Ferramentas de Avaliação de Desempenho de Sistemas Computacionais

Prof. Sérgio L. Cechin
Prof. João C. Netto

Instituto de Informática e PPGC da UFRGS
Av. Bento Gonçalves, 9500 - Bloco *IV*
CEP 91501-970 - Porto Alegre - RS - Brasil
E-Mail: (cechin, netto}@ inf.ufrgs.br

1 Introdução

Para que um novo produto seja aceito, é necessário que os serviços oferecidos sejam de qualidade. Esta necessidade de aceitação faz com que os investidores cerquem-se do maior número possível de elementos que indiquem o sucesso de um novo produto. Esta premissa é verdadeira, também, na área tecnológica, onde os novos produtos devem ser avaliados, antes de sua introdução no mercado. Fatores importantes nesta avaliação são a viabilidade econômica, o desempenho, a segurança de operação, entre outros.

Este curso tem por objetivo apresentar algumas técnicas usadas na *avaliação de desempenho de sistema computacionais*. Serão discutidas alguma ferramentas de avaliação e os resultados que se pode esperar das mesmas.

É importante salientar a principal razão para utilizar a avaliação de desempenho: o baixo custo. Não é raro que o custo associado com a avaliação do sistema real seja muito elevado. Isso é especialmente verdadeiro quando estamos interessados em verificar a viabilidade técnica de um novo produto: como saber se determinada máquina é capaz de solucionar, com um desempenho mínimo especificado, as tarefas para as quais está sendo projetada? Não é aceitável que um sistema seja desenvolvido para responder a determinada tarefa e, no final do projeto, este objetivo não seja alcançado.

As técnicas que formam a avaliação de desempenho não permitem prever como o sistema funcionará. Estas técnicas fornecem resultados, em geral numéricos, que podem ser usados para estimar o comportamento futuro do sistema em desenvolvimento. Assim, a aplicação destas técnicas permite a comparação numérica, por exemplo, de duas alternativas de projeto, antes que sejam usadas, auxiliando na tomada de decisão. A análise de desempenho é realizada através da comparação entre valores numéricos [FER90].

A próxima seção apresenta as principais técnicas de avaliação existentes. Na seção 3 são apresentados as técnicas de medidas elementares para avaliação e a seção 4

discute os conceitos básicos da construção de modelos. A análise de resultados numéricos obtidos de uma ferramenta de avaliação é o tema da seção 5.

2 Técnicas de Avaliação

O objetivo da aplicação das técnicas de avaliação de desempenho é responder a determinadas perguntas. Estas são formuladas durante a tomada de decisão e visam dirigir o desenvolvimento do sistema. Antes de responder a estas perguntas, deve-se aplicar alguma técnica de avaliação. Para estudarmos as técnicas, estas serão divididas conforme aceito por vários pesquisadores [ALL78, FER78, GEL80, LAZ84, BRA87].

As técnicas de avaliação de desempenho podem ser divididas em *técnicas elementares* e *técnicas indiretas*.

As técnicas elementares são aquelas aplicadas diretamente sobre os sistemas reais. Desta forma, requerem que o sistema a ser avaliado exista e possa ser submetido aos testes. Não é um grupo de técnicas que possa ser usado durante o desenvolvimento dos sistemas. Entretanto, quando o sistema existe e sua aplicação é adequada, fornecem resultados bastante próximos da operação real do mesmo.

Para a aplicação destas técnicas, mesmo que o sistema exista, deve ser possível a aplicação dos testes. Por exemplo, não é aceitável efetuar testes no sistema computacional de uma usina nuclear, visando verificar qual o comportamento do mesmo diante de um vazamento radioativo.

Outro fator que deve ser levado em consideração é o da interferência no sistema. Quando um sistema é instrumentado para permitir a extração de medidas de desempenho, este estará operando em condições diferentes daquelas consideradas normais. A aplicação deste tipo de técnica deve ser acompanhado de um estudo que demonstre não ser significativa a interferência da instrumentação na operação do sistema.

As técnicas elementares são *monitoração* e *benchmarks*. Nas monitoração, deseja-se obter um índice de desempenho: um número que expresse o desempenho médio do sistema como um todo. Por exemplo, avaliar um conjunto de máquinas diferentes através da medição de quanto tempo gastam para resolver um algoritmo.

A técnica de benchmark consiste na aplicação de uma carga de trabalho (workload) sobre os sistemas. Esta carga de trabalho deve ser bem conhecida, ou seja, deve ter sido projetada para exercitar aquelas características que deseja-se medir. Por exemplo, a avaliação de unidades de processamento (UCP) será feita pela aplicação de cargas de trabalho formadas por instruções desta unidade. Não faz sentido avaliar a operação de uma unidade de processamento, por exemplo, usando acessos a disco.

As técnicas indiretas representam um grupo importante pois permitem a avaliação de sistemas não existentes, sendo as mais aplicadas no estágio de

desenvolvimento. Como o sistema real não existe, deve-se recorrer a *modelagem de sistemas*, sendo esta a principal característica destas técnicas.

Estas técnicas utilizam-se de ferramentas de previsão, que possibilitam a modelagem dos sistemas segundo algum paradigma. Os resultados fornecidos por estas técnicas estará tão próximo da realidade quanto a precisão do modelo empregado. Isso não significa um modelo detalhado e complexo, mas um modelo que seja suficientemente preciso para fornecer resultados que possibilitem decisões corretas.

As técnicas indiretas usam ferramentas que podem ser analíticas, um conjunto de fórmulas e/ou diagramas que podem ser manipulados adequadamente, ou uma descrição que será "executada" em alguma ferramenta computacional.

As ferramentas analíticas podem ser *estocásticas* ou *probabilísticas*. A "execução" de uma descrição é chamada de *simulação* e pode ser dividida em *simulação de eventos* ou *simulação discretas*. A divisão das técnicas apresentadas está esquematizada na tabela 1.

Tabela 1 – Divisão das Técnicas de Avaliação

Métodos Elementares		Métodos Indiretos			
Monitoração	Benchmark	Simulação		Analíticos	
		Eventos	Discreta	Estocástica	Probabilística

3 Modelagem

Os modelos de sistemas, no âmbito da avaliação de desempenho, são utilizados pelas técnicas indiretas de avaliação. Desta forma, deveriam ser tratadas no contexto daquelas técnicas. Entretanto, devido a importância do assunto, uma vez que a modelagem pode ser usado em vários campos do conhecimento, e devido a importância das técnicas indiretas, pois são as únicas que podem ser aplicadas na etapa de desenvolvimento, a construção de modelos é tratada em capítulo separado.

Um modelo representa o comportamento de uma realidade ou parte dela. Este conceito é geral. Portanto, aplica-se a qualquer área do conhecimento, onde os modelos sejam necessário.

A construção de um modelo é feita a partir dos recursos de modelagem disponíveis. Estes modelos podem ser tão complexos quanto desejado ou possível. Entretanto, a dificuldade de tratamento é crescente com a complexidade dos mesmos. Conforme já mencionado, os modelos devem ter a complexidade adequada a pergunta que desejamos responder. Deve-se tomar cuidado para não construir um modelo excessivamente complexo, o que pode levar a um consumo exagerado dos recursos disponíveis(analíticos ou computacionais).

Em geral, quando construindo um modelo, passamos a olhá-lo como uma representação fiel da realidade. Deve-se evitar este sentimento. Os problemas advindos desta forma de olhar o modelo e do grau de complexidade do mesmo podem ser exemplificado através da modelagem de uma realidade: desenhar um quadrado; usando uma ferramenta de modelagem específica: capaz de desenhar, apenas, círculos. Na figura 1 são apresentadas três tentativas de modelagem.

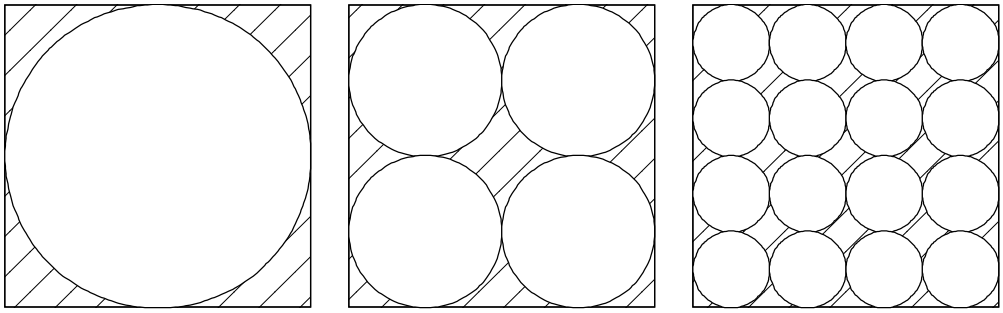

Figura 1 – Modelagem de quadrados usando círculos

Na figura 1, o mesmo quadrado foi modelado usando círculos de diferentes diâmetros. Na primeira (a esquerda), percebe-se que a modelagem não foi muito fiel. Na segunda (no centro), a idéia de quadrado está mais visível. Na última (a direita), pode-se dizer que o resultado é bastante razoável. Se a ferramenta de modelagem permitir, poderiam ser utilizados círculos menores ainda, levando a um modelo bastante semelhante ao objeto real.

Nota-se, entretanto, que o ganho em precisão de modelagem foi acompanhado por uma aumento na complexidade do modelo. Enquanto que na modelagem menos precisa pode-se descrever o modelo através do raio do circulo e das coordenadas do centro, na modelagem mais precisa a descrição deverá fornecer as coordenadas do centro de 16 círculos. Quando menores os círculos, mais coordenadas serão necessárias.

Como observação final do exemplo, percebe-se que, mesmo com os menores círculos possíveis, haverão área não cobertas, dentro do quadrado. Isso significa que, por mais preciso que a modelagem seja, sempre haverão “áreas não cobertas”. Portanto, a análise dos resultados deve ser crítica, lembrando sempre que estamos manipulando um modelo e não o sistema real: os resultados obtidos podem não condizer com a realidade.

Desta forma, ao modelar um sistema, deve-se observar as seguintes atividades:

- estudar e interpretar a realidade a ser modelada: deve-se conhecer a realidade a ser modelada e extrair os pontos importantes, ou seja, aqueles significativos para a construção do modelo desejado;
- listar os pontos a serem avaliados e identificá-los no modelo: garante a convergência da aplicação das técnicas, impedindo a utilização inadequada do modelo;

- escolher a ferramenta mais apropriada para a tarefa de modelagem: uma avaliação criteriosa possibilita a escolha da ferramenta com as características mais adequadas ao modelo;
- construir e verificar a validade do modelo: após construído o modelo, deve-se validá-lo, verificando se os resultado previstos pelo mesmo são confirmados no sistema real. Caso o sistema real não exista ou não esteja disponível, pode-se validar o modelo através de comparação dos resultados de duas técnicas diferentes.
- garantir a confiabilidade dos resultados: é importante que, a cada resultado obtido, seja conhecida a margem de erro associada.

3.1 Tipos de Modelos

Conforme já mencionado, é comum encontrar modelos em diversas área do conhecimento. Desta forma, os tipos de modelos refletem esta realidade.

Os modelos podem ser classificados segundo três categorias: *físicos, analíticos* e *modelos de simulação*.

Os modelos físicos são construções, normalmente em escala, que permitem a avaliação de alguma característica importante do sistema real. Por exemplo, modelos em escala de aviões, permitem a análise do comportamento dos mesmos, quando em vôo. Neste caso, os modelos são colocado em túneis de vento.

A modelagem analítica é feita através de equações matemáticas e diagramas. São exemplos a Teoria das Filas, Redes de Petri e Cadeias de Markov.

Os modelos de simulação são expressões de comportamento em linguagem algorítmica. Em geral, estas descrições podem ser "executadas" em computador.

As técnicas de avaliação de desempenho podem usar alguns tipo de modelo, de acordo com a sua natureza. Assim, por exemplo, as técnicas analíticas devem usar modelos analíticos enquanto que as técnicas de simulação devem usar modelos funcionais. A relação entre os tipos de modelos e suas características está apresentado na figura 2 [FER90].

	Métodos Elementares	**Métodos Indiretos**	
		SIMULAÇÃO	ANALÍTICOS
Sujeito da Avaliação	Sistema real	Modelo funcional	Modelo comportamental
Nível de Abstração	Nenhum	Baixo	Alto
Fator limitante	Tempo de observação	Tempo de simulação	Complex. do algoritmo
Precisão	Real	Alta	Exata

Figura 2 - Características dos Modelos

4 Ferramentas Analíticas de Avaliação

Neste capítulo vamos discutir algumas ferramentas analíticas usadas na avaliação de desempenho: Cadeias de Markov, Redes de Petri e Teoria de Filas.

4.1 Cadeias de Markov

Os modelos obtidos pelo uso de Cadeias de Markov são estocásticos: modelam as probabilidades de tempos de operação.

Os serviços modelados são representados por estados interligados probabilidades de transição. Assim, se existe uma probabilidade de que uma tarefa passe de um serviço para outro, esta deverá ser representada no modelo.

A representação do modelo pode ser feita por um conjunto de equações probabilísticas de transição ou por um modelo gráfico, conforme apresentado na figura 3.

A modelagem através de Cadeias de Markov usa como premissa que o sistema modelado apresenta um comportamento “Markoviano”, ou seja, corresponde a processos em que o estado futuro depende apenas do estado atual, não importando a forma como o sistema chegou a ele. Diz-se que o processo é *memoryless*.

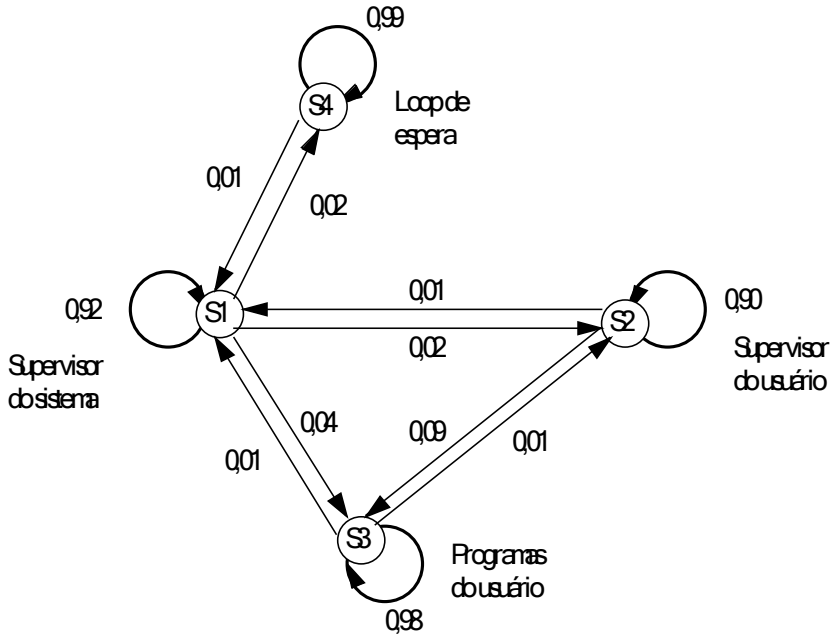

Figura 3 – Exemplo de Cadeia de Markov

A modelagem usando Cadeias de Markov utiliza-se de informações probabilísticas do comportamento do sistema. Da mesma forma, os resultados que podem ser obtidos também são probabilísticos. Por exemplo, pode-se determinar qual a probabilidade de que uma tarefa esteja em um serviço específico. Pode-se determinar, também, qual a probabilidade de que uma tarefa utilize mais do que um valor limite de tempo no seu processamento. Estes modelos são interessantes pois permitem determinar "gargalos" do sistema, determinando-se o grau médio de utilização dos serviços que formam o sistema.

4.2 Redes de Petri

As Redes de Petri são utilizadas na modelagem de sistemas concorrente com restrições de paralelismo. Apresentam propriedades estáticas: *lugares*, *transições* e *arcos dirigidos*; e propriedades dinâmicas: *marcas*.

Os lugares correspondem a condições de operação, que podem estar satisfeitas, quando houver pelo menos uma marca, ou não, quando os lugares estiverem vazios. As transições correspondem a eventos. Estes disparam, quando ocorrem, possibilitando que uma marca existente no lugar origem seja consumida e uma marca seja criada no lugar de destino da transição. Na figura 4 é apresentado um exemplo de Rede de Petri.

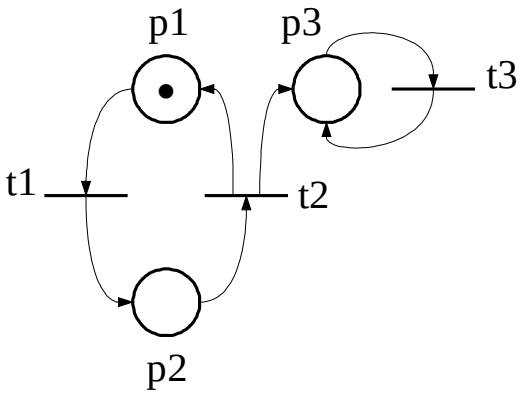

Figura 4 – Exemplo de Rede de Petri

No exemplo da figura 4, existem três lugares (p1, p2 e p3) e três transições (t1, t2 e t3). Observa-se, também, que o único lugar com marca é p1. Desta forma, diz-se que a transição t1 está habilitada a disparar. Se isso ocorrer, a marca de p1 será consumida e será criada nova marca em p2. O comportamento da marca é semelhante ao contador de programa de um processador. Ele indica onde está ocorrendo o processamento.

Entretanto, de forma diferente de uma unidade de processamento, podem haver mais de uma marca na rede. Por exemplo, quando houver uma marca no lugar p2, a transição t2 estará habilitada a disparar. Quando isso ocorrer, a marca de p2 será consumida e serão criadas marcas em p1 e p3. Neste caso, a rede estará operando com duas marcas.

Para que uma transição esteja habilitada a disparar é necessário que todos os lugares de origem contenham marcas. Na figura 5 é apresentado um exemplo em que a transição t2 não está habilitada, apesar de um dos lugares de entrada (ou lugares origem) possuir uma marca.

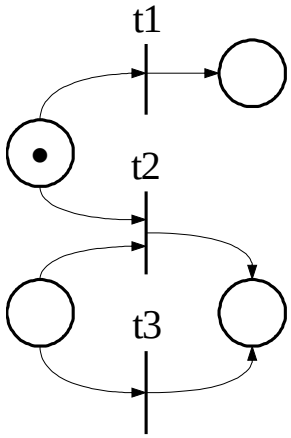

Figura 5 – Exemplo de transição não habilitada

As Redes de Petri podem ser usadas para a modelagem de fenômenos não determinísticos. Isso é feito através do uso de *transições em conflito*: transições que tornam-se habilitadas pelo mesmo conjunto de lugares, ou seja, quando uma estiver habilitada a outra também estará. Neste caso, apenas uma poderá disparar, consumindo a marca que às habilitava, e impedindo o disparo da outra. A escolha da transição que deve disparar é aleatória, podendo-se dizer que existe uma probabilidade de disparo 50% para cada transição. Um exemplo desta situação está representado na figura 6.

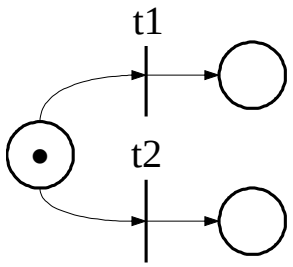

Figura 6 – Exemplo de transições em conflito

Na figura 6, as transições t1 e t2 estão habilitadas simultaneamente, portanto estão em conflito. Se disparar t1, a transição t2 não deverá disparar e vice-e-versa.

Este comportamento nem sempre é adequadamente modelado pelas ferramentas de simulação de Redes de Petri. Se o modelo do sistema tiver necessidade deste tipo de comportamento, deve-se verificar se a ferramenta possibilita este comportamento.

Quando um modelo de Redes de Petri é construído e posteriormente executado (manualmente ou usando alguma ferramenta de simulação), podem ocorrer comportamentos que caracterizam situações de queda de desempenho ou até mesmo falha.

Por exemplo, alguns modelos de Redes de Petri não impõe limitação ao número de marcas que podem ser armazenadas em um lugar. Este acúmulo de marcas significa que o serviço representado por aquela condição não está conseguindo processar na taxa desejada. Se o acúmulo de marcas tende a estabilizar em algum número médio, o sistema é estável. Entretanto, deve-se colocar memória específica para armazenar as tarefas, até que possam ser processadas. O crescimento não limitado do número de marcas indica um sistema instável: não é possível construir um sistema com esta característica, sem correrm os o risco de falha do sistema real.

- as Redes de Petri apresentam as seguintes propriedades:
- modelagem de concorrência e paralelismo;
- natureza dinâmica assíncrona;
- não determinismo;

- possibilidade de modelagem de eventos não primitivos;
- não simultaneidade dos disparos
- modelagem abstrata
- modelagem hierárquica

4.3 Teoria das Filas

A Teoria das Filas é uma técnica para modelagem analítica de sistemas de computador, que permite a análise de desempenho destes sistemas.

Neste contexto, dada uma configuração do sistema e a carga de trabalho pretendida, pode-se avaliar o desempenho do mesmo. Pode-se prever, por exemplo, para cada módulo ou para todo o sistema, a sua taxa de processamento, a sua taxa de utilização e possíveis gargalos.

Dentre as principais área de aplicação desta técnica pode-se citar a de Sistemas Operacionais e a de Redes de Computador. Entretanto, por ser uma técnica geral, sua aplicação não se restringe a área do computação. Pode-se utilizar esta técnica, por exemplo, para a modelagem de complexos industriais.

A técnica da Teoria das Filas é inerentemente estatística, ou seja, suas previsões são sempre do tipo “... em média...”, “... no máximo...”, etc. Desta forma, toda a modelagem é matemática, podendo levar a modelos extremamente complexos e de difícil análise. Entretanto, existem alguns modelos para os quais o desenvolvimento matemático já foi feito. Como resultado destes desenvolvimentos, existem fórmulas tabuladas, o que facilita a aplicação da técnica. Assim, existem alguns padrões conhecidos, para os quais foram desenvolvidas soluções. Para usar a técnica, deve-se classificar o problema em um dos padrões para os quais foram desenvolvidas soluções. Se não existir padrão que possa ser usado na modelagem do problema, pode-se recorrer a duas possíveis soluções: desenvolver o modelo, o que pode demandar muito tempo e requer um profundo conhecimento de estatística, ou recorrer a simulação, usando simuladores como, por exemplo, o GPSS .

4.3.1 Nomenclatura (notação de Kendall)

As filas são classificadas de acordo com um conjunto de seis parâmetros fundamentais. São eles:

- **processo de chegada de tarefas**: indica a forma como as tarefas são submetidas a fila, ou seja, qual a distribuição de probabilidades de chegada de tarefas.
- **tempo de serviço**: indica a distribuição estatística do tempo gasto no processamento de uma tarefa.

- **número de servidores**: uma fila pode ser a entrada de um único servidor. Entretanto, em termos gerais, pode-se ter mais de um servidor, para uma única fila. Este é o caso, por exemplo, das filas de banco: existe apenas uma fila (fila única) mas são vários os servidores (caixas). Na modelagem da teoria, supõe-se que, quando houver mais de um servidor, estes serão idênticos, operam em paralelo e todos têm a mesma prioridade para servir as tarefas.
- **capacidade do sistema**: este parâmetro indica quantas tarefas podem estar, simultaneamente, na fila. Este número compreende aquelas tarefas que estão sendo servidas e aquelas que estão em estado de espera. Caso este parâmetro não seja indicado, supõe-se que seja infinito. Esta premissa corresponde a idealização de um sistema no qual seja desprezível a probabilidade de que alguma tarefa tenha que ser descartada, por falta de espaço de espera.
- **tamanho da população**: corresponde ao número total de tarefas potenciais: aquelas que podem ser aplicadas à fila. É comum considerar-se que este parâmetro é infinito. Novamente, esta idealização pode ser feita quando as tarefas potenciais são em número muito grande, de tal forma que este valor tem pouca influência nas equações estatísticas de comportamento da fila.
- **disciplina de serviço**: corresponde a ordem como as tarefas são servidas. A disciplina de serviço pode ser das mais variadas formas. São exemplos a ordenação FCFS (*First Come First Served*), LCFS (*Last Come First Served*), LCFS-PR (LCFS com preempção), Round-Robin, compartilhamento de tempo, etc. Se este parâmetro não for informado, supõe-se a ordenação FCFS (FIFO).

Os dois parâmetros: processo de chegada de tarefas e tempo de serviço, correspondem a distribuições de probabilidades. Para indicar estas distribuições são usadas letras, da seguinte forma: M, indicando uma distribuição exponencial (memoryless) e IID (chegada de tarefas independente e identicamente distribuídas); D, distribuição determinística (tempos fixos com variância zero); G, para uma distribuição qualquer. Além destas são usadas EK, para distribuição de Erlang com parâmetro K (muito usada para modelagem de sistemas comutados, por exemplo, telefonia) e HK, para distribuição hiperexponencial com parâmetro K.

4.3.2 Parâmetros Básicos

A representação gráfica de uma fila é composta pela espaço reservado para as tarefas que estão em espera e pelos servidores. Na figura 7 está representada uma fila com um único servidor.

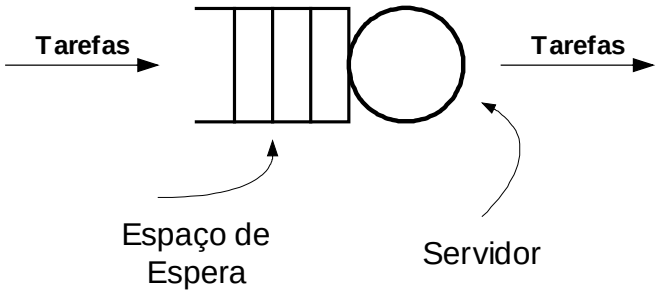

Figura 7 – uma fila com um servidor

Para que a análise possa ser efetuada, o conhecimento de dois parâmetros é fundamental: a taxa de chegada de tarefas (representada por λ) e a taxa de processamento das mesmas (representada por μ).

Dados as taxas λ e μ, pode-se calcular o tempo médio entre chegadas de tarefas e o tempo médio para servir uma tarefa, pelo uso das fórmulas:

$$E(T_q) = \frac{1}{\lambda} \quad \text{e} \quad E(T_s) = \frac{1}{\mu}$$

onde Tq corresponde a variável aleatória do tempo médio de espera de uma tarefa e Ts é a variável aleatória do tempo médio de serviço das tarefas.

A partir dos dois parâmetros λ e μ pode-se calcular a taxa de ocupação do servidor (tempo ocupado em relação ao tempo total) para uma fila M/M/1:

$$\rho = \frac{\lambda}{\mu}$$

Com os três parâmetros e a configuração da fila a ser analisada, pode-se calcular valores médios que caracterizam o desempenho do sistema. A caracterização do desempenho de uma fila é feito, normalmente, pelo cálculo dos VALORES MÉDIOS dos seguintes fatores:

- **número de tarefas no sistema (sistema de fila)**: corresponde ao número médio de tarefas que estão aguardando serem servidas mais as tarefas que estão sendo servidas;
- **número de tarefas na fila (fila de espera)**: é o número médio de tarefas que estão aguardando serem servidas;
- **número de tarefas em serviço**: número médio de tarefas que estão sendo servidas. Como estes valores são médios, o número de tarefas sendo servidas jamais será igual ao número de servidores, uma vez que a probabilidade de que nenhum servidor esteja ocupado é diferente de zero.

Pode-se demonstrar que o número médio de tarefas no sistema é igual a soma do número médio de tarefas aguardando pelo serviço e do número médio de tarefas sendo servidas.

A partir do cálculo dos três parâmetros anteriores e pela aplicação do Teorema de Little, pode-se calcular os tempos médios das tarefas em cada dispositivo. Os tempos que podem ser calculados são:

- **tempo médio de resposta**: corresponde a tempo médio que uma determinada tarefa permanece dentro do sistema (fila de espera mais servidor). Este tempo é calculado a partir do número médio de tarefas no sistema;
- **tempo médio de espera na fila**: indica o tempo médio que uma tarefa deverá aguardar na fila de espera, antes de ser servida. Este tempo é calculado a partir do número médio de tarefas na fila de espera;
- **tempo médio de serviço das tarefas**: é o tempo gasto, em média, para que uma tarefa seja processada. Corresponde ao valor $E(T_s)$.

4.3.3 Relações Importantes

Existem quatro relações entre os parâmetros dos sistemas de filas que devem ser conhecidas: a condição de estabilidade, a dos números de tarefas, a dos tempos e o Teorema de Little (relação entre número de tarefas e o tempo).

Condição de Estabilidade

Esta condição estabelece os requisitos necessários para garantir que a fila de espera não cresça indefinidamente, ou seja, que apresente um ponto de equilíbrio. Este ponto de equilíbrio corresponderá ao número médio de tarefas na fila de espera.

Se o sistema de fila receber, EM MÉDIA, mais tarefas do que é capaz de processar, então a fila de espera, certamente, deverá crescer indefinidamente. Assim, para que isso não ocorra, o número médio de tarefas completadas deverá ser, sempre, maior do que o número médio de tarefas apresentadas ao sistema. Traduzindo em equações, a taxa média de chegada de tarefas (λ) deverá ser menor do que a taxa de processamento das mesmas (μ). A partir disso, pode-se dizer que um sistema de filas será estável se $\rho < 1$, com:

$$\rho = \frac{\lambda}{\mu}$$ se houver apenas um servidor, ou

$$\rho = \frac{\lambda}{m\mu}$$ se houver “m” servidores.

Relação Entre o Número de Tarefas

Conforme já apresentado, o número médio de tarefas no sistema como um todo (fila de espera mais servidor) pode ser calculado pela soma do número médio de tarefas na fila de espera e o número médio de tarefas no servidor. Assim:

$$E(N) = E(N_q) + E(N_s)$$

Se o número de tarefas na fila de espera (Nq) for estatisticamente independente do número de tarefas no servidor (Ns), então pode-se calcular a variância do número de tarefas no sistema como a soma das variâncias de seus componentes:

$$VAR(N) = VAR(N_q) + VAR(N_s)$$

Relação Entre os Tempos

De forma semelhante ao número médio de tarefas, o tempo médio que uma tarefa gasta para percorrer todo o sistema da fila, aguardando e depois sendo servida, pode ser calculado pela soma dos tempos médios gastos nas duas etapas componentes. Desta forma, tempos que:

$$E(T_r) = E(T_q) + E(T_s)$$

A variância apresenta comportamento análogo ao do número de tarefas.

Teorema de Little

Este teorema fornece a relação existente entre o tempo médio que uma tarefa permanece em um sistema qualquer e o número médio de tarefas neste mesmos sistema. Sua formulação é a seguinte:

Número médio de tarefas = taxa de chegada × tempo médio de resposta.

Este teorema é válido nas seguintes condições:

- distribuição de probabilidades arbitrária, para a chegada de tarefas;
- quando não houver perda de tarefas. Isso ocorre sempre que o espaço reservado para a fila de espera for infinito (caso ideal) ou quando o tamanho desta fila for suficientemente grande para que a taxa de perdas seja desprezível.

No caso da perda de tarefas atingir valores que não podem ser desprezados, o teorema ainda é válido, desde que consideremos a taxa efetiva de processamento. Esta taxa corresponde às tarefas efetivamente tratadas pelo sistema. Corresponde a taxa média de chegada de tarefas menos a taxa de perdas.

A partir do uso deste teorema, pode-se calcular os tempos médios de permanência das tarefas em um sistema a partir do número médio de tarefas no sistema, observadas em um determinado instante. No caso de uma fila M/M/1, teremos as seguintes equações:

$E(T_r) = \lambda \times E(N)$ para o tempo médio de resposta;
$E(T_q) = \lambda \times E(N_q)$ para o tempo médio de espera na fila;
$E(T_s) = \lambda \times E(N_s)$ para o tempo médio de serviço da tarefa.

4.3.4 Processos Birth-Death

A chegada de tarefas a um sistema será caracterizada como de Poisson se estas forem IID (Independentes e Identicamente Distribuídas) e a sua distribuição de probabilidades for exponencial. Os processos de Poisson, por sua vez, são um caso particular de processos Birth-Death: processos de Markov no qual as transições de estado estão restritas aos estados vizinhos. Note-se que, para um processo ser Birth-Death, não é necessário uma distribuição exponencial, mas uma distribuição memoryless (o estado futuro dependerá, apenas, do estado atual; o histórico de transições de estados que levaram ao estado presente não influencia na determinação do estado futuro).

A modelagem de filas usando processos Birth-Death é bastante adequada, pois é comum considerar-se que a chegada de tarefas, assim como o processamento, são processos de Poisson.

O estado de sistemas modelados por processos Birth-Death pode ser representado por um número natural, que representará o número de tarefas presentes no sistema.

Se um sistema está no estado $n = i$, então existem i tarefas no mesmo. A chegada de nova tarefa causará a transição para o estado $n = i + 1$ (birth), enquanto que a conclusão de uma tarefa levará o sistema para o estado $n = i - 1$ (death). Estas transições ocorrem com freqüência λ e μ, respectivamente. Na figura 8, estão representados os estados e as transições descritas.

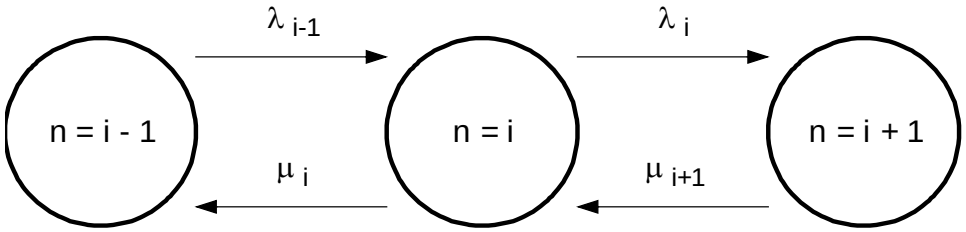

Figura 8 – Processo *Birth-Death*

Neste contexto, pode-se demonstrar que a probabilidade de que um processo birth-death esteja no estado n, será calculado por:

$$p_n = \frac{\lambda_0 \times \lambda_1 \times \ldots \times \lambda_{n-1}}{\mu_1 \times \mu_2 \times \ldots \times \mu_n} \times p_0 ,$$

onde p0 é a probabilidade de que o sistema esteja no estado n = 0, ou seja, não existam tarefas sendo processadas nem na fila de espera. A partir deste resultado e do fato que o somatório de todas as probabilidades deve resultar 1:

$$\sum_{n=0}^{\infty} p_n = 1 ,$$

pode-se determinar o valor de p0 e, consequentemente, uma equação para pn.

A partir da equação de pn e usando somatórios simples para o cálculo dos valores esperados, pode-se determinar os índices do sistema: E(N), E(Nq) e E(Ns).Finalmente, usando o Teorema de Little, calcula-se E(Tr), E(Tq) e E(Ts).

4.3.5 Fila M/M/1 como Processo *Birth-Death*

Neste tipo de fila, a taxa de chegada de tarefas assim como a de conclusão são constantes e independentes do estado do sistema. Temos, então, que para qualquer valor de n: $\lambda n=\lambda$ e $\mu n=\mu$. Desta forma, temos que:

$$p_n = \left(\frac{\lambda}{\mu}\right)^n \times p_0$$

Definindo $\rho = \frac{\lambda}{\mu}$ e usando $\sum_{n=0}^{\infty} p_n = 1$, encontramos:

$p_0 = 1-\rho$,

ou seja,

$p_n = \rho^n \times (1-\rho)$

Cálculo dos Números de Tarefas

Conforme já explicado, a partir do valor de p0 e da equação de pn, pode-se calcular os números médios de tarefas no sistema. Assim, temos:

$E(N) = \sum_{n=0}^{\infty} n \cdot p_n = \frac{\rho}{1-\rho}$ para o número médio de tarefas no sistema;

$E(N_q) = \sum_{n=1}^{\infty} (n-1) \cdot p_n = \frac{\rho^2}{1-\rho}$ para o número médio de tarefas na fila de espera;

$E(N_s) = E(N) - E(N_q) = \rho$ para o número médio de tarefas no servidor.

Cálculo dos Tempos

Os tempos médios que as tarefas permanecem em um determinado módulo serão calculados pela aplicação do Teorema de Little sobre os números de tarefas correspondentes. Desta forma, serão os seguintes:

$E(N) = \lambda \times E(T_r)$ e $E(T_r) = \frac{1}{\mu - \lambda}$ tempo médio das tarefas no sistema;

$E(N_q) = \lambda \times E(T_q)$ e $E(T_q) = \frac{\rho}{\mu - \lambda}$ tempo médio das tarefas na fila de espera;

$E(N_s) = \lambda \times E(T_s)$ e $E(T_s) = \frac{1}{\mu}$ tempo médio das tarefas no servidor.

4.3.6 Conclusão

A técnica descrita aqui é bastante poderosa. Entretanto, requer o reconhecimento, dos padrões já desenvolvidos, no problema sob análise. Isso pode fazer com que o analista descarte esta técnica, por desconhecer os padrões cujas soluções existem.

Outro fator que dificulta o uso da técnica é o pouco conhecimento de estatística. Esta dificuldade aparece com muita ênfase na análise dos resultados, principalmente quando estes são obtidos através de simulação.

O texto apresentado aqui apenas "arranha" a superfície do assunto. Muitos são os textos clássicos existentes, além de artigos em revistas com soluções para novos padrões de filas.

O desenvolvimento apresentado para uma fila M/M/1 pode ser extrapolado, em linhas gerais, para o desenvolvimento de outros tipos de filas. Entretanto, é comum que cada tipo de fila apresente características específicas e que devem ser consideradas no desenvolvimento da solução.

Finalmente, deve-se citar as técnicas de tratamento de Redes de Filas, que permitem a análise de sistemas complexos. Estas técnicas não serão apresentadas aqui, mas podem ser encontradas, por exemplo, em [JAI91].

5 Benchmarks

O que é benchmark?

Benchmark é um teste que mede a performance de um sistema, ou subsistema, em uma tarefa, ou conjunto de subtarefas, bem definidas [SIL 95].

Afirma-se que o melhor benchmark é a própria aplicação. Entretanto, isso não é muito realista: não é sempre possível executar a aplicação na máquina em questão; o programa pode ter sido otimizado para a máquina em questão; desejamos caracterizar velocidade de máquina independente de um particular programa de aplicação. O que se deseja é caracterizar velocidade de máquina independente do programa de aplicação particular. Por conseguinte, o melhor benchmark seria [WEI 91]:

- é representativo para algum tipo de estilo de programação (isto é, programação de sistemas, programação numérica, programação comercial)
- é escrito em linguagem de alto nível, tornando ele portável sobre diferentes plataformas
- pode ser mensurável facilmente
- tem uma ampla distribuição

Benchmarks freqüentemente são suados para mensurar características gerais como gráficos, I/O, computação (inteira e ponto-flutuante), etc. Qualquer aspecto da performance de computadores pode ser objetivo de medidas de benchmarks pelo usuário.

Características desejáveis em um benchmark

Um benchmark deve possuir características semelhantes às das aplicações reais do sistema que se está avaliando. Algumas delas são [ALL 95]:

- Tempo de execução
- Espaço do programa
- Freqüência de instruções
- Estruturas detalhadas
    - Tamanho de bloco básico
    - Estrutura de loop característica
    - Freqüência de desvios e imprevisibilidade
    - Dependência de dados
    - Dependência de controle
    - Múltiplos fluxos de execução
    - Código orientado a objetos

Como são usados os benchmarks?

Geralmente, benchmarks são usados para mensurar a performance de um sistema desconhecido em um atarefa conhecida, ou no mínimo bem definida. Benchmarks também podem ser usados como ferramentas de diagnóstico e monitoração. Executando um benchmarks e comparando os resultados, sob uma configuração conhecida, pode potencialmente apontar as causa da perda de performance. De maneira similar, um desenvolvedor pode executar um benchmark após fazer uma modificação. al;em desses enfoques, benchmarks são freqüentemente usados para garantir um nível mínimo de performance exigido em uma especificação.

Enfoques envolvendo medidas de benchmarks

Geralmente, benchmarks tem o significado de ser uma medida sistemática de alguns aspectos de performance de sistemas de computação. Essas medidas são sistemáticas quanto tentativas são feitas para controlar o ambiente no qual as medidas serão realizadas.

Existem diferentes tipos de códigos usados em estudos científicos de benchmark. Alguns desses códigos são [BER 91]:

Sintéticos: Esses segmentos de código não representam qualquer computação real, mas exercitam várias funções de máquina básicas. Whetstone e Dhrystone são os mais comuns exemplos de tais benchmarks.

Kernel: Esses segmentos de código são geralmente extraídos de grandes programas e representam porções significantes de tempo de execução em aplicações reais.

Algoritmos: Esses segmentos de código representam algoritmos bem definidos do tipo que aparece em textos de computação numérica.

Aplicações: Esses segmentos de código representam programas de aplicação completos de resolução de problemas científicos bem definidos. Benchmarks baseados nesta característica podem não computar o tempo total de aplicações, pois tempos de I/O podem ser excluídos.

Influências nas medidas de benchmarks que não são causadas pela CPU [WEI 91]

- Influência da linguagem de programação

Existem propriedades em linguagens (seqüências de chamadas, semântica de ponteiros, semântica de strings) que têm obvia influência em tempo de execução, mesmo se o programa fonte pareça similar e produza os mesmos resultados.

- Influência do compilador

Diferentes níveis de otimização influenciam no tempo de execução do programa.

- Sistema de bibliotecas Runtime

Quando benchmarks são comparados, devemos levar em consideração que diferentes sistemas podem resultar em diferentes tempos gastos na execução das mesmas funções de bibliotecas matemáticas. Por exemplo, se você está preocupado com a exatidão na medida de computações de ponto-flutuante, é importante saber qual biblioteca está sendo usada com o benchmark.

- Tamanho de memória cache

É importante perceber que a performance aumenta quando o tamanho da memória cache se aproxima ao tamanho relevante do benchmark. Dependendo do tempo de acesso da cache e da memória principal, o efeito do tamanho da cache pode ser considerável. Existem casos que uma deferente ordem de linkagem pode lavar o mesmo benchmark a um aumento do tempo de execução em 5%.

Certifique-se sobre o benchmark

- Não confie em números MIPS se não existir uma clara derivação de como eles foram obtidos.
- Verifique se números MFLOPS podem servir como um benchmark padrão. Esse tipo de benchmark caracteriza sua aplicação?
- Conheça as propriedades dos benchmarks cujos resultados são divulgados.
- Esteja seguro de conhecer todos os fatos relevantes sobre o sistema e sobre o software e benchmark:
  Hardware: freqüência de clock, latência de memória, tamanho de cache.
  Software: linguagem de programação, tamanho de dados, versão do compilador, opções do compilador, bibliotecas Runtime.
- Verifique a listagem do benchmark e certifique-se que nenhuma otimização ilegal foi feita.
- Procure bons relatórios de performance. Bons bechmarks informam todos os detalhes relevantes.

Como interpretar os resultados de um benchmark?

Pode-se cometer muitos erros ao se interpretar os resultados de um benchmark, tanto se eles são executados por você ou por um vendedor de sistemas comerciais. Deve-se levar em conta [SIL 95]:

1) Que benchmark foi usado? Precisa-se do nome, versão, detalhes sobre mudanças feitas, tanto para portabilidade como para melhorar o desempenho.
2) Em que configuração o benchmark rodou?
   a) processador modelo, velocidade, cache, número de CPUs
   b) memória
   c) c) versões do software (Sistema Operacional, Compilador, Aplicações relevantes, etc.)
   d) d) opções e flags usados pelo compilador/loader para montar o benchmark.
   e) e) estado do sistema (mono-usuário, multi-usuário, ativo, inativo, etc)
   f) f) periféricos, como HDs.

3) Qual a relação entre o benchmark e a carga de trabalho que vai ser exigido? Sem saber o que o benchmark está medindo, pode-se pensar que a aplicação rodará no sistema que tiver o melhor resultado do benchmark.

Whetstone

Primeiro programa (1976) da literatura que foi explicitamente projetado para propósitos de bechmark. Whetstone é o projeto de um programa sintético, consistindo de vários módulos onde cada módulo contém declarações de algum tipo particular (isto

é, integer, floating-point, if, calls), terminando com impressão de resultados. Pesos são atribuídos para os diferentes módulos tal que a distribuição de instruções Whetstone para o benchmark sintético seja equivalente a distribuição observada no programa exemplo.

Características Whetstone [WEI 91]

- Whetstone tem uma alta porcentagem de dados em ponto flutuante e operações em ponto-flutuante. Isso é intencional, visto que o objetivo é representar programas numéricos.
- Uma alta porcentagem de tempo de execução é gasto em bibliotecas de funções matemáticas. Características de implementação alteradas nestas bibliotecas podem alterar os valores de benchmark.
- Whetstone quase não usa variáveis locais. Algoritmos, ou características de arquitetura, projetados para suporte a variáveis locais não fazem diferença em tempos de execução Whetstone.
- Whetstone usa alguns dados globais. Características do compilador sobre variáveis globais, ou otimização, podem incrementar performance em Whetstone.
- Whetstone tem uma alta localidade de código. Taxas próximas a 100% de hit rate podem ser esperadas mesmo em pequenos caches. Pela mesma razão, uma simples reordenação de código fonte pode significativamente alterar o tempo de execução em alguns casos.

Dhrystone

Dhrystone é um programa sintético baseado em características de distribuição de linguagem fonte, em programação de sistemas não-numéricos (sistemas operacionais, compiladores, editores, etc). Algumas dessas características são: conter menos loops, declarações computacionais mais simples, maior quantidade de declarações 'if' e chamadas a procedimentos.

Esse software consiste de doze procedimentos incluídos em um loop de medida com 94 declarações. Durante um loop (1 Dhrystone), 103 declarações (versão C) são executadas dinamicamente. Os resultados são geralmente dados em Dhrystones por segundo. Atualmente, Dhrystone é principalmente usado na versão em C (versão 2.1).

Dhrystone foi o primeiro benchmark a explicitamente considerar a localidade dos operandos: variáveis locais e parâmetros são usados mais freqüentemente que variáveis globais. Devemos, além disso, perceber que máquinas que utilizam registradores para armazenar variáveis locais e/ou parâmetros tem um acréscimo em performance de Dhrystone (máquinas RISC).

Características Dhrystone [WEI 91]

- Dhrystone não contém operações em ponto flutuante em seus loops de medida.
- Uma considerável percentagem de tempo de execução é gasto em função de string.

- Dhrystone quase não contém loops dentro do loop de medida principal. Portanto, em processadores com pequena cache de instruções quase todas as intruções acessadas são caches misses.
- Nenhuma tentativa foi feita para evitar otimização de compiladores. O objetivo é que o programa possa refletir um estilo de programação típico; ele possa ser similarmente otimizado como programas normais.

SPEC [SPE 95] [DIX 91]

SPEC (Standard Performance Evaluation Corporation) é uma corporação formada para estabelecer, manter e aprovar um conjunto padronizado de relevantes benchmarks que podem ser aplicados nas mais recentes gerações de computadores de alta performance [SPE 95]. De outro ponto de vista, os usuários poderão beneficiar-se de uma série objetiva de testes orientados a aplicações, aos quais podem servir como pontos de referência comuns a serem considerados durante um processo de avaliação. Enquanto nenhum benchmark pode completamente caracterizar a performance de um sistema em geral, os resultados de uma variedade de benchmarks podem dar uma perspectiva da performance real esperada.

O principal produto da SPEC são pacotes de benchmarks. O código destes programas são desenvolvidos pela SPEC baseados em códigos doados de inúmeras fontes. Logo, SPEC trabalha para garantir a portabilidade e criar ferramentas e workloads significativos para o código escolhido como benchmark.

Dentre os pacotes de benchmarks SPEC disponíveis, 4 são citados a seguir:

SPECINT92 – (CPU intensive integer benchmarks)
SPECFP92 – (CPU intensive floating point benchmarks)
SPECSDM – (UNIX Software Development Workloads)
SPECFS – (System level file server (NFS) workload)

CPU Benchmarks

Existem atualmente 2 pacotes de SPEC benchmarks de computação intensiva, medindo a performance de CPU, sistemas de memória e a geração de código pelo compilador. Eles usam o UNIX como o veículo de portabilidade, mas já estão sendo portados para outros sistemas. A percentagem de tempo gasto com o sistema operacional e funções de I/O são geralmente insignificantes nestes benchmarks.

- **CINT92** (Versão 1.1) – Este pacote contém 6 benchmarks para executar computação inteira; todos eles escritos em C. Os programas são os seguintes:

008.espresso	Projeto Lógico
022.li	Interpretador
023.eqntott	Projeto Lógico
026.compress	Compressão de Dados
072.sc	Planilha Eletrônica
085.gcc	Compilador

- **CFP92** (Versão 1.1) – Este pacote contém 14 benchmarks para executar computação em ponto-flutuante. 12 são escritos em FORTRAN e 2 em C. Os programas são os seguintes:

013.spice2g6	Projeto de Circuitos
015.doduc	Simulação
034.mdljdp2	Química Quântica
039.wave5	Eletromagnetismo
047.tomcatv	Translação Geométrica
048.ora	Ótica
052.alvinn	Robótica
056.ear	Simulação Médica
077.mdljsp2	Química Quântica
078.swm256	Simulação
089.su2cor	Física Quântica
090.hydro2d	Astrofísica
093.nasa7	Kernels NASA
094.fpppp	Química Quântica

Os benchmarks para CPU podem ser usados para medidas em dois modos:

- Medir velocidade
- Medir throughput

Medir velocidade

Os resultados são expressos como o índice da divisão do tempo de clock para executar uma única cópia do benchmark, comparado com o "tempo de referência SPEC" fixado. SPEC definiu médias inteiras e ponto-flutuantes. São elas:

SPECint92 – média geométrica de 6 índices SPEC de CINT92.
SPECfp92 – média geométrica de 14 índices SPEC de CFP92.

Medir throughput

Com esse método de medida, chamado de método de capacidade homogênea, várias cópias de um dados benchmark são executadas. Este método é particularmente útil para sistemas multiprocessador. Os resultados expressam quantos jobs de um tipo em particular (caracterizado por um individual benchmark) podem ser executados em um dado tempo de execução. Foram definidas médias que são:

SPECrate_int92 – média geométrica de 6 índices SPEC de CINT92.
SPECrate_fp92 – média geométrica de 14 índices SPEC de CFP92.

SDM Benchmarks

SDM (Systems Development Multiuser – Versão 1.1) serve para mensurar a capacidade de um sistema em um ambiente multiusuário UNIX. Contrário aos benchmarks de CPU, o benchmark SDM contém scripts do shell UNIX (consistindo de comandos como cd, mkdir, find, etc) que exercitam o sistema operacional bem como a CPU e componentes de I/O do sistema. Esse pacote contém 2 benchmarks:

057.sdet	Representa o ambiente de software baseado em UNIX/C
061.kenbus1	Representa o ambiente UNIX/C de um meio de pesquisa e desenvolvimento

SFS Benchmarks

SFS (System-level File Server – Versão 1.1) foi projetado para fornecer um método de mensurar e relatar performance em servidores de arquivo NFS. Contém o benchmark: 097.LADDIS. Esse benchmark mede performance em termos de resposta NFS e throughput. Ele faz isso gerando um workload NFS sintético baseado em uma abstração de um misto de operações NFS e uma taxa de requisições de operações NFS.

SFS Benchmarks fornece resultados incluindo completa informação de configuração do servidor (hardware e software) e um gráfico do tempo de resposta do servidor versus carga NFS para o misto de operações 097.LADDIS.

6 Bibliografia

[ALL 95] ALLISON, Dennis – About Benchmarks and Workloads. http://umunhum.stanford.edu/embedded/about/about.html. Nov/1995.

[ALL78] ALLEN, A. O. Probalility, Statistics, and Queueing Theory; Whit Computer Science Applications. London: Academic Press. 1978.

[BAR79] BARNES, F. M. Measurement and Modelling Methods for Computer Systems Performance Studies. [S.l.]: Lagton Information Systems, 1979.

[BER 91] BERRY, M.; CYBENKO, G. and LARSON, J. – Scientific benchmark characterizations. Parallel Computing, Vol. 17, No. 10 & 11. Dec/1991.

[BRA87] BRATLEY, P.; FOX, B. SCHRAGEL, L. A Guide to Simulation. New York: Spring-Verlag, 1987.

[DIX 91] DIXIT, Kaivalya M. – The SPEC benchmarks. Parallel Computing, Vol. 17, No. 10 & 11. Dec/1991.

[FER78] FERRARI, D. Computer Systems Performance Evaluation. Englewood Cliffs: Prentice-Hall. 1978.

[FER90] FERNANDES, P. H. L. Modelos para Interconexão de Processadores: Avaliação de Desempenho de Alocação de Recursos. Porto Alegre: CPGCC da UFRGS, 1990. (Dissertação de Mestrado).

[GEL80] GELENBE, E.; MITRANI, I. Analysis and Synthesis of Computer Systems. London: Academic Press. 1980.

[JAI 91] JAIN, Raj. The art of computer systems performance analysis. Littleton: John Wiley & Son, 1991. 685p.

[LAZ84] LAZOWSKA, E. et al. Quantitative System Performance: Computer System Analysis Using Queueing Networks models. Englewood Cliffs: Prentice-Hall. 1984.

[SIL 95] SILL, Dave – Benchmarks FAQ Version 0.6. http://hpwww.epfl.ch/bench.FAQ.html. 28/Mar/1995.

[SPE 95] SPEC FAQ / SPEC Primer. http://hpwww.epfl.ch/bench/SPEC.FAQ.html. 15/Dec/1995.

[SVO76] SVOBODOVA, L Computer Performance Mesaurementt and Evaluation Methods: Analysis and Applications Elsevier Computer Science Library.

[WEI 91] WEICKER, Reinhold P. – A detailed look at some popular benchmarks. Parallel Computing, Vol. 17, No. 10 & 11. Dec/1991.