticos

GSPN - rede de Petri estocástica generalizada [Granda, 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).

Granda e outros elaboram um modelo de GSPN ilimitada para análise de desempenho, no qual definem as condições para criação dos estados e transições e o método para redução da cadeia de Markov (agregação vertical e horizontal) criada a partir da GSPN. A criação de uma cadeia de Markov a partir da GSPN é possível pois GSPN's são isomórficas às cadeias de Markov, podendo ser tratadas de modo idêntico. A metodologia aplicada na criação do modelo é apresentada na figura 3.2.

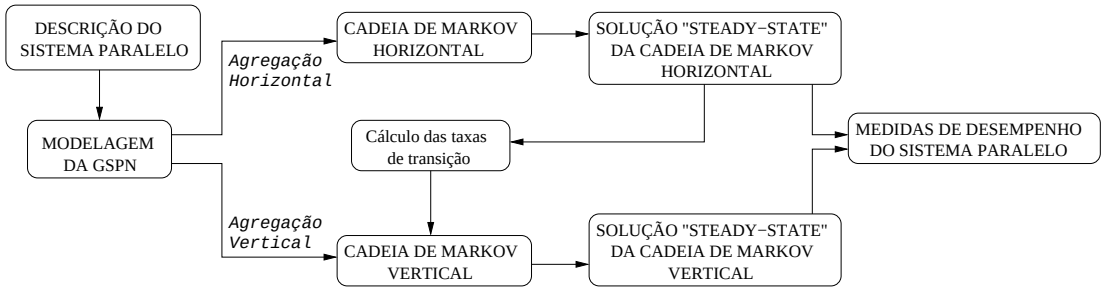

Figura 3.2: Cálculo das métricas de desempenho com GSPN ilimitadas.

São propostos três conjuntos de medidas que podem ser obtidos através deste tipo de rede de Petri: medidas que demonstram o comportamento dinâmico do sistema (tempo gasto em cada ciclo do programa, tempo gasto por cada tarefa no ciclo e tempo de espera), medidas referentes a carga do sistema (tempo gasto por cada processador na execução de tarefas, esforço computacional necessário para execução de cada tarefa) e métricas que demonstram o comportamento do sistema em relação à memória (memória requerida para execução das tarefas e para troca de mensagens).

3.3 Ferramentas para hardware

O termo ferramentas para hardware é utilizado para designar todas as ferramentas que têm por sistema alvo sistemas de hardware. Nesta seção são descritas tais ferramentas.

3.3.1 Ferramentas baseadas em simulação

PAWS [Pease, 1991]

PAWS (Parallel Assessment Window System) é um sistema com uma interface composta por janelas que permite a análise de desempenho de programas em diversas configurações de máquina, como também de vários tipos de máquinas executando aplicações comuns.

O sistema é composto por quatro ferramentas: ferramenta de caracterização da aplicação (que transcreve o código fonte da aplicação escrito em linguagem de alto nível em um grafo de dependência de dados, e permite a avaliação do nível e do grau de paralelismo, além do mapeamento para várias arquiteturas), a ferramenta de caracterização de arquitetura (empregada para análise dos sistemas de hardware), a ferramenta de análise de desempenho (colhe os dados gerados pelas ferramentas anteriores, realiza simulações e fornece ao usuário as medidas de desempenho de seu interesse) e a ferramenta de visualização gráfica (ferramenta utilizada na visualização dos dados provenientes das simulações).

A linguagem intermediária escolhida pelo PAWS é o IF1, uma linguagem gráfica que utiliza grafos de dependência de dados para exibir as arquiteturas das máquinas e grafos de execução das aplicações. Isso permite que linguagens de alto nível sejam traduzidas para IF1, permitindo sua análise com o PAWS. A abordagem utilizada no PAWS é mesma utilizada na maioria das ferramentas de análise de desempenho, que consiste em separar os modelos de arquiteturas dos modelos de programa, permitindo que aplicações sejam testadas (simuladas) em diferentes configurações de arquiteturas.

PDL [Vemuri, 1996]

PDL é uma linguagem de alto nível que possui 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. A linguagem foi originalmente desenvolvida para análise de hardware, mas pode também ser utilizada na análise do conjunto hardware/software.

Um modelo em PDL depende de módulos, portas, condutores e seus atributos, que geram uma estrutura de hardware em que as portas podem emular sinais lógicos, os módulos podem emular portas lógicas e os condutores as ligações entre portas. A especificação de um sistema é realizada através da construção de um código com sintaxe proprietária que, após ser compilado, gera um código intermediário que pode ser analisado através de simulação. Existe também a possibilidade de adição de regras associadas com determinados esquemas de composição de objetos para que sejam obtidas medidas desejadas (como por exemplo o possível custo da máquina).

RSim - The Rice Simulator for ILP Multiprocessors [Hughes, 2002]

Esta é uma ferramenta classificada como um simulador para sistemas de hardware, por ter sido inicialmente desenvolvida para esse fim (simular arquiteturas multiprocessadas com memória compartilhada), mas que pode ser utilizada tanto para análise de sistemas de hardware quanto de software.

RSim, é um simulador desenvolvido no Departamento de Engenharia Elétrica e Computação da Rice University. O simulador, que permite a avaliação de arquiteturas multiprocessadas com memória compartilhada em computadores que exploram ILP ("Instruction Level Parallelism" - paralelismo em nível de instrução), foi concebido quando não haviam simuladores que tratavam ILP e a documentação sobre projetos comerciais era escassa.

Na sua criação, grandes porções de código dos subsistemas de memória e rede foram baseados em um outro simulador criado no mesmo departamento, o RPPT[6]. Como a maioria dos equipamentos existentes no laboratório onde foi criado

[6]Rice Parallel Processing Testbed

o simulador era da fabricante Sun Microsystems, parte do conjunto de instruções do simulador foi baseada em instruções da arquitetura Sparc V9.

O simulador tem a capacidade de modelar processadores que exploram várias funcionalidades, como predição dinâmica e estática de ramificações, "prefetching", "multithreading", entre outras; sendo grande parte dos parâmetros de processador (tamanho da palavra de instrução, número de unidades funcionais, etc.) valores configuráveis pelo usuário.

O simulador apresenta várias estatísticas de execução de programas no sistema, como o número total de ciclos, instruções por ciclo, além de estatísticas sobre o processador, como a utilização das várias unidades funcionais.

Simics [Magnusson, 2002]

Simics é um plataforma de simulação que tem por objetivo modelar sistemas completos. De acordo com os autores, a simulação de sistemas completos deve suportar as fases de projeto, desenvolvimento e testes dos sistemas de hardware e de software, dentro de um ambiente de simulação que se aproxima ao máximo do ambiente real.

O simulador pode modelar desde "desktops" até sistemas multiprocessados, podendo simular até "clusters" e redes de computadores contendo os equipamentos modelados. Simulando os processadores no nível dos conjuntos de instruções, atualmente suporta modelagem para UltraSparc, Alpha, x86, x86-64 (Hammer), PowerPC, Itanium, MIPS e ARM.

Através da união desse simulador com outras ferramentas para análise de desempenho de sistemas paralelos (o conjunto de ferramentas do consórcio SPEC, por exemplo), pode-se obter várias medidas de desempenho dos sistemas modelados, como acontece em [Magnusson, 2002], em que se apresenta uma série de medidas de sistemas (seqüenciais e paralelos) simulados.

3.3.2 Ferramentas baseadas em "*benchmarking*"

FTIO benchmark [Fagerstrom, 2000]

FTIO, desenvolvido na NASA, é um "benchmark" para análise de desempenho do subsistema de entrada/saída, criado com o objetivo de suprir a deficiência do conjunto *NAS Parallel Benchmarks* neste aspecto[7], tomando como base uma série de critérios padrão, como o não favorecimento de uma determinada arquitetura em relação a outra, a escalabilidade em relação ao tamanho do problema e ao número de processadores e necessidade de especificação precisa.

Este "benchmark" utiliza um conjunto de ferramentas, composto por uma biblioteca padrão de passagem de mensagens (MPI - "Message Passing Interface") para comunicação entre os nós de processamento, uma biblioteca em C para cálculo de transformadas de Fourier discretas de ordem dois e um utilitário que obtém informação em contadores de hardware.

A execução do "benchmark" resume-se na extração dos tempos gastos na preparação da matriz que contém os dados para a resolução da transformada, na distribuição das parcelas da matriz entre os nós de processamento, na execução da rotina de resolução e no processo de envio dos dados resultantes ao nó mestre. A rotina de resolução possui uma série de eventos que fazem com que o subsistema de E/S seja utilizado ao extremo, ao ponto de ocorrer degradação no desempenho devido ao "gargalo" do subsistema.

Um aspecto interessante deste "benchmark" é que ele não utiliza o tempo de processador, mas o tempo de barreira (através da função MPI_Wtime()) para calcular o tempo gasto em E/S. Essa opção deve-se ao fato que a comunicação utilizada entre os processadores[8] e as operações de leitura e escrita no MPI serem bloqueantes.

Como resultado a ser utilizado pelo analista ou usuário, o "benchmark" apresenta uma "fração de E/S", fator que indica o quanto (em porcentagem) o processo de E/S consome do tempo total de execução.

[7] O pacote NAS já possuia um "benchmark" destinado a analisar o desempenho de E/S (*NHT-1 I/O Benchmarks*), mas de acordo com os autores, era impreciso.

[8] Opção escolhida pelos programadores do "benchmark".

Linpack [Dongarra, 1988]

Linpack é considerado a origem de todos os "benchmarks", embora Dongarra e sua equipe, quando o desenvolveram, tivessem a intenção de criar apenas uma biblioteca de álgebra linear para ser utilizada em computadores de alto desempenho.

Sendo assim, este "benchmark" consiste em um conjunto de funções que resolvem sistemas densos de equações lineares. Os programas no Linpack podem ser caracterizados por realizarem um grande número de operações de ponto flutuante, pois em sistemas de n incógnitas são necessárias $2n^3/3 + 2n^2$ operações (tanto adições quanto multiplicações). Esses programas fornecem as medidas de desempenho em Mflops (milhões de operações de ponto flutuante por segundo).

Atualmente o Linpack é utilizado amplamente, sozinho e em comparações com resultados gerados por outras ferramentas, como acontece em [Dekker, 1998], cujos resultados gerados pelo Linpack são comparados com resultados analíticos de um modelo. Atualizações desse conjunto foram implementadas para melhor adequação aos sistemas atuais, além da criação de uma versão chamada HPL ("Highly Parallel Computing Linpack"), desenvolvida para explorar a escalabilidade de sistemas paralelos. As diversas distribuições do "benchmark" podem ser adquiridas em *http://www.netlib.org/linpack/*.

Lapack [Dongarra, 1999] e Scalapack [Dongarra, 1995]

Essas duas bibliotecas são atualizações da biblioteca Linpack, e têm como objetivo principal permitir que o conjunto de aplicações do Linpack seja executado de forma eficiente em computadores vetoriais (Lapack) e computadores de memória distribuída (Scalapack). São baseadas em algoritmos de particionamento de blocos, o que minimiza a frequência do movimento de dados entre diferentes níveis de hierarquia de memória e aumenta o desempenho do sistema.

SPEC [Uniejewski, 1989]

SPEC (Standard Performance Evaluation Corporation) é uma corporação formada por grandes fabricantes de computadores, buscando *desenvolver, manter e apoiar um conjunto padronizado de "benchmarks" que possam ser aplicados a mais nova geração de computadores.* Seu pacote é constituído por cerca de 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 versão.

Inicialmente com programas que analisavam o desempenho apenas da CPU, da FPU (unidade de ponto flutuante) e do subsistema de memória, o pacote possui atualmente programas para análise do desempenho de outros subsistemas, como os de comunicação, entrada/saída e armazenamento. Existem também programas que fornecem medidas para aplicações, como é o caso do SPECWeb, que analisa o desempenho de servidores web.

O pacote SPEC define que os algoritmos e programas podem ser adequados pelo usuário que estiver testando o equipamento. Isto, no caso de equipamentos paralelos, permite um ajuste do sistema para a arquitetura da máquina, podendo resultar em um melhor desempenho do que se fosse utilizado um sistema de avaliação genérico.

Outras ferramentas

Existem muitas ferramentas destinadas à análise de sistemas de hardware, cada uma com sua particularidade e característica buscando analisar não apenas o sistema completo, mas também subsistemas. Outros exemplos de ferramentas que analisam sistemas de hardware são os "benchmarks" NAS [Waheed, 1998], TPC [TPC, 2002], Splash [Woo, 1995], Parkbench [Netlib, 2002B] e o simulador SimpleScalar [Austin, 2002].

3.4 Ferramentas para software

Nesta seção são apresentados alguns ambientes e ferramentas para análise de desempenho de sistemas de software. Alguns podem ser aplicados também à análise de hardware, mas possuem como característica principal a aplicação em sistemas de software.

3.4.1 Ferramentas baseadas em simulação

ASiA (Ambiente de Simulação Automático) [Santana, 1996]

ASiA é um ambiente de simulação que permite ao usuário descrever o programa através de um editor gráfico. Neste editor, o usuário pode definir pontos onde os dados estatísticos serão coletados, o formato como eles serão apresentados como saída, além de escolher a linguagem de simulação alvo.

Após a modelagem gráfica, o usuário não necessita mais iteragir com o ambiente. O editor gráfico interpreta as informações inseridas pelo usuário e as organiza em estruturas previamente definidas para fornecer como saída o programa de simulação; no caso de usuários experientes o simulador oferece a possibilidade de edição do código gerado, para adição de módulos específicos.

O objetivo do ASiA, de acordo com seus criadores, é evitar que o usuário tenha que participar do processo de tradução do modelo gráfico para o programa a ser simulado.

3.4.2 P3T+ [Fahringer, 2000]

P3T ("Parameter based Performance Prediction Tool")[Fahringer, 1995] é uma ferramenta desenvolvida como parte do ambiente VFCS ("Vienna Fortran Compilation System"), um ambiente que permite a criação de programas paralelos através de troca de mensagens a partir de programas escritos em Vienna Fortran e HPF (com algumas restrições).

P3T+, o sucessor do P3T, é uma ferramenta que prediz o desempenho de programas paralelos e distribuídos, com diferentes modelos de programação e arquiteturas. Diferentemente do P3T, oferece suporte pleno à linguagem HPF, além de analisar programas escritos em Fortran V90 e bibliotecas de troca de mensagens gerados pelo Vienna. A ferramenta apresenta ao usuário alguns aspectos relacionados ao desempenho do programa, como informações sobre a comunicação (quantidade de dados transferida, número de transferências, e o tempo gasto nelas), distribuição de carga entre as tarefas, tempo de computação e taxa de acerto no acesso à memória cache.

Os dados utilizados pela ferramenta são oriundos de um outro módulo do conjunto Vienna, que realiza "profiling" no programa fonte ignorando as construções paralelas para extração das informações relevantes, como frequência na execução e probabilidades de execução de laços.

3.4.3 Ferramentas baseadas em *benchmarking*

AIMS [Yan, 1995]

AIMS ("Automated Instrumentation and Monitoring System") é um conjunto de ferramentas desenvolvido pela NASA, que pode ser aplicado na avaliação e detecção de falhas no desempenho de sistemas. Tem como característica a instrumentação e análise de programas escritos em Fortran 77 e C que utilizam bibliotecas de troca de mensagens PVM, MPI e NX.

O conjunto é composto de três ferramentas básicas:

- Xinstrument : ferramenta de instrumentação do código fonte; insere chamadas às rotinas de monitoramento de desempenho na aplicação do usuário;
- Biblioteca de monitoramento do desempenho em tempo de execução, que consiste em um conjunto de rotinas de monitoramento que medem e gravam vários dados sobre o desempenho do programa;

- Um módulo que processa os dados sobre o desempenho do programa e os disponibilizam de uma forma amigável para o usuário.

A instrumentação do código fonte pode ser realizada tanto pela ferramenta XInstrument quanto através de linha específica de comando para instrumentação. O usuário pode selecionar quais partes do programa serão instrumentadas (como por exemplo todas as rotinas e todos os eventos de entrada/saída) e suspender a instrumentação a qualquer momento da execução do programa.

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 utilizando o código fonte do programa;
- a execução da ferramenta propriamente dita, quando é criada uma versão instrumentada do programa (aqui várias opções podem ser utilizadas para produção de diferentes tipos de medidas);
- a execução do programa gerado pelo IDTrace, que cria dois novos arquivos, sendo um deles o arquivo que pode ser utilizado por simuladores.

Através da execução do IDTrace, podem ser obtidos traços de eventos de diferentes tipos, como traços de laços na execução, traços sobre memória cache, traços sobre a execução do programa.

Paradyn [Miller, 1995]

Paradyn é uma ferramenta utilizada na análise de desempenho de programas paralelos e distribuídos de grande escala (programas que possuem tempo de

execução elevado, chegando a várias horas ou dias, executados em máquinas com número elevado de nós de processamento).

Realiza a instrumentação do código executável da aplicação dinamicamente, buscando extrair traços dos eventos sem que o usuário precise alterar o código fonte do programa ou necessite de um compilador especial. A instrumentação e a análise de desempenho são realizadas durante a execução do programa aplicação, em tempo real, disponibilizando informações ao usuário assim que elas são obtidas. Existe também a possibilidade da análise dos arquivos gerados após a execução do programa (análise *pós-mortem*).

A ferramenta decide quais dados coletar enquanto o programa está sendo executado, através da biblioteca de instrumentação dinâmica cujo nome é DyninstAPI [Buck, 2000], adicionando pequenas porções de instrumentação no código executável da aplicação para procurar possíveis pontos de degradação do desempenho (como gargalos de comunicação ou entrada/saída). A cada ponto de degradação encontrado a ferramenta adiciona uma quantidade maior de instrumentação para pesquisar o possível problema.

Paradyn usa os conceitos de "*metric-focus grid*" (grade orientada à medida) e "*time-histogram*" (histograma de tempo) para selecionar, analisar e apresentar dados sobre o desempenho. A grade é um conjunto de dois vetores, um que armazena uma lista de medidas de desempenho (como tempo de CPU, taxa de mensagem, taxa de utilização de entrada/saída), e outro que contém uma lista de componentes de programa (processos, disco, canais de comunicação e instâncias de barreira). O produto dos dois vetores produz uma matriz com cada medida listada para cada componente do programa, e o usuário pode selecionar o par ordenado para observar seu comportamento.

Pablo Performance Analysis Environment [Reed, 1994]

Pablo é um conjunto de ferramentas desenvolvido na Universidade de Illinois que apresenta uma diversidade de medidas de avaliação bastante elevada. É composto 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). Tal formato torna a ferramenta difundida por ser o padrão nativo para a ferramenta proprietária para análise de desempenho ParAide, usada nas antigas máquinas Paragon XP/S da Intel. Um outro fato que facilita a difusão da ferramenta é a distribuição de conversores de outros formatos de arquivos de dados de desempenho (como AIMS e PICL) para o formato SDDF.

A biblioteca de instrumentação permite realizar a obtenção dos dados para análise de ciclos na execução do programa, dispositivos de Entrada/Saída e da comunicação usando MPI. A interface para o usuário, SvPablo, permite a observação de grande parte dessas medidas, além da exibição do código fonte com os pontos de instrumentação.

Uma característica positiva da ferramenta é a possibilidade de avaliação do subsistema de comunicação e da troca de mensagens apenas adicionando referência à biblioteca de análise fornecida com a ferramenta, se o programa for implementado com bibliotecas MPI, não sendo necessária nenhuma instrumentação.

PAT - Performance Analysis Tool [Cray T-series]

PAT é uma ferramenta proprietária dos computadores Cray que permite ao usuário realizar vários tipos de análise de desempenho, sem sobrecarregar o processador. A ferramenta pode analisar programas executáveis escritos em várias linguagens, entre elas C e Fortran 90, apenas ligando o arquivo do programa executável com a biblioteca PAT e outro arquivo proprietário da ferramenta. Uma vantagem da ferramenta consiste na não necessidade de recompilar o código do programa para análise. Algumas análises permitidas pela ferramenta permitem visualizar a quantidade de tempo gasto pelas funções do programa alvo, criação e análise de arquivos de dados contendo traços dos eventos ocorridos durante a execução do programa, quantidade de chamadas a rotinas, etc.

A ferramenta faz uso dos contadores de desempenho de hardware dos

computadores Cray para criar os arquivos contendo os traços de eventos do sistema, podendo inserir chamadas para obtenção dos dados de determinadas partes do programa. Os arquivos gerados pela ferramenta podem ser visualizados pela ferramenta VAMPIR [Pallas].

Total View [Etnus, 2002]

Etnus Total View é um conjunto comercial de ferramentas que auxilia o pesquisador na análise de programas seriais e paralelos. Desenvolvida para diversos ambientes UNIX e Linux, possui vários módulos para análise da execução dos programas. Classificado pela empresa que o criou como um "profiler" e como depurador de código, oferece ao usuário medidas de desempenho em relação ao tempo de processamento, comunicação e endereçamento de memória, entre outros.

A interface com o usuário é realizada através do conjunto de ferramentas de visualização. Tais ferramentas permitem ao usuário observar código fonte, tamanho das variáveis, valor das variáveis, tempo de execução das rotinas do programa, situação das unidades de processamento (ociosa/ecupada). Um modo de exibição que se destaca é o referente à troca de mensagens utilizando MPI, com grafos das mensagens trocadas entre as unidades de processamento e informações sobre o tamanho e tipo das mensagens enviadas.

Vampir/Vampirtrace [Pallas]

Vampir é um conjunto comercial de ferramentas para extração de traços de eventos de programas desenvolvida pela PALLAS GmbH. O conjunto é utilizado para extração de traços na comunicação de programas que utilizam biblioteca MPI, e são construídos com linguagens C ou Fortran.

A instrumentação é realizada através da adição da biblioteca VampirTrace no programa onde se deseja realizar as medições. Os arquivos contendo os traços da comunicação são colhidos a partir da execução do programa instrumentado.

O conjunto possui uma ferramenta com três módulos para visualização das

medidas de desempenho: Process State Display, Statistics Display, Timeline Display. O primeiro mostra todos os processos como caixas e mostra o estado de cada um em relação ao tempo. O segundo apresenta as estatísticas totais do arquivo de dados em forma de gráficos de torta. O último apresenta os estados de processos durante o período de análise e a comunicação entre processos.

Não é possível adicionar análise de linhas específicas do programa, mas é possível analisar o desempenho de conjuntos de rotinas. Por ser uma ferramenta comercial, pode ser executada em várias plataformas. Os arquivos de dados podem ser gravados em formato binário ou formato simples (texto), e para tal exigem uma grande quantidade de espaço em disco para que as informações sejam guardadas (caso o programa alvo realize muita comunicação).

VT - Visualization Tool [Browne, 1997]

VT, *Visualization Tool*, é uma ferramenta que faz parte dos pacotes distribuídos com os computadores IBM SP, que pode tanto analisar eventos após a execução dos programas instrumentados como também monitorar a execução do programa durante sua execução.

Para análise pós-execução, é necessário apenas executar o programa com a opção de "tracing" habilitada para que os dados sejam gerados. VT pode realizar a análise de dados gerados por comunicação ponto a ponto e coletiva, estatísticas do núcleo do sistema operacional AIX, além de realizar a análise de dados gerados por instrumentação do código da aplicação.

Os dados são criados com o formato específico, que podem depois ser analisados através do Painel de Controle. A visualização dos dados pode acontecer de diversas formas, tal como instantânea (informação em um ponto específico da análise) ou em pequenas quantidades de tempo ("streaming"). A parte de visualização consiste de um grupo de analisadores que apresentam dados complexos do sistema de uma forma facilmente interpretada pelo usuário, como gráficos de barra e em formato circular.

3.5 Ferramentas de Visualização

Um conjunto especial de ferramentas são as de visualização, que não geram os dados para análise ou predição de desempenho, mas apenas tratam os dados já existentes, gerados por outras ferramentas, e apresentando-os ao usuário de forma amigável.

Além das ferramentas descritas a seguir podem ser citadas algumas implementações específicas, tais como XPVM, XMPI, UPSHOT e NUPSHOT, as quais, em geral, se comportam de modo semelhantes às quais aqui descritas.

PARADE [Statsko, 1995]

PARADE é um ambiente para criação e animação de programas paralelos e distribuídos. A ferramenta de animação, chamada POLKA, é utilizada para visualizar programas escritos em diferentes linguagens e baseados em diferentes arquiteturas.

A ferramenta de animação permite ao usuário configurar o ambiente para visualizar programas paralelos sobre um grande conjunto de dados, podendo analisar aplicações escritas utilizando as bibliotecas MPI, PVM e programas que fazem uso de "threads", nas linguagens C e HPF.

ParaGraph e ParaGraph+ [Heath, 1995]

Paragraph é uma ferramenta gráfica para visualização do comportamento do desempenho de programas paralelos em computadores paralelos e distribuídos. Esta ferramenta permite um "playback" do programa paralelo baseado nos traços de execução que foram coletados por outras ferramentas durante a execução do programa. A informação pode ser visualizada de várias formas, como por exemplo utilização de processador em relação ao tempo, ou tráfego de comunicação em relação ao tempo, etc.

Paragraph+ é a implementação comercial da ferramenta Paragraph, distribuída pela empresa Pallas GmbH.

Capítulo 4

Classificação das técnicas e ferramentas

No capítulo dois foram descritas e discutidas as estratégias de classificação de técnicas e ferramentas para análise de desempenho. No capítulo seguinte várias técnicas e ferramentas foram apresentadas, com breves descrições sobre suas características e particularidades.

Neste capítulo apresenta-se a classificação das técnicas e ferramentas para análise de desempenho de sistemas tendo por base as estratégias apresentadas no capítulo 2. As características das técnicas e ferramentas são apresentadas no formato de tabelas e, quando necessário, comentários são tecidos com o objetivo de complementar essas informações. São apresentadas também considerações sobre o processo de seleção da técnica ou ferramenta apropriada para uma dada situação.

As técnicas e ferramentas estão dispostas de acordo com o sistema de aplicação (hardware ou software) e seu tipo. Para cada ferramenta ou técnica é apresentado um quadro com suas características em forma tabular, que pode ser facilmente consultado para a identificação de sua adequação a cada problema. Mesmo com a apresentação das características de cada técnica e ferramenta individualmente neste capítulo, uma tabela de referência rápida, reunindo as principais informações de cada ferramenta ou técnica, é apresentada no Apêndice A.

4.1 Técnicas aplicáveis a sistemas de hardware

4.1.1 Análise de subsistema de Entrada/Saída

[Seção 3.1.1] [Ganger, 1998]

Nome	Análise de subsistema de E/S
Sistema aplicado	hardware
Abordagem para obtenção dos resultados	simulação (criação do subsistema) benchmarking (coleta dos dados)
Forma de medição ou instrumentação	não se aplica na etapa de simulação monitoração por software (coleta dos dados)
Tipos de dados gerados pela análise	não se aplica na etapa de simulação extração de traços de eventos (coleta dos dados)
Arquitetura	–
Informações fornecidas	número de requisições de E/S, utilização da CPU, tempo médio de operação, tempos médios de espera, acesso e resposta nos processos de E/S
Análise (sistema alvo)	subsistema de E/S, isolado e em contato com o sistema completo

Tabela 4.1: Características da técnica de análise de subsistema de E/S.

Esta técnica pode ser dividida em duas etapas distintas: a primeira consiste na coleta dos dados contendo os traços dos eventos, realizada através da instrumentação do sistema operacional e conseqüente monitoração por software. A segunda é a simulação dos eventos propriamente dita.

Na primeira fase são coletados os traços da atividade do sistema, através da monitoração realizada pelo sistema operacional. Essa extração de traços não altera significativamente o desempenho do sistema (chegando a, no máximo, 0,1% de degradação). Após a extração dos traços um pós-processador escolhe quais os dados serão utilizados na etapa de simulação; para que isso seja possível, a ferramenta utilizada captura também eventos auxiliares do sistema.

Nesta fase o simulador é utilizado para fornecer as medidas de desempenho. É realizada a simulação tanto do dispositivo isolado (direcionado por eventos de E/S) quanto do dispositivo conectado a um sistema completo (direcionado por eventos de sistemas).

4.1.2 Análise do desempenho da memória em sistemas DSM

[Seção 3.1.3] [Abandah, 1998]

Nome	Análise de desempenho de memória em sistemas DSM
Sistema aplicado	hardware
Abordagem para obtenção dos resultados	benchmarking
Forma de medição ou instrumentação	–
Tipos de dados gerados pela análise	medição de intervalo de tempos
Arquitetura	SIMD, MIMD (estrutura dos benchmarks)
Informações fornecidas	tempo de acesso à memória (local, cache e compartilhada), largura de banda necessária na execução dos programas, tempo de sobregarga para sincronismo
Análise (sistema alvo)	subsistema de memória, subsistema de escalonamento (sist. operacional)

Tabela 4.2: Características da técnica de análise de desempenho de sistemas DSM.

O desempenho do subsistema de memória é avaliado em duas situações: uma quando apenas um processador tem acesso ao subsistema, representando o melhor caso, em que não existe disputa para acessar os dispositivos (apenas um processador escreve e lê nas memórias); e outra quando vários processadores disputam o acesso às memórias e aos meios de comunicação (entre os vários "hypernodes" e, dentro de cada um, entre as unidades funcionais).

A degradação de desempenho resultante da disputa entre os processadores é avaliada confrontando os dados gerados pelas medições nos dois casos.

4.1.3 Análise do tempo de comunicação e determinação do caminho baseada em desempenho

[Seção 3.1.3] [Kim, 1999]

Nome	Análise do tempo de comunicação
Sistema aplicado	hardware
Abordagem para obtenção dos resultados	benchmarking
Forma de medição ou instrumentação	não se aplica
Tipos de dados gerados pela análise	medição de intervalo de tempos
Arquitetura	–
Informações fornecidas	atraso no tempo de comunicação, speedup para diversos tamanhos de mensagens
Análise (sistema alvo)	subsistema de comunicação (várias formas de comunicação)

Tabela 4.3: Características da técnica de determinação do caminho de comunicação.

Nesta técnica vários "benchmarks" diferentes são utilizados na análise do desempenho do subsistema de comunicação. A seleção do melhor caminho (meio físico de comunicação) a ser seguido é realizada dinamicamente tendo como base o comportamento do subsistema de comunicação durante a execução dos programas.

Três são as opções de seleção dos caminhos a serem percorridos durante a execução dos programas: a manutenção do caminho original, a seleção de um único caminho, quando esse é sempre o de melhor desempenho, e a agregação de meios, quando o desempenho deles varia durante a execução.

4.1.4 Caracterização do desempenho de um computador paralelo vetorial com memória compartilhada

[Seção 3.1.2] [Dekker, 1998]

Nome	Caracterização do desempenho de um computador paralelo vetorial com memória compartilhada
Sistema aplicado	hardware
Abordagem para obtenção dos resultados	analítico
Forma de medição ou instrumentação	não se aplica
Tipos de dados gerados pela análise	não se aplica
Arquitetura	SIMD (computador vetorial)
Informações fornecidas	medidas sobre escalabilidade do sistema, pico de desempenho, tempo de acesso à memória
Análise (sistema alvo)	sistema computacional

Tabela 4.4: Características da técnica de análise e caracterização de computador paralelo e vetorial.

Nesta técnica, os dados sobre o desempenho e a escalabilidade do computador paralelo vetorial apresentados são resultantes da modelagem matemática. Deve-se salientar que o desempenho do sistema é analisado apenas em relação ao número de operações de ponto flutuante (picos de desempenho) e não são considerados fatores que degradam o desempenho como em um sistema real, entre eles o tempo necessário para escalonamento, E/S, conflitos de memória e rede, entre outros.

4.1.5 Estudo de casos de "prefetching" de instruções em "cache"

[Seção 3.1.1] [Hsu, 1998]

Nome	Estudo de casos de "prefetching" de instruções em "cache"
Sistema aplicado	hardware
Abordagem para obtenção dos resultados	simulação benchmarking
Forma de medição ou instrumentação	monitoração por hardware
Tipos de dados gerados pela análise	medição de intervalo de tempos extração de traços de eventos
Arquitetura	não se aplica
Informações fornecidas	eficiência da técnica, taxa de erro de prefetch, tráfego para memória
Análise (sistema alvo)	subsistema de memória (cache)

Tabela 4.5: Características da técnica de análise de métodos de "prefetching".

Os procedimentos na técnica são divididos em dois blocos. No primeiro os dez "benchmarks" são executados em um computador real, com o monitor de desempenho de hardware[1]. No outro bloco são realizadas simulações tomando como fonte traços das instruções de um computador Cray Y-MP, utilizando a organização original da memória cache e outras situações consideradas importantes pelos analistas.

Como o desempenho da ação de "prefetching" não pode ser obtido de uma forma exata, os autores apresentam uma medida chamada *eficiência de prefetching*, baseada em alguns valores médios. Para cada linha de instrução levada ao cache, o monitor é acessado e dados são colhidos no início do processo e no momento em que pela primeira vez a linha é referenciada; o número de erros é então calculado e dividido por um valor médio de pulsos de relógio gastos por instrução, resultando na *eficiência de prefetching*.

[1] "HPM – Hardware Performance Monitor" – monitor oferecido pelo fabricante no computador Y-MP.

4.1.6 Paralelização do benchmark STAP

[Seção 3.1.3] [Hwang, 1999]

Nome	Paralelização do benchmark STAP
Sistema aplicado	hardware
Abordagem para obtenção dos resultados	benchmarking e analítico (para alguns sistemas)
Forma de medição ou instrumentação	não se aplica
Tipos de dados gerados pela análise	medição de intervalo de tempos
Arquitetura	SIMD (programa desenvolvido para essa arquitetura)
Informações fornecidas	speedup, eficiência, tempo na execução (discos, comunicação), memória utilizada, tempo de atraso para "start-up" do programa
Análise (sistema alvo)	sistema completo (subsistema de E/S, comunicação e memória interligados)

Tabela 4.6: Características da técnica de paralelização do benchmark STAP.

Esta técnica é classificada como uma técnica aplicável a sistemas de hardware porque, apesar dos autores realizarem a paralelização de um programa (o STAP), toda a avaliação de desempenho é voltada ao sistema de hardware. Os autores focam o trabalho nas alterações do desempenho devido à fatores como tamanho da máquina (número de processadores), velocidade de processamento, atraso em comunicação (troca de mensagens), e tamanho da banda agregada destinada à comunicação.

Os dados dos sistemas que estavam em desenvolvimento foram apresentados através de modelagem analítica, tendo como base informações fornecidas por seus fabricantes. Comparando os dados da técnica com dados recentes [TOP500, 2002] pode-se concluir que, devido à imprecisão no modelo dos computadores que estavam sendo construídos, alguns dados apresentados não descrevem a realidade. Tal fato não torna a técnica inválida, mas é um bom exemplo de que a escolha de dados imprecisos pode levar à obtenção de resultados espúrios.

4.1.7 Teoria de filas aplicada no cálculo de limitantes

[Seção 3.1.2] [Balsamo, 1998]

Nome	Cálculo dos limitantes de desempenho através de teoria de filas
Sistema aplicado	hardware
Abordagem para obtenção dos resultados	analítico
Forma de medição ou instrumentação	não se aplica
Tipos de dados gerados pela análise	não se aplica
Arquitetura	MIMD
Informações fornecidas	tempo de resposta dos processos e das tarefas, atraso para sincronismo, tamanho da fila de distribuição e carga de trabalho
Análise (sistema alvo)	sistema computacional, sistema de escalonamento de processos

Tabela 4.7: Características da técnica de análise de desempenho usando teoria de filas.

A aplicação de teoria de filas não se restringe apenas a sistemas de hardware, mas também a sistemas de software. A utilização de redes de filas e redes de filas estendidas também é ampla quando os modelos para análise de desempenho são baseados em simulação.

4.2 Técnicas aplicáveis a sistemas de software

4.2.1 Conversão do código executável em grafo de execução

[Seção 3.2.1] [Manacero Jr., 1997]

Nome	Conversão do código em grafo de execução
Sistema aplicado	software
Abordagem para obtenção dos resultados	simulação
Forma de medição ou instrumentação	análise de fluxo pela simulação do programa reescrito
Tipos de dados gerados pela análise	traços de eventos, profiling
Arquitetura	SIMD, MIMD
Informações fornecidas	tempo de processamento do programa, velocidade média de processamento, velocidade média por cliente, tempo médio de espera por cliente, "speedup"
Análise (sistema alvo)	programas paralelos

Tabela 4.8: Características da técnica de conversão de código em grafo de execução.

O método proposto é baseado na reescrita do código executável em um grafo de execução, e na posterior simulação do grafo gerado, para extração dos dados sobre o desempenho do programa no modelo de máquina pré-determinado.

De acordo com o autor, algumas vantagens desta proposta são a precisão obtida com a reescrita do código executável na forma de grafo de execução e o baixo custo de simulação (devido à não necessidade da máquina paralela para execução).

4.2.2 GSPN - rede de Petri estocástica generalizada

[Seção 3.2.3] [Granda, 1992]

Nome	GSPN
Sistema aplicado	software (também pode ser aplicado para análise de hardware)
Abordagem para obtenção dos resultados	analítico, simulação
Forma de medição ou instrumentação	não se aplica
Tipos de dados gerados pela análise	não se aplica
Arquitetura	sistemas paralelos
Informações fornecidas	medidas do comportamento do sistema (tempo gasto por ciclo, tempo gasto por cada tarefa no ciclo, tempo de espera entre os eventos)
Análise (sistema alvo)	sistemas multiprocessados

Tabela 4.9: Características da técnica de análise de desempenho usando GSPNs.

O motivo principal que levou Granda a propor uma técnica baseada em redes de Petri estocásticas generalizadas foi a limitação das redes de Petri comuns em modelar sistemas paralelos complexos que possuiam lugares ilimitados (e conseqüente explosão do número de estados analisados). A similaridade desse tipo de rede de Petri com cadeias de Markov foi outro fator importante na escolha da técnica (os estados de uma GSPN correspondem aos estados de sua cadeia de Markov equivalente).

A transformação da rede de Petri em cadeia de Markov acontece quando são aplicadas as técnicas de agregação de estados (horizontal e vertical), seguindo o diagrama mostrado na descrição da técnica.

4.2.3 Modelagem de limitantes de desempenho

[Seção 3.2.1] [Lui, 1998]

Nome	Modelagem de limitantes de desempenho
Sistema aplicado	software
Abordagem para obtenção dos resultados	simulação, analítico
Forma de medição ou instrumentação	não se aplica
Tipos de dados gerados pela análise	não se aplica
Arquitetura	–
Informações fornecidas	tempos de resposta nos casos de limitantes inferior e superior
Análise (sistema alvo)	programas paralelos que usam paradigma "fork-join"

Tabela 4.10: Características da técnica de modelagem de limitantes de desempenho.

Os autores apresentam neste trabalho uma metodologia para analisar os limitantes superiores e inferiores de desempenho.

Na técnica, os autores mesclam dois tipos de metodologia para análise de desempenho: metodologia analítica e de simulação. Isso ocorre pois o problema não pode ser resolvido diretamente utilizando análise numérica devido ao número de estados do programa que utiliza fork-join serem infinitos.

4.3 Ferramentas aplicáveis a sistemas de hardware

4.3.1 FTIO benchmark

[Seção 3.3.2] [Fagerstrom, 2000]

Nome	FTIO benchmark
Sistema aplicado	hardware
Abordagem para obtenção dos resultados	benchmarking
Forma de medição ou instrumentação	monitoração por software (perfex)
Tipos de dados gerados pela análise	medição de intervalo de tempos e contagem de eventos (perfex)
Arquitetura	–
Informações fornecidas	fração do tempo total de execução dispendido em E/S, ciclos gastos por operação de ponto flutuante
Análise (sistema alvo)	subsistema de Entrada/Saída

Tabela 4.11: Características do benchmark FTIO.

Este "benchmark" foi desenvolvido para analisar o subsistema de entrada e saída. Um dos componentes da ferramenta, o utilitário *perfex*, é responsável por duas características na classificação: a monitoração por software e a contagem de eventos. A ferramenta ainda realiza medição de intervalo de tempos ao analisar o tempo dispendido com entrada/saída.

Os parâmetros de entrada da ferramenta, que é disparada com auxílio do gerenciador de filas denominado PBS, determinam as dimensões da matriz usada como padrão de entrada da ferramenta. A distribuição dos dados entre os processadores é realizada na fase inicial do "benchmark", sendo que cada processador envolvido na análise recebe um bloco diferente da matriz.

Durante a execução da ferramenta existe uma redistribuição dos dados da matriz entre os processos. Isto faz com que ocorra uma grande troca de informações entre os processadores, permitindo melhor análise do sistema de entrada/saída.

4.3.2 Linpack e variações

[Seção 3.3.2] [Dongarra, 1988]

Nome	Linpack, lapack, scalapack, hplinpack
Sistema aplicado	hardware
Abordagem para obtenção dos resultados	benchmarking
Forma de medição ou instrumentação	–
Tipos de dados gerados pela análise	contagem de eventos, medição de intervalo de tempos
Arquitetura	SISD, SIMD e MIMD
Informações fornecidas	número de operações (em GFlops) máximo, tamanho da matriz em que o número de operações é máximo, número de operações com a matriz reduzida (metade da original)
Análise (sistema alvo)	sistema computacional

Tabela 4.12: Características do benchmark linpack e suas variações.

A versão original do "benchmark" Linpack (matriz 100x100), passado um período de sua criação, deixou de ser o maior "benchmark" na história da análise de desempeno para ser uma ferramenta esquecida. Tal fato deu-se devido a vários fabricantes conseguirem colocar na memória "cache" todas as estruturas utilizadas durante a execução da ferramenta. Para contornar esse problema, o tamanho da matriz precisou ser alterado, tornando o Linpack um "benchmark" aceito novamente pela comunidade.

Apesar desse fato, os dados fornecidos pelo "benchmark" Linpack continuaram (e ainda continuam) servindo de base para comparação entre diferentes computadores de todo o mundo, como acontece em *"Top 500 Report"* [TOP500, 2002].

4.3.3 PAWS - Parallel Assessment Window System

[Seção 3.3.1] [Pease, 1991]

Nome	PAWS
Sistema aplicado	hardware, software, ambos
Abordagem para obtenção dos resultados	simulação
Forma de medição ou instrumentação	não se aplica
Tipos de dados gerados pela análise	profiling, medição de intervalo de tempos
Arquitetura	SISD, SIMD, MISD, MIMD (de acordo com os autores)
Informações fornecidas	curvas sobre speedup, perfil do paralelismo, perfil da execução dos programas
Análise (sistema alvo)	sistema completo ou subsistemas

Tabela 4.13: Características da ferramenta PAWS.

Um fator interessante do PAWS reside na abordagem usada no sistema, que separa os modelos de arquiteturas da máquina dos modelos de programa, abordagem utilizada em grande parte das ferramentas de análise de desempenho, permitindo que máquinas diferentes e algoritmos diferentes sejam analisados.

Para analisar os sistemas de hardware (computadores), a ferramenta de caracterização de arquitetura divide a arquitetura de um tipo específico de sistema em quatro categorias: computação, transporte de dados e comunicação, entrada/saída e controle. Cada categoria pode ser reparticionada em subsistemas repetidamente até que seja tão pequeno que possa ser temporizada.

4.3.4 PDL - Performance Description Language

[Seção 3.3.1] [Vemuri, 1996]

Nome	PDL - Performance Description Language
Sistema aplicado	hardware, software, ambos
Abordagem para obtenção dos resultados	simulação
Forma de medição ou instrumentação	não se aplica
Tipos de dados gerados pela análise	não se aplica
Arquitetura	qualquer, a princípio
Informações fornecidas	ciclos por instrução, tempo de processador gasto por cada instrução, custo dos procedimentos
Análise (sistema alvo)	sistema completo

Tabela 4.14: Características da linguagem e do simulador PDL.

O simulador é baseado na linguagem de descrição PDL. Uma característica de destaque deste simulador é a possibilidade de definir separadamente os modelos de hardware e software, e ainda incluir os custos relativos a cada um dos itens modelados.

Alguns sistemas modelados com a linguagem PDL estavam disponíveis para testes, mas a incompatibilidade das versões apresentadas com os sistemas disponíveis para testes impediram uma análise mais detalhada sobre o funcionamento do simulador.

4.3.5 RSim - The Rice Simulator for ILP Multiprocessors

[Seção 3.3.1] [Hughes, 2002]

Nome	RSim
Sistema aplicado	hardware (software)
Abordagem para obtenção dos resultados	simulação
Forma de medição ou instrumentação	não se aplica
Tipos de dados gerados pela análise	não se aplica
Arquitetura	SIMD, MIMD
Informações fornecidas	estatísticas sobre o processador (utilização, tempo de uso, etc.), memória, "cache" e rede de conexão
Análise (sistema alvo)	sistemas de hardware (computador e subsistemas), sistemas de software

Tabela 4.15: Características da ferramenta de simulação RSim.

O ambiente RSim consiste de dois elementos: a aplicação paralela a ser simulada e o simulador. O simulador é composto de três módulos: os módulos de processador, sistema de memória e rede de conexão. O módulo de processador simula as fases de decodificação, edição, execução e saída do pipeline para cada instrução. O módulo de sistema de memória simula o comportamento do subsistema de memória incluindo "cache", "buffers", protocolo de coerência, e a memória principal. Já o módulo de rede é usado para simular a rede de conexão através de topologias definidas pelo usuário e algoritmos de roteamento.

Em relação às estatísticas de desempenho, o simulador apresenta tanto métricas de desempenho global, como o número total de ciclos de execução e número de instruções por ciclo que um programa atinge no sistema simulado, quanto métricas particulares, como a utilização de unidades funcionais no processador e comportamento na predição de laços.

Outras métricas apresentadas pela ferramenta são atraso médio das várias

classes de operações de memória, utilização de barramento, utilização do "buffer" de escrita e tráfego nos canais de comunicação.

Para várias métricas o simulador ainda disponibiliza outras funcionalidades, incluindo valores médios, desvios padrão e histogramas sobre o comportamento das medidas durante a execução de programas.

4.3.6 Simics

[Seção 3.3.1] [Magnusson, 2002]

Nome	Simics
Sistema aplicado	hardware (software)
Abordagem para obtenção dos resultados	simulação
Forma de medição ou instrumentação	não se aplica
Tipos de dados gerados pela análise	não se aplica
Arquitetura	qualquer
Informações fornecidas	número de instruções por unidade de processamento, número de operações por segundo, número de operações por segundo em cada unidade de processamento
Análise (sistema alvo)	sistemas de hardware

Tabela 4.16: Características do simulador Simics.

Esta é uma ferramenta que não foi desenvolvida para avaliação de desempenho, mas tem sido utilizada por analistas com o intuito de testar determinadas plataformas antes de adquiri-las. Ela também pode ser utilizada para análise de sistemas de software em determinados ambientes simulados que se aproximem do ambiente real de execução.

Uma característica que torna vantajosa a utilização da ferramenta é a possibilidade de simular em um único ambiente várias arquiteturas e computadores de vários fabricantes, além da possibilidade da alteração de módulos dos sistemas simulados.

4.3.7 SPEC

[Seção 3.3.2] [Uniejewski, 1989]

Nome	SPEC (conjunto padronizado de benchmarks)
Sistema aplicado	hardware/software
Abordagem para obtenção dos resultados	benchmarking
Forma de medição ou instrumentação	não se aplica
Tipos de dados gerados pela análise	medição de intervalo de tempos
Arquitetura	qualquer
Informações fornecidas	velocidade de processamtento, vazão, medidas comparativas em SPECs
Análise (sistema alvo)	sistema completo e vários subsistemas, como CPU, subsistema de memória, compiladores e sistemas dedicados (cliente/servidor).

Tabela 4.17: Características do conjunto SPEC.

O conjunto SPEC conta atualmente com ferramentas que analisam o desempenho de sistemas computacionais completos ou de subsistemas, como CPU, memória, vídeo, e também de alguns sistemas de software, como servidores de e-mail, "web servers" e servidores de aplicação.

4.4 Ferramentas aplicáveis a sistemas de software

4.4.1 AIMS

[Seção 3.4.3] [Yan, 1995]

Nome	AIMS
Sistema aplicado	software
Abordagem para obtenção dos resultados	benchmarking
Forma de medição ou instrumentação	instrumentação do código fonte, monitoração por software
Tipos de dados gerados pela análise	profiling, medição de intervalo de tempos, extração de traços de eventos
Arquitetura	MIMD, SIMD
Informações fornecidas	tempo de execução do programa e de subrotinas, eventos de E/S e troca de mensagens
Análise (sistema alvo)	sistema de software

Tabela 4.18: Características da ferramenta AIMS.

Os dados sobre o desempenho dos programas são gerados a partir da execução do programa compilado referenciando as bibliotecas de monitoramento. A ferramenta gera arquivos separados para cada processador e, após a execução, une os arquivos para a criação de um arquivo totalizador das estatísticas ou das informações.

Os arquivos então podem ser analisados através do módulo de visualização (VK - Visual Kernel). Essa ferramenta possui um controle parecido com o de um "playback" e, literalmente, reproduz a execução, exibindo os dados sobre o desempenho. O usuário pode, se desejar, adicionar pontos de análise particulares para verificar a situação do sistema.

As informações oferecidas pelo conjunto são os eventos de entrada e saída, eventos de troca de mensagens, tempos de execução das sub-rotinas e do programa. Aspectos negativos sobre o conjunto são o número reduzido de plataformas onde a ferramenta pode ser executada e alguns erros na amostragem das medidas durante a fase de análise de desempenho.

A ferramenta AIMS é muito versátil quanto à instrumentação, permitindo em alguns casos a instrumentação automática sem a necessidade da adição manual de código pelo usuário. Outra característica é a possibilidade da edição dos arquivos de configuração, e conseqüente alteração no comportamento da ferramenta de monitoramento, em tempo de execução.

4.4.2 ASiA

[Seção 3.4.1] [Santana, 1996]

Nome	ASiA
Sistema aplicado	software (hardware)
Abordagem para obtenção dos resultados	simulação
Forma de medição ou instrumentação	não se aplica
Tipos de dados gerados pela análise	não se aplica
Arquitetura	qualquer
Informações fornecidas	utilização, período médio ocupado, comprimento médio da fila, número de vezes que a fila foi encontrada vazia, comprimento máximo e mínimo da fila (para cada recurso)
Análise (sistema alvo)	sistemas de software

Tabela 4.19: Características do Ambiente ASiA.

A análise de desempenho através deste ambiente de simulação deve ser realizada em quatro estágios básicos:

- edição gráfica: interface com o usuário, em que se modela o programa a ser simulado; o editor gráfico é responsável pela organização do grafo em estruturas do simulador;
- geração da aplicação: utiliza os dados do editor gráfico para gerar o programa de simulação e realizar a simulação propriamente dita;
- análise dos resultados: análise dos dados fornecidos pelo estágio de simulação;
- saída gráfica: apresenta ao usuário os resultados oriundos da simulação através de ambientes gráficos ou histogramas.

Além das informações oferecidas pelo simulador, ainda existe a possibilidade do usuário adicionar instruções para coleta de outras métricas, alterando o código fonte do programa simulado.

4.4.3 IDTrace

[Seção 3.4.3] [Pierce, 1994]

Nome	IDTrace
Sistema aplicado	software
Abordagem para obtenção dos resultados	benchmarking (execução para obtenção dos traços)
Forma de medição ou instrumentação	instrumentação do código executável
Tipos de dados gerados pela análise	extração de dados de eventos
Arquitetura	SISD
Informações fornecidas	traços de eventos da execução do programa, laços, acesso à memória cache
Análise (sistema alvo)	sistema de software

Tabela 4.20: Características da ferramenta IDTrace.

Apesar de não ser uma ferramenta aplicável a sistemas paralelos, esta foi escolhida para estar no conjunto de ferramentas de avaliação que realiza "benchmarking" por ser um exemplo de ferramenta dedicada à extração de traços de eventos, gerando dados para posteriores simulações.

4.4.4 P3T+

[Seção 3.4.2] [Fahringer, 2000]

Nome	P3T+
Sistema aplicado	software
Abordagem para obtenção dos resultados	simulação
Forma de medição ou instrumentação	não se aplica
Tipos de dados gerados pela análise	não se aplica
Arquitetura	SPMD, MPMD
Informações fornecidas	distribuição de carga, informações sobre comunicação, tempo de computação, taxa de acerto de acesso à memória cache
Análise (sistema alvo)	software

Tabela 4.21: Características da Ferramenta P3T+

As técnicas de predição utilizadas na ferramenta são baseadas na modelagem de espaços de iteração de laços, modelos de acesso a vetores e modelos de distribuição de dados. As informações sobre a comunicação são oriundas da simulação do comportamento da biblioteca de comunicação e os tempos de execução são oriundos de "kernels" pré-avaliados. Para uma melhor precisão dos resultados, aspectos do sistema alvo são tratados cuidadosamente, como o tamanho e o número de linhas de memória cache disponíveis, tempos de disparo, tempo de transferência de mensagens por byte, etc.

O objetivo principal desta ferramenta é apresentar ao usuário diferentes construções de programas paralelos e distribuídos, e seus respectivos comportamentos em relação ao desempenho.

4.4.5 Pablo Performance Analysis Environment

[Seção 3.4.3] [Reed, 1994]

Nome	Pablo/SvPablo
Sistema aplicado	software
Abordagem para obtenção dos resultados	benchmarking
Forma de medição ou instrumentação	modificação do código fonte e do executável
Tipos de dados gerados pela análise	extração de traços de eventos, profiling
Arquitetura	não se aplica
Informações fornecidas	tempo de processamento, contagem de disparos de rotinas, comportamento em relação à troca de mensagens, utilização de E/S
Análise (sistema alvo)	software

Tabela 4.22: Características das Ferramentas Pablo/SvPablo.

A instrumentação do software é realizada através da inserção das bibliotecas contendo extensões que realizam a extração de traços, ciclos e troca de mensagens. A extração de traços de troca de mensagens através de MPI é realizada apenas com a referência à biblioteca proprietária no momento da execução. A extração de traços sobre o processamento e ciclos é realizada automaticamente em apenas algumas arquiteturas, pois em estações Unix, IBM RS 6000 e SPs por exemplo, a instrumentação deve ser realizada manualmente. Quando o alvo da avaliação do sistema for a utilização de dispositivos de E/S, todas as instrumentações (em qualquer arquitetura) devem ser realizadas manualmente, pois o sistema não provê instrumentação automática para esses casos.

A análise e verificação dos dados gerados pela instrumentação deve ser realizada através da utilização de comandos específicos, no caso de traços de E/S e através do componente "Pablo Analysis GUI", quando os dados estiverem dispostos no formato SDDF. O GUI é um conjunto capaz de processar arquivos SDDF composto de quatro módulos, que permitem a criação de grafos para disponibilização dos dados

de desempenho.

Apesar de permitir a criação de vários tipos de grafos para análise das medidas de desempenho, existem relatos [Browne, 1997] sobre o não funcionamento de alguns dos tipos de grafos oferecidos pelo componente.

Outra opção para análise de desempenho no conjunto Pablo é o componente chamado SvPablo. Com ele o usuário pode organizar o código fonte do programa, além de organizar os arquivos de traços criados em outras avaliações. Com esse componente, após o arquivo dos traços do programa ser aberto, o usuário pode verificar as estatísticas de cada rotina instrumentada e de cada laço analisado.

4.4.6 Paradyn

[Seção 3.4.3] [Miller, 1995]

Nome	Paradyn
Sistema aplicado	software
Abordagem para obtenção dos resultados	benchmarking
Forma de medição ou instrumentação	modificação do código executável
Tipos de dados gerados pela análise	extração de traços de eventos
Arquitetura	todas
Informações fornecidas	tempo de CPU, barreira, informações sobre quantidades de E/S, comunicação e sincronismo
Análise (sistema alvo)	sistema completo, subsistemas de E/S, sincronismo e comunicação, CPU

Tabela 4.23: Características da ferramenta Paradyn.

A ferramenta, que realiza a instrumentação através da adição de contadores e da utilização de temporizadores, tem como objetivo principal identificar os pontos do programa em que o desempenho sofre degradação. Para isso utiliza o modelo de pesquisa w^3 (de *why–where–when*), que busca localizar problemas de desempenho respondendo a três perguntas, que geram os seguintes eixos:

1. "Por que" a aplicação executa de uma forma ruim?
2. "Onde" o problema de desempenho ocorre?
3. "Quando" o problema ocorre?

Através desses três eixos, a ferramenta busca apresentar os locais exatos em que ocorre a degradação do desempenho dos programas. Após a instrumentação e a execução do programa, quando usado o Performance Consultant, a ferramenta apresenta ao usuário um histograma sobre o comportamento do sistema, as características do par ordenado selecionado pelo usuário (se houver) e uma estrutura como uma árvore contendo os possíveis motivos para os problemas de desempenho. O

comportamento da ferramenta pode ser alterado pelo usuário, através do ajuste das variáveis (limiares) utilizadas na criação das hipóteses de teste.

Para usuários avançados existe a possibilidade de especificar quais dados de desempenho o Paradyn deverá coletar. Esse processo é realizado em duas etapas: seleção do tipo de dado a ser coletado e escolha de quais partes do programa esses dados serão coletados. Os tipos de dados de desempenho que podem ser coletados incluem medidas como tempo de CPU, tempo de barreira, e informações sobre quantidades relevantes de E/S, comunicação e operações de sincronismo. Os dados do desempenho da aplicação podem ser apresentados de várias formas, incluindo gráfico de barras, histogramas e tabelas.

4.4.7 PAT - Performance Analysis Tool

[Seção 3.4.3] [Cray T-series]

Nome	PAT - Performance Analysis Tool
Sistema aplicado	software
Abordagem para obtenção dos resultados	benchmarking
Forma de medição ou instrumentação	monitoração por hardware
Tipos de dados gerados pela análise	extração de traços de eventos
Arquitetura	MIMD
Informações fornecidas	eventos da execução do programa
Análise (sistema alvo)	sistema de software

Tabela 4.24: Características da ferramenta PAT.

Esse é um exemplo de uma ferramenta proprietária que faz uso de um dispositivo específico de um computador. A ferramenta PAT utiliza um hardware específico, disponível nos computadores Cray T3E para extração dos dados para análise de desempenho, sendo, portanto aplicável apenas nesses sistemas.

Apesar deste tipo de técnica ou ferramenta (utilização de dispositivos de hardware) para análise de desempenho ser considerada inviável devido ao seu alto custo, ela se torna adequada quando o dispositivo de coleta de dados se encontra disponível.

4.4.8 Total View

[Seção 3.4.3] [Etnus, 2002]

Nome	Total View (Etnus)
Sistema aplicado	software
Abordagem para obtenção dos resultados	benchmarking
Forma de medição ou instrumentação	monitoração por software
Tipos de dados gerados pela análise	profiling, extração de traços de eventos
Arquitetura	SIMD, MIMD
Informações fornecidas	tempo de execução do programa e subrotinas, tamanho e valores de variáveis, dados sobre comunicação e endereçamento de memória
Análise (sistema alvo)	sistema de software

Tabela 4.25: Características do conjunto de ferramentas Etnus TotalView.

Totalview oferece ao usuário algumas funcionalidades durante e execução dos programas[2], dentre elas se destacam:

- possibilidade de verificação e alteração dos valores de variáveis;
- inserção de "breakpoints" normais e condicionais (acionados apenas se determinada condição for satisfeita);
- insersção de barreiras de sincronismo (para processos paralelos);
- inserção de pontos para avaliação do comportamento do programa.

Para usuários avançados, a ferramenta ainda oferece a possibilidade de editar endereçamentos de memória utilizados pelo programa.

[2]Os programas devem ter sua execução suspensa para utilização das funcionalidades

4.4.9 Vampir/Vampirtrace

[Seção 3.4.3] [Pallas]

Nome	Vampir
Sistema aplicado	software
Abordagem para obtenção dos resultados	benchmarking
Forma de medição ou instrumentação	instrumentação do código fonte
Tipos de dados gerados pela análise	extração de traços de eventos, profiling
Arquitetura	SIMD, MIMD
Informações fornecidas	porcentagens dos processos em relação ao tempo total do programa, quantidade de dados transferida, mensagens trocadas pelos processos
Análise (sistema alvo)	sistema de software que utilizam biblioteca MPI

Tabela 4.26: Características da ferramenta Vampir.

Vampir pode apresentar vários tipos de dados sobre a execução do programa alvo, cada uma de uma maneira distinta. Dentre as opções estão a animação das atividades do programa em um determinado intervalo e o comportamento das atividades de comunicação ao longo do tempo.

Apesar da biblioteca Vampirtrace instrumentar apenas aplicações escritas com MPI, a ferramenta Vampir exibe os dados dos arquivos de traços gerados por aplicações MPI, PVM e Parmacs.

No momento de gerar os arquivos de traços, Vampirtrace se infiltra na interface de "profiling" do MPI e coleta as informações necessárias, sem gerar grande sobrecarga de instrumentação.

4.5 Instruções para uso das tabelas apresentadas nesse capítulo (e no apêndice A)

Para que as tabelas possam ser utilizadas de forma correta, o usuário ou analista precisa ter conhecimento do problema a ser resolvido ou da aplicação a ser avaliada. A escolha da ferramenta ou técnica para avaliação do desempenho torna-se menos trabalhosa a medida em que a quantidade de informações que descrevem o problema aumenta.

Através das informações sobre o problema, o analista pode seguir um procedimento para seleção da técnica ou ferramenta a ser aplicada. Esse procedimento é realizado através dos seguintes passos:

- verificação do tipo de sistema, incluindo sua arquitetura;
- verificação da existência física e da disponibilidade do sistema a ser avaliado (para escolha da abordagem de avaliação);
- qual o sistema ou subsistema alvo da avaliação;
- quais informações devem ser fornecidas na avaliação.

Ao seguir esse procedimento, pode-se assegurar que a ferramenta escolhida, será no mínimo adequada, podendo ser a ótima em grande parte das ocasiões. Observe-se que a determinação de uma ferramenta ótima depende de fatores como disponibilidade de acesso à mesma.

Para ilustrar o uso dessas tabelas e, principalmente, das várias estratégias de classificação, considere 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.

Quarto, que tipo de dados são do interesse do usuário, a partir da descrição das ferramentas neste capítulo. 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.

Capítulo 5

Testes, resultados e considerações

5.1 Considerações iniciais

Neste capítulo apresenta-se testes realizados em algumas das ferramentas mencionadas no texto. Os objetivos destes testes são a compreensão da funcionalidade das ferramentas e da facilidade de sua aplicação para realizar a análise de desempenho. Outro fator analisado é a coerência entre as informações fornecidas pelos autores e as características reais das ferramentas (como sistemas aplicados, medidas fornecidas, etc.).

5.2 Ambiente utilizado para os testes

O ambiente utilizado na avaliação das ferramentas foi o "cluster" de computadores com sistema operacional Linux disponível no LAC/DEE – Laboratório Aplicado de Computação do Departamento de Engenharia Elétrica.

O "cluster", composto de seis computadores, foi escolhido com o objetivo de testar as ferramentas em um ambiente contendo computadores de diversas configurações, nas condições normais de funcionamento. As máquinas estão conectadas através da rede local do departamento, a qual é compartilhada com computadores não utilizados nos testes. As configurações dos computadores utilizados nos testes

são apresentadas na tabela 5.1.

Nome	Processador	Memória	Distribuição Linux
wolverine	P4 - 1.8 Ghz	512 MB	RedHat 7.3
ciclope	P4 - 1.8 Ghz	512 MB	RedHat 7.3
starmie	P3 - 1.0 Ghz	128 MB	Conectiva 6.0
quincas	P3 - 1.0 Ghz	128 MB	Conectiva 6.0
dartanhan	P2 - 500 Mhz	128 MB	Conectiva 6.0
aramis	P2 - 500 Mhz	128 MB	Conectiva 6.0

Tabela 5.1: Características dos computadores pertencentes ao "cluster".

5.3 Testes realizados

Aplicaram-se dois testes às ferramentas. O primeiro (programa *energy*, usado no cálculo de energia de cadeias de polímeros) foi adotado pela facilidade de acesso e pela familiaridade do autor com o mesmo. O segundo programa foi escolhido por fazer parte do Linpack (*xhpl*, voltado para aplicações de alto desempenho), sendo mais acessível por outros pesquisadores e, portanto, permitindo uma maior padronização dos testes.

A seguir apresentam-se os resultados obtidos nesses testes, com considerações sobre as ferramentas utilizadas na análise de desempenho dos mesmos.

5.3.1 HPL - High Performance Linpack

Os testes com o HPL tiveram como objetivo a avaliação de uma ferramenta destinada à medição de sistemas de hardware, que no caso fornece resultados a partir de "benchmarks".

Aspectos Funcionais

O "benchmark" requer que alguns pacotes estejam previamente instalados; são eles:

- biblioteca de comunicação MPI (versão 1.1 ou superior);
- implementações de BLAS ("Basic Linear Algebra Subprograms") ou VSIPL ("Vector Signal Image Processing Library");

O "benchmark" resolve sistemas lineares de ordem n do tipo $Ax = b$; primeiro por um processo de decomposição LU, depois por pivoteamento parcial das linhas. Desde que o fator triangular inferior L seja aplicado a b, após todo o processo de decomposição a solução é obtida pela resolução do sistema triangular superior $Ux = y$.

O modo em que a troca de dados é realizada pode ser selecionado de um conjunto de seis opções; cada uma delas representa um procedimento diferente de distribuição dos dados entre os processos.

Considerações sobre os testes

Aplicou-se dois conjuntos de testes. O primeiro tendo como base o arquivo de configuração padrão fornecido com a ferramenta e o segundo com o arquivo de configuração alterado. O arquivo de configuração, chamado HPL.dat, permite ao usuário ou analista configurar alguns itens utilizados pelo programa, como tamanho do problema, configuração dos processos e características do algoritmo.

No primeiro conjunto, o "benchmark" foi empregado na resolução de sistemas com ordem de matriz $1x4$, $4x1$ e $2x2$. Essas dimensões implicam na criação, sempre, de quatro processos (isso porque a ferramenta obriga que o número mínimo de processos seja igual ou maior ao produto do número de linhas pelo número de colunas).

Quando executado em apenas um computador, ocorreu uma queda significativa no desempenho, pois o programa paralelo, com quatro processos, foi executado em apenas uma unidade de processamento, como vistos na figura 5.1 e tabela 5.2.

Com o aumento no número de computadores cada máquina executou menos processos, o que implicou numa progressão esperada para o desempenho, até três máquinas.

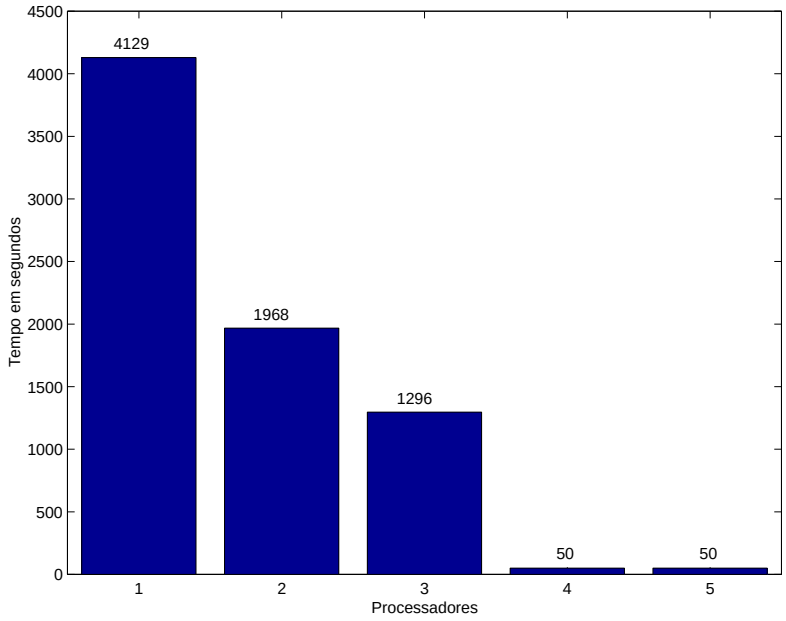

Figura 5.1: Tempos de execução utilizando o arquivo HPL.dat padrão.

Ao executar o programa em quatro unidades de processamento (configuração ideal para execução do "benchmark"), observou-se uma inesperada melhora de desempenho, com um ganho de quase 26 vezes em relação ao tempo de execução do programa para três processadores. Isso ocorre pelo melhor casamento entre o número de processos e a arquitetura do sistema, evitando um consumo de tempo excessivo na espera em barreiras de sincronismo, que ocorria quando os processos disputavam a mesma CPU.

A igualdade nas métricas de desempenho para quatro e cinco processadores ocorre porque o "benchmark" utiliza no máximo quatro processadores (como explicado anteriormente).

Número de Processadores	Speedup Relativo	Taxa máxima (em mflops)
1	–	0,4015
2	2,098	0,3612
3	3,186	0,3818
4	82,58	1,310
5	82,58	1,310

Tabela 5.2: Medidas obtidas na execução do programa Xhpl.

No segundo conjunto de testes, o tamanho do problema (números de linhas e colunas) foi adequado para cada situação. Assim, para cinco processadores foram utilizados sistemas de ordem $1x5$ e $5x1$. Com essas características, o tamanho do problema também aumentava com o número de unidades de processamento, não gerando diferenças significativas entre os resultados (todos os experimentos obtiveram valores máximos de desempenho em torno de $1,2 \times 10^{-3}$ mflops a $1,4 \times 10^{-3}$ mflops.

A partir dos testes aplicados constata-se que a ferramenta apresenta resultados compatíveis aos previstos por seus autores. Apesar de apresentar poucas medidas de desempenho, mostrou-se uma ferramenta de fácil manuseio e aplicação, coerente com o que é esperado de uma ferramenta para medição de desempenho de computadores.

5.3.2 Paradyn

Os testes com o Paradyn tiveram como objetivo principal a verificação de funcionalidade de uma ferramenta para análise de software. O Paradyn, como já descrito nos capítulos três e quatro, faz o "benchmarking" de programas fornecendo medidas de desempenho (essencialmente gargalos de execução) através da instrumentação dinâmica do código durante sua execução, sendo portanto um interessante caso de estudo.

Aspectos funcionais

Como citado anteriormente, no momento da execução do programa, a ferramenta utiliza o modelo de pesquisa w^3, descrito a seguir.

O eixo "por que"

O eixo "por que" representa o grande conjunto de problemas que podem tornar vagarosa a execução do programa em teste. Problemas de desempenho em potencial são representados por hipóteses e testes. Hipóteses representam os tipos fundamentais de problemas de desempenho que ocorrem em programas paralelos, independente do programa estudado e do algoritmo utilizado. As hipóteses podem ser refinadas através de outras hipóteses 5.2. As dependências entre as hipóteses definem um grafo dirigido, e o ponto de degradação do desempenho pode ser encontrado através do caminho de busca deste grafo.

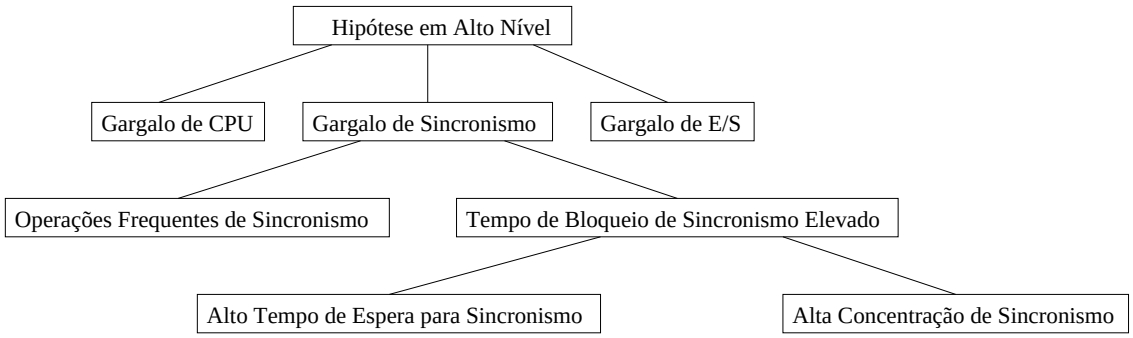

Figura 5.2: Exemplo do eixo "por que" com algumas hipóteses.

Os testes são funções booleanas que avaliam a validade das hipóteses, e são expressos por limiares e medidas calculadas pelo módulo de instrumentação. A figura 5.2 apresenta, como já indicado, um grafo de busca do eixo "por que". Nele, o local de degradação do desempenho encontrado por sua análise está situado em *tempo elevado na barreira de sincronismo*.

O eixo "onde"

No eixo "onde" é definido o problema específico do programa paralelo, ou seja, as partes do programa em que o problema de desempenho pode residir.

Enquanto que a busca através do eixo "por que" procura classificar o tipo de problema de desempenho, a busca através do eixo "onde" tem por objetivo especificar o local exato (rotina ou função) em que a degradação de desempenho ocorre.

Um exemplo é apresentado na figura 5.3, onde a ferramenta tenta detalhar o problema, aumentando a precisão da busca, agora procurando em algum semáforo, módulo de troca de mensagens ou barreiras. A ferramenta também mostra em qual CPU está ocorrendo a degradação.

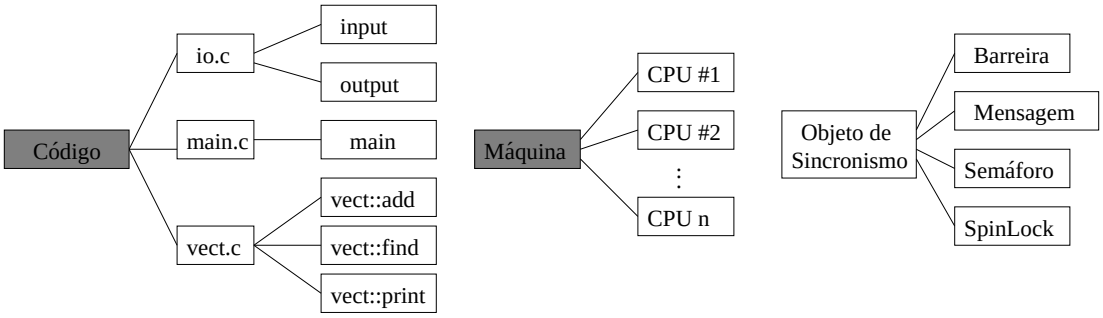

Figura 5.3: Exemplo do eixo "onde" com três hierarquias de classes.

O eixo "quando"

Os programas possuem diferentes fases de execução e o eixo "quando" apresenta os períodos de execução dessas diferentes fases. Por exemplo, um programa simples pode ter três fases de execução: inicialização, processamento e saída. Quando um programa muda de fase podem ocorrer mudanças drásticas no seu comportamento. Este eixo envolve testes de hipóteses na busca por problemas nos diferentes intervalos de execução do programa.

Considerações sobre os testes

Para que a avaliação de desempenho possa ser realizada no momento da execução do programa, é necessário que o "daemon" do Paradyn esteja instalado em todas as máquinas nas quais o programa alvo será executado (no caso de ambiente com memória distribuída) ou em um lugar comum a todos os nós de processamento (em ambiente com memória compartilhada).

Nos testes, a ferramenta foi aplicada na análise de programas seriais e programas que utilizavam as bibliotecas PVM, MPI e pthreads (energy, xhpl e gauss-jacobi, respectivamente).

Apesar da informação de que a ferramenta avalia o desempenho de programas que utilizam "threads" ter sido veiculada em alguns textos, após contato com um dos pesquisadores envolvidos com seu desenvolvimento foi possível averiguar que esta função ainda não está disponível para todas plataformas, estando disponível apenas para computadores Sparc com sistema operacional Solaris. Outro aspecto a ser considerado é a remoção do item *pvmd* no menu de definição do programa (a opção fazia parte do menu de opções em versões anteriores).

Figura 5.4: Janela para definição do programa que será avaliado.

Várias tentativas de execução de programas escritos em PVM no ambiente Linux foram realizadas selecionando outra opção de "daemon", mas nenhuma delas foi bem sucedida. De acordo com os criadores da ferramenta, é necessário alterar o arquivo de configuração para que programas em PVM possam ser executados normalmente (a alteração não é uma tarefa simples, pois envolve várias definições da biblioteca de comunicação e do sistema operacional).

Quando aplicada a um programa baseado em MPI (o programa *xhpl*), a ferramenta apresentou-se como uma boa alternativa para avaliação de desempenho, pois permitiu a seleção de várias métricas e formas de apresentação, além de não oferecer nenhum trabalho extra ao usuário no momento da avaliação.

Foi necessário apenas selecionar o programa alvo (figura 5.4), selecionar

o "daemon" e as métricas que seriam exibidas. A seleção de métricas e tipo de visualização puderam ser alteradas no decorrer da execução da análise, através do menu de opções.

Na opção de visualização por histograma, ainda foi possível selecionar o trecho em que se desejava avaliar através da opção de zoom contida no ambiente.

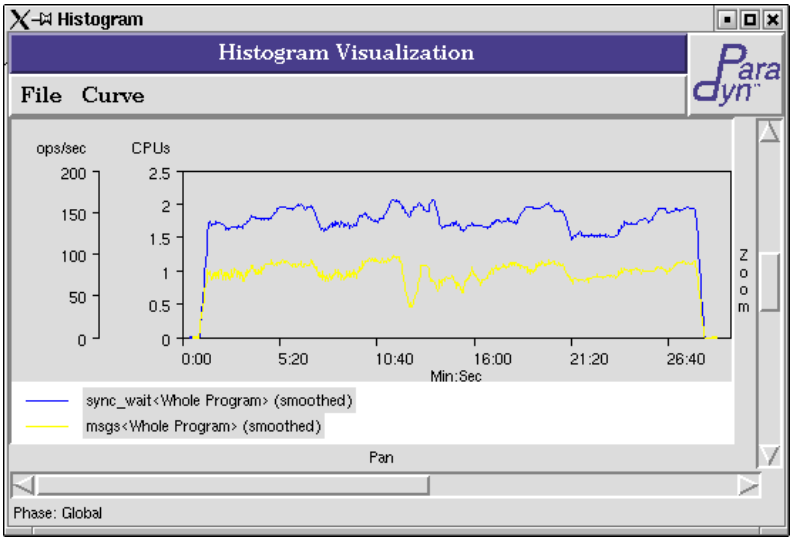

Figura 5.5: Exemplo de gráfico apresentado pela ferramenta.

Em algumas execuções a ferramenta apresentou problemas em conciliar os dispositivos de visualização gráfica e consulta de desempenho ("Performance Consultant"). Nesses casos, os "daemons" eram terminados quando as duas janelas eram mostradas na tela.

Paradyn possui uma interface de fácil utilização. O manual do usuário explica claramente as várias opções disponíveis para avaliação de desempenho. Uma inconveniência é o tamanho do gráfico exibido no "Performance Consultant" (ver figura 5.6), que na maioria das vezes era exageradamente grande, não podendo ser visualizado sem a necessidade de alteração nas configurações das janelas do dispositivo.

Além da facilidade de uso e entendimento, a equipe de desenvolvimento do Paradyn mantém um serviço de suporte que, sempre que consultado, prontamente

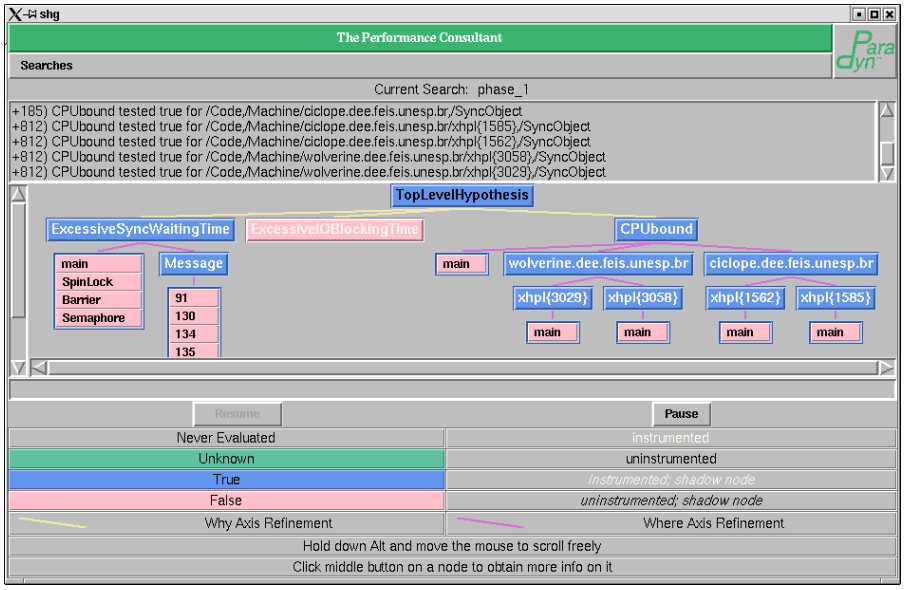

Figura 5.6: Módulo de análise de problemas - Performance Consultant.

respondeu às dúvidas e questões sobre o mesmo. Essa característica dificilmente apareceu com as demais ferramentas examinadas.

5.3.3 Vampir/Vampirtrace

Outra ferramenta examinada na categoria de ferramentas baseadas em "benchmarking" para sistemas de software é o Vampir. Os testes realizados com essa ferramenta tiveram a mesma finalidade daqueles realizados com o Paradyn.

Aspectos Funcionais

Vampir é uma ferramenta que permite ao usuário observar arquivos de "tracing" com o comportamento da execução e comunicação de programas escritos em MPI. Isso é possível com a aplicação da ferramenta Vampirtrace para extração dos dados do programa alvo.

A ferramenta Vampirtrace é utilizada na extração de informações do programa e na criação do arquivo de traços. Para que os dados sejam gerados, é necessário

que o programa alvo, após instrumentado, seja compilado referenciando a biblioteca de "tracing".

A extração de dados pode ser interrompida e reiniciada em qualquer ponto da execução, através de alteração no código fonte. A ferramenta não analise linhas de comando específicas do código fonte. Todavia, se o usuário desejar avaliar algum ponto específico do programa, pode fazê-lo através da inserção de chamadas às rotinas do Vampirtrace.

Como mencionado anteriormente, a quantidade de dados armazenados é muito expressiva. Na tentativa de diminuir esse problema, os criadores da ferramenta produziram um filtro que limita os dados armazenados. Tal filtro é configurado pelo usuário antes da execução do programa, e é aplicado logo no início do processo de "tracing".

Considerações sobre os testes

Vampir e Vampirtrace são fáceis de instalar e são compatíveis com um grande número de arquiteturas. Para realização dos testes foi preciso apenas descompactar os arquivos fornecidos pelo fabricante e configurar variáveis de ambiente utilizadas no funcionamento das ferramentas, inclusive variáveis referentes às bibliotecas de comunicação.

Os testes aqui apresentados foram aplicados a um programa fornecido junto com a ferramenta, chamado vtJacobic, que é aplicado na solução de problemas através do método de Jacobi (decomposição). Este programa é um programa já instrumentado para possibilitar o acesso às rotinas do Vampirtrace. O programa testado não sofreu nenhuma modificação no código, ou seja, foi executado da forma como foi distribuído pelo fabricante.

Ao contrário do Paradyn, não foi necessário instalar a ferramenta em um lugar comum para todos os computadores utilizados nos testes. Nos gráficos apresentados a seguir mostra-se a execução do programa exemplo utilizando dois e cinco processadores.

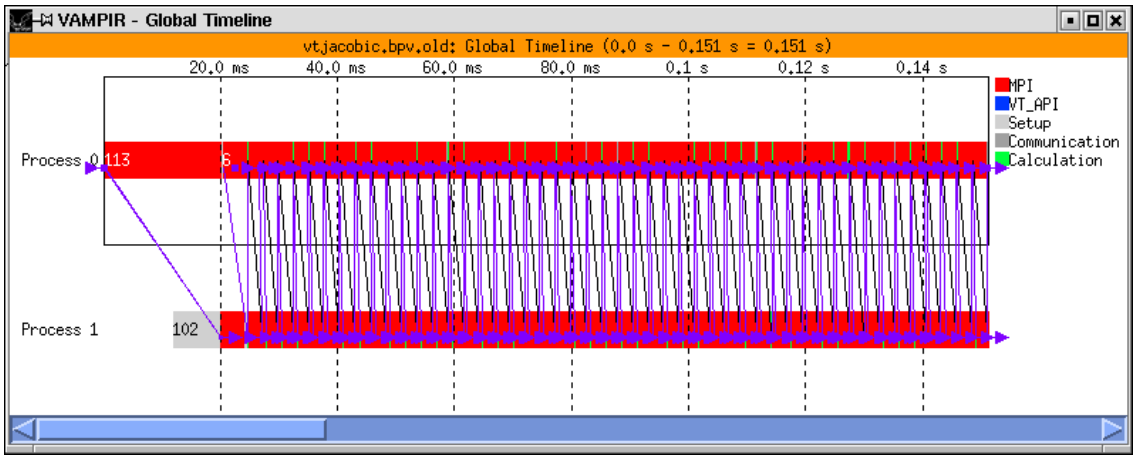

Figura 5.7: Parte do aspecto global da execução do programa.

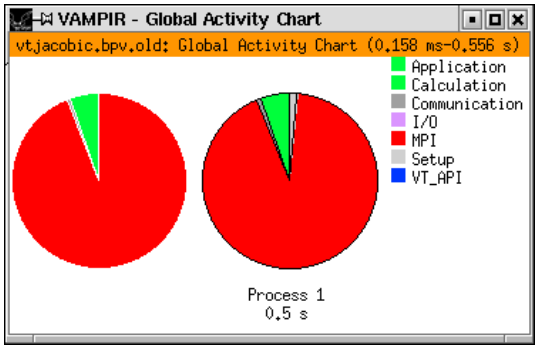

Figura 5.8: Gráfico que demonstra a execução das rotinas em cada processo.

A ferramenta Vampir possui uma interface amigável com o usuário, e todas as funções básicas encontram-se na barra principal de menus. Como no Paradyn, é possível selecionar um determinado intervalo na barra de tempo do gráfico de execução para análise mais apurada do comportamento do programa (como apresentado na figura 5.7).

O usuário pode selecionar outros meios de exibição dos dados sobre desempenho global do programa, como o gráfico de torta apresentado na figura 5.8, ou ainda selecionar qual processo deseja avaliar de forma mais precisa, como mostra a figura 5.9.

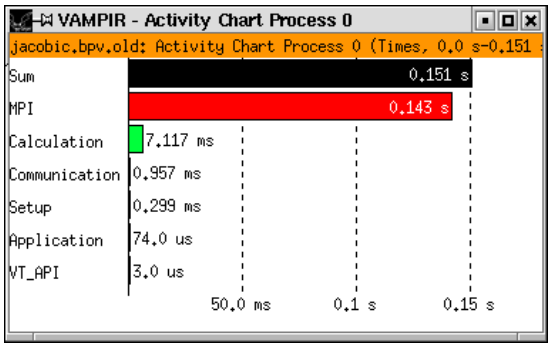

Figura 5.9: Dados sobre um processo específico.

Um aspecto a ser considerado antes do início dos testes com a ferramenta foi a demora na resposta de um pedido para envio de uma nova licença, devido a problemas ocorridos com o arquivo original. A solução foi a solicitação de um novo conjunto (programa + licença) através da "home page" do fabricante. A resposta para a solicitação inicial chegou após quatro meses do envio da mensagem.

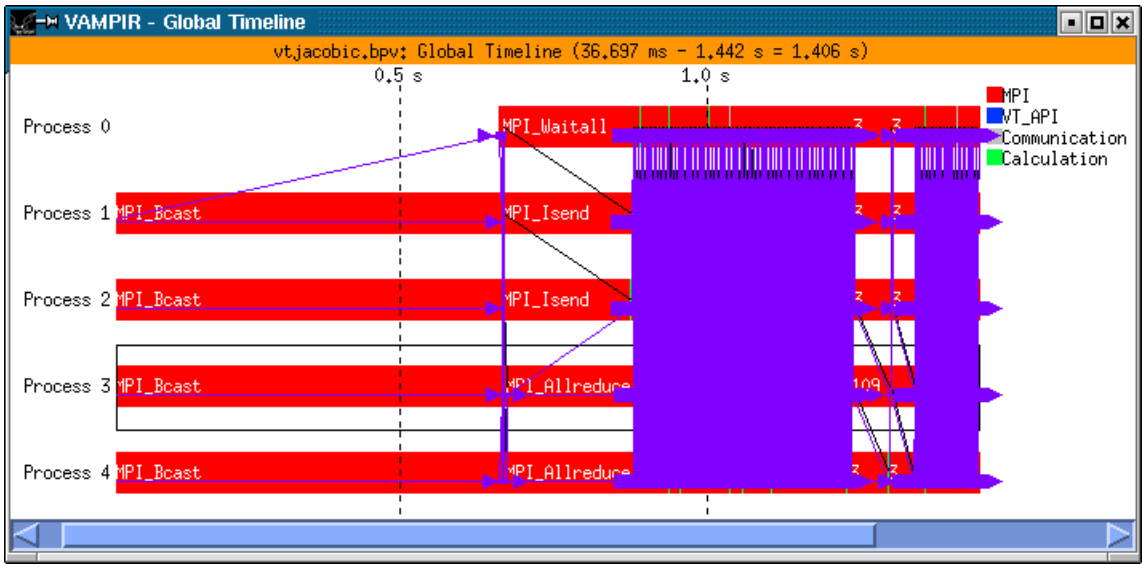

Figura 5.10: Dados sobre a execução do programa em 5 processadores.

A visibilidade de alguns gráficos (especialmente os de aspectos globais) fica comprometida à medida que o número de processadores e da atividade instrumentada (troca de mensagens, por exemplo) cresce. Um exemplo desse problema é apresentado

na figura 5.10. Tal problema pode ser contornado através de ferramentas de "zoom" e seleção de processos.

5.4 Comparação entre as ferramentas descritas

5.4.1 Ferramentas aplicadas a sistemas de hardware

Quando o alvo da análise de desempenho é o sistema completo, e o sistema se encontra disponível, a utilização do conjunto Linpack e suas variações mostra-se como uma das melhores alternativas para avaliação de desempenho. Dentre todas as ferramentas analisadas, o Linpack (ou variações) pode ser considerado a ferramenta de maior facilidade de utilização para o usuário, possuindo, entre outras funcionalidades, versões portáveis para várias plataformas e sistemas operacionais, incluindo uma versão multiplataforma escrita em Java. Uma característica que pode ser considerada negativa, mas não tirando o mérito do conjunto, é o fato dele apresentar poucas medidas de desempenho.

Caso o usuário deseje analisar também subsistemas, o pacote SPEC é então indicado. Ao contrário do Linpack, que possui apenas subrotinas que permitem analisar o sistema completo, o pacote SPEC possui um grande conjunto de ferramentas e programas para análise do sistema completo e de subsistemas. Cada pacote fornecido pela organização é composto de vários programas, como é o caso do pacote SPEC CPU2000, com mais de 20 aplicações para análise do sistema completo.

Em relação à disponibilidade de código para "download" e assistência oferecidas pelas ferramentas, ambas possuem descrições e procedimentos detalhados de instalação e configuração de suas ferramentas. Um fator negativo de ambas as ferramentas é a demora na resposta ao usuário quando mensagens são enviadas para as pessoas responsáveis pelo atendimento à dúvidas.

Outras ferramentas são utilizadas também para análise de subsistemas, como é o caso do "benchmark" FTIO, pertencente ao conjunto NAS, utilizado para análise de sistemas de entrada e saída. Como o pacote é mantido pela NASA, versões para avaliação são apenas disponíveis para determinados setores da indústria e da

comunidade científica. Os pedidos para fornecimento da ferramenta (juntamente com todo o conjunto NAS) para testes nesse trabalho foram negados.

O usuário ou analista deve ter cuidado especial no momento da escolha da ferramenta para avaliação de desempenho pois, além das características descritas no texto, ainda existe a possibilidade de existirem sistemas orientados a "benchmarks", que são sistemas criados especificamente com o objetivo de obter altos índices de desempenho em determinadas ferramentas que fazem "benchmarking" dos equipamentos.

Se o sistema a ser avaliado não estiver disponível, ferramentas baseadas em simulação são indicadas. Dentre todas as ferramentas avaliadas, PDL e PAWS mostraram-se projetos que dariam certo se tivessem continuidade pois, apesar dos autores prometerem novas versões, não existem relatos de atualizações das ferramentas. O projeto do ambiente PAWS foi aproveitado e está sendo aplicado no desenvolvimento do ambiente P3T, que atualmente funciona como uma ferramenta que interpreta os dados de outros aplicativos e apresenta ao usuário.

As ferramentas Simics e RSim mostraram-se aplicáveis tanto na avaliação de desempenho do sistema completo quanto de subsistemas. Algumas diferenças entre as duas ferramentas: o fato do RSim permitir apenas simulação de multiprocessadores com memória compartilhada, enquanto que o Simics, de acordo com seus autores, permite simular desde computadores seriais até redes de computadores interligados e o fato do Simics ser uma ferramenta comercial (criada no meio acadêmico, é comercializada atualmente pela empresa Virtutech), enquanto que o RSim ainda é uma ferramenta distribuída como "open source".

Não foi possível avaliar a ferramenta Simics pois até a data em que este texto foi escrito ainda não havia resposta sobre como licenciar a ferramenta.

5.4.2 Ferramentas aplicadas a sistemas de software

Uma grande diversidade de ferramentas encontra-se disponível para auxiliar o usuário ou analista quando o objetivo é avaliar sistemas de software (programas).

Dentre as ferramentas que buscam avaliar o desempenho de sistemas de software através de "benchmarking", a grande maioria avalia o desempenho dos programas após sua execução, como AIMS, Pablo, Total View e Vampir. Diferente das outras ferramentas, o Paradyn avalia o desempenho dos programas também durante a execução.

Uma característica comum a essas ferramentas é o fato de todas serem aplicáveis na avaliação de desempenho de programas que utilizam bibliotecas de comunicação PVM e MPI.

Apesar dessas ferramentas serem aplicadas ao mesmo conjunto de programas e utilizarem o mesmo tipo de abordagem, fornecem ao usuário ou analista diferentes métricas e informações sobre o desempenho dos programas.

Pablo permite ao usuário acompanhar as características do programa diretamente no código fonte da aplicação. Nos exemplos apresentados com a ferramenta, apenas dois conjuntos de medidas são disponibilizadas ao usuário: as medidas de contagem de acessos e da duração.

AIMS fornece um número maior de medidas de desempenho que o Pablo, e ainda permite ao usuário acompanhar o processo de execução do programa. Apresenta também o comportamento dos nós de processamento e de trocas de mensagens entre os processadores. Sendo assim, pode ser considerada uma ferramenta mais completa que Pablo.

A ferramenta Vampir, além de mostrar a troca de mensagens entre os processadores, apresenta também a quantidade de dados (em bytes) trocada pelos processadores e o quanto cada rotina ligada à biblioteca de comunicação ("send", "receive", "barrier", p.ex.) consome do tempo total de execução.

Como caracterizado pela empresa que o criou, o Etnus Total View é um "debugger". Com ele o usuário pode verificar o desempenho do programa, valores de variáveis e informações sobre a troca de mensagens entre os processos. O usuário acompanha o processo através da exibição do código em linguagem "assembly", caso não possua o código fonte, ou no próprio código caso esteja disponível e for compilado corretamente.

Deve-se salientar que, para que todas as ferramentas até então citadas possam funcionar corretamente, existe a necessidade de alguma alteração no código fonte ou no processo de compilação, o que não acontece com o Paradyn.

O Paradyn, ao contrário das outras, não necessita de pré-edição do código ou ação sobre ele, pois realiza a instrumentação automaticamente. Dentre todas as ferramentas publicamente disponíveis, o Paradyn é a que disponibiliza maior quantidade de métricas ao usuário, além de permitir que o usuário acompanhe o comportamento do programa durante a execução.

Capítulo 6

Conclusões e trabalhos futuros

Neste trabalho apresentou-se uma classificação de ferramentas e técnicas disponíveis para análise de desempenho de sistemas paralelos e distribuídos, segundo diversas estratégias de classificação. Um conjunto representativo de estratégias foi selecionado com o objetivo de abranger, se não todas, grande parte das técnicas e ferramentas existentes nos meios acadêmico e comercial. Aquelas que aqui não foram citadas podem ser facilmente enquadradas através de analogia com as já consideradas. Aqui também se apresentou diversas ferramentas e técnicas para a análise e predição do desempenho de sistemas paralelos, comparando-as segundo sua aplicabilidade para um melhor casamento com o problema a ser examinado.

A comparação mencionada no parágrafo anterior deve ser entendida apenas como uma amostra do que pode ser encontrado no mercado, não podendo ser vista como um resultado conclusivo. Isso porque não existe um padrão universal para desempenho ou medida de desempenho, não existem ferramentas e técnicas boas ou ruins, mas sim ferramentas e técnicas que melhor se adequam a determinado problema.

6.1 Conclusões

A classificação apresentada neste trabalho busca auxiliar usuários e desenvolvedores de sistemas paralelos no momento da escolha da ferramenta para análise

de desempenho. A simplicidade na descrição das estratégias, técnicas e ferramentas permite que esse trabalho seja utilizado como referência não só por pessoas ligadas à área de análise de desempenho, mas principalmente por pessoas de outras áreas, como a engenharia elétrica, em que o desempenho das aplicações (tempo de execução, por exemplo) é de suma importância, justificando a importância para a engenharia.

Em relação à área de análise de desempenho, este trabalho funciona como uma revisão bibliográfica de métricas aplicadas para avaliação, além do objetivo principal, que é o de classificar técnicas e ferramentas.

Com relação aos objetivos propostos nesse trabalho, pode-se considerar que foram atingidos, uma vez que apresentou-se um mecanismo para a classificação de técnicas e ferramentas para análise de desempenho, baseada em diferentes abordagens de classificação, que é abrangente o bastante para ser de fácil uso e aplicação. Além disso, realizaram-se diversos testes que permitiram a geração de informações adicionais para a comparação de algumas das ferramentas disponíveis no mercado.

6.2 Perspectivas para futuros trabalhos

Sendo concluída essa fase do trabalho, pode-se vislumbrar vários caminhos para sua continuidade. Dentre os quais destacam-se:

- a criação de um repositório de técnicas e ferramentas para análise de desempenho, que uma vez disponibilizado permitirá que usuários possam ter acesso às informações contidas nesse trabalho, e assim utilizá-las como um referencial no momento da escolha;
- a manutenção da busca e reposição de novas ferramentas, pois a necessidade de atualização do banco de dados, em consequência do constante surgimento de técnicas e ferramentas para análise de desempenho, é evidente, evitando assim que o trabalho aqui realizado não se torne obsoleto com o aparecimento de novas aplicações;
- a identificação de uma metodologia realmente unificadora para classificação de técnicas e ferramentas para análise de desempenho, uma vez que a atual

situação, com várias propostas parcialmente aceitas (ou usadas) acaba por criar uma certa confusão no momento de escolha da ferramenta adequada para cada situação;

- aumento na base de comparações entre ferramentas, propiciando uma base neutra para comparação entre as mesmas, que forneça informações coerentes e não viciadas sobre facilidade de uso, precisão e variedade de resultados oferecidos, por exemplo.

REFERÊNCIAS BIBLIOGRÁFICAS

[Abandah, 1998] Abandah, G.A. and Davidson, E.S. (1998). Characterizing distributed shared memory performance: a case study of the Convex SPP1000. Em *IEEE Trans. on Parallel and Distributed Systems*, volume 9, number 2, pages 206–216.

[Amdahl, 1967] Amdahl, G. M. (1967). Validity of single processor approach to achieving large scale computing capability. Em *AFIPS Conference Procedings*, volume 30, AFIPS Press, Reston – Va, pages 483–485.

[Aquilani, 2000] Aquilani, F.; Balsamo, S. and Inverardi, P. (2000). An Approach to Performance Evaluation of Software Architectures. *Research Report CS-2000-3*, Dipartimento di Informatica Università Ca' Foscari di Venezia, March.

[Austin, 2002] Austin, T.; L. E. and Ernst, D. (2002). Simplescalar: An infrastructure for computer system modeling. Em *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. Em *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. *http://www.cs.utk.edu/ browne/perftools-review/*

[Bruschi, 2002] Bruschi, S. (2002) ASDA: Um Ambiente de Simulação Distribuída Automático. *Tese de Doutorado, ICMSC-USP*, São Carlos, SP – Brasil.

[Buck, 2000] Buck, B. and Hollingsworth, J. K. (2000). An API for Runtime Code Patching. Em *Journal of High Performance Computing Applications*, volume 14:4, pages 317–329.

[Cain, 2000] Cain, H. W., et alii (2000). A Callgraph-Based Search Strategy for Automated Performance Diagnosis. Em *Euro-Par 2000*, Munich, Germany, August 2000.

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

[Chang, 2000] Chang, R. K. C. and Lam, S. (2000). A novel approach to queue stability analysis of polling models Em *Performance Evaluation*, volume 40, pages 27–46.

[Chen, 1994] Chen, P. M. and Patterson, D. A.(1994). A new approach to I/O performance evaluation - self-scaling I/O bechmarks, predicted I/O performance. Em *ACM Transactions on Computer Systems*, volume 12, n. 4, pages 309–339.

[Cray T-series] Cray T-series. Cray T-series Man Pages. Cray Inc.

[Cracken, 1998] Cezar, F.A.M.C. (1998). Paralelização de um código para Geração de Estatísticas Conformacionais de Cadeias Polissacarídicas. *Trabalho de conclusão de curso*, IBILCE / UNESP, S.J. do Rio Preto, SP – Brasil.

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

[Dongarra, 1988] Dongarra, J. J. (1988). Performance of various computers using standard linear equations software. *Thechnical Report 23*, Argonne Nat. Lab.

[Dongarra, 1995] Dongarra, J. J. and Walker, D. W. (1995). Software libraries for linear algebra computations on high performance computers. *SIAM Review 37*, pages 151–180.

[Dongarra, 1999] Dongarra, J. J. et alii (1999). LAPACK Users' Guide Third Edition. *http://www.netlib.org/lapack/*

[Duncan, 1990] Duncan, R. (1990). A Survey of Parallel Computer Architectures. Em *IEEE Computer*, pages 5–15.

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

[Etnus, 2002] Etnus (2002). Totalview 4.1 multiprocess debugger. *http://www.etnus.com*, Natick, MA, EUA.

[Fagerstrom, 2000] Fagerstrom, F. C. and Kuszmaul, C. L. (2000) The FTIO Benchmark. *NAS Technical Report NAS-00-004*, NASA Ames Research Center, Moffett Field, CA, EUA.

[Fahringer, 1995] Fahringer, T. (1995). Estimating and optimising performance from parallel programs. Em *Computer*, special issue n. 28, pages 47–56.

[Fahringer, 2000] Fahringer, T. et alii (2000). Evaluation of P3T+: A Performance Estimator for Distributed and Parallel Applications. Em *Proceedings of 14th International Parallel and Distributed Processing Symposium (IPDPS'00)*, Cancun – México, pages 229–234.

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

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

[Graham, 1982] Graham, S. L.; Kessler, P. B. and McKusick, M. K. (1982) Gprof: a call graph execution profiler. Em *ACM Sigplan Notices*, volume 17, n. 6, pages 120–126.

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

[Gustafson, 1988] Gustafson, J. (1988). Reevaluating amdah's law. Em *Communications of the ACM*, volume 31, n. 5, pages 532–533.

[Heath, 1995] Heath, M.T.; Maloney, A.D. and Rover D.T. (1995). The visual display of parallel performance data. Em *Computer*, special issue n. 28, pages 21–28.

[Hondroudakis, 1995] Hondroudakis, A., et alii (1995). Performance evaluation and visualization with vispat. Em *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. Em *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. Em *Computer*, volume 35, n. 2, pages 40–49.

[Hwang, 1993] Hwang, K. (1993). *Advanced Computer Architecture: Parallelism, Scalability, Programmability.* Mc Graw Hill, 1st edition.

[Hwang, 1996] Hwang, K.; Xu, Z. and Arakawa, M. (1996). Benchmark evaluation of the IBM SP2 for parallel signal processing Em *IEEE Trans. on Parallel and Distributed Systems*, volume 7, n. 5, pages 522–535.

[Hwang, 1999] Hwang, K., et alii (1999). Resource scaling effects on mpp performance: the stap benchmark implications. Em *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, 1999] Kim. J. and Lilja, D.J. (1999). Performance-based path determination for interprocessor communication in distributed computing systems. Em *IEEE Trans. on Parallel and Distributed Systems*, volume 10, n. 3, pages 316–327.

[Kim, 1996] Kim, J. and Shin, K. (1996). Execution time analysis of communicating tasks in distributed systems. Em *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. Em *Information and Software Technology*, volume 36, n. 7, pages 457–464.

[Lui, 1998] Lui, J. C. S.; Muntz, R. and Towsley, D. (1998). Computing performance bounds of fork-join parallel programs under a multiprocessing environment. Em *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. Em *IEEE Computer*, volume 35, n. 2, pages 50–58.

[Manacero Jr., 1997] Manacero Jr., A. (1997). Predição do desempenho de programas paralelos por simulação do grafo de execução. *Tese de Doutorado, UNICAMP*, Campinas, SP – Brasil.

[Marcari Jr., 1999] Marcari Jr., E. (1999). Paralelização de um código para Geração de Estatísticas Conformacionais de Cadeias Polissacarídicas – Fase II. *Trabalho de conclusão de curso*, IBILCE / UNESP, S.J. do Rio Preto, SP – Brasil.

[Miller, 1992] Miller, B. and Holligsworth, J. (1992). Parallel Program Performance Metrics: a Comparation and Validation. Em *Supercomputing*, Minneapolis.

[Miller, 1995] Miller, B. et alii (1995). The Paradyn Parallel Performance Peasurement Tools. Em *IEEE Computer*, volume 28, n. 11, pages 37–46.

[Netlib, 2002] Netlib (2002). Lapack – linear algebra package and the scalapack project. *http://www.netlib.org*.

[Netlib, 2002B] Netlib (2002). PARKBENCH (PARallel Kernels and BENCHmarks). *http://www.netlib.org/parkbench/*.

[Pallas] Pallas GmbH - High Performance Products. Vampirtrace User's and Installation-Guide. *http://www.pallas.de*.

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

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

[Reed, 1994] Reed, D. (1994). Experimental analysis of parallel systems: Techniques and open problems. Em *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. Em *IEEE Parallel and Distributed Technology*, volume 4, n. 1, pages 43–56.

[Santana, 1994] Santana, R.H.C. et alii (1994). Técnicas para Avaliação de Desempenho de Sistemas Computacionais. *Notas Didáticas do ICMC - USP*, número 14, 31 pgs.

[Santana, 1996] Santana, R.H.C. et alii (1996). Graphical User–interface for an Automatic Simulation System. *Proceedings of the 1996 Summer Computer Simulation Conference* (SCSC'96), Oregon – USA, p.139–143.

[Statsko, 1995] Stasko, J. T. (1995). The PARADE Environment for Visualizing Parallel Program Executions: A Progress Report. *Technical Report GIT-GVU-95-03 – January/1995*, Georgia Institute of Technology.

[Tatsumi, 2002] Tatsumi, E.S. (2002). Análise e avaliação do uso da ferramenta ParSMPL em um ambiente de simulação distribuída automático – ASDA. *Monografia de Qualificação (Mestrado em Ciências da Computação e Matemática Computacional)*, São Carlos, 2002.

[TOP500, 2002] TOP 500 Supercomputer sites (2002). TOP500 List for November 2002. *http://www.top500.org/list/2002/11/*.

[TPC, 2002] TPC - Transaction Processing Performance Council (2002). *http://www.tpc.org/*.

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

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

[Vemuri, 1998] Vemuri, R. (1998). PDL Reference Manual and Applications *http://www.ececs.uc.edu/ ddel/projects/pdl/pdl.html*

[Waheed, 1998] Waheed, A. and Yan, J. (1998) Parallelization of NAS Benchmarks for Shared Memory Multiprocessors *http://www.nas.nasa.gov/Research/Reports/Techreports/1998/nas-98-010-abstract.html*.

[Woo, 1995] Woo, S. et alii (1995) The SPLASH2 Programs: Characterization and Methodological Considerations. *Proceedings of the Twenty-first International Symposium on Computer Architecture*, June.

[Yan, 1995] Yan, J. C.; Sarukkai S. R. and Mehra, P. (1995). Performance measurement, visualization and modeling of parallel and distributed programs using the AIMS toolkit. Em *Software - Practice and Experience*, volume 25, n. 4, pages 429–461.

[Yan, 1996] Yan J. C. and Sarukkai, S. R. (1996). Analyzing Parallel Program Performance Using Normalized Performance Indices and Trace Transformation Techniques. Em *Parallel Computing*, volume 22, n. 9, pages 1215–1237.

[Zuberek, 1989] Zuberek, W. M. (1989). Performance evaluation using unbounded timed Petri nets Em *Proc. of the third intl. Workshop on Petri nets and Performance models*, Kyoto, Japan, pages 180–186.

Apêndice A

Tabelas para referência rápida

Neste apêndice apresentam-se tabelas contendo informações sobre as técnicas e ferramentas descritas e examinadas nos capítulos três e quatro. O objetivo deste apêndice é funcionar como um guia de referência rápida para o usuário no momento da escolha de qual técnica ou ferramenta se adequa ao seu problema.

(TABELA NA PRÓXIMA PÁGINA)

Nome da Técnica	H/S	Abordagem	Instrum./Medição	Tipos de dados	Dados
Análise de desempenho de programa aplicado em física computacional	S	benchmarking	modificação do cod. fonte	contagem de eventos, medição de intervalo de tempos, extração de traços de eventos	tempo do programa e subrotinas, número de acessos a rotinas, dados sobre a troca de mensagens
Análise de subsistema de Entrada/Saída	H	simulação, benchmarking	monitoração por software	extração de traços de eventos	uso de CPU, pedidos, tempos médios de operação, espera e resposta de E/S
Análise do desempenho da memória em sistemas DSM	H	benchmarking	não se aplica	medição de intervalo de tempos	tempo de acesso à memória, sobrecarga no sincronismo, largura de banda
Análise do tempo de comunicação e determinação ...	H	benchmarking	não se aplica	medição de intervalo de tempos	speedup, atraso no tempo de comunicação
Caracterização do desempenho de um computador paralelo...	H	analítico	não se aplica	não se aplica	medidas da escalabilidade, tempo de acesso à memória, "picos" de desempenho
Conversão do código executável em grafo de execução	S	simulação	não se aplica	não se aplica	tempos médios de processamento e espera, velocidade média, speedup
Estudo de casos de "prefetching" de instruções em "cache"	H	simulação, benchmarking	monitoração por hardware	medição de intervalo de tempos, extração de traços de eventos	eficiência, razão de erro no prefetch, tráfego em memória

Tabela A.1: Características das técnicas descritas no texto (parte 1).

Nome da Técnica	H/S	Abordagem	Instrum./Medição	Tipos de dados	Dados
GSPN	S	analítico, simulação	não se aplica	não se aplica	tempo gasto por ciclo, tempo de espera entre os eventos
Modelagem de limitantes de desempenho	S	simulação, analítico	não se aplica	não se aplica	tempos de resposta, limitantes superior e inferior
Paralelização do benchmark STAP	H	benchmarking, analítico	não se aplica	medição de intervalo de tempos	speedup, eficiência, tempo de execução, memória utilizada
Teoria de filas aplicada no cálculo de limitantes	H	analítico	não se aplica	não se aplica	tempo de resposta dos processos e tarefas, tempo de atraso devido a sincronismo

Tabela A.2: Características das técnicas descritas no texto (parte 2).

Nome da Ferramenta	H/S	Abordagem	Instrum./Medição	Tipos de dados	Dados
AIMS	S	benchmarking	instrumentação do código fonte, monitoração por software	profiling, medição de intervalo de tempos, extração de traços de eventos	tempo de execução do programa e subrotinas, eventos de E/S e troca de mensagens
ASiA	S(H)	simulação	não se aplica	não se aplica	utilização, período médio ocupado, informações sobre filas (para cada recurso)
FTIO	H	benchmarking	monitoração por software	medição de intervalo de tempos, contagem de eventos	fração total de execução gasto em E/S, ciclos gastos por operação de ponto flutuante
IDTrace	S	benchmarking	instrumentação do código executável	extração de dados de eventos	traços de eventos da execução do programa, laços, acesso à memória cache
Linpack e variações	H	benchmarking	não se aplica	contagem de eventos, medição de intervalo de tempos	número de operações (em GFlops) máximo, tamanho da matriz em que o número de operações é máximo, número de operações com a matriz reduzida
P3T+	S	simulação	não se aplica	não se aplica	informações sobre carga, comunicação, tempo de computação, taxa de acerto de acesso à memória cache

Tabela A.3: Características das ferramentas descritas no texto (parte 1).

Nome da Ferramenta	H/S	Abordagem	Instrum./Medição	Tipos de dados	Dados
Pablo	S	benchmarking	modificação do código fonte e do executável	extração de traços de eventos, profiling	tempo de processamento, contagem de disparos de rotinas, comportamento em relação à troca de mensagens, utilização de E/S
Paradyn	S	benchmarking	modificação do código executável	extração de traços de eventos	tempo de CPU, barreira, informações sobre quantidades de E/S, comunicação e sincronismo
PAT	S	benchmarking	monitoração por hardware	extração de traços de eventos	eventos da execução do programa
PAWS	S/H	simulação	não se aplica	profiling, medição de intervalo de tempos	curvas sobre speedup, perfil do paralelismo, perfil da execução dos programas
PDL	S/H	simulação	não se aplica	não se aplica	ciclos por instrução, tempo de processador gasto por cada instrução, custo dos procedimentos
RSim	H/S	simulação	não se aplica	não se aplica	estatísticas sobre o processador (utilização, tempo de uso, etc.), memória, "cache" e rede de conexão

Tabela A.4: Características das ferramentas descritas no texto (parte 2).

Nome da Ferramenta	H/S	Abordagem	Instrum./Medição	Tipos de dados	Dados
Simics	H/S	simulação	não se aplica	não se aplica	número de instruções por unidade de processamento, número de operações por segundo, número de operações por segundo em cada unidade de processamento
SPEC	H/S	benchmarking	não se aplica	medição de intervalo de tempos	velocidade de processamtento, vazão, medidas comparativas em SPECs
TotalView	S	benchmarking	monitoração por software	profiling, extração de traços de eventos	tempo de execução do programa e sub-rotinas, tamanho e valores de variáveis, dados sobre comunicação e endereçamento de memória
Vampir/Vampirtrace	S	benchmarking	instrumentação do código fonte	extração de traços de eventos, profiling	porcentagens dos processos em relação ao tempo total do programa, quantidade de dados transferida, mensagens trocadas pelos processos

Tabela A.5: Características das ferramentas descritas no texto (parte 3).