
UNIVERSIDADE FEDERAL DE PERNAMBUCO
GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO
CENTRO DE INFORMÁTICA

# ANÁLISE DE DESEMPENHO DE SISTEMAS OLTP UTILIZANDO O BENCHMARK TPC-C

TRABALHO DE GRADUAÇÃO

**Aluno:** Marcelo Rodrigues Nunes Mendes (mrnm@cin.ufpe.br)
**Orientador:** Paulo Romero Martins Maciel (prmm@cin.ufpe.br)

Recife, 5 de outubro de 2006.

# Sumário

Sumário....1
Resumo....4
Abstract....5
1. Introdução....6
1.1. Motivação....6
1.2. Estrutura da Monografia....7
2. O ambiente OLTP....9
2.1. Noções de Arquitetura de Computador....10
2.1.1. Hierarquia de Memória....12
2.2. Subsistema de I/O....14
2.2.1. Discos Magnéticos....14
2.2.2. Padrões de Acesso a Dados....15
2.2.3. RAID....16
2.3. SGBD....18
3. Análise de Desempenho e Planejamento de Capacidade....19
3.1. Introdução....19
3.2. Noção de Sistema e de Modelo....19
3.2.1. Sistema....19
3.2.2. Modelo....20
3.3. Uma Abordagem para Análise de Desempenho....20
3.4. Técnicas para Análise de Desempenho....21
3.5. Seleção de Métricas....23
3.6. Caracterização de Carga de Trabalho....24

3.7. Benchmarking ..... 25
3.8. Planejamento de Capacidade ..... 26
4. Fundamentos de Modelagem de Sistemas ..... 28
4.1. Introdução à Teoria das Filas ..... 28
4.1.1. Notação ..... 28
4.1.2. Lei de Little ..... 30
4.2. Sistemas Computacionais e Redes de Filas ..... 31
4.2.1. Descrição dos Clientes ..... 32
4.2.2. Descrição dos Centros de Serviço ..... 33
4.2.3. Modelos de Redes de Filas e Teoria das Filas ..... 33
4.2.4. Limitações e Aplicabilidade de Teoria das Filas ..... 34
4.3. Leis Operacionais ..... 35
4.3.1. Lei da Utilização ..... 36
4.3.2. Lei do Fluxo Forçado ..... 36
4.3.3. Conseqüências da Lei de Little ..... 38
4.4. Estimativa de Parâmetros com Modelos de Regressão ..... 39
5. O Benchmark TPC-C ..... 42
5.1. Visão Geral ..... 42
5.2. Layout do Banco de Dados ..... 43
5.3. As Transações ..... 44
5.3.1. New Order ..... 44
5.3.2. Payment ..... 44
5.3.3. Delivery ..... 45
5.3.4. Order Status ..... 45
5.3.5. Stock Level ..... 45

5.4. Ambiente de Execução ..... 46
5.5. Métricas de Desempenho e Escala do Sistema ..... 46
6. Modelagem do Ambiente ..... 50
6.1. Parametrização ..... 51
6.1.1. Descrição dos Experimentos ..... 51
6.1.2. Cálculo de Demanda dos Recursos ..... 54
6.2. Avaliação de Resultados ..... 60
6.2.1. Validação do Modelo ..... 60
6.2.2. Limitações do modelo ..... 63
7. Conclusões ..... 64
Referências Bibliográficas ..... 65
Apêndice A: Tuning do Sistema ..... 67
A1: Configurações do Processador ..... 67
A2: Configurações do Sistema de I/O ..... 67
A3: Configurações do Sistema Operacional ..... 68
A4: Configurações do SQL Server ..... 68

# Resumo

O presente trabalho apresenta um estudo de desempenho de um ambiente OLTP típico, utilizando o *benchmark* TPC-C como carga de trabalho. O objetivo é prever variações no desempenho do sistema em decorrência de mudanças de hardware. No decorrer do trabalho, são discutidos os diversos fatores que influenciam o desempenho de um sistema OLTP, bem como os conceitos fundamentais de modelagem com redes de filas.

**Palavras chaves**: Análise de Desempenho, Banco de Dados, OLTP, Redes de Filas, Benchmark, TPC-C.

# Abstract

This work presents a performance study for OLTP systems, using the TPC-C benchmark as workload. The objective is to predict performance variations due to hardware changes. The several factors that influence the performance of a OLTP system are discussed, as well as the fundamental concepts of queuing networks modeling.

**Keywords**: Performance Evaluation, Database Systems, OLTP, Queuing Networks, Benchmark, TPC-C.

# 1. INTRODUÇÃO

A avaliação de desempenho está presente em todos os momentos do ciclo de vida de um sistema computacional. Na hora de projetar, produzir ou implantar um sistema, o objetivo final é sempre o mesmo: escolher dentre diversas alternativas aquela que proporcione o melhor desempenho, com o menor custo possível.

Entretanto, não existe um meio universal com o qual possamos avaliar o desempenho das diversas classes de sistemas computacionais. Cada aplicação possui características próprias, o que faz com que cada estudo de desempenho seja único. A caracterização da carga (*workload*), a seleção de métricas, a escolha da técnica de avaliação, tudo isso deve ser feito de maneira específica para cada caso, levando-se em consideração as características do sistema objeto de estudo.

Quando a intenção é comparar o desempenho de sistemas distintos, costuma-se lançar mão de *benchmarks*, que são aplicativos projetados para utilizar ao máximo os recursos do sistema. Por limitar a diversidade de operações realizadas, um *benchmark* permite uma comparação inequívoca entre dois sistemas, quanto a suas capacidades em realizar essas operações.

Com o objetivo de avaliar o desempenho de sistemas OLTP, será utilizada a carga de trabalho do *benchmark* TPC-C. Criada pela TPC (*Transaction Processing Performance Council*), organização independente e sem fins lucrativos, a carga consiste em uma miscelânea de transações que simulam as atividades encontradas em um ambiente OLTP complexo. A TPC tem como membros associados grandes fabricantes de hardware e software, como IBM, Microsoft, Oracle, dentre outros, o que confere credibilidade à organização e aos *benchmarks* por ela produzidos.

## 1.1. Motivação

A administração e manutenção de um ambiente OLTP é uma tarefa complexa. Um dos aspectos fundamentais que os administradores de TI se deparam é o *provisionamento de recursos*, isto é: quanto deve ser investido em

determinado recurso de modo a proporcionar o funcionamento ótimo do sistema, dentro das limitações de orçamento. Isso significa responder a questões do tipo “para aumentar o desempenho do sistema é melhor aumentar o poder de processamento ou a quantidade de memória ou quem sabe o número de discos”?

No entanto, essa não é uma tarefa fácil. Os obstáculos são muitos e vão desde a falta de ferramentas adequadas até à ausência de expertise dos administradores de TI. O cenário mais comum torna-se então o do sobre-provisionamento, no qual se fornece uma quantidade de recursos além da necessária para a operação do sistema.

O objetivo desse trabalho é descrever o processo de planejamento de capacidade para sistemas OLTP, propondo um modelo que permita prever ganhos/perdas de desempenho em decorrência de mudanças de hardware. Embora o trabalho foque especificamente na carga do *benchmark* TPC-C, os conceitos e técnicas utilizados podem ser facilmente aplicados em outras situações do mundo real.

## 1.2. Estrutura da Monografia

O restante do trabalho está organizado da seguinte forma:

- O Capítulo 2 apresenta os conceitos gerais e as características de um ambiente OLTP típico. É feita uma breve revisão de aspectos de arquitetura de computadores e sistemas de gerenciamento de banco de dados (SGBD);
- O Capítulo 3 fornece o *background* para o entendimento da área de análise de desempenho. São apresentadas as técnicas comumente utilizadas e as principais métricas. O capítulo ainda provê explanações sobre caracterização de carga de trabalho, *benchmarking* e planejamento de capacidade;
- O Capítulo 4 introduz os conceitos fundamentais da modelagem analítica usando Redes de Filas. São descritas a notação e também as leis que regem os sistemas de filas. No fim do capítulo

é feita ainda uma introdução à estimativa de parâmetros com modelos de regressão linear;

- O Capítulo 5 descreve em detalhes o *benchmark* TPC-C, expondo o layout do banco de dados, as transações que compõem a carga de trabalho e as métricas utilizadas;
- O Capítulo 6 apresenta a modelagem do ambiente TPC-C, utilizando redes de filas. São descritas as técnicas e ferramentas empregadas na construção do modelo. O modelo é então validado, comparando as saídas por ele produzidas com as obtidas em experimentos;
- Por fim, no Capítulo 7 são feitas as conclusões, apontando as contribuições do presente trabalho bem como sugestões para trabalhos futuros.

# 2. O AMBIENTE OLTP

Sistemas de processamento de transação formam provavelmente o tipo mais comum de sistemas de banco de dados em uso nos dias de hoje. Também conhecidos como sistemas OLTP (*On-Line Transaction Processing*), eles são empregados para registrar as atividades de uma organização. Exemplos de aplicações OLTP incluem: sistemas de reserva de passagem aérea, sistemas bancários, sistemas de acompanhamento de mercado de ações, etc. Embora, sejam empregados nas mais diversas áreas tais sistemas compartilham algumas características comuns:

- Grande quantidade de usuários – O sistema deve suportar o acesso simultâneo de muitos usuários;
- Restrições sobre o tempo de resposta – O sistema deve responder às requisições dos usuários em um tempo inferior a um valor especificado;
- Disponibilidade – Devido à importância que têm para as organizações, sistemas OLTP normalmente devem estar em operação 24 horas por dia, 7 dias por semana;
- Padrão de acesso a dados – Em um sistema OLTP típico, as transações são compostas por um misto de leituras e escritas. Os dados são lidos por meio de consultas e escritos por meio de inserções, atualizações e exclusões;

Cada um desses fatores implica em necessidades específicas para que o sistema opere de maneira normal. Em particular, o número de usuários (que determina a intensidade da carga submetida ao sistema) e o tempo gasto para responder às suas requisições (um critério para a avaliação do desempenho do sistema) têm uma maior relevância para esse trabalho.

Para analisar o desempenho de sistemas OLTP, se faz necessário um entendimento dos seus componentes, sejam eles de hardware (CPU, memória, I/O, etc) ou de software (Sistema Operacional, SGBD, etc). Desta feita,

estaremos vendo nesse capítulo, os principais elementos que compõem um ambiente OLTP típico.

## 2.1. Noções de Arquitetura de Computador

Nessa seção será feita uma breve introdução aos conceitos fundamentais de arquitetura de computadores. . A intenção aqui não é de modo algum prover um material completo sobre o assunto, mas fornecer ao leitor o conhecimento necessário ao entendimento das explanações que se seguirão no decorrer do trabalho Aqueles que tiverem interesse no assunto e desejarem obter mais informações devem recorrer à literatura especializada.

A figura abaixo mostra os componentes de um PC típico:

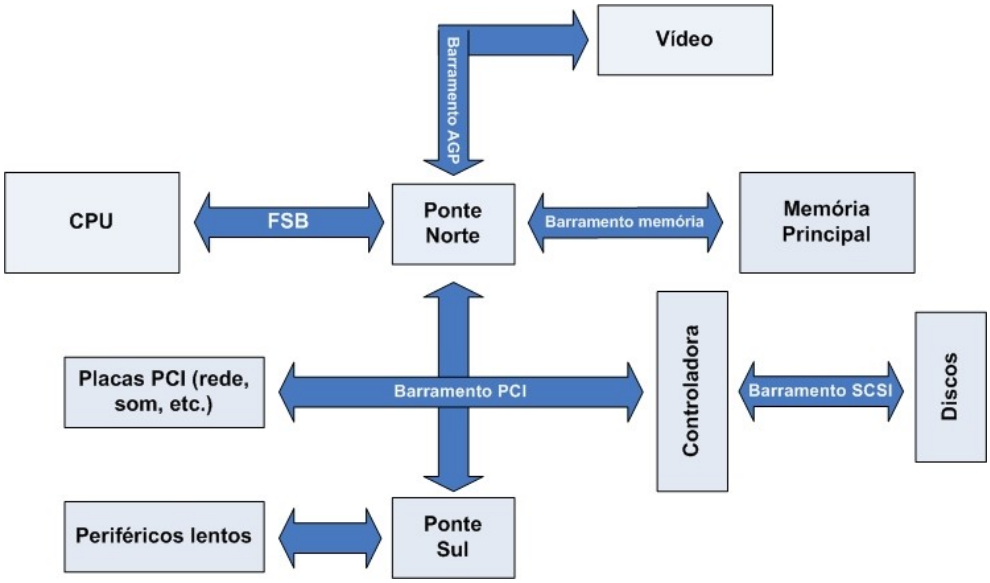



Figura 1. Arquitetura de um PC

O processador (CPU) é o componente principal. É ele que controla as outras partes do computador permitindo a execução de funções como operações matemáticas e manipulação de dados.

Os programas que rodam sobre essa estrutura de hardware nada mais são que conjuntos de instruções binárias. Esses normalmente não ficam armazenados no processador (a exceção são os sistemas dedicados, em que o processador executa um conjunto bastante reduzido de tarefas), mas na

memória principal, cuja função é justamente armazenar as instruções e os dados dos programas que devem ser executados pelo processador.

Independentemente da aplicação, todos os processadores trabalham de um modo parecido. Para a execução de uma instrução, basicamente os mesmos passos são realizados. Primeiramente, a instrução é buscada na memória (*fetch*). Em seguida, a instrução é decodificada (isto é, o processador verifica se a instrução é válida – se faz parte do seu *instruction set* – e caso seja, define quais os próximos passos para a execução da instrução). Depois o processador busca os dados necessários à execução da instrução (operandos). Por fim, a instrução é executada. É claro que o modo de execução varia de uma instrução para outra e melhorias foram desenvolvidas com o passar dos anos (como a inclusão da técnica de ***pipeline*** ou o uso de ***arquiteturas superescalares*** e de memórias cache), mas, no geral, a lógica se mantém a mesma.

Os barramentos servem de conexão entre os diversos componentes do computador, permitindo a troca de dados entre eles. Tradicionalmente, os barramentos são classificados em três categorias [5]:

- Barramentos processador-memória – são extremamente rápidos e, como o nome sugere, servem para a comunicação entre o(s) processador(es) e a memória. É comum usar o termo FSB (*Front-Side Bus*) para designar esse tipo de barramento;
- Barramentos de entrada-saída – são utilizados para a comunicação do processador com os periféricos (discos, portas seriais, etc.);
- Barramentos de *backplane* – são barramentos intermediários entre os dispositivos (ou outros barramentos) e o barramento processador memória.

A figura a seguir mostra um exemplo de organização hierárquica de um conjunto de barramentos:

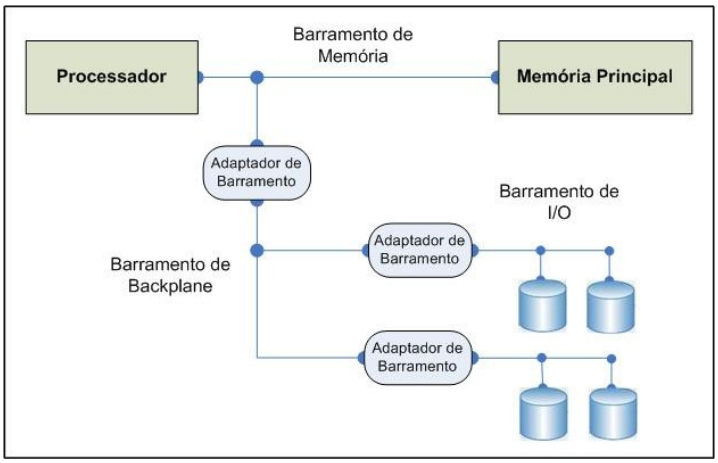



Figura 2.: Organização de barramentos

Nesse caso, usa-se um barramento dedicado para a troca de dados entre o processador e a memória principal. Um barramento de *backplane* faz a interface entre o barramento processador-memória e o barramento de I/O (um exemplo de *backplane* seria o barramento PCI). O barramento de I/O (SCSI, por exemplo) faz a comunicação com os discos.

A ponte norte é o circuito mais importante presente na placa-mãe. Dentro dela encontram-se integrados o controlador de memória, a ponte barramento local-PCI e a ponte barramento local-AGP. A ponte sul é responsável por fazer a interface com os periféricos mais lentos (IDE, USB, porta serial, etc.).

### 2.1.1. Hierarquia de Memória

Como foi visto há pouco, durante a execução de uma instrução, o processador está constantemente acessando a memória, seja na busca das próprias instruções, seja na busca por operandos. O problema é que a velocidade de operação do processador é significativamente maior que a da memória. Isso faz com que em determinados momentos o processador tenha que esperar pela memória, prejudicando o desempenho do sistema. Felizmente, os sistemas computacionais obedecem ao chamado ***princípio da localidade***. Esse princípio estabelece que:

i. Se um item é referenciado, ele tende a ser referenciado novamente dentro de um curto espaço de tempo (localidade temporal);

ii. Se um item é referenciado, itens próximos a eles deverão ser referenciados em breve (localidade espacial).

Isso significa que em um dado instante, um programa usa somente uma pequena parte do seu espaço de endereçamento. Para tirar proveito desse fato, a memória dos computadores é organizada sob a forma de uma hierarquia, onde os níveis mais próximos ao processador são mais rápidos, reduzindo as perdas de desempenho decorrentes de estados de espera. O nível mais próximo é formado pela ***memória cache***. Por serem construídas a partir da tecnologia SRAM (memória estática de acesso randômico), as memórias cache são caras, o que limita o seu tamanho. Já a memória principal é construída a partir da tecnologia DRAM (memória dinâmica de acesso randômico), cujo custo por bit é bem menor, o que lhe possibilita uma maior capacidade. O último nível, chamado de memória secundária, é formado pelos discos magnéticos, que é uma tecnologia mais barata, porém muito mais lenta. Diversos fatores influenciam no desempenho do sistema de memórias, mas de maneira geral, quanto mais memória dos níveis superiores estiver disponível, melhor. No fim das contas, o objetivo de um sistema de memória hierárquico é apresentar ao usuário uma capacidade de memória próxima à disponibilizada pela tecnologia mais barata, e um tempo de acesso próximo ao disponibilizado pela tecnologia mais cara [5].

Uma hierarquia de memória pode ser formada por vários níveis, mas o fato é que os dados são sempre copiados entre dois níveis adjacentes por vez. Se o dado não puder ser encontrado no nível superior diz-se que houve uma ***falta***. A taxa de faltas corresponde à fração dos acessos à memória não encontrados no nível superior. Para reduzir a taxa de faltas da memória cache, é comum o uso de blocos maiores ou o aumento do seu ***grau de associatividade***.

Do mesmo modo que a memória cache serve como armazenamento temporário à memória principal, esta última pode agir como uma “cache” para a

memória secundária. Essa técnica, conhecida como ***memória virtual***, permite que vários programas compartilhem a memória principal sem que um interfira no outro, e ainda que um programa “enxergue” mais memória do que realmente existe. O conceito de memória virtual é particularmente importante no contexto OLTP onde grandes quantidades de dados em armazenamento secundário devem ser manipuladas, sendo constantemente levadas e retiradas da memória principal.

## 2.2. Subsistema de I/O

Como vimos na seção anterior, a memória secundária é várias ordens de grandeza mais lenta que a memória principal. Por esse motivo, um subsistema de I/O sobrecarregado ou mal configurado pode rapidamente degradar o desempenho de todo o sistema, o que reforça a importância de mantê-lo otimizado. Nessa seção será dada uma visão geral dos componentes do subsistema de I/O comumente utilizados em ambientes OLTP.

### 2.2.1. Discos Magnéticos

Os discos magnéticos são de longe o meio de armazenamento permanente mais utilizado em sistemas computacionais. Embora não seja possível efetivamente otimizar um disco magnético é importante conhecer as suas características e limitações para que seja possível configurar o subsistema de I/O de uma maneira que atenda às necessidades de desempenho pré-estabelecidas.

Fisicamente falando, um disco é composto por um conjunto de superfícies magnéticas empilhadas. Cada uma dessas superfícies armazena dados em trilhas concêntricas, que são, por sua vez, divididas em setores. Para cada superfície existe um braço mecânico com uma cabeça de leitura/escrita.

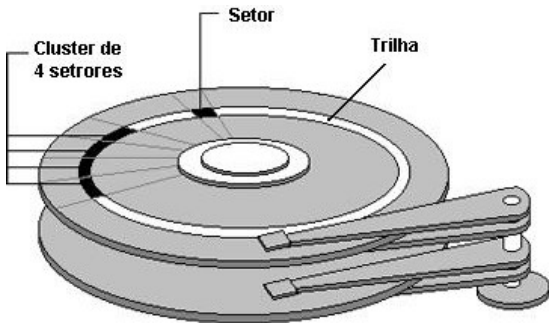



Figura 3.: Estrutura física de um disco magnético.

Durante uma operação de I/O, todos os braços se movimentam simultaneamente na direção de onde se encontra o dado solicitado. Esse movimento dos braços ocorre em dois sentidos: radial e rotacional. O movimento radial corresponde ao deslocamento do braço mecânico de uma trilha a outra. O movimento rotacional corresponde ao deslocamento do braço mecânico de um setor a outro de uma mesma trilha. Só depois que a cabeça de leitura/escrita se encontra sobre a trilha e o setor corretos é que é iniciada a transferência dos dados. Desse modo, o tempo necessário para que um disco complete uma solicitação de I/O é a soma de:

- **Tempo de busca (*seek time*) –** é o tempo necessário para que o braço se desloque da trilha atual para a trilha em que se encontra o dado solicitado;
- **Latência rotacional –** é o tempo necessário para que o braço se desloque do setor atual para o setor em que se encontra o dado solicitado;
- **Tempo de transferência –** é o tempo necessário para transferir eletronicamente os dados do disco para a controladora;

### 2.2.2. Padrões de Acesso a Dados

As solicitações de I/O podem acontecer basicamente de duas formas: randômica e seqüencial. Quando o a acesso aos dados se dá de forma randômica, o braço mecânico se move também de maneira randômica, o que aumenta o tempo de busca e limita o desempenho. Quando o acesso aos

dados é feito de maneira seqüencial, o tempo de busca é reduzido significativamente, o que permite uma taxa de processamento de I/O muito maior. Só para ter uma idéia do impacto que o padrão de acesso a dados tem sobre o desempenho do subsistema de I/O, considere a especificação de um disco SCSI moderno [10]:

**Tabela 1: Parâmetros de um disco SCSI**

| Tempos de Disco | milissegundos |
|---|---|
| Tempo de busca médio | 3,8 |
| Tempo de busca entre trilhas adjacentes | 0,3 |
| Latência rotacional | 2 |

Como pode ser visto, uma operação randômica necessita de 5,8 milissegundos, o que limita a taxa de I/O a um máximo teórico de aproximadamente 170 por segundo. Uma operação seqüencial necessita de apenas 2,3 milissegundos, o que proporciona uma taxa máxima de I/O de aproximadamente 435 por segundo – um desempenho duas vezes e meia melhor que o acesso randômico.

Na tentativa de reduzir o impacto das operações randômicas, muitas controladoras de disco implementam um algoritmo conhecido como ***elevator sorting***. Esse algoritmo consiste no reordenamento das solicitações de I/O pendentes em fila, visando à redução dos movimentos do braço mecânico.

### 2.2.3. RAID

Sistemas OLTP têm como característica o fato de exigirem bastante do subsistema de I/O. Por esse motivo, é comum o uso de múltiplos discos visando atender a essa grande demanda. RAID (*Redundant Array of Independent Disks*) é uma tecnologia que permite a junção de dois ou mais discos com o objetivo de oferecer mais capacidade de armazenamento, mais desempenho e/ou tolerância a falhas. Para o sistema operacional e para as aplicações de usuário, um *array* RAID é visto como um único disco lógico.

RAID pode ser implementado tanto via hardware quanto via software. A solução via hardware tem um desempenho superior por não consumir processamento do sistema principal, mas incorre em custos adicionais com a aquisição de equipamento dedicado.

Na tecnologia RAID existe o conceito de **níveis**, que correspondem às diversas maneiras de se organizar os discos. Cada nível tem características peculiares que o tornam mais ou menos adequados às necessidades de desempenho e tolerância a falhas. Vejamos então os principais níveis RAID:

- **RAID 0 (*striping*) –** consiste em dividir os dados em pedaços (*stripes*) e distribuí-los nos discos que compõem o *array*. Desse modo, um item de dado pode ser acessado em paralelo. A desvantagem é o fato de ele não prover tolerância a falhas, o que implica que se apenas um dos discos falhar, todo o *array* estará comprometido e os dados serão perdidos;
- **RAID 1 (*mirroring*) –** consiste em duplicar os dados de um disco em um outro. Dessa maneira, se um dos discos falhar, os dados não serão perdidos. Como toda solução de redundância, a desvantagem é a necessidade de dobrar os recursos sem ter nenhum ganho de capacidade;
- **RAID 5 –** nesse nível, como no RAID 0, os dados são distribuídos entre os discos, mas, além disso, ele provê tolerância a falhas por meio de bits de paridade, que são também espalhados entre os discos. A vantagem dessa técnica é que ela provê tolerância a falhas a um custo menor (para n discos, tem-se uma capacidade de n-1). A desvantagem fica pelo fator desempenho, já que uma operação de escrita implica na leitura e escrita tanto do dado quanto da paridade (uma operação de escrita incorre, portanto, em quatro operações de I/O);
- **RAID 0+1 –** é uma combinação dos níveis 0 e 1, provendo *striping* e redundância. Seu uso é recomendado quando se tem um grande volume de dados (o que inviabiliza o nível 1), é exigida tolerância a

falhas (o que inviabiliza o nível 0) e mais de 10 por cento das operações de I/O são de escrita (o que inviabiliza o nível 5).

## 2.3. SGBD

Um Sistema de Gerenciamento de Banco de Dados (SGBD) é uma coleção de programas que possibilita a definição, construção e manipulação de bancos de dados [4]. Definir um banco de dados envolve especificar os tipos de dados, as estruturas e as restrições para os dados que serão armazenados. Construir um banco de dados é o processo de armazenar, de alguma maneira, os dados. Manipular o banco de dados inclui funções como submissão de consultas, atualização dos seus dados ou geração de relatórios. As funções desempenhadas por um SGBD incluem:

- **Processamento e Otimização de Consultas –** uma consulta expressa em linguagem de alto nível como SQL deve, primeiro, ser examinada, analisada, validada e otimizada, para só então ser executada;
- **Processamento de Transações –** em um ambiente OLTP, a manipulação do banco de dados se dá por meio de transações. Uma transação é uma unidade lógica de processamento, formada por uma ou mais operações de acesso a dados. O SGBD deve garantir que as transações tenham um conjunto de propriedades conhecido como ACID (Atomicidade, Consistência, Independência e Durabilidade). Para tal, ele deve escalonar as transações, provendo controle de concorrência;
- **Recuperação de Falhas –** é responsabilidade de SGBD manter a consistência do banco de dados mesmo após falhas. Para fazer isso, o SGBD deve manter informações sobre as alterações executadas pelas transações em um arquivo de *log*. A partir do *log* é possível refazer ou desfazer os efeitos das transações.

# 3. ANÁLISE DE DESEMPENHO E PLANEJAMENTO DE CAPACIDADE

## 3.1. Introdução

A importância da análise de desempenho se estende desde o projeto até a aquisição e o uso dos sistemas. Algumas de suas aplicações incluem:

- Comparar sistemas entre si e definir qual é o mais adequado aos requisitos de desempenho impostos;
- Prever o desempenho do sistema frente a futuras mudanças na carga ou no próprio sistema;
- Identificar gargalos de desempenho;
- Propor mudanças de configuração de modo a obter um melhor desempenho (*tuning*).

Quaisquer pessoas associadas a um sistema, sejam elas engenheiros de software, analistas de sistemas, desenvolvedores ou gerentes de TI devem ser capazes de especificar de maneira clara os seus requisitos de desempenho. O objetivo desse capítulo é dar uma visão geral das técnicas e procedimentos comumente usados para a avaliação de desempenho e planejamento de capacidade de sistemas computacionais.

## 3.2. Noção de Sistema e de Modelo

É importante que antes que comecemos a falar de modelagem de sistemas, a noção exata de sistema e modelo esteja clara na mente do leitor.

### 3.2.1. Sistema

Segundo a definição do IEEE, um **sistema** é uma combinação de componentes que agem em conjunto para realizar uma função que não seria possível se feita por qualquer das partes individuais. No presente trabalho, o termo sistema será empregado para denotar o conjunto de elementos de hardware e software utilizados para o processamento de uma carga OLTP.

### 3.2.2. Modelo

Um modelo é algo que simula o comportamento de um sistema. É uma abstração de algo real. A modelagem matemática consiste em definir o conjunto de variáveis de entrada e variáveis de saída associadas a um sistema, bem como estabelecer o relacionamento entre elas. Um modelo na maioria das vezes aproxima o comportamento do sistema real. A qualidade do modelo é dada justamente por quão bem ele prevê a resposta do sistema (variáveis de saída) a um estímulo (variáveis de entrada). A figura abaixo ilustra o relacionamento entre sistema e modelo:

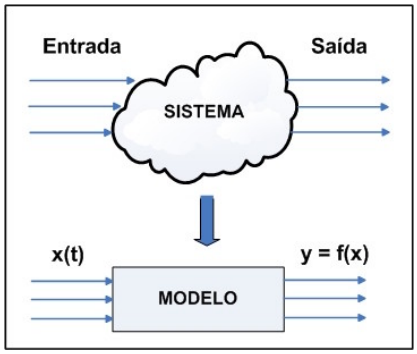



Figura 4.: sistema e modelo.

## 3.3. Uma Abordagem para Análise de Desempenho

Embora cada estudo de desempenho seja um caso particular, com suas próprias técnicas e métricas, existem procedimentos comuns, que podem ser empregados na maioria das situações no intuito de sistematizar o estudo. Em [6] é sugerida uma abordagem para a condução de estudos de desempenho. A seguir são mostrados os principais passos dessa abordagem:

1. **Definição de objetivos e do sistema –** consiste em definir exatamente o que será estudado e em que nível. Por exemplo, se o objetivo é mensurar o efeito do processador em um ambiente OLTP, o sistema será todo o hardware e software envolvidos. No entanto, se o objetivo é saber, dentro de uma CPU, qual ALU (Unidade Lógica e Aritmética) é a mais eficiente, o sistema em estudo será a CPU;

2. **Escolha de métricas –** consiste em definir quais critérios serão utilizados para medir o desempenho do sistema. Exemplos de métricas incluem: tempo de resposta, throughput, taxa de erros, entre outros;
3. **Listagem de parâmetros e seleção de fatores –** consiste em identificar quais parâmetros afetam o desempenho do sistema e em seguida escolher dentre eles aqueles que sofrerão variação durante o estudo (os fatores);
4. **Escolha da técnica de avaliação –** existem basicamente três técnicas usadas para a avaliação de desempenho de sistemas: modelagem analítica, simulações e medições. A escolha de uma delas depende de uma série de fatores que serão discutidos na próxima seção;
5. **Caracterização da carga (*workload*) –** descrever o que (e de que modo) será requisitado do sistema. Por exemplo, em um sistema de banco de dados, a carga seria formada por um conjunto de transações que buscam e/ou atualizam dados;
6. **Análise e interpretação de dados –** nesse passo, são analisados os dados obtidos em medições e simulações. Mais importantes que a coleta de dados em si, são as conclusões que o analista tira a partir deles;
7. **Apresentação de Resultados –** por fim cabe ao engenheiro de desempenho comunicar de forma clara os resultados obtidos durante o estudo.

Além dos passos citados acima, vale citar a importância de validar os resultados obtidos. Em geral, deve-se comparar os resultados obtidos usando uma técnica (ex. simulação) com os obtidos com uma outra (ex. modelagem). Nas próximas seções veremos em mais detalhes alguns conceitos apresentados aqui.

## 3.4. Técnicas para Análise de Desempenho

A escolha da técnica adequada é um passo fundamental num projeto de avaliação de desempenho. Como mencionado na seção anterior, existem três

técnicas e a escolha de uma delas é determinada por alguns fatores como custo, tempo ou precisão:

- **Modelagem Analítica –** consiste em desenvolver um modelo matemático que represente um sistema com suas respectivas entradas e saídas. Pode ser usada para prever o desempenho de sistemas ainda em fase de projeto. Quando usa essa técnica, o analista faz uso de um bom número de suposições e simplificações, o que normalmente tem conseqüências negativas no nível de precisão dos resultados. Em contrapartida, essas limitações são compensadas pelo fato de a modelagem analítica exigir pouca instrumentação e pouco tempo;
- **Simulação –** nessa técnica são desenvolvidos modelos de simulação os quais consistem basicamente em programas que representam o comportamento de sistemas reais. Esses modelos podem ser implementados tanto em linguagens de propósito geral como em linguagens específicas de simulação. Por empregar um nível maior de detalhes, o uso de simulações oferece resultados mais precisos que a modelagem analítica. No entanto, o desenvolvimento de modelos de simulação não é uma tarefa fácil, visto que requer um bom nível de conhecimento do sistema em estudo, e das linguagens e ferramentas utilizadas;
- **Medições –** consiste em monitorar o sistema com o intuito de obter dados de desempenho. Diferentemente das outras duas, essa técnica só pode usada quando o sistema objeto de estudo já existe. O tempo necessário para realizar as medições varia muito e não deve ser subestimado. Além do mais, essa costuma ser a técnica mais custosa, pois requer equipamentos e instrumentos reais. Apesar das desvantagens citadas, as medições podem produzir resultados precisos se o ambiente for configurado corretamente e não exigem um conhecimento muito profundo do sistema. Outra vantagem do uso de medições é o fato de os seus resultados serem mais facilmente aceitos, visto que eles são obtidos diretamente a partir de um sistema real.

Como regra geral, recomenda-se o uso de mais de uma técnica durante um estudo de desempenho de modo que os resultados obtidos a partir de uma delas sejam validados pelos obtidos com a outra. Nesse trabalho será utilizada a modelagem analítica para a projeção de ganhos ou perdas de desempenho em função de mudanças no sistema e medições para validar os resultados.

## 3.5. Seleção de Métricas

Métricas podem ser agrupadas em três categorias: **reação, produtividade e utilização**. A primeira diz respeito ao tempo que o sistema leva para executar o serviço. A segunda se refere à taxa de execução dos serviços. A última está relacionada à utilização de um recurso.

Para cada estudo de desempenho, um conjunto de métricas deve ser selecionado. A escolha dever ser feita levando em consideração os objetivos do estudo e as características do sistema. Por exemplo, se o objetivo é analisar qual a capacidade de processamento de pacotes de um roteador, duas métricas apropriadas seriam a taxa de pacotes processadas por unidade de tempo, e o tempo médio de processamento por pacote.

A seguir, são mostradas algumas das métricas mais utilizadas na avaliação de sistemas computacionais:

- ***Throughput* (taxa de serviço) –** é a taxa em que as requisições são processadas por um sistema. Ex. **MIPS** (milhões de instruções por segundo) ou **FLOPS** (instruções de ponto flutuante por segundo), para CPUs; **transações por segundo**, para sistemas OLTP; **bps** (bits por segundo), para redes de comunicação, etc.;
- **Tempo de resposta –** corresponde ao intervalo de tempo entre o momento em que uma requisição é submetida ao sistema e o momento em que o sistema responde a requisição;
- **Utilização –** porcentagem do tempo em que os recursos de um sistema encontram-se ocupados. O recurso com a maior utilização é dito ser o **gargalo** do sistema;

- **Taxa de erros –** indica a freqüência com que o comportamento do sistema se apresenta fora do esperado;
- **Disponibilidade –** a porção de tempo em que o sistema fica à disposição dos usuários para atender às suas requisições;

## 3.6. Caracterização de Carga de Trabalho

O processo de caracterização de carga de trabalho é sem dúvida uma parte crucial em um projeto de avaliação de desempenho. Resultados incorretos podem ser obtidos se a carga não for descrita em conformidade com a realidade observada. Para que isso não ocorra, convém seguir uma metodologia na hora de caracterizar a carga de trabalho:

1. **Definir a perspectiva da análise –** definição do que será objeto de estudo (ex. em um ambiente OLTP é comum a existência de servidores Web intermediários entre os clientes e o servidor principal. Se eles devem ou não ser incluídos na análise depende do objetivo do estudo);
2. **Identificar os componentes básicos –** consiste em definir quais são as transações ou requisições que compõem a carga;
3. **Definir o conjunto de parâmetros que melhor descrevem a carga –** definição de características de intensidade (ex. número de clientes, tempo entre interações, etc.) e demanda (ex. quanto uma requisição exige de CPU ou de I/O) da carga;
4. **Monitorar o sistema e coletar dados –** consiste na monitoração do sistema por um certo intervalo de tempo, coletando dados que permitam obter os valores dos parâmetros previamente definidos;
5. **Particionar a carga –** consiste em dividir a carga em classes homogêneas. Os componentes pertencentes a uma mesma classe compartilham alguma característica particular (como modo de processamento, grau de utilização ou padrão de acesso ao recurso etc.);

6. **Construir o modelo de carga –** consiste na caracterização de cada um dos parâmetros. Diversas técnicas podem ser utilizadas: uso de valores médios, especificação de medidas de dispersão, uso de histogramas, etc.

## 3.7. Benchmarking

Com a evolução dos computadores, devida em grande parte a mudanças de arquitetura, a comparação entre sistemas feita meramente com base em suas especificações técnicas vem mostrando-se inadequada. Um exemplo bastante atual é o fato de processadores AMD Athlon obterem desempenho similar ao de processadores Intel Pentium 4, mesmo funcionando a freqüências mais baixas. Além disso, um sistema pode ser mais eficiente que outro na execução de um conjunto de operações, mas ter um desempenho pior em um outro conjunto de operações. Por esses e outros motivos, faz-se necessário a definição de mecanismos que permitam a comparação entre dois sistemas quanto a sua capacidade em realizar determinadas operações. É nesse contexto que surge o conceito de *benchmarking*.

Segundo [6], ***benchmarking*** é o processo de comparação de desempenho entre dois ou mais sistemas. A carga de trabalho usada na comparação é chamada de ***benchmark***. O termo *benchmark* é também usado para designar o programa ou conjunto de programas que geram a carga. *Benchmarks* são projetados para testar características particulares dos sistemas, como por exemplo, o poder de processamento ou o desempenho gráfico ou ainda a capacidade do subsistema de I/O. Os *benchmarks* costumam ser classificados em ***sintéticos*** e ***de aplicação***. *Benchmarks* sintéticos são programas que executam um conjunto reduzido de instruções, testando componentes muito específicos de um sistema. *Benchmarks* de aplicação simulam um ambiente real, dando uma melhor noção do desempenho do sistema como um todo.

O nascimento do *benchmarking* se deu no começo da década de 80. Naquela época, os processadores eram os componentes mais caros dos sistemas, de maneira que se costumava denotar o desempenho do sistema

como o desempenho do próprio processador. Foram desenvolvidos, então, os primeiros programas sintéticos, com o intuito de testar a capacidade de processamento dos sistemas. Exemplos dessa primeira geração de *benchmarks* incluem o **Dhrystone** (desempenho aritmético com inteiros), **Whetstone** (desempenho aritmético de ponto flutuante) e **Linpack** (operações com sistemas de equações lineares). No entanto, com o passar dos anos ficou claro que não era suficiente medir o desempenho somente do processador. Além do mais, era comum os fabricantes realizarem otimizações nos compiladores específicas para os conjuntos de instruções desses *benchmarks*, o que gerava resultados distantes dos obtidos em aplicações do mundo real. Essa falta de credibilidade ocasionou o surgimento das primeiras organizações de *benchmark*, entidades neutras, que normalmente têm como membros os grandes fabricantes de hardware e software. Dentre elas, as mais conhecidas são a **SPEC** (*Standard Performance Evaluation Corporation*), e a **TPC** (*Transaction Processing Performance Council*). A SPEC é responsável por diversos *benchmarks*, voltados para várias áreas, dentre elas desempenho de CPU, de servidor web, de servidor de banco de dados, de aplicações cientificas, etc. Já a TPC mantém *benchmarks* voltados ao desempenho de sistemas de bancos de dados. Essas organizações ajudaram a recuperar a credibilidade dos *benchmarks*, visto que a publicação de resultados é sujeita a rígidas regras, passando, inclusive, por procedimentos de auditoria.

## 3.8. Planejamento de Capacidade

Planejamento de capacidade consiste em garantir que uma quantidade suficiente de recursos estará disponível para atender a condições futuras de carga de trabalho.

O primeiro passo no processo de planejamento de capacidade é ***instrumentar o sistema com monitores*** com o intuito de coletar dados de desempenho que permitam ***caracterizar a carga de trabalho atual***. O passo seguinte é ***projetar a carga de trabalho no futuro***. Isso pode ser feito observando o sistema por um longo período de tempo, na procura de padrões que descrevam a evolução da carga. Essa projeção é então alimentada como

entrada em um ***modelo desenvolvido para prever o desempenho do sistema sob diversas configurações***. O último passo, conhecido como ***sizing***, consiste em escolher a melhor configuração dentre as testadas.

# 4. FUNDAMENTOS DE MODELAGEM DE SISTEMAS

## 4.1. Introdução à Teoria das Filas

A teoria das filas é uma técnica de modelagem analítica bastante utilizada para a avaliação de desempenho de sistemas computacionais. O cenário de teoria das filas pode ser descrito desse modo: um recurso (também chamado de **centro de serviço**) é compartilhado por um conjunto de clientes (ou *jobs*) que de tempos em tempos vão a esse recurso para receberem seu serviço. Se ao solicitar o serviço o cliente encontrar o recurso disponível, ele será imediatamente atendido. Caso contrário, se o recurso já estiver atendendo outro(s) cliente(s) no momento da solicitação, ele deverá aguardar em uma fila. Cenários como esse são muito comuns dentro e fora da computação, o que faz com que a teoria das filas tenha aplicações nas mais diversas áreas, como, redes de comunicação, sistemas telefônicos e, claro, avaliação de desempenho de sistemas computacionais.

Nas próximas seções serão mostrados os conceitos fundamentais de teoria das filas: sua notação, suas aplicações e suas limitações.

### 4.1.1. Notação

A notação hoje amplamente utilizada para descrever filas foi criada por Kendall e é composta de seis parâmetros:

A/S/m/B/K/SD

- A (***A**rrival Process*) **–** o processo de chegada das requisições ao centro de serviço;
- S (***S**ervice Time Distribution*) **–** a distribuição que determina tempo que o centro precisa para atender as requisições;
- m **–** o número de servidores, que são considerados parte de um mesmo sistema de filas caso sejam todos idênticos;

- B (***B****uffer*) – o número máximo de clientes que o sistema aceita. Esse número pode ser limitado por questões de espaço ou para limitar o tempo de espera;
- K – o número de clientes que podem porventura vir a solicitar serviço do sistema;
- SD (***S****ervice* ***D****iscipline*) – A ordem em que os clientes são atendidos. Alguns exemplos são FIFO (*First-In, First-Out* – primeiro a chegar, primeiro a ser servido), LIFO (*Last-In, First-Out* – último a chegar, primeiro a ser servido), Round Robin (cada cliente recebe uma mesma quantidade de tempo de serviço no centro), dentre outros;

Os processos de chegada e de serviço são representados por uma letra, que indica qual a distribuição que eles seguem:

- M – Exponencial;
- $M^{[x]}$ – Exponencial em rajadas;
- $E_k$ – Erlang, com parâmetro k;
- $H_k$ – Hiper-exponencial, com parâmetro k;
- D – Determinístico;
- G – Geral;

A letra M, usada para indicar a distribuição exponencial, vem de "*memoryless*" ou de "Markovian", para indicar que valores assumidos no passado não influenciam nos valores futuros. Quando vem acompanhado por um sobrescrito, significa que as chegadas (ou serviços) acontecem em rajadas, isto é cada chegada (ou serviço) consiste, na realidade, em um grupo de requisições. Nesse caso, x representa o tamanho do grupo, que também é uma variável aleatória. A distribuição determinística denota um valor constante. Uma distribuição geral significa que não se conhece exatamente qual a distribuição, e o resultado é válido para qualquer distribuição.

Como um exemplo, **M/M/2/50/5000/FIFO**, indica um sistema de fila no qual os tempos de chegada e de serviço são exponencialmente distribuídos, tem 2 servidores, e que suporta um máximo de 50 requisições, de uma população total de 5000 clientes, servidos em um regime de primeiro a chegar, primeiro a sair.

É comum abreviar a notação, e considerar que não existe limitação de buffer, a população de clientes é infinita e a política de serviço é FIFO. Assim, costuma-se escrever apenas os três primeiros parâmetros para representar uma fila. Os tipos mais comuns de filas são os seguintes:

- M/M/1 – esse tipo de fila é usado para representar sistemas de um único processador ou qualquer outro recurso individual;
- M/M/m – esse tipo de fila pode ser usado para modelar centros de serviço compostos por vários dispositivos idênticos, como por exemplo, um subsistema de I/O com vários discos. O sistema é formado por m servidores, cada um com uma taxa de serviço μ, atendendo a clientes que chegam a uma taxa λ. Se em um dado momento o número de clientes n no sistema for menor que m, um novo cliente que chegue será imediatamente atendido. Somente se todos os m servidores estiverem ocupados o novo cliente será posto em fila;

### 4.1.2. Lei de Little

A lei de Little é uma das mais importantes leis da teoria das filas. Ela estabelece que o número médio de requisições em um sistema é igual ao produto entre a taxa de chegada e o tempo médio que a requisição passa no sistema.

$$E[N] = \lambda \times E[R]$$

Apesar de ser uma lei intuitivamente razoável, trata-se de um resultado excepcional, se considerarmos que ele é válido para a grande maioria das situações, não sendo necessário nenhum conhecimento

adicional sobre os processos de chegada e saída. A única exigência é que o número de requisições que chegam ao sistema seja igual ao número de requisições que têm seus serviços completados. E isso implica que o throughput (taxa de serviço) do sistema seja igual à taxa de chegada, donde tiramos:

$$E[N] = X \times E[R]$$

Pode-se ainda obter a partir da Lei de Little outros relacionamentos:

$$E[N_q] = \lambda \times E[w_q]$$

$$E[N_S] = \lambda \times E[s]$$

Onde:

$E[N_q]$ – Número médio de requisições na fila de espera;

$E[w_q]$ – Tempo médio de espera na fila;

$E[N_s]$ – Número médio de requisições recebendo serviço;

E[s] – Tempo médio de serviço;

Outra característica interessante da Lei de Little é o fato de ela poder ser aplicada nos mais diferentes níveis de um sistema, desde o nível de um único recurso, até o nível de todo o sistema.

Como exemplo da aplicação da Lei de Little, suponha que um disco tenha sido monitorado por um tempo e que se tenha verificado que a taxa de chegada seja de 150 requisições por segundo e que o tempo médio de serviço seja de 10 milisegundos. É possível determinar o número médio de requisições no disco como:

E[N] = 150 * 0,01 = 1,5 requisições

## 4.2. Sistemas Computacionais e Redes de Filas

Na seção anterior, foram vistas as características de sistemas modelados como uma única fila. No entanto, a maioria dos sistemas

computacionais consiste em um conjunto de centros de serviço (filas), pelos quais as requisições vão passando. Esse tipo de modelo, no qual uma requisição sai de uma fila e entra em outra é conhecido como **rede de filas**.

Tradicionalmente, as redes de filas são classificadas como abertas ou fechadas. Em uma **rede de filas aberta**, as requisições entram no sistema, são processadas, e em seguida deixam o sistema. Isso significa que o número de requisições presentes no sistema varia com o tempo. Em uma **rede de filas fechada**, as requisições não entram nem deixam o sistema, mas ficam circulando de uma fila a outra. O número de requisições é, portanto, constante.

Para caracterizar uma rede de filas, é preciso definir três elementos: as requisições (clientes), os centros de serviço e as demandas. As demandas dos recursos podem ser obtidas por meio de medições, conforme será mostrado mais adiante. A seguir veremos em mais detalhes os outros dois elementos.

### 4.2.1. Descrição dos Clientes

A intensidade da carga de trabalho pode ser descrita de três maneiras, dependendo das características dos clientes:

- **Carga de transação –** nesse tipo de carga, os clientes entram e saem do modelo, e a intensidade é dada pela taxa de chegada λ dos mesmos;
- **Carga batch –** nesse tipo de carga, o número de clientes N permanece constante, e a intensidade é dada justamente por N. Quando uma requisição é processada, outra imediatamente entra no sistema para ser processada;
- **Carga de terminal –** nesse tipo de carga, o número de clientes N também é constante, mas a intensidade é dada por N e também por um parâmetro adicional Z, que indica o tempo médio entre duas solicitações de um mesmo cliente (esse parâmetro é conhecido como *thinking time*).

Note que cargas de transação são modeladas como redes de filas abertas, enquanto cargas batch e de terminal são modeladas como redes de filas fechadas.

### 4.2.2. Descrição dos Centros de Serviço

Os centros de serviço podem ser de três tipos:

- **Centros de serviço com capacidade fixa (FCSC – *Fixed-Capacity Service Centers*) –** o tempo de serviço desse tipo de centro não varia em função do número de requisições (obviamente o tempo total que a requisição passa no centro cresce com o tamanho da fila);
- **Centros de serviço dependentes de carga (*Load-Dependent Service Centers*) –** nesse tipo de centro, a taxa de serviço depende do número de requisições pendentes (ou seja, da carga). Um exemplo são as filas M/M/m, nas quais a taxa de serviço cresce até que todos os m servidores estejam ocupados;
- **Centros de Delay –** são centros em que não existe fila. Qualquer requisição que nele chegue é imediatamente atendida. São utilizados para modelar recursos em que não existe competição, como por exemplo, terminais.

### 4.2.3. Modelos de Redes de Filas e Teoria das Filas

A modelagem de sistemas computacionais por meio de redes de filas usa apenas um subconjunto dos conceitos e técnicas disponíveis em Teoria das Filas. Grande parte dos trabalhos publicados em Teoria das Filas foca na modelagem de um único centro de serviço com características complexas. Nesses casos, são empregadas técnicas sofisticadas e medições mais precisas (utilizando distribuições em vez do comportamento médio).

No caso de redes de filas, normalmente se utilizam centros de serviços mais simples, sobre os quais são feitas algumas suposições e/ou simplificações. A vantagem aqui é que não é necessário tomar conhecimento de características complexas do sistema (conhecimento esse que muitas vezes

não está disponível por falta, por exemplo, de ferramentas adequadas). Veremos na Seção 4.3. como muitos dos problemas de Teoria das Filas podem ser resolvidos com algumas regras simples.

### 4.2.4. Limitações e Aplicabilidade de Teoria das Filas

Existe uma série de características dos sistemas computacionais que não podem ser facilmente modeladas utilizando Teoria das Filas. A seguir uma lista delas:

- **Tempo de serviço não-exponencial –** a maioria dos modelos propostos em Teoria das Filas assume que os tempos de serviços são distribuídos exponencialmente. Quando essa não é a realidade do sistema que está sendo modelado, a Teoria das Filas mostra-se inadequada;
- **Chegadas em rajadas –** não é possível modelar sistemas cujas requisições cheguem em rajadas, a menos que os tempos de chegada dos grupos estejam distribuídos exponencialmente;
- **Sincronização –** a Teoria das Filas não dispõe de ferramental para lidar com questões de sincronização entre processos;
- **Condições de Disputa e Acesso exclusivo (bloqueio) –** condições de disputa por um recurso ferem o princípio da independência entre requisições, essencial para a modelagem com filas;
- **Regime de chegada dependente de carga –** atualmente, boa parte dos sistemas realiza balanceamento de carga. Esse comportamento é difícil de ser modelado;
- **Uso simultâneo de mais de um recurso –** é muito difícil representar o fato de uma requisição precisar de mais de um recurso ao mesmo tempo.

Diante de tantas limitações, pode parecer à primeira vista que a Teoria das Filas não tem aplicação prática para a análise de desempenho de sistemas computacionais. No entanto, a verdade é que boa parte das características

mencionadas há pouco são captadas implicitamente durante o processo de parametrização do sistema. Por exemplo, quando medimos a utilização da CPU estamos incluindo não somente o tempo que ela passa realizando trabalho útil, mas também o tempo em que os processos aguardam por sincronização ou acesso exclusivo. Desse modo, indiretamente, o modelo incorpora aspectos de sincronização e bloqueio, produzindo, no fim das contas, um bom resultado. É claro que a intenção pode ser, por exemplo, fazer um estudo do efeito de bloqueios no desempenho do sistema. Em casos como esse, é necessário enriquecer o modelo com procedimentos que permitam representar as características consideradas relevantes para o estudo. De maneira geral, a aplicabilidade da Teoria das Filas depende de quais características e de que nível de detalhes se deseja representar.

## 4.3. Leis Operacionais

Muitos problemas de teoria das filas podem ser resolvidos por meio de algumas relações simples, sem que para isso seja necessário um conhecimento exato sobre as distribuições dos tempos de chegada e de serviço. Nessa seção, serão mostradas algumas dessas relações, conhecidas como **leis operacionais**.

Seja um sistema modelado como um único centro de serviço (isto é, o sistema consiste em um único recurso). Se observarmos esse sistema por um determinado período de tempo, podemos mensurar as seguintes quantidades:

T – o tempo total de observação do sistema;

A – o número total de requisições que chegaram ao sistema;

C – o número total de requisições processadas pelo sistema;

B – o total de tempo em que o sistema encontra-se ocupado atendendo às requisições;

Esses valores podem ser obtidos diretamente, por meio de simples monitoração do sistema – daí o termo ***operacional***. Eles servem de variáveis independentes em equações que permitem estabelecer um conjunto adicional de quantidades:

λ – a taxa de chegada de requisições. λ = A/T;

X – o *throughput* ou taxa de processamento. X = C/T;

U – utilização do sistema. U = B/T;

S – o tempo de serviço médio de uma requisição. S = B/C;

### 4.3.1. Lei da Utilização

Perceba que podemos ainda expressar a utilização do recurso de uma outra maneira, como:

$$U = \frac{B}{T} = \frac{B}{C} \times \frac{C}{T} \Rightarrow$$

$$\mathbf{U = X \times S}$$

Essa relação é chamada de **Lei de Utilização**.

### 4.3.2. Lei do Fluxo Forçado

Quando a Lei de Little foi apresentada na Seção 4.1, foi salientado o fato de que ela pode ser aplicada nos mais diferentes níveis de um sistema. O conceito de “requisição” varia de acordo com o nível que estamos focando. Por exemplo, ao nível dos discos, uma requisição é uma operação de I/O. Quando se considera o sistema OLTP como um todo, uma requisição é uma solicitação de processamento de transação, que envolve todos os componentes do sistema. A relação entre os diferentes níveis de um sistema, é expressa pela Lei do Fluxo Forçado (*Forced Flow Law*), que determina que os fluxos em todas as partes do sistema mantêm uma relação de proporcionalidade entre si.

Se observarmos o processamento das requisições não somente no nível do sistema (doravante chamadas de transações), mas também no nível dos dispositivos que o compõem, poderemos estabelecer o número médio de visitas que uma transação faz a cada um dos dispositivos:

$$V_K = \frac{C_K}{C}$$

Isto é, o número de visitas $V_K$ por transação ao dispositivo k é igual à razão entre o número de requisições processadas pelo dispositivo k e o número total de requisições processadas pelo sistema. Se quisermos saber o throughput do dispositivo, teremos:

$$X_K = \frac{C_K}{T}$$

Da relação anterior, temos que:

$$C_K = V_K \times C$$

Logo:

$$X_K = V_K \times \frac{C}{T} \Rightarrow$$

$$\mathbf{X_k = X \times V_k}$$

Essa é a **lei do fluxo forçado**. O cálculo da utilização do dispositivo é simples e direto:

$$U_K = X_K \times S_K = X \times V_K \times S_K \Rightarrow$$
$$U_K = X \times D_K$$

$D_K$ é a demanda total de serviço de uma transação no dispositivo k. Define-se o ***gargalo*** do sistema como sendo o dispositivo com a maior demanda $D_K$. Para tornar mais claro em que situações essas leis podem ser aplicadas, vejamos um exemplo:

**Exemplo**: Um sistema submetido a uma carga OLTP é observado por um determinado período de tempo. Os dados obtidos durante a medição, revelam um throughput de 1500 transações por segundo. A demanda de CPU de uma transação é de 0,4 milissegundos. O sistema conta com dois discos. O disco 1 atendeu em média a 150 requisições de I/O por segundo e o disco 2 100 requisições por segundo. O tempo médio de serviço do disco 1 é de 5 milissegundos e o do disco 2 é de 7 milissegundos. Qual dispositivo é o gargalo do sistema?

$$U_{D1} = X_{D1} \times S_{D1} = 150 \times 0{,}005 = 75\%$$
$$U_{D2} = X_{D2} \times S_{D2} = 120 \times 0{,}007 = 84\%$$
$$U_{CPU} = X_{CPU} \times S_{CPU} = 1500 \times 0{,}0004 = 60\%$$

Assim, o disco 2 é o gargalo do sistema. Interessante notar que apesar de o disco 1 atender a mais requisições que o disco 2, a demanda – e conseqüentemente a utilização – desse último é maior por ele precisar de mais tempo para atender a uma requisição (tempo de serviço maior).

### 4.3.3. Conseqüências da Lei de Little

Observando-se um dispositivo do ponto de vista externo, o tamanho da fila de requisições corresponde, não somente ao às requisições em espera na fila do dispositivo, mas inclui também aquela recebendo serviço do mesmo. Dessa maneira, podemos reescrever a Lei de Little como:

$$Q_K = X_K \times R_K$$

Ora, o tempo de resposta é dado por:

$$R_K = S_K \times (1 + Q_K)$$

Em palavras, isso quer dizer que quando a requisição chega ao dispositivo ela precisa esperar pelo processamento das outras Q requisições (incluindo a que está recebendo serviço) para só depois ser processada. Como conseqüência, o tamanho da fila é dado por:

$$Q_K = X_K \times S_K \times (1 + Q_K) = U_K \times (1 + Q_K) \Rightarrow$$
$$Q_K = \frac{U_K}{(1 - U_K)}$$

Essa relação retrata o efeito direto da utilização sobre o tamanho da fila (e conseqüentemente sobre o tempo de resposta). A Figura 5 mostra que quando a utilização do dispositivo atinge certo ponto, o crescimento da fila torna-se exponencial. É por isso que se deve evitar manter os dispositivos constantemente próximos à sua capacidade máxima.

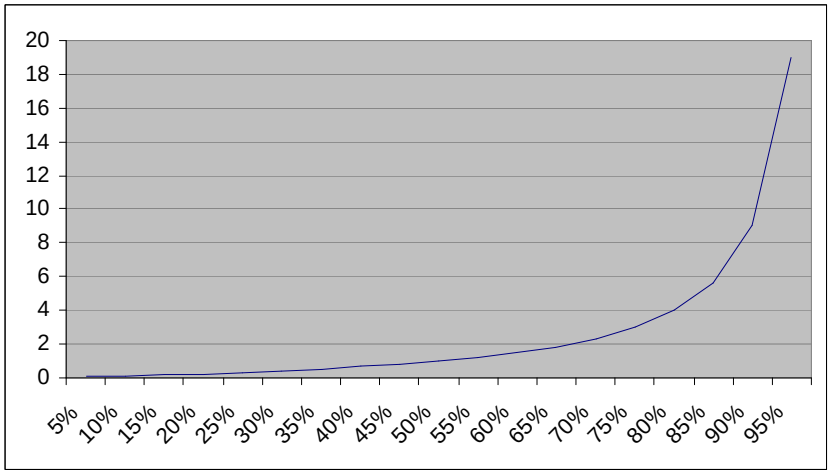



Figura 5.: Tamanho da fila vs. Utilização.

## 4.4. Estimativa de Parâmetros com Modelos de Regressão

Não é raro encontrar situações em que o comportamento de um parâmetro não pode ser facilmente caracterizado, tornando necessário o uso de modelos estatísticos. **Modelos de regressão** permitem estabelecer um relacionamento entre duas ou mais variáveis aleatórias. Eles são empregados para estimar o valor de uma variável como uma função de outras. A variável estimada é chamada de **variável de resposta** ou **dependente** e as variáveis usadas para estimar o valor da primeira são chamadas de **variáveis independentes** ou **fatores**.

O objetivo principal de um modelo de regressão é minimizar a diferença entre os dados amostrados e a curva que representa a função por ele estabelecida. Essa curva pode ser linear ou não. Para esse trabalho, o foco será nos modelos de regressão linear. Seja, então, o seguinte relacionamento linear entre duas variáveis aleatórias:

$$\hat{y} = \alpha + \beta x$$

Onde:

$\hat{y}$ é o valor previsto da variável de resposta para o fator x.

α e β são os parâmetros do modelo de regressão.

Dado um conjunto de pares $\{(x_1, y_1), ..., (x_n,y_n)\}$, o valor estimado de y para um certo $x_i$ é:

$$\hat{y}_i = \alpha + \beta x_i$$

E a diferença (erro) entre o valor previsto e o valor real de y é

$$e_i = y_i - \alpha - \beta x_i$$

Pode-se demonstrar que o melhor modelo possível, que mais se aproxima dos dados reais, é aquele que minimiza a soma dos quadrados dos erros (SSE):

$$SSE = \sum_{i=1}^{n} e_i^{\,2}$$

E que os parâmetros de regressão que minimizam esse erro são dados por:

$$\alpha = \overline{y} - \beta \overline{x}$$

$$\beta = \frac{\sum xy - n\overline{x}\,\overline{y}}{\sum x^2 - n\left(\overline{x}\right)^2}$$

Onde:

$\overline{x}$ é média das variáveis independentes

$\overline{y}$ é a média das variáveis de resposta

A principal medida de qualidade de um modelo de regressão é o **coeficiente de determinação**, $R^2$. Ele indica qual a fração da variação de y é explicada pelo modelo de regressão. Quanto maior o seu valor, melhor é o modelo. Ele pode ser calculado pela fórmula:

$$R^2 = \frac{SST - SSE}{SST}$$

Onde:

**SST** é a soma dos quadrados totais, que representa a variação dos valores $y_i$ para a sua média, sendo dada por:

$$SST = \sum_{i=1}^{n} \left( y_i - \overline{y} \right)^2 = \left( \sum_{i=1}^{n} y^2 \right) - n\left( \overline{y} \right)^2$$

**SSE** é a soma dos quadrados dos erros, mencionada há pouco, e é dada pela fórmula:

$$SSE = \sum y^2 - \alpha \sum y - \beta \sum xy$$

# 5. O BENCHMARK TPC-C

## 5.1. Visão Geral

O *benchmark* TPC-C é uma carga de trabalho, composta por um conjunto de cinco transações de leitura e escrita, que simula as atividades encontradas em um ambiente OLTP complexo[12]. O propósito do TPC-C é prover dados relevantes de desempenho que auxiliem os usuários a comparar de maneira objetiva dois ou mais sistemas quanto a sua capacidade de processar transações.

O ambiente simulado pelo TPC-C consiste em um sistema de processamento de transações usado para registrar as atividades de uma empresa atacadista que produz, vende e distribui seus produtos. Essa companhia encontra-se geograficamente distribuída, de maneira que suas vendas estão espalhadas por um número de armazéns (*warehouses*) e distritos (*districts*). O tamanho da organização é definido pelo número de armazéns que ela possui – conforme ela cresce mais armazéns e distritos precisam ser criados. Cada armazém mantém estoque para todos os cem mil produtos vendidos pela companhia e cada um dos seus distritos atende a três mil clientes (*customers*). A figura a seguir ilustra a hierarquia entre armazéns, distritos e clientes:

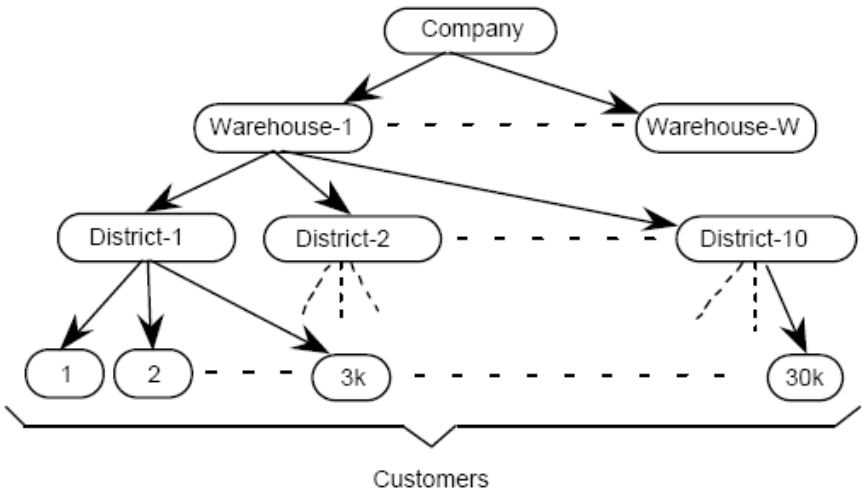



Figura 6.: Relação entre armazéns, distritos e clientes [12].

O sistema da companhia é usado para:

- Registrar solicitações de compras de clientes;
- Consultar a situação de pedidos feitos previamente;
- Registrar pagamentos dos clientes;
- Processar entregas de pedidos;
- Consultar o nível de estoque de produtos;

Como será mostrado na Seção 5.3, cada uma dessas ações corresponde a uma das transações que compõem a carga de trabalho do *benchmark*.

## 5.2. Layout do Banco de Dados

O banco de dados especificado pelo TPC-C é composto por nove tabelas. A figura abaixo mostra essas tabelas e os seus respectivos relacionamentos:

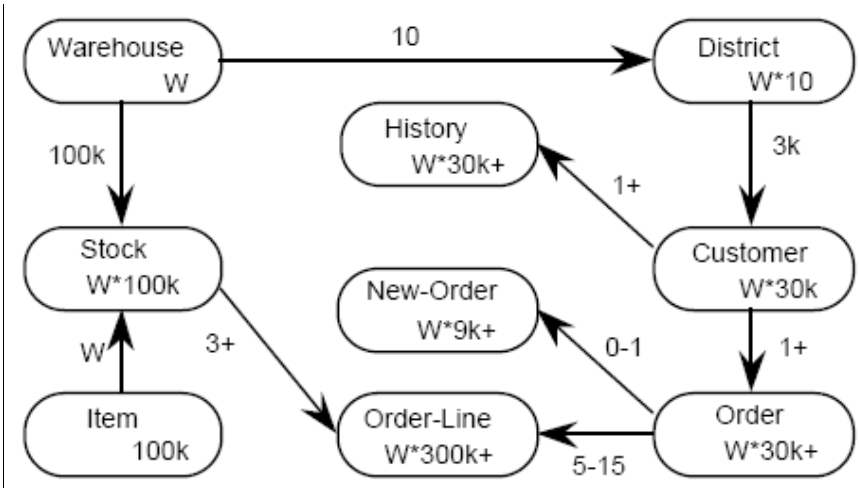



Figura 7.: Diagrama E-R do banco de dados TPC-C [12].

- Os números dentro das entidades indicam a cardinalidade das tabelas (número de linhas). Por exemplo, W indica que a tabela *warehouse* é populada com W itens de dados. A tabela *district*, por sua vez, tem 10*W linhas. Interessante notar que, excetuando-se a tabela *Item*, cuja cardinalidade é fixada em 100 mil linhas, todas as outras tabelas têm

suas cardinalidades proporcionais ao número de linhas da tabela *warehouse*;

- Os números próximos às linhas de relacionamento representam a cardinalidade dos relacionamentos;

## 5.3. As Transações

A carga de trabalho TPC-C é formada por cinco transações. Nas próximas seções elas serão brevemente descritas, destacando as operações executadas e os padrões de acesso a dados de cada uma delas.

### 5.3.1. New Order

A transação *New Order* consiste no registro de uma operação de compra de um cliente. As operações executadas incluem:

- Inserção de registros nas tabelas *Order*, *New Order* e *Order Line*;
- Para cada um dos itens que compõem a compra, atualização do nível de estoque (tabela *Stock*);
- Consulta de informações das tabelas *Warehouse*, *District*, *Customer* e *Item*;

É uma transação executada com uma freqüência alta, que impõe uma carga média ao sistema por meio de operações de leitura e de escrita. Como veremos mais adiante, trata-se da principal transação do *benchmark* TPC-C.

### 5.3.2. Payment

A transação *Payment* consiste no registro do pagamento de uma compra. As operações executadas incluem:

- Atualização de informações de saldo do cliente (tabela *Customer*), do distrito (tabela *District*) e do armazém (tabela *Warehouse*);
- Insere um novo registro na tabela *History*;

Trata-se de uma transação de peso leve, com alta freqüência de execução, composta por operações de leitura e escrita.

### 5.3.3. Delivery

A transação *Delivery* registra a entrega de um pedido. Ela consiste no processamento em batch de dez novos pedidos. Para cada distrito de um armazém, as seguintes operações são executadas:

- Busca e exclusão do registro mais antigo da tabela *New Order* que esteja associado ao distrito em questão;
- Atualização de registros nas tabelas *Order*, *OrderLine* e *Customer*;

Trata-se de uma transação de leitura e escrita, com baixa freqüência de execução, que impõe uma carga moderada ao sistema.

### 5.3.4. Order Status

A transação *Order Status* representa uma consulta da situação da última compra de um cliente. As operações executadas são as seguintes:

- Seleção de informações do cliente a partir da tabela *Customer*;
- Seleção de informações de compra a partir das tabelas *Order* e *Order Line*;

É uma transação somente-leitura, executada com baixa freqüência, que impõe uma carga moderada ao sistema.

### 5.3.5. Stock Level

A transação *Stock Level* é utilizada para determinar, dentre os itens vendidos recentemente, quais estão com o estoque abaixo de um determinado limite. Ela é composta das seguintes operações:

- Seleção das últimas vinte compras de um distrito (tabela *District)*;
- Contagem dos itens cujo estoque encontra-se abaixo do limite estabelecido (tabelas *Stock* e *Order Line*);

Trata-se de uma transação somente-leitura com baixa freqüência de execução, mas que impõe uma carga pesada ao sistema.

## 5.4. Ambiente de Execução

O ambiente de teste TPC-C é em constituído, em situações normais, de três elementos:

- ***System Under Test* (SUT) –** é o sistema alvo da carga. As métricas de desempenho reportadas ao fim de um teste TPC-C indicam a capacidade de processamento de transações do SUT;
- ***Web Clients* –** são os *front-ends*. Representam servidores HTTP, servindo de intermediários entre os usuários e o servidor de banco de dados;
- ***Remote Terminal Emulators* (RTE) –** são os emuladores de usuários, que geram carga sob a forma de chamadas às transações disponíveis no servidor de banco de dados.

A figura abaixo mostra a relação entre SUT, *web clients* e RTEs:

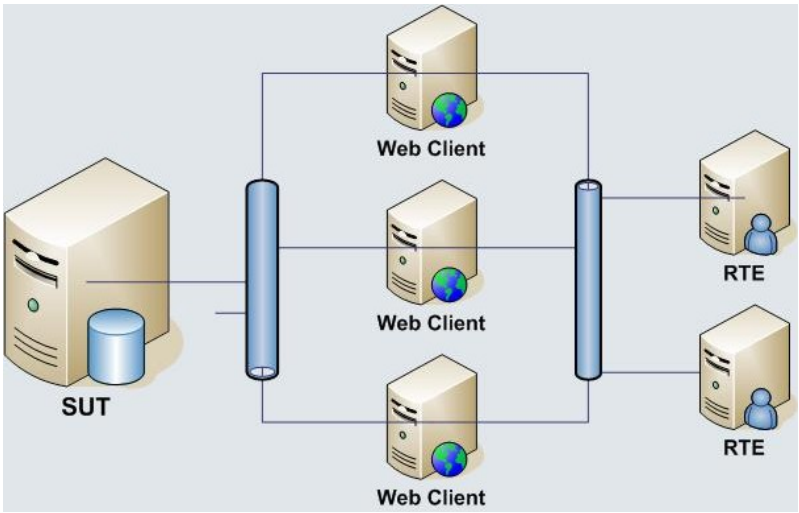



Figura 8.: Ambiente de Testes TPC-C

## 5.5. Métricas de Desempenho e Escala do Sistema

O *throughput* reportado ao final da execução de um teste TPC-C está diretamente relacionado ao número de usuários emulados que submetem transações ao sistema em teste (SUT) – a carga é proporcional ao número de usuários. Por sua vez, segundo as regras do *benchmark*, o número de usuários

acessando o sistema, deve corresponder a dez vezes o número de *warehouses* do banco de dados. Desse modo, **a cardinalidade da tabela *warehouse* é o fator de escala do benchmark TPC-C**, determinando a carga submetida ao sistema.

A métrica utilizada pelo TPC-C é o tpmC, que mede o número de transações do tipo *New Order* completadas por minuto. Apesar de a métrica refletir o throughput de apenas um único tipo de transação, o processamento das outras transações influencia no tempo de resposta da transação *New Order*, o que garante que o desempenho do sistema seja testado por meio de todas as transações.

A Figura 9 mostra o ciclo de operações executado por cada usuário emulado pelo RTE:

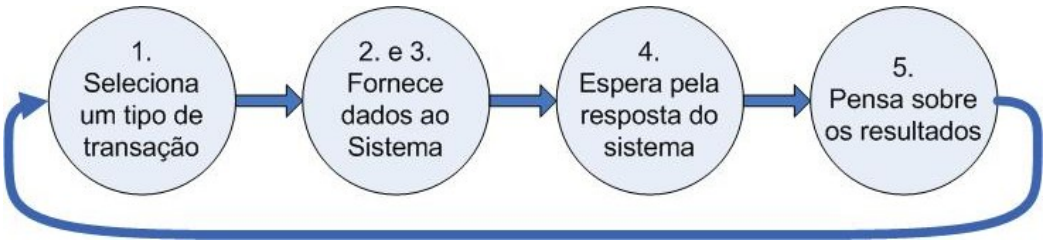



Figura 9.: Ciclo de execução dos usuários emulados.

1. Seleciona um tipo de transação dentre as cinco disponíveis. A escolha da transação a ser requisitada é feita de acordo com a distribuição mostrada na Tabela 2;
2. Espera pelo atraso do menu (***Menu Delay***);
3. Fornece os campos de entrada necessários à execução da transação (***Keying Time***);
4. Espera pela resposta do sistema (***Response Time***);
5. Pensa a respeito dos resultados retornados pelo SUT (***Thinking Time***).

Assim, o tempo total gasto para que uma solicitação de um usuário seja completada (*Completion Time*) é dado por:

$$CT_{txn} = MT_{txn} + KT_{txn} + RT_{txn} + TT_{txn}$$

Os tempos MT, KT e TT têm seus valores mínimos especificados para cada transação segundo a Tabela 3 (obviamente, o valor RT depende da capacidade do sistema).

**Tabela 2: Distribuição de Execução das Transações**

| Transação | % Mínima |
|---|---|
| New Order | - |
| Payment | 43,0 |
| Delivery | 4,0 |
| Order Status | 4,0 |
| Stock Level | 4,0 |

**Tabela 3: Menu/Keying/Thinking Times das Transações**

| Transação | MT | KT | TT |
|---|---|---|---|
| New Order | 0,1 | 18 | 12 |
| Payment | 0,1 | 3 | 12 |
| Delivery | 0,1 | 2 | 5 |
| Order Status | 0,1 | 2 | 10 |
| Stock Level | 0,1 | 2 | 5 |

Essas regras impostas pelo *benchmark* impõem um limite teórico ao número máximo de tpmC que pode ser obtido com um certo número de *warehouses*.

Seja:

- tpmc = número de transações do tipo *New Order* completadas por minuto;
- X = número de transações completadas por minuto;
- U = número de usuários;

O *throughput* X do sistema é dado pelo inverso da média dos tempos de serviço de todas as transações, multiplicado pelo número de usuários. O tpmc é obtido multiplicando a probabilidade de escolha da transação New Order pelo *throughput*:

$$X = \frac{60}{AVG(CT_{TXN})} \times U$$

$$tpmC = P_{NO} \times X \times U$$

O limite teórico corresponde à situação ideal na qual o sistema responde instantaneamente às requisições dos usuários ($RT_{txn}$=0). Logo:

$CT_{NO} = MT_{NO} + KT_{NO} + RT_{NO} + TT_{No} = 0,1+18+0+12 = 30,1$

$CT_{P} = MT_{P} + KT_{P} + RT_{txn} + TT_{P} = 0,1+3+0+12 = 15,1$

$CT_{D} = MT_{D} + KT_{D} + RT_{D} + TT_{D} = 0,1+2+0+5 = 7,1$

$CT_{OS} = MT_{OS} + KT_{OS} + RT_{OS} + TT_{OS} = 0,1+2+0+10 = 12,1$

$CT_{SL} = MT_{SL} + KT_{SL} + RT_{SL} + TT_{SL} = 0,1+18+0+12 = 7,1$

$$AVG(CT_{TXN}) = \sum P_{TXN} \times CT_{TXN} = 21,09 \Rightarrow$$

$$X_{\max} = \frac{60}{21,09} \times U \Rightarrow$$

$$tpmC_{\max} = P_{NO} \times X_{\max} = 0,45 \times \frac{60}{21,09} \times U \cong 1,28 \times U$$

*Como U = 10 * W,*

***tpmc$_{max}$ = 12,8 * W***

Ou seja, para um fator de escala de W *warehouses*, o máximo tpmC que pode ser obtido é de 12,8 * W.

# 6. MODELAGEM DO AMBIENTE

Considerando-se todo o ambiente, formado por geradores de carga e servidor de banco de dados, o *workload* TPC-C é classificada como uma carga de terminal, em que o número de usuários se mantém constante, e esses “pensam” entre interações consecutivas. No entanto, do ponto de vista do SUT, o número de usuários é variável, o que o caracteriza como uma carga de transação. A figura abaixo ilustra esse fato e mostra o modelo que será utilizado:

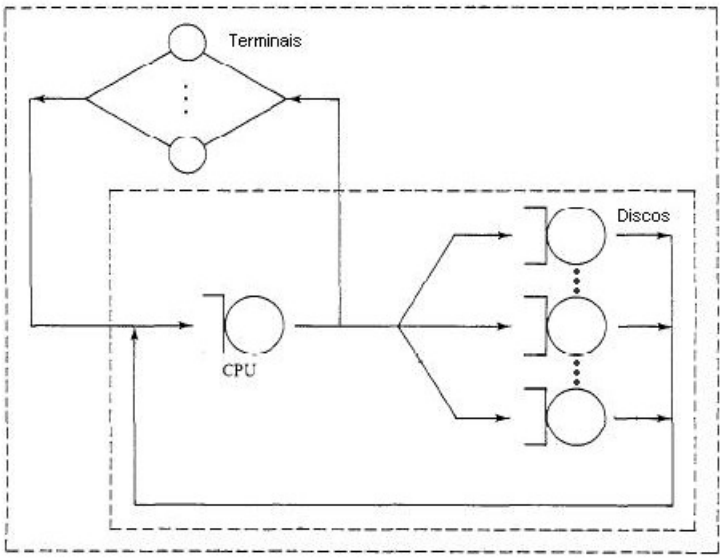



Figura 10.: Rede de Filas representando o SUT e os terminais.

O SUT será modelado como uma rede de filas aberta (***open queueing network***), com dois tipos de centros de serviço: CPU e disco. Uma solicitação que chega ao SUT entra na fila de CPU. Se necessitar de dados armazenados em memória secundária, ela será passada à fila dos discos. Após sair do disco, ela volta à CPU para processamento adicional, para só então deixar o sistema.

Serão estudados os efeitos de três fatores sobre o desempenho do sistema:

- Poder de CPU;
- Tamanho da memória;

- Quantidade de discos.

## 6.1. Parametrização

A parametrização do modelo consiste basicamente em definir as demandas de recursos das transações que compõem a carga de trabalho. Alguns parâmetros podem ser obtidos diretamente a partir de medições feitas durantes experimentos. Outros precisam ser induzidos a partir desses últimos por meio de leis operacionais.

Nas próximas seções serão descritos os experimentos e os resultados neles obtidos. No fim do capítulo, o modelo será validado comparando a saída por ele produzida com as obtidas em experimentos executados em configurações distintas daquelas utilizadas para parametrizar o sistema.

### 6.1.1. Descrição dos Experimentos

Para obter a demanda por recursos da carga TPC-C, foi necessária a realização de experimentos, durante os quais eram coletados dados de desempenho. A figura abaixo ilustra o ambiente em que os testes TPC-C foram realizados:

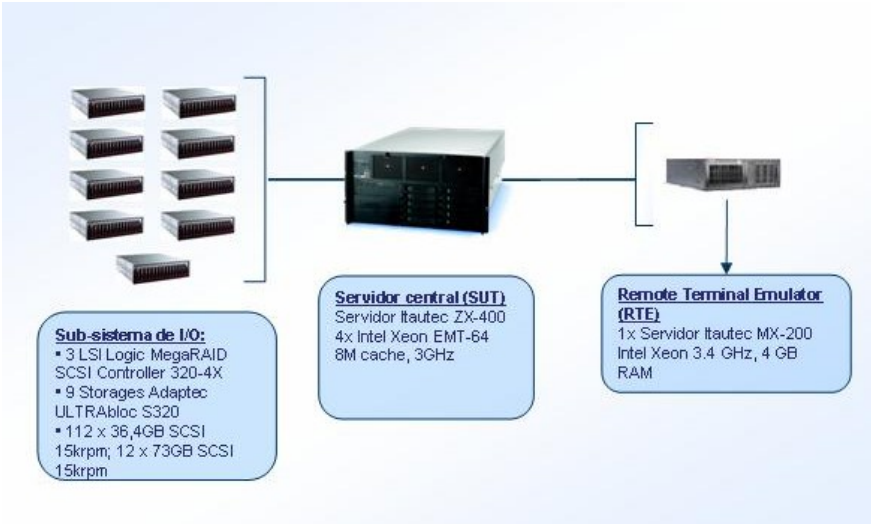



Figura 11.: Ambiente de Testes TPC-C.

Como pode ser visto na figura acima, o ambiente é formado por dois componentes: o SUT e o RTE. Eles estavam ligados por meio de uma rede Gigabit Ethernet.

O SUT é um Servidor Itautec ZX-400 com 4 processadores Intel Xeon 3,0 GHz e 32 GB de memória. Ele foi ligado a 112 discos SCSI de 36GB, 15krpm (usados para armazenar os arquivos de dados do banco), organizados em 8 *arrays* RAID 0 e 12 discos SCSI de 73GB, 15krpm (para armazenar o log do banco), organizados em 1 *array* RAID 0+1. Nele foi instalado o sistema operacional Windows 2003 Server Enterprise Edition, com Service Pack 1 e o SGBD SQL Server 2005 Standard Edition.

O RTE é um servidor Itautec MX-200 com um processador Intel Xeon 3,4 GHz e 4 GB de memória. Nele foi instalado o sistema operacional Windows 2003 Server Standard Edition, com Service Pack 1. Para gerar carga no SUT, foi utilizado o software BenchCraft, que é fornecido no kit TPC-C da Microsoft.

Durante os testes, todos os recursos estavam fisicamente disponíveis. A modificação de hardware do SUT foi simulada via software, limitando a quantidade de recursos que o SQL Server tinha à sua disposição. O tamanho da memória e o número de CPUs são limitados por meio de dois parâmetros de configuração do SQL Server, respectivamente: "*max server memory*" e "*affinity mask*". O número de discos era definido no momento da criação do banco de dados.

Para a coleta de dados de desempenho, foi utilizada a ferramenta *System Monitor*, nativa do Windows. Ela permite a coleta de diversos contadores de desempenho, com *overhead* mínimo. Para o presente trabalho, os seguintes contadores de desempenho foram monitorados:

- **(CPU) % *Processor Time* –** indica a utilização das CPUs do sistema;
- **(SQL Server) *Buffer Cache Hit Ratio* –** porcentagem de requisições que referenciam páginas já presentes na memória, sem a necessidade de operações de I/O;

- **(Disco)** ***Avg. Disk Queue Length*** **–** o número médio de requisições de I/O esperando na fila;
- **(Disco)** ***Avg. Sec/Transfer*** **–** o tempo de resposta médio de uma requisição de I/O, em segundos;
- **(Disco)** ***Disk Transfers/sec*** **–** a taxa de requisições de I/O por segundo;

Os contadores de CPU são coletados para cada processador do sistema. Os contadores de disco são coletados para cada *array* de discos em RAID, que para o sistema operacional são vistos como um único disco lógico.

Os testes foram conduzidos da seguinte maneira:

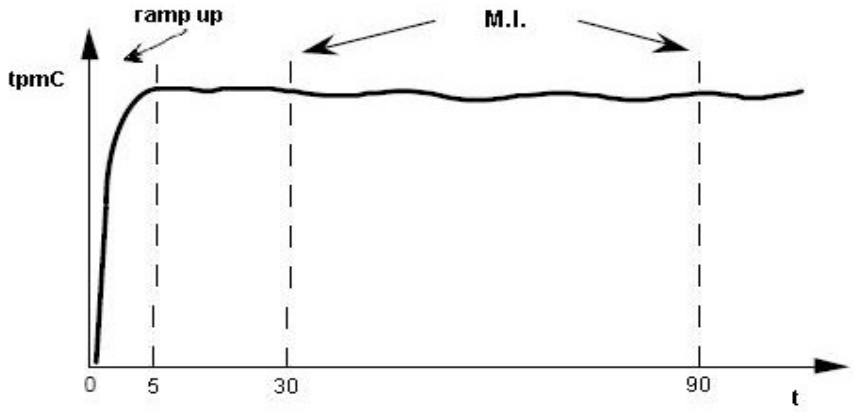



Figura 12.: Duração dos experimentos

Os primeiros cinco minutos eram reservados para que os usuários se conectassem ao SUT (*ramp up*). Em seguida, ocorria uma folga de 25 minutos, para deixar que o sistema alcançasse um estado estável (*steady state*). Após esses primeiros trinta minutos, o sistema começava a ser monitorado. Nesse intervalo de medição (MI), que durava 1 hora, ocorria a coleta de dados de desempenho do SUT. Ao fim do MI, era verificado o número total de transações completadas C, número esse utilizado na parametrização do modelo.

### 6.1.2. Cálculo de Demanda dos Recursos

Experimentos realizados e dados coletados, o que resta agora é determinar, utilizando as leis de teoria das filas descritas no Capítulo 4, as demandas de recursos da carga TPC-C. Vejamos, então, como isso feito.

#### 6.1.2.1. CPU

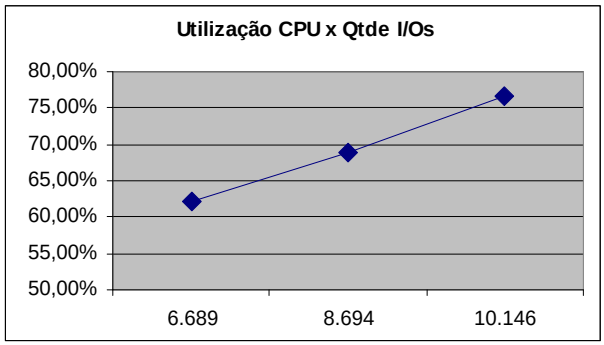
Conforme, vimos no Capítulo 4, podemos obter a demanda de um recurso, se forem conhecidos a sua utilização e o *throughput* do sistema ($D_K = U_K/X$). A utilização da CPU pode ser obtida com a ferramenta de monitoração do Windows. O *throughput* é informado pela ferramenta de geração de carga ao fim do MI. O problema é que a demanda por CPU de uma transação TPC-C se divide em dois componentes:

- A demanda de processamento das operações que compõem a transação (seleções, atualizações, junções, operações aritméticas, etc.);
- A demanda (*overhead*) de processamento das operações de I/O;

Desse modo:

$$D_{CPU} = D_{CPU\_PROC} + D_{CPU\_IO}$$

Ou seja, a demanda de CPU depende não somente da intensidade da carga, mas também de um fator que varia de maneira independente dela – a taxa de solicitação de I/O. Ora, lembrando-se da Lei do Fluxo Forçado, sabemos que $D_{CPU\_IO} = V_{IO} * S_{CPU\_IO}$. Aqui, $V_{IO}$ nada mais é que o número de operações de I/O executadas pela transação. Isso significa que a demanda de I/O tem efeito direto na demanda da CPU. O gráfico a seguir mostra os resultados de três experimentos e ilustra bem esse fato:




Podemos ver que mesmo mantendo o número de usuários constante (intensidade de carga inalterada), se a quantidade de operações de I/O cresce, a demanda por CPU também aumenta. Por outro lado, a demanda de processamento das operações aritméticas das transações se mantém a mesma, independentemente da quantidade de operações de I/O. Como a quantidade de I/Os pode ser obtida diretamente por meio de experimentos (ou prevista pelo modelo, como veremos mais adiante), para saber a demanda média de CPU de uma transação, fica faltando ainda a determinação de duas variáveis, a saber, $D_{CPU\_PROC}$ e $S_{CPU\_IO}$.

Temos, então, duas variáveis e apenas uma equação, o que em princípio impossibilita a determinação dos seus valores. Felizmente, podemos estabelecer o valor de $S_{CPU\_IO}$ a partir de dois experimentos consecutivos, variando a quantidade de I/O e verificando qual o efeito sobre a utilização da CPU.

Sejam, então, dois experimentos 1 e 2, cuja demanda de I/O foi acrescida reduzindo a quantidade de memória disponível para o buffer de dados. O tempo de serviço médio para processamento de operação de I/O $S_{CPU\_IO}$ é obtido da seguinte maneira:

$$\begin{cases} D1_{CPU} = D_{CPU\_PROC} + V1_{IO} \times S_{CPU\_IO} \\ D2_{CPU} = D_{CPU\_PROC} + V2_{IO} \times S_{CPU\_IO} \end{cases} \Rightarrow \frac{U2_{CPU}}{X_2} - \frac{U1_{CPU}}{X_1} = (V2_{IO} - V1_{IO}) \times S_{CPU\_IO} \Rightarrow$$

$$S_{CPU\_IO} = \frac{X_1 \times U2_{CPU} - X_2 \times U1_{CPU}}{X_1 \times X_2 \times (V2_{IO} - V1_{IO})}$$

A tabela abaixo mostra os resultados de dois experimentos realizados com o objetivo de determinar $S_{CPU\_IO}$:

**Tabela 3: Resultados de experimentos I/O vs Utilização CPU.**

| Contador | Experimento 1 (8 GB) | Experimento 2 (6 GB) |
|---|---|---|
| Throughput Txn (X) | 893,01 | 880,43 |
| Utilização CPU (U) | 68,89% | 76,54% |
| Throughput I/O ($X_{IO}$) | 8.704,30 | 10.146,18 |

A partir desses números, $S_{CPU\_IO}$ foi calculada em **55,113 µs** (microssegundos). A demanda de processamento $D_{CPU\_PROC}$ pode ser obtida substituindo esse valor na fórmula a seguir:

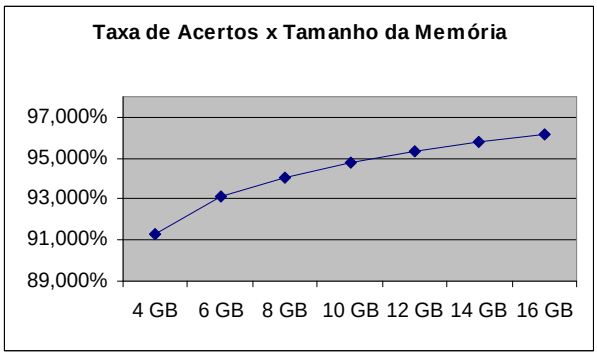
$$D_{CPU\_PROC} = \frac{U_{CPU}}{X} - V_{IO} \times S_{CPU\_IO}$$

Donde tiramos que **$D_{CPU\_PROC}$ = 234,2 µs** (microssegundos). Esses valores serão utilizados pelo modelo na hora de determinar a utilização da CPU ou o tamanho da fila sob outras condições de carga.

#### 6.1.2.2. O Efeito da Memória

Conforme pudemos perceber, a memória desempenha papel fundamental em cargas OLTP. Ela é utilizada essencialmente como armazenamento temporário para os dados armazenados em disco (buffer de dados). Quando se tem mais memória principal, ocorrem menos requisições de I/O, o que reduz a carga não só para o subsistema de I/O, mas também para a CPU, como vimos há pouco. O objetivo, então, é determinar o efeito do tamanho da memória sobre a quantidade de I/Os (e conseqüentemente, sobre a utilização da CPU). Obviamente o tamanho do banco de dados é também um fator preponderante sobre a taxa de acertos dos acessos ao buffer de dados e por isso deve ser levado em consideração na modelagem.

Para caracterizar o efeito de variações na relação tamanho da memória/escala do banco sobre a taxa de acertos, um modelo de regressão foi desenvolvido. Uma série de experimentos foi conduzida com o intuito de obter os parâmetros necessários à criação desse modelo. O primeiro passo foi decidir que tipo de modelo seria mais adequado (Linear? Curvilíneo?). O gráfico abaixo exibe a evolução da taxa de acertos com o aumento de memória, para um fator de escala fixo:




Visualmente, o gráfico aparenta exprimir um crescimento logarítmico conforme cresce a razão entre memória e escala. Assim, a taxa de acerto pode ser expressa da seguinte maneira:

$Sejam$:

- $Variável\ independente: X = \left( \frac{Tamanho\_memoria}{Escala\_banco} \right)$
- $Variável\ de\ resposta: Y = taxa\_de\_acertos$

$Então$:

$$Y = \alpha + \beta \times \ln(X)$$

Para facilitar os cálculos, a variável X foi substituída por X':

$$X' = \ln(X) \quad \Rightarrow \quad Y = \alpha + \beta \times X'$$

Perceba que agora temos uma relação linear entre Y e X', o que nos permite usar as equações apresentadas na Seção 4.4 para obter os parâmetros α e β. A tabela a seguir mostra os resultados utilizados para a criação do modelo de regressão linear:

**Tabela 4: Relação escala x memória x taxa de acerto.**

| Escala (*warehouses*) | Tamanho Memória (milhares de páginas) | ln(memória/escala) | Taxa de Acerto (%) |
|---|---|---|---|
| 2000 | 768 | -1,363 | 93,00 |
| 2000 | 768 | -1,363 | 93,32 |
| 2000 | 1024 | -0,669 | 94,03 |
| 2000 | 1024 | -0,669 | 94,08 |
| 2000 | 1536 | -0,264 | 95,51 |
| 2000 | 1536 | -0,264 | 95,23 |
| 2500 | 1024 | -0,893 | 93,25 |

O tamanho da memória disponível está expressa em milhares de páginas (a troca de dados entre a memória principal e o armazenamento secundário é feita em páginas de 8K). O experimento com maior número de *warehouses* (2500) foi feito para demonstrar que o relacionamento se mantém para outros fatores de escalas. Com base nos resultados dos experimentos, o seguinte relacionamento linear foi estabelecido:

$$Taxa\_de\_acerto = 96{,}47 + 3{,}68 \times \ln\left(\frac{Tamanho\_Memoria}{Escala\_Banco}\right)$$

O coeficiente de determinação, $R^2$, que mede quão bem a regressão se adequa aos dados coletados, foi de 99,9996% para o intervalo de medição.

#### 6.1.2.3. Discos

Sistemas OLTP são extremamente sensíveis a altas latências do subsistema de I/O. Por isso, é de grande importância conhecer (e respeitar) os limites impostos por esse subsistema (de maneira geral, recomenda-se manter a sua utilização abaixo de 85%). Nesse trabalho, para fins de análise, apenas os discos são avaliados, não levando em consideração o comportamento de

controladoras ou barramentos. A análise tem por objetivo prever a utilização dos discos e os seus tempos de resposta em certas condições de carga; os fatores determinantes dessas duas variáveis são:

- O número de discos (N);
- A taxa de requisição de operações de I/O ($X_{IO}$);
- O tempo de serviço médio de uma operação de I/O ($S_{IO}$);

O número de discos é conhecido a priori. A taxa de requisição pode ser obtida diretamente através de monitoração do sistema. Já o tempo de serviço não é conhecido, pois depende de uma série de fatores, como características do disco e da carga. No entanto, sabe-se que o tempo de resposta é dado pela equação:

$$R_K = S_K \times (1 + Q_K)$$

De modo que, o tempo médio de serviço de uma operação de I/O pode ser calculado como:

$$S_K = \frac{R_K}{(1 + Q_K)}$$

Como a ferramenta utilizada para monitorar o sistema disponibiliza os contadores que aparecem como variáveis independentes na equação, $S_{IO}$ é calculado de maneira simples e direta. Durante os vários experimentos, o valor médio obtido para $S_{IO}$ foi de 3,5 ms (milissegundos), com um desvio padrão de 0,043. Essa pequena variação se deve a diversos fatores, dentre eles flutuações na própria carga e até mesmo imprecisões na coleta dos dados. Vale ressaltar que o valor encontrado é válido de maneira muito específica para o tipo de disco utilizado nos testes e para a carga TPC-C. Em um ambiente com discos mais rápidos e um padrão de acesso mais seqüencial, o resultado seria um tempo de serviço menor – e vice-versa, com discos mais lentos ou um padrão de acesso mais randômico, o tempo de serviço seria maior.

## 6.2. Avaliação de Resultados

Nessa seção serão mostrados os resultados produzidos durante os experimentos. Os resultados não serão expressos na métrica tpmC, porque a TPC restringe a publicação de resultados não auditados. Além disso, o contrato de licença do SQL Server proíbe a publicação de resultados de qualquer *benchmark* sem a prévia autorização da Microsoft. A maioria dos fabricantes de SGBDs comerciais inclui uma cláusula similar (conhecida como cláusula De Witt), com o intuito de se proteger de resultados obtidos em ambientes mal-configurados, que venham a prejudicar a imagem dos seus produtos. Isso, no entanto, não prejudica em nada a apresentação dos resultados, visto que a intenção não é mostrar números absolutos, mas sim tendências provenientes de modificações de hardware. Para isso, os resultados serão expostos sob a forma de utilização dos recursos ou dos seus tempos de resposta.

### 6.2.1. Validação do Modelo

A validação do modelo se deu por meio da realização de dois experimentos. Em ambos, o sistema foi aumentado de escala, tanto no que diz respeito ao tamanho da base, quanto na quantidade de recursos de hardware. Um banco maior, de 4000 *warehouses* foi criado para ser usado em ambos os testes. Ele foi distribuído entre todos os 112 discos.

No primeiro deles, o tamanho da memória disponível foi aumentada até o máximo (32 GB) e apenas um processador foi utilizado. A intenção é verificar até que ponto o excesso de memória ameniza a carga sobre a CPU e o subsistema de I/O. A previsão do modelo é dada a seguir:

$$Taxa\_de\_Acerto = 96{,}65 \Rightarrow$$

$$\begin{cases} V_{IO} = 162 \times (1 - 0{,}965) = 5{,}427 \\ X_{IO} = X \times V_{IO} = 9.768{,}60 \end{cases} \Rightarrow$$

$$\begin{cases} D_{CPU} = 234{,}2 \times 10^{-6} + 5{,}427 \times 55{,}113 \times 10^{-6} = 533{,}30\,\mu s \Rightarrow \\ U_{CPU} = X \times D_{CPU} = 95{,}99\% \end{cases}$$

$$\begin{cases} U_{DISCO} = \dfrac{X_{IO} \times S_{IO}}{N_{DISCOS}} = \dfrac{9.768{,}60 \times 0{,}0035}{112} = 30{,}53\% \\ Q_{DISCO} = \dfrac{U_{DISCO}}{(1 - U_{DISCO})} = 0{,}44 \\ RT_{DISCO} = 0{,}0035 \times (1 + 0{,}44) = 0{,}005s \\ X_{DISCO} = \dfrac{X_{IO}}{N_{DISCOS}} = 87{,}220 \end{cases}$$

Segundo, o modelo, com 32 GB de memória, o sistema agüenta a carga imposta, embora a CPU encontre-se muito próxima da sua capacidade máxima. Os resultados obtidos no teste foram os seguintes:

| Contador de Desempenho | Valor |
|---|---|
| Taxa de acertos memória | 96,275% |
| Utilização CPU | 96,81% |
| Fila de Disco (média) | 0,449 |
| I/Os por segundo (média) | 88,64 |
| Segundos por I/O (média) | 0,005s |

Tabela 5: Resultados do 1º experimento de validação.

Como pode ser visto, o resultado previsto pelo modelo foi bastante próximo ao obtido no experimento. O erro da taxa de acertos foi de 0,38%. O erro da utilização de CPU foi de 0,85%. O erro da fila de disco foi de 2% e da taxa de I/O foi de 1,6%.

Já no segundo experimento, 12 GB de memória estavam disponíveis e 2 processadores foram utilizados. A previsão do modelo é a seguinte:

$$Taxa\_de\_Acerto = 93{,}04 \Rightarrow$$

$$\begin{cases} V_{IO} = 162 \times (1 - 0{,}9304) = 11{,}275 \\ X_{IO} = X \times V_{IO} = 20.295{,}36 \end{cases} \Rightarrow$$

$$\begin{cases} D_{CPU} = \dfrac{234{,}2 \times 10^{-6} + 11{,}275 \times 55{,}113 \times 10^{-6}}{2} = 427{,}80\mu s \Rightarrow \\ U_{CPU} = X \times D_{CPU} = 77{,}00\% \end{cases}$$

$$\begin{cases} U_{DISCO} = \dfrac{X_{IO} \times S_{IO}}{N_{DISCOS}} = \dfrac{20.295{,}36 \times 0{,}0035}{112} = 63{,}42\% \\ Q_{DISCO} = \dfrac{U_{DISCO}}{(1 - U_{DISCO})} = 1{,}73 \\ RT_{DISCO} = 0{,}0035 \times (1 + 1{,}73) = 0{,}0096s \\ X_{DISCO} = \dfrac{X_{IO}}{N_{DISCOS}} = 181{,}208 \end{cases}$$

E os resultados obtidos no teste são mostrados abaixo:

| Contador de Desempenho | Valor |
|---|---|
| Taxa de acertos memória | 92,978% |
| Utilização CPU | 83,301% |
| Fila de Disco (média) | 1,78 |
| I/Os por segundo (média) | 186,06 |
| Segundos por I/O (média) | 0,010s |

Tabela 6: Resultados do 2º experimento de validação.

A Tabela 6 mostra que a previsão da taxa de acerto foi boa (erro de apenas 0,1%). A previsão também foi satisfatória para os contadores de disco: o erro do tamanho médio da fila de disco foi de 2,8%, o da taxa de I/O foi de 2,6% e o do tempo médio de I/O de 4%. Somente a utilização da CPU teve uma diferença mais significativa, de 7,6%. Uma explicação para essa diferença pode ser o fato de sistemas com dois processadores não terem efetivamente duas vezes mais poder de processamento que um que possua apenas um processador. Isso porque questões como utilização do barramento (FSB) e coerência de cache aumentam o seu CPI (ciclos por instrução). Quando um sistema possui mais de um processador, a latência do FSB aumenta devido a condições de competição. Além disso, cada processador possui a sua própria memória cache, e os procedimentos necessários para mantê-las coerentes entre si incorre em um *overhead*. O efeito desses fatores não foi considerado no modelo e é provável que seja essa a causa da diferença entre o que foi previsto e o que foi obtido no teste de validação.

### 6.2.2. Limitações do modelo

O último experimento de validação deixou bem claro a principal limitação do modelo proposto: a deficiência em prever o efeito de fatores intrínsecos à arquitetura interna do sistema. Por tratar o sistema de maneira bastante simplificada – apenas como um conjunto de processadores e discos – o modelo se mostra inadequado na hora de prever o resultado de variações em:

- Cache do processador – o tamanho da cache, o seu grau de associatividade, o seu tamanho de bloco, enfim, todos os parâmetros de cache têm conseqüências diretas sobre o poder de processamento de uma CPU, na medida em que reduzem ou aumentam o seu CPI (ciclos por instrução);
- Velocidade do FSB – a velocidade do barramento entre processador e memória também influencia no seu CPI. Aumentar a velocidade do FSB reduz a latência do acesso à memória;

Além das limitações acima, a previsão dada pelo modelo para o subsistema de I/O é específica para o tipo dos discos utilizados. O modelo prevê o resultado de variações apenas na quantidade de discos, não em características dos mesmos como, latência rotacional, ou *seek time*. É claro que os parâmetros do modelo podem ser modificados para se ajustar a esse tipo de mudança, mas isso exigiria novos experimentos.

De maneira geral, o modelo oferece boas previsões quando a intenção é o aumento de escala dos recursos de hardware disponíveis. Quando ocorrem também mudanças de arquitetura, os resultados do modelo tendem a se distanciar da realidade.

# 7. CONCLUSÕES

É muito difícil saber ao certo quanto uma mudança de configuração irá afetar o desempenho de um sistema. Esse trabalho teve como objetivo justamente prever o efeito de mudanças de hardware no desempenho de sistemas de processamento de transação, servindo como um auxílio no processo de planejamento de capacidade para ambientes OLTP. Algumas conclusões importantes podem ser tiradas desse estudo:

- Para melhorar o desempenho de um sistema, deve-se sempre focar no componente que tenha a maior demanda (gargalo), pois os benefícios são maiores nesse caso;
- Em sistemas OLTP, a memória é um componente-chave. Por reduzir tanto a demanda de processamento quanto a demanda de I/O, quase sempre a adição de mais memória ao sistema será a solução que proverá o melhor custo-benefício;
- O modelo proposto neste trabalho mostrou-se bastante preciso na previsão dos resultados obtidos com o acréscimo de recursos ao sistema, mas inadequado para prever o resultado de mudanças em características intrínsecas dos mesmos (ex. cache do processador, velocidade de rotação dos discos, etc.);

Uma sugestão para trabalhos futuros é justamente que sejam feitos estudos sobre o efeito de componentes internos do processador (como cache, FSB) e de parâmetros de discos (como velocidade de rotação, seek time, etc.) no desempenho do sistema. Desse modo, a previsão de desempenho poderia ser feita a partir da própria especificação técnica dos dispositivos.

Outra sugestão seria realizar os experimentos em um ambiente de código aberto e incluir código de *tracing* no SGBD de modo a obter dados mais precisos sobre quanto tempo é gasto por cada transação em cada recurso, o que permitiria a construção de modelos mais detalhados.

# REFERÊNCIAS BIBLIOGRÁFICAS


[1] ALLEN, A.O. **– Probability, Statistics, and Queuing Theory With Computer Science Applications, 2nd edition**. Academic Press, 1990

[2] CASSANDRAS. C.G., STÉPHANE L. **– Introduction to Discrete Event System**. Kluwer Academic Publishers, 1999

[3] DU, X., ZHANG X., DONG, Y., ZHANG, L. **– Architectural Effects of Symmetric Multiprocessors on TPC-C Commercial Workload**

[4] ELMASRI, R., NAVATHE, S.B. **– Fundamentals of Database Systems, 3rd Edition**. Addison-Wesley, 2000

[5] HENNESSY, J.L., PATTERSON, D.A. **– Organização e Projeto de Computadores – A Interface Hardware/Software**

[6] JAIN, R. **– The Art of Computer Systems Performance Analysis**.

[7] LEUTENEGGER, S.T., DIAS, D. **– A Modeling Study of the TPC-C Benchmark**

[8] NARAYANAN, D., THERESKA, E., AILAMAKI, A. **– Continuous Resource monitoring for self-predicting DBMS**

[9] RIBAS, J.C.C., DIAS, R.A. **– Compreendendo e Caracterizando Carga de Trabalho**

[10] SEAGATE **–** Especificação do disco SCSI Cheetah 15K.4. Disponível em: http://www.seagate.com/cda/products/discsales/enterprise/tech/0,1084,656,00.html **–** Último acesso em 27/09/06.

[11] SPEC **–** Standard Performance Evaluation Corporation. Disponível em: http://www.spec.org/ - Último acesso em 25/09/06.

[12] TPC Benchmark C, Standard Specification, Revision 5.6 Disponível em: http://www.tpc.org - Último acesso em 08/09/2006.

[13] TSUEI, T.F., PACKER, A.N., KO, K.T. **– Database Buffer Size Investigation for OLTP Workloads**

[14] WHALEN, E., GARCIA, M., DELUCA, S.A., THOMPSON, D. **– SQL Server 2000 Performance Tuning**. Microsoft Press, 2001

[15] WILIS, A. **– Performance Evaluation Techniques**.

[16] WIKIPEDIA **– Benchmark (Computing)**
Disponível em: http://en.wikipedia.org/wiki/Benchmark_(computing) – Último acesso em 10/09/2006

# APÊNDICE A: TUNING DO SISTEMA

O texto a seguir descreve os ajustes feitos no sistema antes da realização dos experimentos. Estão incluídas configurações de parâmetros de hardware e de software (sistema operacional e SGBD). É importante ressaltar que algumas das configurações expostas são bastante específicas, e trazem benefícios somente para ambientes com determinadas características. Elas não devem, portanto, ser tomadas como modelo geral de configuração para sistemas de processamento de transação. Em alguns casos, aplicá-las pode, inclusive, prejudicar o desempenho do sistema. É preciso entender os efeitos de cada um dos ajustes antes de decidir pelo seu uso.

## A1: Configurações do Processador

Dois parâmetros foram modificados na BIOS do SUT

- **Hardware Prefetching –** recuperação automática de conjuntos linhas da cache utilizando heurísticas para a detecção de padrões nas requisições anteriores. DESABILITADO;
- **Adjacent Cache Line –** mecanismo que permite a recuperação de duas linhas consecutivas de 64 bytes da memória principal para o cache L2. Foi HABILITADO.

Além desses dois parâmetros, o suporte a Hyper Threading foi mantido (por default, o HT vem habilitado).

## A2: Configurações do Sistema de I/O

Para armazenar os dados do banco, foram criados 8 *arrays* RAID 0, cada um com 14 discos de 36 GB, 15krpm. O *stripe size* dos *arrays* foi configurado em 64 KB e o suporte a *write-back* foi habilitado. Para o log do banco, foi criado 1 array RAID 0+1, com 12 discos de 73 GB, 15 krpm. O stripe foi de 64 KB. Como o log de um banco de dados precisa de tolerância a falhas, o suporte a *write-back* foi desabilitado.

## A3: Configurações do Sistema Operacional

- **Selecão do serviços**

Foram mantidos somente os serviços necessários à execução dos testes. Todos os outros serviços foram desligados. A seguir a lista:

1. .NET Runtime Optimization Service v2.0.50727_x64
2. .NET Runtime Optimization Service v2.0.50727_x86
3. Application Experience Lookup Service
4. COM+ Event System
5. Event Log
6. Logical Disk Manager
7. Network Connections
8. Network Location Awareness
9. Plug and Play
10. Remote Procedure Call
11. Remote Procedure Call Locator
12. Security Accounts Manager
13. System Event Notification
14. Windows Management Instrumentation

- **Modificação do registro do Windows**

Dentro da chave:

```
[HKEY_LOCAL_MACHINE\System\CurrentControlSet\Control\Session Manager]
```

Criação da chave "`I/O System`" e dentro dela, criação da dword `CountOperations`, com valor `0`. Isso desabilita a contagem de operações de I/O pelo sistema operacional.

## A4: Configurações do SQL Server

- **Parâmetros do SQL Server**

Os seguintes parâmetros foram configurados:

| Param | valor |
|---|---|
| affinity mask | * |
| awe enabled | 1 |

| | |
|---|---|
| lightweight pooling | 1 |
| max degree of parallelism | 1 |
| max worker threads | 400 |
| max server memory | * |
| min server memory | 2048 |
| network packet size | 4096 |
| priority boost | 1 |
| recovery interval (min) | 32767 |
| show advanced options | 1 |

* parâmetros que variaram de um teste para outro.

- **SQL Server Startup Options**

O SQL Server foi iniciado utilizando as seguintes opções e *trace-flags*:

```
"C:\Program Files\Microsoft SQL
Server\MSSQL.1\MSSQL\Binn\sqlservr.exe" -e"C:\Program
Files\Microsoft SQL Server\MSSQL.1\MSSQL\LOG\errorlog" -c -x
-g150 -t661 -t677 -t697 -t827 -t834 -t836 -t1229 -t8011 -
t8012 -t8018 -t8019 -t8020 -t8193 -t8710 -t8744 -t3502.
```