Universidade Federal de Pernambuco
Centro de Ciências Exatas e da Natureza
Centro de Informática

Pós-graduação em Ciência da Computação

DBT-5: Uma Implementação de Código Aberto do TPC-E para Avaliação de Desempenho de Sistemas de Processamento Transacional

Rilson Oscar do Nascimento

Dissertação de Mestrado

Recife
29 de Agosto de 2008

Universidade Federal de Pernambuco
Centro de Ciências Exatas e da Natureza
Centro de Informática

Rilson Oscar do Nascimento

DBT-5: Uma Implementação de Código Aberto do TPC-E para Avaliação de Desempenho de Sistemas de Processamento Transacional

Trabalho apresentado ao Programa de Pós-graduação em Ciência da Computação do Centro de Informática da Universidade Federal de Pernambuco como requisito parcial para obtenção do grau de Mestre em Ciência da Computação.

Orientador: *Prof. Dr. Paulo Romero Martins Maciel*

Recife
29 de Agosto de 2008

À minha família.

Agradecimentos

Gostaria de agradecer a Prof^{a} Ana Carolina Salgado e ao Prof. Paulo Romero Martins Maciel o voto de confiança dado através de suas cartas de recomendação para ingresso no programa de Mestrado.

Aos colegas de trabalho e de turma tenho especial apreço, pois participamos de momentos importantes nesta empreitada acadêmica. Como representantes saúdo Guilherme Amorim, Carlos Eduardo Pires e Antônio Ricardo Cavalcante.

Agradeço à Lu, minha esposa, pelo seu amor, incentivo e por compreender minha ausência em incontáveis momentos. Aos meus filhos Beatriz e Pedro de forma especial sou grato, pois sem seus carinhos e brincadeiras não encontraria o equilíbrio necessário para dar andamento ao trabalho.

Por fim, sem a aprovação do Criador nada se completaria; agradeço a Ele pelo sustento contínuo.

-Rilson Nascimento

Resumo

O TPC-E é o novo benchmark aprovado pelo conselho TPC e modela uma carga de trabalho de processamento de transações *online* centrada no Sistema Gerenciador de Banco de Dados (SGBD). O TPC-E simula um sistema de corretagem de ações com múltiplos tipos de transação e foi projetado para balancear o uso de disco e de processamento. Os benchmarks são ferramentas importantes para os fornecedores de SGBD, pois podem ser utilizados no processo de controle de qualidade destes sistemas. Os objetivos principais são: diagnosticar problemas de desempenho ou regressão nos produtos existentes e auxiliar na engenharia de novos produtos e novas funcionalidades. O usuário final também pode se utilizar de benchmarks para auxiliá-lo na decisão de investimento em sistemas computacionais, através de comparativos de desempenho entre sistemas concorrentes. Neste trabalho é apresentado o DBT-5, uma implementação de código aberto do benchmark TPC-E. Sua arquitetura e implementação são descritas e o resultado da validação é apresentado, seguindo os requisitos de auditoria da especificação TPC-E. Por fim, dois estudos de caso são mostrados. O primeiro avalia o desempenho dos *schedulers* de E/S do Linux executando os sistemas de arquivos EXT2, EXT3 e ReiserFS. O segundo analisa o comportamento de E/S do DBT-5 utilizando um *tracer*.

Palavras-chave: Avaliação de Desempenho, Benchmarking, Carga de Trabalho de Banco de Dados

Abstract

The TPC-E is the new benchmark recently approved by the TPC council and is patterned after an online transaction processing workload centered in the Database Management System (DBMS). TPC-E models a brokerage house company with mutliple transaction types and was designed to balance I/O and processor usage. Benchmarks are invaluable tools for DBMS makers because they can be used as quality assurance tools to diagnose performance problems or regressions in existing products. Also, benchmarks can be used in the performance engineering of new products or new features. End users can employ benchmarks to help them in the decision of buying a new system by doing performance comparisons between competitors. In this work DBT-5 is presented; an open source fair use TPC-E implementation. Its architecture and implementation are described and the validation results are presented, following the audit rules of the TPC-E specification. To conclude, two study cases were conducted. The first evluates the performance of the Linux I/O schedulers on three file systems, EXT2, EXT3 and ReiserFS. The second study evaluates the DBT-5 I/O behavior by using a tracer.

It is designed to exercise an On-Line Transaction Processing workload of a brokerage firm and be representative of current database server work. In this work the DBT-5 tool is presented, a fair usage open-source implementation of the TPC-E benchmark. In addition to reporting on the design and implementation details of the tool, experimental results on a system running the PostgreSQL database engine are also described. The significance of this work is that it provides an environment where recent innovations in the OLTP workload field can be evaluated.

Keywords: Benchmark, Database Workload, Performance Evaluation

Sumário

Lista de Figuras

Lista de Tabelas

CAPÍTULO 1

Introdução

It is not simply how busy you are, but why you are busy. The bee is praised; the mosquito is swatted.

— (Marie O'Conner)

Neste capítulo é apresentada uma introdução aos Benchmarks de Banco de Dados com uma breve descrição de sua evolução no tempo. Em seguida é apresentada uma introdução ao Benchmark TPC-E. Na seqüência, são apresentados os objetivos deste trabalho e a sua relevância. Em seguida, são apresentados alguns trabalhos relacionados. Finalmente, é apresentada a organização da dissertação.

1.1 Motivação

O requisito não-funcional Desempenho é uma faceta significativa dos sistemas de processamento de transação. O tempo de resposta é crítico para usuários finais assim como a vazão transacional é importante para administradores de sistemas gerenciadores de banco de dados (SGBD). Em um cenário corporativo, as empresas não podem dispender recursos em sistemas que são incapazes de suster de maneira satisfatória seu negócio. Desta forma, escalabilidade e desempenho têm um papel fundamental na decisão de investir em sistemas computacionais. No processo de decisão os administradores necessitam de instrumentos que reportem o desempenho dos sistemas de forma que comparações racionais possam ser feitas entre diferentes sistemas.

Como resposta a esta necessidade, os fabricantes de sistemas computacionais formaram um consórcio independente chamado Transaction Processing Performance Council (TPC)[tpp08]. O TPC define modelos de cargas de trabalho de Sistema Gerenciador de Banco de Dados (SGBD) que, devidamente empregados, podem ser usados para realizar comparações coerentes entre diversos sistemas[BP97].

O TPC Benchmark E (TPC-E)[TPC07] é um dos benchmarks do TPC que modelam uma carga de trabalho de processamento de transação online. O TPC-E simula um sistema de corretagem de ações com múltiplos tipos de transações e foi projetado para balancear o uso de disco e de processamento. Como seu predecessor, o TPC Benchmark C (TPC-C)[TPC92], o TPC-E

especifica também um tipo de acesso não-uniforme as tabelas para melhor modelar a atividade online transacional. O TPC-E objetiva substituir o TPC-C pois este já tem mais de dez anos de uso. A carga de trabalho DBT-5[dNWM07] é baseada no TPC-E, ou ainda, na especificação do TPC-E.

Além do seu caráter comercial, os benchmarks da TPC têm sido amplamente usados na academia, pois têm demonstradao serem meios significativos para representação dos sistemas que visam modelar, possuem grande aceitação pela indústria e foram criados por uma entidade neutra de reconhecida competência. Llanos implementou o TPC-C para ser utilizado na avaliação de sistemas distribuídos e paralelos [LP06]; Leutenegger e Dias modelaram the TPC-C[LD03] para estudar seus padrões de acesso; Sivasubramaniam criou um sintetizador de carga de trabalho baseado no benchmark TPC-H[TPC99a].

O amplo uso dos benchmarks por parte de administradores e desenvolvedores de SGBD, principalmente àqueles relacionados com SGBD de código aberto, motiva a criação de novos benchmarks que sejam mais representativos dos sistemas atuais. Prover uma ferramenta que auxilie o dia-a-dia desses profissionais é um fator motivante e importante para realização deste trabalho.

1.2 Evolução dos Benchmarks de Banco de Dados

Várias organizações definem padrões de benchmarks. Para o domínio de banco de dados, os mais proeminentes são o TPC (Transaction Processing Performance Council - Conselho de Desempenho de Processamento Transacional) e o SPEC (System Performance Evaluation Cooperative - Cooperativa de Avaliação de Desempenho de Sistemas). O TPC é um consórcio sem fins lucrativos de vendedores de hardware e software que define benchmarks na área de processamento transacional e data warehousing. O SPEC também é um consórcio de vendedores que defines benchmarks na área científica, *desktop*, componentes entre outros. Os benchmarks da TPC têm como característica o rigor da especificação quanto a medição, precificação e forma de rodar o teste. Particularmente a concepção de uma métrica que associe preço e desempenho é interessante por que facilita a comparação entre diferentes sistemas e arquiteturas.

A história dos benchmarks de banco de dados coincide com o início da organização TPC. A formalização dos primeiros benchmarks pela TPC, DebitCredit e TP1, foram chamados de TPC-A[TPC89] e TPC-B[TPC90], tendo suas respectivas especificações e resultados ainda disponíveis no site da TPC[tpp08]. Antes do advento dos benchmarks formais, i.e., aqueles formalmente especificados por organizações reconhecidas, as empresas criavam e mantinham suas próprias ferramentas e processos para comparação de desempenho. Porém, não era comum a publicação de resultados na impressa para evitar embaraço, haja vista que era difícil conseguir

credibilidade e imparcialidade com benchmarks criados pela próprias empresas. Ainda assim, quando números comparativos era publicados por terceiros ou competidores, os perdedores geralmente acusavam problemas e tentavam desacreditar o benchmark. Este eventos causavam as chamadas guerras de benchmark [Gra93]. Este embate se dá quando um competidor A perdia para um B numa disputa de avaliação de desempenho e então usando especialistas conseguia melhorar seus números, obtendo vitória sobre B. B por sua vez usava da mesma estratégia para obter números ainda melhores sobre A e assim por diante.

Benchmarketing é uma variação da guerra de benchmarks [Gra93]. Para cada sistema, há um benchmark em particular no qual o sistema tem seu melhor desempenho. O departamento de marketing ou uma empresa externa de marketing assim tentava fazer deste benchmark o "padrão"de mercado, omitindo os detalhes internos deste. Ao mesmo tempo exacerbava os pontos fortes do seu banco, omitindo os negativos. Esta atividade criava ceticismo e acabava por descreditar todos os resultados publicados e o próprio benchmark em si. As empresas perceberam que a criação de uma organização independente, que unisse representantes de todas as empresas, seria uma alternativa interessante para gerar benchmarks neutros e resultados confiáveis. Neste contexto nasce a organização TPC em 1988, fundada por Omri Serlin.

Os primeiros dois benchmarks da TPC já citados, o TPC-A e o TPC-B, reduziram drasticamente a guerra de benchmarks [Gra93]. Os clientes adquiriram confiança tanto no rigoroso processo de implementação quanto na auditoria de publicação dos benchmarks da TPC e passaram a requerer resultados TPC para compra de sistemas. A Tabela 1.1 mostra os benchmarks criados ou em desenvolvimento pela TPC até a presente data.

Grandes clientes de hardware do Brasil, como o Governo Federal, utilizam os resultados dos benchmarks TPC para avaliação de possíveis fornecedores em suas licitações ou pregões. Os clientes e também os próprios vendedores podem se valer dos benchmarks pelo menos das seguintes formas:

- *Comparar diferentes sistemas de hardware e software*. Os benchmarks podem ser usados para avaliar o desempenho relativo de diferentes sistemas em hardware distintos rodando a mesma aplicação. Esta seria a clássica situação competitiva entre dois vendedores de hardware.

- *Comparar diferentes sistemas de software em uma mesma máquina*. O objetivo é avaliar dois ou mais produtos de softwares no mesmo hardware. Esta seria a situação típica de competição entre dois fabricantes de software. Exemplo: comparação de dois SGBD numa mesma máquina.

- *Comparar máquinas diferentes dentro de uma mesma família*. Os benchmarks podem ser utilizados para avaliar o desempenho relativo de máquinas dentro de uma mesma família ou linha de hardware.

Tabela 1.1 Benchmarks da TPC

Benchmark	Ano Criação	Obsoleto?	Em Desenvolvimento?	Domínio
TPC-A	1989	Sim	Não	OLTP[1]
TPC-B	1990	Sim	Não	estresse
TPC-D[TPC95]	1995	Sim	Não	Suporte a Decisão
TPC-R[TPC99b]	1999	Sim	Não	Suporte a Decisão
TPC-W[TPC99c]	1999	Sim	Não	Comércio Web
TPC-C	1992	Não	Não	OLTP
TPC-H[TPC99a]	1999	Não	Não	Suporte a Decisão
TPC-App[TPC05]	2005	Não	Não	Servidor de Aplicação
TPC-E	2007	Não	Não	OLTP
TPC-DS[TPC08]	–	Não	Sim	Suporte a Decisão

- *Comparar releases diferentes de um mesmo produto em uma mesma máquina*. Os benchmarks podem ser utilizados pelos fabricantes de software (ex: SGDB) para avaliar o desempenho entre diferentes versões ou releases, com o objetivo de avaliar possíveis regressões ou melhorias.

Além dos benchmarks da TPC, outros mostraram-se também como importantes marcos na área, como o Wisconsin[BDT83] e o AS3AP[TOB93]. A seguir são descritas as principais características dos principais benchmarks mencionados até aqui neste capítulo. O TPC-E é descrito em um capítulo específico por ser objeto deste trabalho.

TPC-A. O TPC-A foi o primeiro benchmark criado pelo TPC depois de um ano de sua fundação. Foi baseado no teste DebitCredit desenvolvido por Jim Gray [A$^+$85], naquele então funcionário da Tandem Computers. Este teste propõe um emulação bem elaborada de um sistema de suporte a atendimento bancário de um grande banco com múltiplas agências. Os parâmetros-chave para o banco emulado incluíam quantidade de contas-cliente, quantidade de agências bancárias, quantidade de atendentes por agência, entre outros.

A IBM levou a cabo esta emulação para analisar gargalos de desempenho e considerações preço-desempenho que levou o Bank of America a adotar os mini-computadores da IBM. No DebitCredit só havia um tipo de transação, que representava um depósito/retirada de um correntista. Para dar mais realismo a emulação, 15% de todas as transações de entrada envolviam contas entre agências diferentes. A intenção era representar um cliente usando uma agência diferente daquela a qual ele estava diretmente ligado. A escalabilidade era assegurada pela

Tabela 1.2 TPC-A comparado com o DebitCredit

Item	DebitCredit	TPC-A
Autoridade	Nenhuma - Resultado publicado na imprensa	Aprovado pela TPC
Propriedades do Sistema	Log duplicado	Testes ACID
Chegada de Transação	Não Especificado	Aleatório (Distribuição Exp. Neg.[2])
Relatório de Resultado	Não Especificado	Relatório detalhado obrigatório; auditoria recomendada

fixação da taxa de geração de transações por atendente (100 segundos por transação) e, para cada transação por segundo, 100.000 contas, 10 agências e 100 atendentes. As mais importantes inovações do DebitCredit foram sua abordagem não tendenciosa, a possibilidade de comparar sistemas levando em conta seu custo, e, o domínio público dos requerimentos do DebitCredit.

O TPC-A se diferenciava do DebitCredit em vários aspectos. Os principais estão sumarizados na Tabela 1.2. O TPC realizou simplificações e melhorias no DebitCredit. As simplificações, por alguns, eram vistas negativamente pois diminuía a representatividade do benchmark; por outros, estas visavam dar um enfoque mais diverso o que permitia um uso mais amplo. Por outro lado, as melhorias ou requisitos introduzidos tornaram o TPC-A mais rígido, impondo uma implementação mais representativa da carga de trabalho modelada (ex: taxa de chegada não uniforme). O TPC-A definia duas métricas principais, o tpsA (transação por segundo A; o "A"serve para indicar que a origem da métrica e o TPC-A e não outro benchmark qualquer) e o custo-por-tpsA.

A partir do aceite e disponibilização da especificação TPC-A no site da TPC os resultados oficiais começaram a ser publicados por grandes empresas de hardware e software, como DEC, HP e IBM, que serviram como sinalização de aprovação por parte da indústria no benchmark recém-criado.

TPC-B. O TPC-B usa o mesmo perfil de transação, requerimentos ACID, e fórmulas do TPC-A[Gra93], mas define processos geradores de transações *batch*, no lugar de emulação de terminal, para criação de transações de entrada. Dado que nesta configuração o conceito de usuário não existe, o termo "tempo de resposta"foi substituído por "tempo em permanência", que indica essencialmente quanto tempo a transação permaneceu no servidor de banco de dados. Desta forma, o TPC-B é uma simplificação ou subconjunto do TPC-A, em que a entidade usuário foi extraída, restando apenas a parte *batch*.

Os benchmarks TPC-A e TPC-B tornaram-se obsoletos a medida que seus modelos eram então considerados simples demais[LGKK93]. Tanto em relação à complexidade da transação,

funcionalidade requerida do banco e quanto à funcionalidade de comunicação já não era representativo dos sistemas OLTP atuais. Foi neste contexto que os membros do TPC perceberam a necessidade de criação de um novo benchmark, o TPC-C. Afora os já mencionados pontos negativos do TPC-A e TPC-B, o TPC-C foi concebido para resolver outras questões, tais como:

1. O uso de características cujo propósito era melhorar o desempenho do TPC-A ou TPC-B, sem ter qualquer relevância nas aplicações do mundo real.

2. A arquitetura antiquada do TPC-A, que se baseava em terminais burros. A arquitetura mais moderna naquele então era a cliente-servidor.

3. O modelo falho de precificação do TPC. Embora a presença da métrica preço-desempenho ofereça a vantagem de poder comparar sistemas diferentes, as empresas faziam uso de artifícios para baixar o preço do sistema e conseguir melhor preço-desempenho. A especificação TPC-A obriga a inclusão de custo com manutenção de cinco anos; as empresas incluíam este valor, mas consideravam condições de preço que não ocorriam no mundo real, por serem desvantajosas para o cliente, apenas com o intuito de baixar o preço-desempenho. Exemplo: dar um desconto significativo na manutenção de cinco anos considerando que o cliente pague adiantado o valor total da manutenção.

TPC-C. O benchmark TPC-C é composto por um conjunto de cinco transações de leitura e escrita, que simula as atividades encontradas em um ambiente OLTP complexo[TPC92]. O propósito do TPC-C é fornecer 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 a empresa cresce, mais armazéns e distritos precisam ser criados. Cada armazém mantém estoque para cem mil produtos vendidos pela companhia e cada um dos seus distritos atende a três mil clientes (*customers*).

O banco de dados especificado pelo TPC-C é composto por nove tabelas. A Figura 1.1 mostra essas tabelas e os seus respectivos relacionamentos. 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

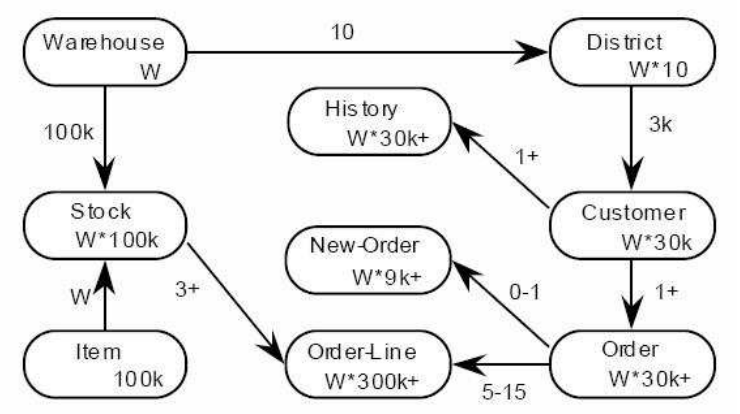

Figura 1.1 Tabelas do TPC-C e seus relacionamentos

tabela warehouse. Os números próximos às linhas de relacionamento representam a cardinalidade dos relacionamentos.

A carga de trabalho TPC-C é formada por cinco transações.

- 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.

- 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.

- 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.

- 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.

- 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.

TPC-DS. O TPC-DS é o único benchmark em estágio de desenvolvimento da TPC. Os demais benchmarks por estarem ativos e concluídos estão numa fase chamada de manutenção, tendo cada um o grupo de trabalho específico dentro da TPC. O TPC-DS visa representar o estado-da-arte dos sistemas de suporte à decisão, contando com um espectro de consultas complexas e requisitos rigorosos.

O benchmark [TPC08] modela sistemas de suporte à decisão que:

- examinam grandes volumes de dados,

- respondem questões reais de negócio,

- executam consultas de complexidades e requerimentos diversos, tais como ad-hoc, relatórios, OLAP iterativo e data mining,

- são caracterizadas por alta utilização tanto de CPU quando de Disco,

- são periodicamente sincronizadas com fontes de sistemas OLTP através de funções de manutenção.

O TPC-DS modela uma indústria que gerencia, vende e distribui produtos. Utilizando o modelo de negócio de uma grande loja de varejo tendo múltiplas lojas localizadas em toda a nação, o TPC-DS enseja ser representativo de processos tais como:

- registro de compras de clientes de qualquer canal de venda,

- modificação de preços de acordo com promoções,

- manutenção do inventário,

- criação de páginas web dinâmicas,

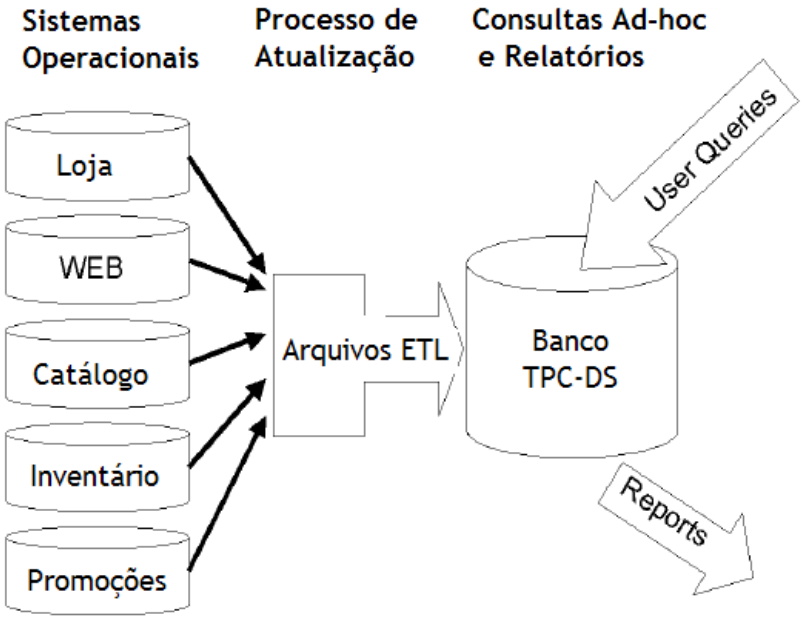

Figura 1.2 Modelo de Negócio do benchmark TPC-DS

- manutenção de perfis de clientes (Gerenciamento de Relacionamento com Clientes).

A Figura 1.2 mostra o processo de negócio no qual o TPC-DS se baseia.

A carga de trabalho do TPC-DS consiste de três disciplinas diferentes: Carga do banco de dados, Execução de Consultas and Manutenção de Dados[PNW07]. A etapa de execução de consultas é realizada duas vezes, uma antes e uma após a etapa de manutenção de dados. O TPC-DS possui 99 templates de consultas que, semelhantemente ao TPC-H, possuem variáveis de ligação que podem receber diferentes valores. O objetivo é simular um ambiente multi-usuário concorrente no qual cada usuário pode enviar diferentes parâmetros para a mesma consulta.

AS3AP. O benchmark AS3AP [TOB92] possui como características principais a medição do desempenho das principais funcionalidades de um banco de dados, e a variação de tipos de carga de trabalho aplicados. O benchmark é dividido em módulos que podem ser executados separadamente ou em conjunto. Cada módulo é responsável por testar uma certa funcionalidade. Algumas organizações de interesse público, como universidades e laboratórios de pesquisa, criaram implementações de código aberto do benchmark AS3AP, como o Open Source Database Benchmark [osd01], da Compaq Computer Corporation.

O benchmark OSDB (Open Source Database Benchmark) foi criado com o objetivo inicial de avaliar a taxa de I/O e o poder de processamento da plataforma GNU Linux/Alpha. Sua

Tabela 1.3 Testes mono-usuário do AS3AP

Nome	Descrição
selectivity 25%	seleção de 25% das linha de uma tabela
join_2	junção com 2 tabelas
proj_100	projeção de 100 linhas (operador distinct)
proj_10pct	projeção de 10% das linhas de uma tabela (operador distinct)
agg_simple_report	Agrupamento simples
agg_subtotal_report	Agrupamento complexo
refer_integrity	Tentativa de inserção de chave duplicada
update_btree	Atualização de campo chave
update_alpha	Atualização de campo alfanumérico
bulk_modify	Atualização de 1000 linhas
bulk_delete	Remoção de 1000 linhas

implementação é baseada no benchmark AS3AP, diferindo em alguns aspectos: quantidade de métricas retornadas e número de módulos. Se por um lado a análise dos resultados gerados pelo benchmark AS3AP baseia-se em uma única métrica (tamanho máximo do banco de dados suficiente para completar o teste em menos de 12 horas), o OSDB possibilita a comparação através de métricas tais quais: tempo de resposta das consultas e número de linhas retornadas por segundo.

Este benchmark é dividido em três módulos: carga e estrutura, mono-usuário e multi-usuário. O módulo de carga e estrutura inclui a criação e carga de tabelas a partir de dados armazenados em arquivos texto, além da criação de índices clusterizados e não-clusterizados (apenas B-tree). No módulo mono-usuário é testado o desempenho de seleções, junções, projeções, agregações e atualizações. A ordem de execução das consultas é definida de forma a não favorecer a utilização de dados em cache. Uma listagem dos testes se encontram na Tabela 1.3.

No módulo multi-usuário, os testes simulam perfis de carga de trabalho diferentes. Para cada tipo, um determinado número de processos é executado concorrentemente, simulando uma massa de usuários conectados. A quantidade de usuários é um quarto do tamanho do banco de dados. Os perfis de carga de trabalho incluem:

- Perfil OLTP, considerando a base de dados de 512MB, 127 usuários executam operações de atualização em uma única tabela. O outro usuário executa um conjunto de 08 (oito) consultas pré-definidas sobre uma mesma tabela;

- Perfil misto, 01 (um) usuário executa um conjunto de operações incluindo atualizações e consultas, enquanto que os demais executam o conjunto de consultas do perfil OLTP.

O AS3AP pode ser também caracterizado como um conjunto de micro-benchmarks que auxiliam na avaliação do otimizador de consultas.Um estudo comparativo entre banco de dados de código aberto, PostgreSQL e MySQL foi realizado usando o OSDB pelo autor e outros colegas [PdNS07], e foi constatado sua utilidade como ferramenta de suporte a engenharia de desempenho destes bancos a medida que foi possível identificar vários pontos de melhoria em cada SGBD.

1.3 Objetivos e Contribuições

O objetivo desta dissertação é apresentar uma ferramenta para avaliação de sistemas computacionais, particularmente, os SGBD. A idéia é colocar à disposição um ambiente para realização de testes de desempenho que catalisem melhorias ou inovações nos próprios SGBD. Embora o alvo da carga de trabalho seja o banco de dados, é importante salientar que todos os outros sub-sistemas auxiliares, sejam de software ou de hardware, são também avaliados, e, portanto, podem ser caracterizados através da ferramenta. No tocante à avaliação de desempenho, o subsistema de armazenamento é um componente importante, e vem sendo avaliado por diversos trabalhos científicos atuais, como veremos na seção de trabalhos relacionados. Neste sentido, o DBT-5 se torna uma ferramenta útil pois simula uma carga de trabalho representativa e pode ser um mecanismo importante para melhoria de tais sistemas.

Embora o DBT-5 seja correspondente à especificação TPC-E, seus resultados ou métricas não podem ser publicamente anunciados como resultados do TPC-E, ou ainda publicamente comparados com resultados oficiais do TPC. A razão é que o DBT-5 é uma implementação *fair-use* do TPC-E, dessa forma, deve-se orientar pelo documento Diretrizes do TPC[tpp08] no que tange ao uso de especificação da TPC para desenvolvimento de benchmarks não-TPC.

A idéia de implementar o TPC-E para o PostgreSQL, criando a primeira versão de código-aberto deste benchmark, foi inicialmente testada pelo autor como possível projeto do Google Summer of Code 2007. Este evento patrocinado pelo Google visa atrair desenvolvedores para a comunidade de código-aberto.

A aceitação do projeto pelo Google além de validar o conceito inicial do trabalho foi útil para viabilizar recursos importantes para implementação do projeto. O projeto foi associado a então organização OSDL, hoje *Linux Foundation*, que disponibizou acesso remoto aos seus servidores para desenvolvimento e teste do DBT-5.

Em 2007 foi produzida uma apresentação falando sobre DBT-5 para submissão no principal

congresso do PostgreSQL, o PgCon - *PostgreSQL Conference for Users and Developers*. Uma vez aceita a apresentação, o autor contou com o patrocínio do Google para custeio de todas as despesas para apresentação do trabalho no PgCon 2007 (Ottawa, Canadá).

Ainda em 2007 foi escrito o primeiro artigo sobre o DBT-5[dNWM07]. Neste mesmo ano este artigo foi aceito na sexta edição do WPerformance, evento integrante do XXVII Congresso da Sociedade Brasileira de Computação (SBC). Em 2008 um versão ampliada do artigo original foi submetida à sétima edição do WPerformance.

Ainda em 2008 uma versão ampliada e revisada foi confeccionada e submetida ao períodico *Computing and Informatics* (CAI) e encontra-se em processo de avaliação desde Junho.

Este trabalho está organizado como segue. No Capítulo 2 o Benchmark TPC-E é apresentado e sua principais características. No Capítulo 3 são apresentados a arquitetura e a implementação da carga de trabalho desenvolvida. O Capítulo 4 mostra os trabalhos relacionados à presente dissertação. O Capítulo 5 foca na validação da carga de trabalho. O Capítulo 6 apresenta um estudo de caso para mostrar a viabilidade do trabalho. Finalmente o Capítulo 7 traz um resumo do trabalho desenvolvido, mostrando como os objetivos propostos foram atingidos, as dificuldades encontradas, conclusões e recomendações para trabalhos futuros.

CAPÍTULO 2

Fundamentação Teórica

Common sense is the pitfall of performance analysis

— (Neil J. Gunther)

2.1 Avaliação de Desempenho

Os objetivos gerais da avaliação de desempenho são a modelagem, análise e síntese de sistemas e sua dinâmica. Mede-se e modela-se o comportamento temporal de sistemas reais; definem-se medidas características de desempenho e; desenvolvem-se regras que garantem a qualidade do serviço[Her01].

Embora as técnicas para avaliação de desempenho de sistemas tem recebido grande atenção pela organizações que desenvolvem hardware e software, em muitas situações a determinação das características de desempenho é adiada para uma estágio em que o custo associado às alterações do projeto se torna muito elevado. Para redução destes relativos, é importante que as alternativas e projeto sejam avaliadas durante as diversas fases do ciclo de vida dos sistemas.

Uma visão geral da avaliação de desempenho é apresentada na Figura 2.1. A Carga de Trabalho pode ser observada como o somatório de todos serviços e atividades necessárias. A Qualidade de Serviço (QdS) requerida representa os requisitos de usuário em relação a tempo de resposta, vazão, entre outros.

As restrições representam fatores de pressão que precisam também ser considerados pelo sistema, como funcionalidade, orçamento, tecnologia etc. O sistema ideal é então aquele que acomoda tais fatores de pressão e atende os requisitos funcionais e de QdS.

O conjunto de técnicas disponíveis para avaliação de desempenho pode ser classificado fundamentalmente em métodos de Medição, Simulação e Modelagem Analítica [Lil00]. A Figura 2.2 mostra uma classificação das técnicas de avaliação de desempenho.

Medições reais podem promover melhores resultados desde que o sistema já exista ou ainda possuam as ferramentas adequadas, visto que simplificações (abstrações) podem ser necessárias. Resultados providos através de medições usualmente são bem aceitos, pois são oriundos do sistema real.

Entretanto, medições de sistemas reais não são muito flexíveis, pois o objeto do estudo

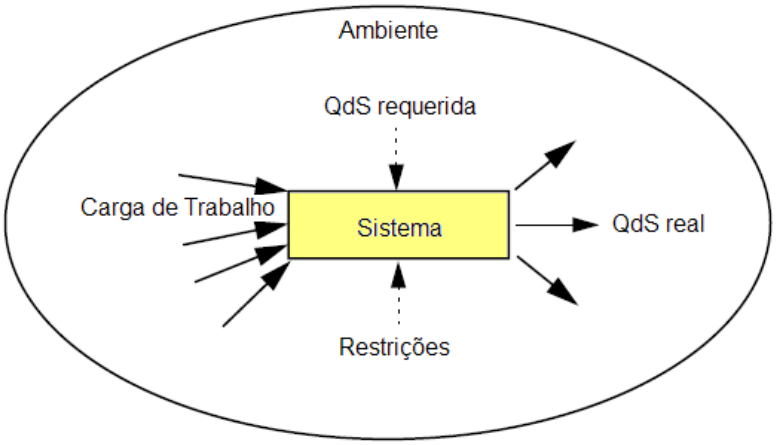

Figura 2.1 Ambiente de Avaliação de Desempenho

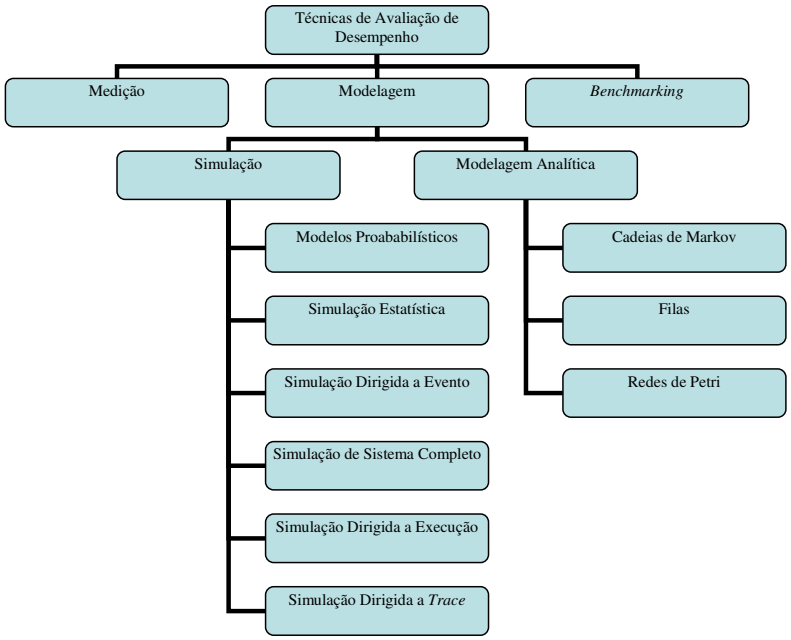

Figura 2.2 Classificação das Técnicas de Avaliação de Desempenho

é aquele sistema específico, o que pode tornar inviável a realização de certos experimentos. Dessa forma, embora a técnica de medição possa produzir resultados de maior exatidão, as dificuldades associadas ao processo podem limitar a utilização desta técnica.

A simulação de um sistema é realizada considerando uma representação (modelo) do sistema.

Dependendo da complexidade do sistema, e o nível de detalhe do modelo, da disponibilidade de um sistema real (o sistema pode estar em fase de projeto) ou ainda da dificuldade de implementação dos cenários de interesse no sistema, a simulação pode ser a única alternativa de avaliação.

O terceiro tratamento pode ser genericamente denomindao de análise. A análise também se baseia em modelos, contudo são modelos matemáticos. Os modelos utilizados para análise de desempenho podem ainda ser caracterizados como modelos analíticos (baseados em *closed forms*) ou modelos numéricos.

A escolha da técnica depende do problema a ser resolvido, de qualquer forma, o profissional de avaliação deve ter a habilidade de usar qualquer das técnicas e a experiência de saber quando aplicar a(s) mais apropriada(s).

Isto significa dizer que as técnicas podem (e devem) ser usadas em conjunto, principalmente quando principalmente quando se estar em um estágio de validação, assim como devem ser utilizadas de forma cooperativa.

Nas seções a seguir descreve-se em mais detalhes algumas das técnicas apresentadas.

2.1.1 Medição

A medição de desempenho é usada para compreender o sistema que já está construído ou prototipado. Os objetivos gerais da medição são [JEL+06]:

- realizar melhorias no desempenho do sistema (*tuning*),
- realizar melhorias na aplicação,
- validar modelos de desempenho construídos anteriormente (ex: RdP ou rede de filas),
- influenciar o projeto de futuros sistemas.

Este processo envolve as seguintes etapas principais[JEL+06]:

1. identificação de gargalos,
2. conhecimento da aplicação que roda no sistema e de como o sistema se comporta diante da carga de trabalho da aplicação,

3. melhorar o sistema para melhor atender a carga de trabalho.

A medição de desempenho corresponde à monitoração do sistema enquanto este está sob a ação de uma carga de trabalho[Jai91]. Para ter resultados significativos, a carga de trabalho então deve ser cuidadosamente selecionada.

Embora a carga de trabalho real seja a melhor escolha, em muitas situações pode não ser uma alternativa viável. Seja por questão de segurança ou para evitar perturbações no ambiente de produção, a carga real nem sempre está acessível etc. Nestes casos, outros tipos de cargas de trabalho podem ser utilizadas, tais como:

- *kernels*,
- programas sintéticos,
- benchmarks.

Tão importante quanto a escolha da carga de trabalho é a seleção de qual estratégia de medição utilizar. As diferentes estratégias de medição têm em sua base o conceito de *evento*. Um evento é uma mudança no estado do sistema; a definição precisa depende da métrica. Um acesso a disco, uma referência de memória, uma comunicação em rede são exemplos de eventos.

Pode-se classificar as métricas em categorias baseadas no tipo de evento que compreende a métrica:

- Métricas de contagem de evento. Estas métricas simplesmente contam a quantidade de vezes que um determinado evento ocorre. Exemplo: quantidade de requisições de E/S.
- Métricas de evento secundário. Tais métricas registram os valores de um parâmetro secundário quando um dado evento acontece. Exemplo: para calcular o quantidade média de requisições de E/S enfileiradas em um buffer, será necessário registrar as requisições a medida que estas são adicionadas ou removidas da fila. Assim, os eventos a serem monitorados serão as operações de enfileiramento e desenfileiramento e a métrica será a quantidade de requisições na fila.
- *Profiles*. São usados para caracterizar o comportamento de um programa ou sistema em relação a algum recurso do sistema. Um *profile* é capaz de, por exemplo, identificar em detalhes em quais operações o programa gasta mais tempo de CPU ou relatar sobre o uso da hierarquia de memória.

Diferentes estratégias de medição atendem diferentes tipo de eventos:

- Dirigida a evento. É a estratégia mais simples de todas. Registra a informação necessária para calcular a métrica sempre que um evento pré-selecionado ocorrer. O nível de perturbação da instrumentalização no sistema vai depender da freqüência do evento.

- Rastreio. É uma estratégia similar a dirigida a evento, sendo mais abrangente em sua coleta de dados. Além da contagem do evento em si, parte do estado do sistema também é percebido, de forma a caracterizar cada evento. Como conseqüência, é mais intrusivo ao sistema.

- Amostragem. Esta estratégia registra a porção do sistema necessária para determinar a métrica de interesse em intervalos fixos, independente da ocorrência de um evento. Sua perturbação está relacionada a freqüência de amostragem e não a captura do evento.

2.1.2 Benchmarking

A técnica de Benchmarking pode ser colocada entre medição e simulação, pois guarda similaridades com ambas. O contexto onde os benchmarks são utilizados torna-os interessantes devido a esta característica híbrida. Por exemplo, se alguém está interessado em saber como um computador novo se comporta, rodando uma aplicação existente, a melhor forma, como já comentado anteriormente, é rodar esta aplicação no computador. Entretanto, esta atividade nem sempre é viável, pois dependendo do tamanho da aplicação a quantidade de tempo e esforço envolvido para portar/ajustar/configurar/instalar a aplicação é substancial. Opta-se então em criar um modelo da aplicação, um programa de benchmark. Deriva-se um modelo menos complexo e mais controlável da aplicação, mas que capture suas características mais importantes.

Pode-se distinguir pelos menos dois tipos de benchmarks, embora seja possível categorizá-los de outras formas (como é mostrado adiante): benchmarks genéricos e benchmarks customizados. O exemplo dado no primeiro parágrafo desta subseção se enquadra como benchmark customizado; o benchmark é feito sob medida, para um aplicação específica.

De outro lado, os benchmarks genéricos tentam abranger domínios de aplicação, como sistemas transacionais, sistemas web e sistemas de apoio à decisão. Alguns exemplos proeminentes de benchmarks genéricos são aqueles criados pelas organização TPC e o *Standard Performance Evaluation Cooperative* (SPEC)[spe08].

Em um passado recente o uso de benchmarks genéricos era controverso dada a dificuldade de gerar benchmarks representativos[JEL+06]. A falta de regras claras e de um ambiente de testes neutro também contribuíram negativamente (mais detalhes em [HP07]). Para uma contextualização histórico-evolutiva dos benchmarks vide Capítulo 1, Seção "Evolução dos Benchmarks de Banco de Dados".

Os benchmarks abrangem diversos domínios, a Tabela 2.1 mostra os mais populares em

Tabela 2.1 Lista dos Benchmarks mais utilizados

Categoria	Sub-categoria	Benchmark
CPU	Monoprocessador	SPEC CPU 2000, PERFECT CLUB
		SciMark, ASCI
	Multiprocessador	SPLASH, NPB, ASCI
Multimedia		EEMBC benchmarks
Embutidos		BDTI benchmarks
Proces. Sinal Digital		MediaBench, MiBench
Java	Cliente	SPECjvm98, CaffeineMark, Morphmark
	Servidor	SPECjBB2000, VolanoMark
	Científico	Java Grande Forum, SciMark
Proces. Transação	OLTP	TPC-C, TPC-E
	DSS	TPC-App, TPC-H
Servidor Web		SPECWeb99, TPC-App
		VolanoMark
Servidor Correio		SPECmail2001, IMAP2003
Sist. Arq. Rede		SPEC SFS/LADDIS
PC		SYSMARK, WinBench
		MacBench

diversos tipos de cargas de trabalho.

Benchmarks também podem ser divididos entre os que utilizam um quantidade fixa de recursos computacionais, os que usam uma quantidade fixa de tempo e os que utilizam recursos e duração variáveis.

2.1.2.1 Benchmarks de Computação Fixa

No mundo físico velocidade é definida como a distância percorrida por unidade de tempo. Para sistemas computacionais, porém, não há uma definição matemática para a distância percorrida por um sistema; em vez disso, utiliza-se medidas relativas a algum sistema base.

Pela definição de uma quantidade fixa de computação pra ser executada pelo benchmark pode-se usar o tempo requerido para realizar esta computação como medida relativa de desem-

penho [Lil00].

Este conceito de fixação de quantidade de computação levou a uma consideração sobre como o desempenho geral de um sistema pode ser melhorado por mudanças em um único componente deste sistema. Gene Amdahl, em 1967, ponderou pela primeira vez sobre esta questão.

Amdahl argumentou que, essencialmente, a melhoria geral observada no desempenho de um sistema está limitado aquela porção do sistema que não é afetada por qualquer mudança feita no sistema. Tal consideração ficou conhecida como Lei de Amdahl. Uma "lição"importante desta lei é que os esforços de melhoria no sistema devem ser direcionados para as porções deste que mais são executadas, que, por conseguinte, terão maior impacto no desempenho.

2.1.2.2 Benchmarks de Duração Fixa

Os benchmarks de computação fixa são mais intuitivos dos que os de duração fixa, pois se encaixam melhor na idéia de melhoria de desempenho. Por exemplo: se um benchmark de computação fixa rodar em um sistema mais rápido é esperado que conclua a carga de trabalho proposta em menos tempo.

Há porém um classe de aplicações no qual quantificar o que é realizado num período de tempo fixo é importante, dado as grandes dimensões e complexidade do problema. Um programa científico para cálculo de previsão de tempo é um exemplo de tal aplicação.

Quando um usuário adquire um sistema mais rápido, ele espera resolver uma parte maior do problema, dada a mesma quantidade de tempo, em comparação com o sistema anterior. Dessa forma, fixa-se a duração do benchmark enquanto a quantidade de computação pode variar. Ao final do tempo alocado, a quantidade de computação realizada é medida e é usada parâmetro de velocidade relativa entre diferentes sistemas.

O benchmark SLALOM[GREC91] (Scalable, Language-independent, Ames Laboratory, One-minute Measurement) foi o primeiro a implementar a estratégia de computação variável e duração fixa. Como os autores jocosamente explicam, é incorreto perguntar quanto tempo um computador leva para rodar SLALOM, pois sua duração é sempre fixa em um minuto.

A pergunta certa a ser feita é qual o tamanho do problema que o computador irá resolver quando sob a carga do SLALOM. Esta é uma das vantagens deste tipo de benchmark, pois é auto-escalável. O tamanho do problema se ajusta ao tamanho do sistema sendo testado.

2.1.2.3 Benchmarks de Duração e Computação Variáveis

Nesta terceira estratégia de benchmarking a duração e a quantidade de computação são variáveis. Logo, a medida de desempenho de interesse envolve tanto a duração quanto a computação [Lil00].

O benchmark HINT[GS95] se enquadra nesta categoria. Criado por um dos autores de SLALOM, é considerado uma evolução deste. HINT define rigorosamente a qualidade de uma solução para um dado problema matemático (equação matemática) que deve ser resolvido pelo programa objeto do bechmark. A solução pode ser continuamente melhorada, bastando se ter mais tempo e poder computacional.

O nível de qualidade da solução e a quantidade de tempo gasto para alcançá-la são utilizados como medida de desempenho. O benchmark HINT expressa sua métrica de desempenho como QUIPS (quality improvements per second - melhorias de qualidade por segundo).

2.1.3 Simulação

Simulação é um processo pelo qual um sistema é avaliado numericamente, e os dados deste processo são usados para estimar métricas de interesse. Simulação é o equivalente computacional ao experimento laboratorial[CL99]. Em lugar dos aparatos físicos, tem-se programas de software que capturam as iterações entre os componentes do sistema. A aleatoriedade natural é substituída por um gerados de números aleatórios. Embora não seja a "coisa real", simulação apresenta-se como uma técnica eficiente e com muitos recursos.

Simulação tem a vantagem de ser menos custoso do que a construção do sistema real. Além disso, é mais flexível para medição de desempenho do que o sistema real, pois no sistema simulado pode se mudar parâmetros com mais rapidez que de outra forma seria muito difícil ou impossível num sistema real. Exemplo: mudar o tamanho do cache L2 de um processador para determinar o efeito no desempenho do sistema.

Entretanto, uma limitação da simulação está nas simplificações assumidas para modelagem do sistema real. Um conflito de escolha deve ser cuidadosamente considerado, acurácia da simulação versus o tempo requerido para programá-la e executá-la. Determinar o nível de detalhe necessário na construção de um simulador é mais uma arte do que uma ciência[Lil00].

Deve-se estar atento que o modelo de simulação seja validado através de outras técnicas de avaliação de desempenho, para que a confiabilidade dos resultados seja garantida.

Pode-se então elicitar os passos para construção de uma simulação desta forma[FM03]:

1. determinar se o problema requer uma simulação,

2. formular um modelo para resolver o problema,

3. formular um modelo de simulação do problema,

4. implementar o model,

5. projetar experimentos de simulação,

6. validar o modelo,

7. realizar experimentos.

Um projeto de simulação típico irá gastar mais tempo nos passos 2, 3 e 4. A formulação do modelo trata de definir os elementos críticos do sistema real e suas interações. Uma vez que estes elementos sejam identificados e definidos (matematicamente, comportamentalmente, funcionalmente), assim como suas interações (causa e efeito, predecessor e sucessor, dependências, fluxos de dados e fluxos de controle) tem-se a estrutura do modelo.

As linguagens de programação mais utilizadas no desenvolvimento de simuladores incluem GASP IV [APH73], baseado em FORTRAN; GPSS [Sch90] (General-Purpose Simulation System - Sistema de Simulação de Uso Geral); Simscript [KVM68]; e SLAM II [Pri86].

2.1.4 Modelagem Analítica

2.1.4.1 Redes de Petri (RdP)

A teoria inicial das RdPs foi pensada por Carl Adam Petri em 1962, na Alemanha, da sua tese de doutoramento *Kommunikation mit automaten* (Comunicação com Autômatos)[Pet62].

RdPs são uma ferramenta gráfica e matemática de modelagem que pode ser aplicada para representar diversos tipos de sistemas, fornecendo um bom nível de abstração em relação a outros modelos gráficos [Pet81].

Uma RdP também pode ser vista como um grafo orientado que permite modelar propriedades estáticas de um sistema de eventos discretos, constituído de dois tipos de nós: transições (correspondem a eventos que caracterizam mudanças de estados do sistema), e lugares (correspondem a condições que devem ser satisfeitas) interligados por arcos direcionados ponderados. Marcas podem estar associadas aos lugares da rede [Pet81].

A Figura 2.3 mostra um exemplo de uma RdP. Os lugares são representados por círculos e as transições por barras ou traços. Este dois elementos são os vértices do grafo associado às RdPs. Or vértices são interligados por arcos dirigidos. Podem haver ainda inscrições tais como nomes, marcas, expressões, guardas, entre outros.

Para cada representação gráfica de uma RdP existe uma representação algébrica equivalente. Ela contém o conjunto de lugares, transições, arcos e informações adicionais como inscrições.

Existem diversas classificações para RdPs na literatura. A seguinte classificação se encontra em [MLC96]:

- RdPs de baixo nível.
  - RdPs Binárias ou Condição-Evento.

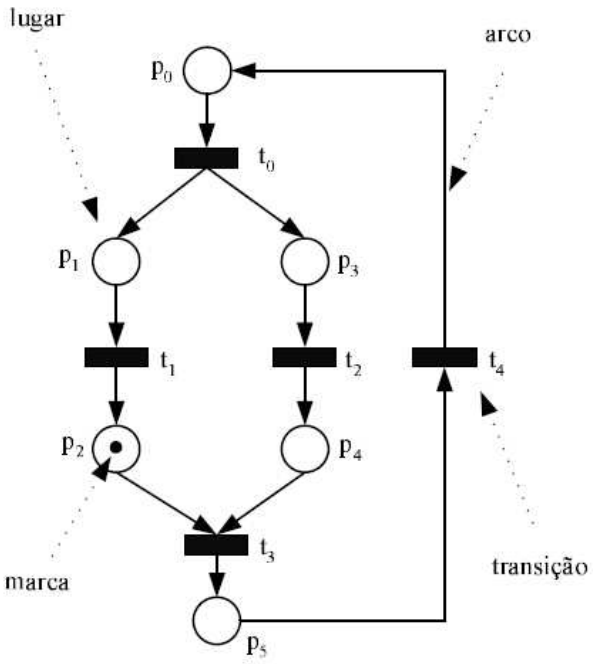

Figura 2.3 Exemplo de uma Rede de Petri

  - RdPs Lugar/Transição.

- RdPs de alto nível.

  - RdPs Predicado-Transição.
  - RdPs Coloridas.
  - RdPs Hierárquicas.

As principais diferenças entre as RdPs de alto nível e as de baixo nível: as RdPs de alto nível possuem marcas com um tipo e podem representar objetos estruturados; expressões formais (também chamadas inscrições) podem ser adicionadas aos arcos do grafo[DHP+93]. Nas RdPs de baixo nível, as marcas são tipos de dados não estruturados (ex: um inteiro não negativo).

Existem várias definições para RdPs. A definição formalmente apresentada aqui utiliza a teoria dos *bags*[Pet81]:

Definição 2.1 (Estrutura das Redes de Petri em Bags)**.** Uma RdP é uma quíntupla $N = \langle P, T, I, O, K \rangle$ onde

- $P = \{p_0, p_1, \ldots, p_n\}$ é um conjunto finito não vazio de lugares.
- $T = \{t_0, t_1, \ldots, t_m\}$ é um conjunto finito não vazio de transições, onde $P \cap T = \emptyset$.

- $I : T \rightarrow P^{\infty}$ é um conjunto de bags que representa o mapeamento de transições em lugares de entrada.

- $O : T \rightarrow P^{\infty}$ é um conjunto de bags que representa o mapeamento de transições em lugares de saída.

- $K : P \rightarrow \mathbb{N} \cup \omega$ é um vetor de capacidades associadas aos lugares. O símbolo ω é utilizado para representar que o lugar tem capacidade infinita.

RdPs têm sido amplamente utilizadas para a modelagem analítica pois:

- A representação gráfica ajuda na visualização de sistemas complexos.

- Conflitos e buffers podem ser modelados de forma fácil e eficiente.

- Travamentos do sistema (deadlocks) podem ser detectados.

- Várias extensões de RdPs permitem análises qualitativas e quantitativas da utilização de recursos, falhas, etc.

- Os modelos de RdPs representam uma ferramenta de modelagem hierárquica com bases matemáticas bem desenvolvidas.

- Permite a modelagem através de vários níveis de abstração.

- Possibilita a modelagem de sistemas paralelos e concorrentes.

- Sincronização de eventos.

- Verificação metódica de propriedades do sistema.

As RdPs de interesse para avaliação de desempenho são as *Generalized Stochastic Petri Nets* (GSPN). Esta extensão incorpora o conceito de tempo aos modelos, com isto aspectos como tempo de duração das atividades e tempos de espera são estudados. Há diversas formas de associar tempo as RdPs, sendo a mais comum aquela que vincula o tempo à transição. Existem três tipos de tempos nas RdPs: determinístico, intervalar e estocástico.

Nas GSPN o tempo é estocástico, ou seja, é modelado por funções de distribuição probabilística. A solução de um modelo GSPN se dá através de simulação exaustiva ou através da geração da cadeia de Makov (vide próxima subseção) correspondente.

2.1.4.2 Cadeias de Markov

Cadeias de Markov é uma ferramenta poderosa, flexível e eficiente para descrição e análise de propriedades de sistemas computacionais dinâmicos. Medidas como Desempenho e Dependabilidade podem ser facilmente derivadas. Além disso, a Cadeia de Markov lança a base teórica para a teoria das filas[BGdMT06].

A Cadeia de Markov constitue um tipo particular de processo estocástico com estados discretos; o parâmetro tempo, porém, pode assumir valores discretos ou contínuos. A propriedade Markoviana de interesse é a sua ausência de memória, no qual o comportamento futuro não é condicionado a fatos passados. Esta propriedade se traduz matematicamente na seguinte fórmula:

$$P(X_{t_{n+1}} \leq s_{n+1} | X_{t_n} = s_n, X_{t_{n-1}} = s_{n-1}, \ldots, X_{t_0} = s_0) = P(X_{t_{n+1}} \leq s_{n+1} | X_{t_n} = s_n).$$

É possível representar o comportamento de um sistema descrevendo todos os diferentes estados que este sistema venha a apresentar e indicando as transições possíveis de um estado para outro durante sua execução; cada transição tem uma probabilidade associada. O conjunto das probabilidade de transição usualmente são sumarizadas em uma matriz de transição estocástica, sobre a qual um exemplo será dado mais a frente.

À Cadeia de Markov está associada um conjunto de estados, sendo que apenas um único estado poderá ser assumido a cada passo; a evolução do sistema é representada por transições do processo de um estado para outro, considerando as probabilidades associadas.

Como já comentado, as Cadeias de Markov são classificadas em dois tipos (em funçao do parâmetro tempo): tempo-contínuo e tempo-discreto. As Cadeias de Markov de tempo contínuo são chamadas CTMC (*Continuous-time Markov Chains*) e as de tempo-discreto DTMC (*Discrete-time Markov Chains*).

As DTMC possuem espaço de estados discreto, finito, ou infinito contável. Graficamente uma DTMC de espaço finito é representada por um diagrama de transição de estado, um grafo finito dirigido no qual um estado i da cadeia é desenhado como um vértice, e uma transição de passo simples de um estado i para um estado j é marcado como uma seta ligando os dois estados.

Basicamente, para modelar um sistema via Cadeias de Markov o primeiro passo é construir o espaço de estados, pela identificação de todos os possíveis estados que o sistema pode assumir. Num segundo passo, as conexões entre os estados (as transições) devem ser identificadas. Em seguida, o modelo é parametrizado, isto é, as probabilidades associadas a cada transição são especificadas.

Depois destes passos teremos um modelo como mostra a Figura 2.4 e sua matriz estocástica representada na Equação 2.1.

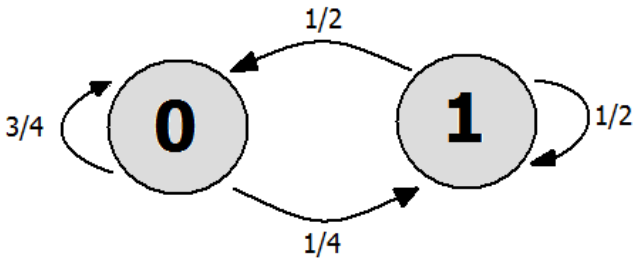

Figura 2.4 Exemplo de uma Cadeia de Markov de tempo-discreto

$$P^{(1)} = \begin{pmatrix} \frac{3}{4} & \frac{1}{4} \\ \frac{1}{2} & \frac{1}{2} \end{pmatrix} \tag{2.1}$$

Depois que o modelo é construído, é preciso resolvê-lo. A solução compreende a abstração de um conjunto de equações lineares (conhecidas como equações de Chapman-Kolmogorov) a partir do espaço de estados; a resolução destas equações visa alcançar as probabilidades estacionárias para cada estado do sistema. A partir daí é possível calculas as medidas de desempenho desejadas.

Uma vez que o modelo esteja resolvido (construído, parametrizado, validado e solucionado) este pode ser alterado para predizer o desempenho de determinada mudança no sistema.

Cadeia de Markov serve tanto para descrever um sistema quanto para predizê-lo. Por sua versatilidade a aplicabilidade dos modelos de Markov é extensa. Entretanto, também possui suas limitações. Uma de suas limitações primárias está no fato de ser suscetível a um fenômeno chamado "explosão de estados", no qual a complexidade computacional para resolver o modelo se torna muitas vezes impraticável.

2.1.4.3 Teoria das Filas

O cenário de teoria das filas pode ser descrito desse modo: um recurso ou centro de serviço é compartilhado por um conjunto de clientes que, com determinada freqüência, usam este recurso para execução de um 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 torna a teoria das filas interessante como ferramenta em diversas áreas, como, redes de comunicação, sistemas telefônicos e, particularmente, avaliação de desempenho.

Os conceitos fundamentais de teoria das filas são apresentados a seguir.

A notação matemática mais utilizada para descrever filas foi proposta por D. G. Kendall em 1953:

$$A/S/m/B/K/SD$$

- *A* (*Arrival Process* - Processo de Chegada) - O processo de chegada das requisições ao centro de serviço,
- *S* (*Service Time Distribution*) - A distribuição que determina o tempo que o centro precisa para atender as requisições,
- *m* - O quantidade de servidores, que são considerados parte de um mesmo sistema de filas caso sejam todos idênticos,
- *B* (*Buffer*) - A quantidade máxima de clientes que o sistema suporta. Esse número pode ser limitado por questões de espaço ou para limitar o tempo de espera,
- *K* - A quantidade de clientes que podem vir a solicitar serviço do sistema,
- *SD* (*Service Discipline*) - A ordem em que os clientes são atendidos. Alguns exemplos são FIFO (*First-In, First-Out* - Primeiro a entrar, Primeiro a sair), LIFO (*Last-In, First-Out* - Último a entrar, Primeiro a sair), *Round Robin* (cada cliente recebe uma mesma quantidade de tempo de serviço no centros, entre outros.

Os processos de chegada e de serviço são representados por uma letra, que indica qual distribuição de probabilidade é seguida:

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

A notação *M* se traduz por *Markovian* ou *memoryless*, para indicar que valores assumidos no passado não têm relevância. A distribuição determinística assume uma valor constante. Uma distribuição geral indica que não se sabe ao certo qual distribuição é utilizada.

Como exemplo de fila, $M/M/2/50/5000/FIFO$ 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 FIFO.

É comum abreviar a notação e considerar que não existe limitação de buffer, que a população de clientes é infinita e que o regime da file é FIFO. Desta forma, costumeiramente escreve-se apenas os três primeiros parâmetros, como $M/M/1$.

2.2 Sistemas de Arquivos Linux

Nesta seção são descritos os sistemas de arquivos utilizados no estudo de caso do Capítulo 7.

Ext2. O primeiro sistema de arquivos implementado no Linux foi o Minix FS. Possuía um sistema de arquivos de 64 MB e os nomes de arquivos eram limitados a 14 caracteres. O Ext substituiu o Minix em 1992, possuindo capacidade de 2 GB e permitindo arquivos com até 255 caracteres. Entretanto, o Ext tinha deficiências na modificação de inodes, sendo o Ext2 desenvolvido para solucionar este problema [Wik08a].

O espaço no Ext2 é dividido em blocos e organizado em grupos de blocos; cada grupo de blocos contém um superbloco, um *bitmap* de grupo de blocos, um *bitmap* de inode, seguido dos blocos de dados. O Ext2 tem capacidade de até 32 TiB quando o tamanho do bloco é 8 KiB.

Ext3. O Ext3 é um sistema de arquivo Linux com suporte à *journaling*. Sendo uma extensão do Ext2, seu principal diferencial é o *journaling*, que aumenta a confiabilidade do sistema e elimina a necessidade de checagem dos arquivos após uma parada inesperada.

Por permitir *upgrade* online a partir do Ext2, passou a ser bastante utilizado por várias distribuições Linux como sistema de arquivos *default*. O Ext3 também permite crescimento online do sistema de arquivos e o uso de árvore-H na estruturação da árvore de diretórios. Por sua compatibilidade com o Ext2, ferramentas existentes de manutenção e reparo do Ext2 também podem ser utilizadas no Ext3 com poucas mudanças [Wik08b].

O Ext3 possui três níveis de *journaling* que promovem diferentes graus de risco:

1. *Journal*. Este é o modo mais seguro de operação, no qual tanto o conteúdo dos arquivos quanto os seus metadados são escritos no *journal* antes de serem escritos no sistema de arquivo principal. Pode haver perda de desempenho neste nível pois os dados precisam ser escritos duas vezes.

2. *Ordered*. Neste nível somente os metadados são escritos no *journal*, mas é condição obrigatória que o conteúdo do arquivo seja escrito em disco antes que seus metadados sejam gravados no *journal*. Neste nível pode haver arquivos corrompidos quando o sistema pára no meio de uma operação de sobreescrit que não podem ser recuperados.

3. *Writeback*. Neste nível somente os metadados são escritos no *journal*. O conteúdo do arquivo pode ser gravado em disco antes ou depois da operação do *journal*. A possibilidade de corrupção de arquivos é mais elevada porém o desempenho esperado é mais alto.

ReiserFS. Foi introduzido em 2001 no Linux 2.4.1. ReiserFS foi o primeiro sistema de arquivos Linux a suportar *journaling*. Quando do seu lançamento, possuia caracterísitcas não encontradas em outros sistemas: *journaling* apenas de metadados, redimensionamento *online* e *tail packing*, que reduz a fragmentação interna.

O desempenho do ReiserFS no manuseio de pequenos arquivos (< 4 KiB) é uma ordem de grandeza maior do que do Ext2 ou Ext3. Por este motivo é um sistema de arquivos indicado para servidores de email, cache HTTP e outras aplicações que necessitam de desempenho em pequenos arquivos.

2.3 Escalonadores de E/S

A técnica de escalonamento de E/S consiste em classificar ou reordenar as operações de E/S antes de submetê-las ao subsistema de E/S. O escalonador é um interlocutor entre a camada de bloco e os drivers de dispositivo de baixo nível. Do ponto de vista do escalonador, a camada de bloco é um produtor de requisições de E/S e os drivers de dispositivos são consumidores [PH04]. Apesar de não ser um componente obrigatório do sistema operacional, sua presença e atuação pode melhorar significantemente o desempenho do subsistema de E/S. Pode-se dizer assim que o único propósito do escalonador é quanto ao desempenho [Rob08].

O kernel Linux atual conta com quatro escalonadores de E/S: Deadline, Anticipatory, Complete Fair Queueing e no-op, sendo possível definir qual opção utilizar durante a inicialização do sistema. Isto se dá através de *elevator flags* que são passadas ao *kernel*. O escalonador Deadline tem esse nome por que assinala a cada requisição de leitura um prazo de execução. Sua lógica garante que as requisições de leitura serão atendidas dentro de um tempo determinado. Embora as requisições de escrita não possuam prazos associados, o escalonador Deadline também objetiva garantir que as escritas enfileiradas não se privem de execução.

O escalonador Anticipator (as) tem como principal critério reduzir as operações de busca (seek). Introduz assim um componente controlado de espera no algoritmo de expedição de E/S [PH04]. O objetivo é, baseado no princípio da localidade, gerar uma espera em uma requisição de leitura permitindo que outras requisições de leitura adjacentes se enfileirem. Desta forma, esta pequena espera iria evitar operações de busca adicionais já que mais de uma requisição será atendida na mesma operação.

O escalonador no-op funciona de forma a colocar todas as requisições de E/S em um fila

FIFO simples, desordenada, implementando tão-somente a mescla de requisições. Assume portanto que o desempenho da E/S está ou será otimizada pela camada de bloco ou eventualmente por uma controladora inteligente externa. O Complete Fair Queueing (CFQ) tem como objetivo assegurar uma alocação bem distribuída da largura de banda de E/S entre as diversas requisições de E/S; desde o kernel 2.6.18, o CFQ é o escalonador padrão do Linux.

CAPÍTULO 3

TPC-E

Humanity is acquiring all the right technology for all the wrong reasons

— (R. Buckminster Fuller)

Neste capítulo é apresentada uma sinopse do TPC-E. O benchmark TPC-E exercita uma variedade de transações on-line, simulando as aplicações OLTP atuais. Por este motivo, algumas características do TPC-E, como o esquema do banco de dados e as transações, são mais complexas do que as do seu predecessor, o TPC-C. O sistema principal sob teste é o SGBD, onde as transações são executadas.

O TPC-E modela uma empresa de corretagem de ações que interage com clientes e com o mercado de ações, a bolsa de valores. Os clientes enviam ordens ou requisições a corretora; o mercado envia respostas sobre as ordens executadas, vide Figura 3.1. O banco de dados central é o repositório de dados destas três entidades, clientes, corretora e mercado. O banco de dados consiste de 33 tabelas, organizadas em quatro conjuntos identificados por Mercado, Cliente, Corretora e Dimensões, como mostra a Tabela 3.1.

O conjunto Dimensão contém dados auxiliares usados pelas outras tabelas, como endereços, códigos postais e taxas. O tamanho do banco de dados é definido em função da cardinalidade da tabela CUSTOMERS, i.e., todas as outras tabelas são carregadas tendo como parâmetro a cardinalidade da tabela CUSTOMERS. A vazão máxima alcançável também depende do tamanho da tabelas CUSTOMERS, ou, de outra forma, do tamanho do banco de dados gerado.

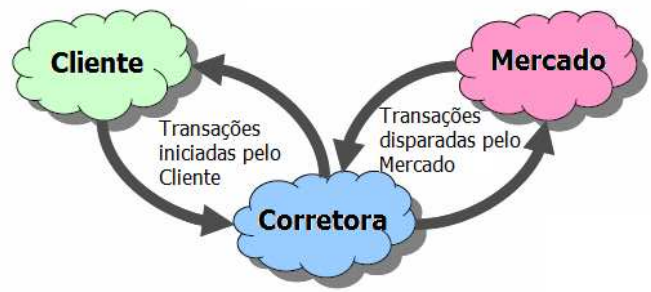

Figura 3.1 Modelo de Negócio TPC-E

Assim, para obter um desempenho determinado, a tabela CUSTOMERS deve ser carregada apropriadamente.

O TPC-E define 12 transações; 10 delas têm uma composição específica que deve ser respeitada durante a execução do benchmark. Esta composição é um valor percentual estipulado para cada transação que define sua freqüência de execução durante um teste. As transações diferem também em relação ao tipo de acesso (somente-leitura ou leitura-escrita), como mostra a Tabela 3.2.

Durante um teste, pelo menos 90% de cada transação deve ter um tempo de resposta igual ou menor ao indicado pelo atributo "90% Tempo de Resposta"da Tabela 3.2.

3.1 Esquema do Banco de Dados

Como já mostrado na Tabela 3.1 o banco TPC-E é composto de 33 tabelas organizado em 4 conjuntos. Para garantir definições de tabela uniformes entre vários SGBD, a especificação define os tipos de dados a serem utilizados. Os tipos são definidos com nomes genéricos e com uma breve explanação de forma a serem facilmente correlacionados com os tipos de dados existentes em cada SGBD. Por exemplo: CHAR(n), NUM(m,[n]) e BOOLEAN.

Também são definidos meta-tipos com o intuito de facilitar a notação na especificação. Sua implementação não é obrigatória, mas quando é feita usa-se entidades chamadas Tipos Definidos pelo Usuário, facilmente encontradas nos SGBD. Alguns exemplos de meta-tipos: IDENT_T, que é definido como NUM(11); TRADE_T, que é definido como NUM(15). As colunas em cada tabela possuem prefixos para facilitar sua identificação: AP_ para tabela ACCOUNT_PERMISSION e TH_ para TRADE_HISTORY. Também são definidos os papéis da Chave Primária e Chave Estrangeira e outras restrições, como NOT NULL.

3.2 Escalonamento e População do Banco de Dados

A quantidade de clientes da corretora define a vazão máxima da carga de trabalho. Para aumentar a vazão, mais clientes são necessários. Como já comentado, a cardinalidade da tabela CUSTOMER é a base do tamanho do banco e do escalonamento do TPC-E. O testador deve calcular o tamanho do banco baseado na vazão que se quer alcançar, para mais informações consulte a seção **Regras de Execução e Métricas** neste capítulo.

O TPC-E possui três tipos de dimensionamento de tabelas:

- Tabelas Fixas. Estas tabelas têm sempre a mesma cardinalidade independente do tamanho do banco e da vazão. Exemplo: TRADE_TYPE tem sempre 5 linhas.

Tabela 3.1 Tabelas TPC-E

Categoria	Nome da Tabela
Cliente	ACCOUNT_PERMISSION
	CUSTOMER
	CUSTOMER_ACCOUNT
	CUSTOMER_TAXRATE
	HOLDING, HOLDING_HISTORY
	HOLDING_SUMMARY
	WATCH_ITEM
	WATCH_LIST
Corretora	BROKER, CHARGE
	CASH_TRANSACTION
	COMMISSION_RATE
	SETTLEMENT
	TRADE, TRADE_HISTORY
	TRADE_REQUEST
	TRADE_TYPE
Mercado	COMPANY, COMPANY_COMPETITOR
	DAILY_MARKET
	EXCHANGE, FINANCIAL
	INDUSTRY
	LAST_TRADE
	NEWS_ITEM
	NEWS_XREF
	SECTOR
	SECURITY
Dimensão	ADDRESS, STATUS_TYPE
	TAXRATE
	ZIP_CODE

Tabela 3.2 Transações TPC-E

Transação	Acesso	Freqüência%	90% Tempo de Resposta
Trade-Order	L/E	10.1	2s
Trade-Result	L/E	10	2s
Trade-Lookup	L	8	3s
Trade-Update	L/E	2	3s
Trade-Status	L	19	1s
Customer Position	L	13	3s
Broker Volume	L	4.9	3s
Security Detail	L	14	3s
Market Feed	L/E	1	2s
Market Watch	L	18	3s
Data Maintenance	L/E	-	-
Trade-Cleanup	L/E	-	-

- Tabelas Escalonadas. Estas tabelas têm suas cardinalidades relacionadas a cardinalidade da tabela CUSTOMER. As transações podem atualizar dados nestas tabelas, mas seu tamanho permanece constante.

- Tabelas Progressivas. Estas tabelas têm suas cardinalidades iniciais relacionadas a CUSTOMER, mas aumentam de tamanho durante a execução do benchmark.

3.3 Transações

O núcleo de cada transação TPC-E roda no SGBD, mas a lógica da transação interage com vários componentes do ambiente de benchmark. Uma transação é composta do EGenTxnHarness e da invocação de um ou mais *Frames*. O EGenTxnHarness é uma peça de código de software provida pela TPC que defines a lógica da transação no que tange a sua parte fora do SGBD. O código do EGenTxnHarness não pode ser alterado. O *Frame* é a lógica da transação definida dentro do SGBD; usualmente um *Frame* é implementado como um Procedimento Armazenado dentro do banco. O que define uma Transação no TPC-E é a junção da parte contida no EGenTxnHarness com a parte contida no SGBD, como mostra a Figura 3.2.

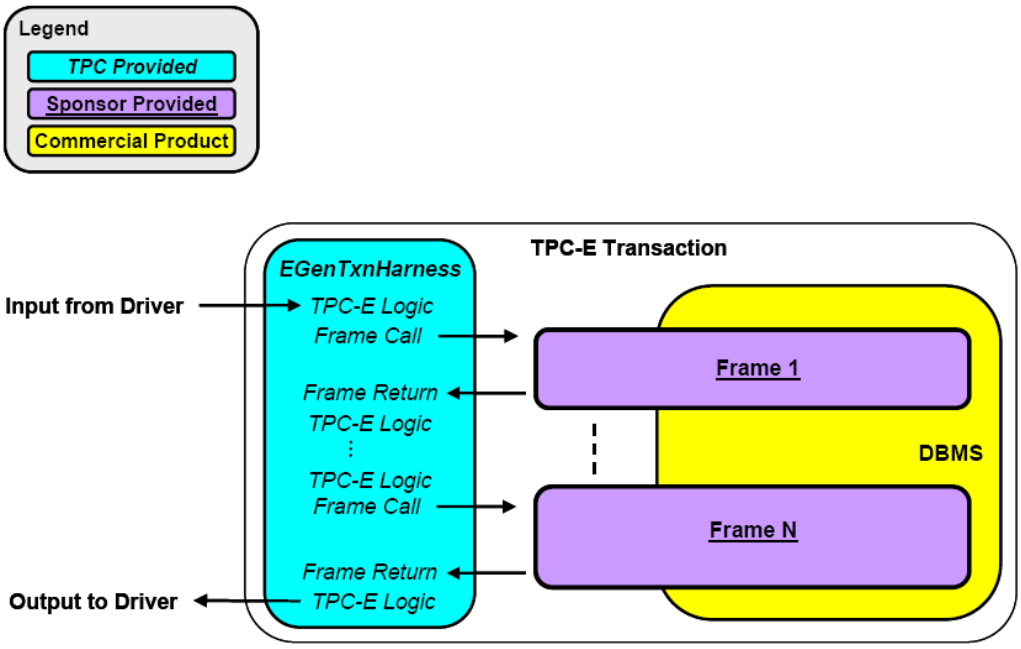

Figura 3.2 Transação TPC-E

Broker-Volume A transação Broker-Volumen foi projetada para emular um processamento atualizado realizado pela corretora para apresentar o potencial de desempenho de vários corretores. Pode-se associar tal transação à uma geração de relatório gerencial com informações de volume dos corretores.

Customer-Position A transação Customer-Position emula o processo que relata o perfil do cliente e sumariza sua situação atual na corretora baseado no valor atual do mercado para seus ativos. Esta transação representa o relatório gerado para um cliente que quer saber seu valor de mercado atualizado.

Market-Feed Esta transação emula o processo de rastreamento da atividade corrente do mercado; representa o processamento realizado na corretora que disponibiliza o preço atualizado dos ativos diretamente da bolsa de valores.

Market-Watch A transação Market-Watch emula o processo de monitoramento do desempenho global do mercado. A transação se inicia por uma entrada do cliente informando um conjunto de ativos dos quais se quer saber sua atual tendência (baixista ou altista[1]). Este con-

[1]Os termos baixista e altista fazem parte do jargão financeiro e servem para indicar, respectivamente, a tendên-

junto de ativos monitorados podem ser baseados na carteira atual do cliente, na lista de possíveis ativos a serem considerados ou de um setor particular na qual as empresas são categorizadas (ex: mineração, telecomunicações).

Security-Detail Esta transação visa representar o processo de acesso às informações detalhadas de um ativo em particular. O objetivo é emular um cliente realizando pesquisas sobre um ativo antes de tomar uma decisão sobre uma possível compra/venda.

Trade-Lookup A transação Trade-Lookup foi projetada para emular a obtenção de informação por um cliente ou um corretor que responda questões pertinentes a um conjunto de operações. Os diferentes conjuntos de operações são escolhidos de tal forma que represente uma destas atividades:

- análise geral do mercado,
- revisão de operações de um certo período no tempo (extrato de conta na corretora),
- análise de desempenho passado de um determinado ativo,
- análise do histórico de custódia de um determinado cliente.

Trade-Order Esta transação emula o processo de compra ou venda de um ativo por um cliente, corretor ou terceiro autorizado. Caso agente da operação não seja o proprietário da conta, a transação irá verificar se este tem autorização apropriada para executar tal operação. A transação permite ainda que o agente compre ou venda a preço de mercado ou limite a compra/venda a um determinado preço. A transação também estima o impacto financeiro da operação proposta pela provisão de dados de perda/ganho, impostos e corretagem. Tal informação permite que o agente avalie se é plausível operar tal ativo e então decidir-se por submeter a transação ou cancelá-la.

Trade-Result Trade-Result representa o processo de execução de uma operação no mercado de ações. Representa as seguintes atividades da corretora: recebimento da confirmação final da operação pelo mercado; e recebimento do preço do ativo. O valor em custódia do cliente é atualizado para refletir a realização da operação. Estimativas geradas de comissão ou corretagem, emolumentos e outros valores, no início da operação, são atualizadas com valores reais. Por fim, as informações sobre tal operação são armazenadas para eventual consulta de histórico.

---

cia de queda ou de elevação no preço de um ativo

Trade-Status A transação Trade-Status emula o processo de atualização do *status* de um conjunto particular de operações. Representa o cliente consultando um sumário de atividades recentes realativo a determinada(s) conta(s).

Trade-Update Esta transação representa o processo de realização de correções ou atualizações em um conjunto de operações. Quando um cliente ou corretor analisa uma ou mais operações, pode entender que pequenas correções são necessárias, como por exemplo para atualizar o nome do corretor que efetuou a operação. Vários tipos de atividades podem gerar correções ou atualizações:

- análise geral do mercado;
- revisão de operações de um certo período no tempo (extrato de conta na corretora);
- análise de desempenho passado de um determinado ativo;
- análise do histórico de custódia de um determinado cliente.

Data-Maintenance Esta transação é representativa de um ação de manutenção periódica realizada em dados secundários, Exemplo: um cliente precisa atualizar seu endereço de e-mail ou código postal. Esta transação ocorre com baixa freqüência em comparação com as demais, haja vista que tais modificações na vida real também ocorrem com baixa recorrência.

Trade-Cleanup Esta transação é utilizada pela cancelar operações pendentes do banco de dados.

3.3.1 Tipos de Operação

Como anteriormente comentado, há dois tipos de operações a serem realizadas sob os ativos: Compra (50% do tempo) e Venda (50% do tempo). Estas são decompostas em operação a preço de mercado (60%) ou com preço limitado (40%).

Para operações de mercado, os dois tipos possíveis são Market-Buy (30%) e Market-Sell (30%). Para operações com preço limitado os tipos são Limit-Buy (20%), Limit-Sell (10%) e Stop-Loss (10%).

Market-Buy e Market-Sell são operações de, respectivamente, compra e venda de um ativo a preço corrente do mercado. Limit-Buy é uma operação de compra na qual o preço do mercado está abaixo ou é igual ao preço-limite especificado. Limit-Sell é uma operação de venda na

Example Database Footprint				
Table	Column	Frame		
		1	2*	3*
CUSTOMER_ACCOUNT	CA_BAL	Reference		
	CA_C_ID	Return		
	CA_TAX_ST	Return		
HOLDING	H_PRICE		Return	
	H_QTY		Modify	
	Row(s)		Remove *	
	1 row		Add *	
TRADE_HISTORY	1 row			Add
Transaction Control		Start	Rollback *	Commit

Figura 3.3 Diagrama de Transação

qual o preço de mercado está acima ou é igual ao preço-limite especificado. Stop-Loss é uma operação de venda que ocorre quando (ou se) o preço do mercado cai em relação ao preço-limite especificado.

A especificação TPC-E define através de diagramas o que cada transação afeta nas tabelas do banco de dados. Tal diagrama representa de forma gráfica as interações requeridas pela transação para que sua função seja realizada. A Figura 3.3 representa um exemplo de um diagrama tal qual surge na especificação.

Esta representação pictórica visa facilitar a tarefa do implementador do benchmark, como referência rápida, e também serve como ferramenta didática para entendimento da função de cada transação. Como se vê na Figura 3.3, as transações são compostas de um ou mais *Frames* que condicionalmente (indicado pelo asterisco após o número do *frame*) ou obrigatoriamente são executados durante a execução da transação.

É importante salientar que até então usamos o termo transação como uma entidade lógica, que usual e fisicamente é representada por Procedimentos Armazenados (*stored procedures*) no Banco de Dados. Na implementação do DBT-5, as *stored procedures* representam os *frames* que aparecem no diagrama; ou seja, uma transação lógica qualquer é composta de uma ou mais transações físicas representadas por *stored procedures* no banco de dados. No Apêndice, pode-se encontrar a implementação das *stored procedures* em SQL. Como foi utilizado o SGBD PostgreSQL [pos08], o dialeto SQL utilizado foi o pl-pgsql [plp08].

Além dos diagramas transacionais, a especificação define ainda o pseudo-código de cada

```
{
   start transaction
   // Should return 0 to 40 rows
   select
      broker_name[] = B_NAME,
      volume[]      = sum(TR_QTY * TR_BID_PRICE)
   from
      TRADE_REQUEST,
      SECTOR,
      INDUSTRY
      COMPANY,
      BROKER,
      SECURITY
   where
      TR_B_ID = B_ID and
      TR_S_SYMB = S_SYMB and
      S_CO_ID = CO_ID and
      CO_IN_ID = IN_ID and
      SC_ID = IN_SC_ID and
      B_NAME in (broker_list) and
      SC_NAME = sector_name
   group by
      B_NAME
   order by
      2 DESC

   list_len = row_count
   commit transaction
}
```

Figura 3.4 Pseudo-código da transação Broker-Volume

Frame de cada transação. Um exemplo de pseudo-código pode ser visto na Figura 3.4. A especificação determina ainda que a implementação efetiva de tal pseudo-código seja funcionalmente equivalente a este. A verificação desta propriedade cabe ao Auditor.

O Auditor é um agente definido na especificação que objetiva validar a implementação do benchmark e também validar sua execução. Esse agente é necessário somente em validações ou publicações formais do benchmark. No caso do DBT-5, que tem como meta ser uma ferramenta de código aberto, para facilitar avaliações extra-oficiais, o Auditor foi representado por um especialista em benchmark de código aberto que auxiliou na validação das transações. Mais detalhes na seção Auditoria e Relatórios.

3.4 System Under Test (SUT) e Driver

O TPC-E é um metamodelo de um ambiente OLTP real. Para entender o escopo de avaliação do TPC-E é necessário entender o ambiente e o nível de abstração. Em ambiente real, usuários se conectam a corretora através da rede usando uma miríade de dispositivos possíveis, tais como PCs e celulares. A corretora também possui a capacidade de conectar-se a terceiros pela rede, como ao mercado de ações. A partir do ambiente real pode-se abstrair os compo-

nentes funcionais principais, como mostra a Figura 3.6. Um usuário utiliza algum dispositivo para se conectar, via rede, ao serviço de apresentação. Como é típico em um ambiente C2B (*Customer-to-Business*), a camada de apresentação proporciona: uma forma de navegação dos serviços disponíveis, seleção da operação desejada, entrada de dados e leitura de resultados. Um exemplo prático de tal cenário é um cliente usando um computador pessoal para se conectar a um sítio web para conduzir alguma operação. A corretora, por sua vez, iria também se conectar a uma companhia externa, tal como o mercado de ações. Como é típico de um ambiente B2B (*Business-to-Business*), o serviço de apresentação não é necessário neste caso. Em vez disso, os dados podem ser trocados diretamente sem a necessidade de ter um formato legível às pessas. Independente de como os dados chegam a corretora, estes irão eventualmente passar por um gerenciador de conexões, onde ocorre multiplexação/demultiplexação de conexões.

Um passo crítico na lógica de negócio ocorre quando os dados são transferidos para processamento pelo gerenciador de banco de dados. Esta transferência pode ser implementada por um código de interface com o banco de dados para que se agregue todos os dados para envio a lógica de aplicação presente no banco (exemplo: procedimentos armazenados). Por sua vez, esta lógica de aplicação que é executada sob o contexto do SGBD utiliza os serviços do SGBD para realizar as tarefas necessárias e retornar os resultados apropriados.

Por definição, o TPC-E é centrado no banco de dados. Assim, embora a camada de apresentação seja parte importante de uma solução C2B completa, esta foi abstraída da carga de trabalho TPC-E. Pela experiência prática com o benchmark predecessor, o TPC-C, nota-se que a camada de apresentação não é um limitador do desempenho, pois pode ser escalonado até que se obtenha o volume de carga de trabalho requerida. Desta forma, para focar no que realmente está sendo avaliado e para facilitar o *modus operandi* do benchmark, a camada ou serviços de apresentação (*Presentation Services*) não são parte funcional na configuração do teste.

O papel do Cliente é aquele de um decisor e provedor de dados para transações, que envolve a escolha da transação e o preenchimento dos seus campos de entrada. Porém, a ausência da camada de apresentação simplifica a emulação do usuário. Os processos de decisão e de geração de dados são ainda essenciais e são realizados pelo benchmark, mas a emulação de tempos de digitação e espera para "pensar"não são mais necessários, como ocorria no TPC-C.

Os seguintes componentes funcionais podem ser derivados a partir da abstração para compor a configuração de hardware/software do teste TPC-E, como mostra a Figura 3.5.

- Driving e Geração de Relatórios. Funcionalidade desenvolvida pelo testador para configurar, administrar e executar um teste TPC-E. Também responsável pela coleta de dados e geração de relatórios de desempenho. O código escrito pelo testador obrigatoriamente deve invocar o EGenDriver por uma interface definida pelo TPC, como mostra a Figura 3.5.

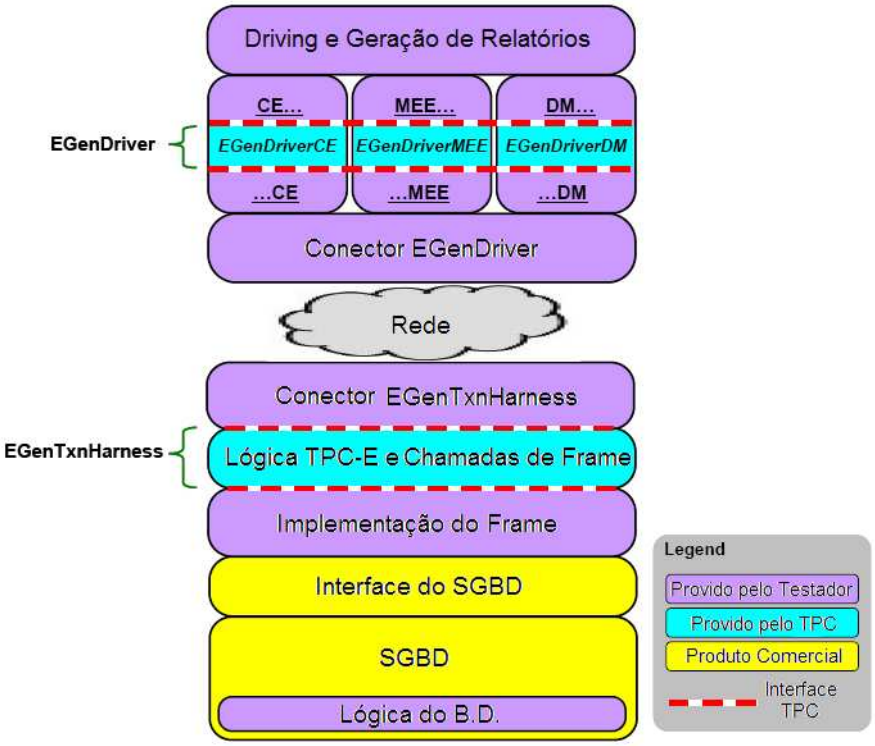

Figura 3.5 Componentes Funcionais do TPC-E

- CE (Customer Emulator). Emulador do Cliente. Funcionalidade desenvolvida pelo testador para configurar, administrar e executar o emulador do Cliente. O testador deve invocar o EGenDriverCE, que é provido pelo TPC e não pode ser alterado.

- MEE (Market-Exchange Emulator). Emulador do Mercado de Ações. Funcionalidade desenvolvida pelo testador para configurar, administrar e executar o emulador de Mercado. O testador deve invocar o EGenDriverMEE, que é provido pelo TPC e não pode ser alterado.

- DM (Data-Maintenance Emulator). Emulador de Manutenção de Dados. Funcionalidade desenvolvida pelo testador para configurar, administrar e executar uma vez por minuto o emulador de Manutenção de Dados. O testador deve invocar o EGenDriverDM, que é provido pelo TPC e não pode ser alterado.

- Interface TPC. É um classe C++ projetada para troca de dados e transferência de controle de execução.

- EGenDriver. Código-fonte C++ desenvolvida pela TPC que implementa funcionalidade essencial para execução do teste. Seu uso é obrigatório. É composto das seguintes partes: EGenDriverCE (Emulador de Cliente), EGenDriverMEE (Emulador de Mercado) e EGenDriverDM (Emulador de Manutenção de Dados).

- Conector EGenDriver. Funcionalidade desenvolvida pelo testador que adere à inteface definida pelo TPC. Este conector é invocado de dentro do EGenDriver por sua interface. É responsável por enviar os dados gerados pelo EGenDriver e receber os dados resultantes correspondentes do conector EGenTxnHarness pela rede.

- Rede. Funcionalidade provida pelo testador que deve suportar a comunicação usando um protocolo padrão de indústria (ex: TCP/IP). A presença e uso da rede é obrigatória, ainda que todos os componentes do teste rodem numa mesma máquina.

- Conector EGenTxnHarness. Funcionalidade desenvolvida pelo testador que é responsável por receber/enviar dados de/para o conector EGenDriver pela rede. Este conector disponibiliza os dados para o EGenTxnHarness pela invocação da interface TPC.

- EGenTxnHarness. Componente de software escrito em C++ provida pela TPC que contém a lógica de chamadas dos *Frames*, disponibilizando para os *Frames* as portas de entrada e saída. Seu uso é obrigatório em um teste TPC-E.

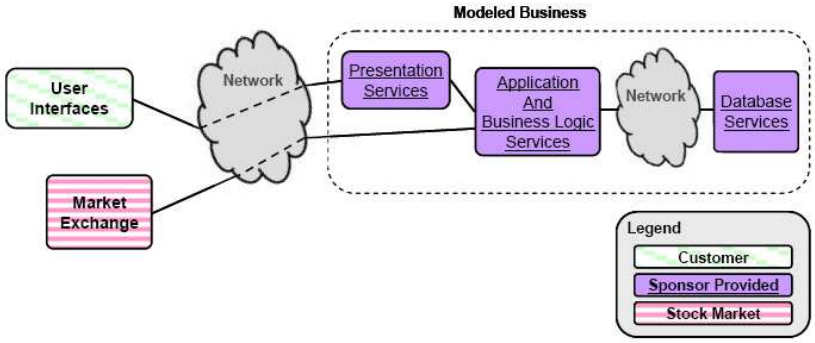

Figura 3.6 Abstração do Ambiente OLTP real

3.5 Regras de Execução e Métricas

Para publicar um resultado oficial TPC-E, os patrocinadores do teste devem incluir as métricas primárias que a especificação enseja: a taxa de vazão, expressa em tpsE (transação por segundo); o preço/desempenho, que leva em conta o custo do sistemas avaliado; e a data de disponibilidade do sistemas, que informa quando os produtos usados no teste estarão comercialmente disponíveis. O tpsE representa a quantidade de transações Trade-Result executadas por segundo durante o teste. Diferentemente do TPC-C, o TPC-E não define *layouts* de tela, nem emulação de tempo de digitação ou pensamento. A ausência destas características conduz a uma simulação mais simplificada mas não menos representativa. EGen é um pacote de software fornecido pelo TPC, que acompanha o benchmark TPC-E, projetado para facilitar a implementação deste. Os principais componentes do EGen são o EGenLoader, EGenDriver e EGenTxnHarness.

O EGenLoader é usado ara gerar dados para o banco de dados; já vem acompanhado por dois carregadores: um que gera arquivos-texto, e outro que carrega diretamente um banco de dados Microsoft SQL Server. O EGenLoader pode ser estendido para suportas a carga direta de outros SGBD. No DBT-5, o EGenLoader foi estendido para permitir uma carga direta de uma banco PostgreSQL, como será mostrado no capítulo seguinte.

O EGenDriver facilita a implementação de um *driver*, que é o componente de software que atua como coordenador da emulação. O EGenDriver possui os seguintes componentes: EGenDriverCE (emulador do cliente), EGenDriverMEE (emulador de mercado) e EGenDriverDM (emulador das transações de manutenção do banco).

O EGenTxnHarness é uma classe C++ que defines a lógica transacional no que tange ao cliente. Este componente não pode ser alterado pelos testadores. Esta lógica é usada em conjunto com as transações presentes do banco de dados que definem a lógica do servidor.

Um dos principais objetivos do TPC-E é usar dados que sejam representativos de aplicações transacionais reais. Desta forma, o banco é carregado com distribuição de dados baseado no censo americano e canadense de 2000 e nos mercados de ações da bolsa de Nova Iorque e NASDAQ. É muito importante mencionar que o TPC-E foi projetado para assegurar um ambiente de testes que não privilegie nenhum participante em particular. Isto é alcançado pela provisão do pacote de software já citado, EGen, que obrigatoriamente deve ser utilizado para implementar o benchmark. Este é um passo importante para assegurar um processo de teste justo, evitando erros cometidos por antigos benchmarks, no qual os resultados eram vistos com suspeição por serem alvos dos *jogos de benchmark* [Jai91].

3.6 Auditoria e Relatórios

Antes de um resultado TPC-E ser publicado, este precisa ser revisto por um Auditor independente certificado pela TPC. Os dois fatores, independência e certificação, são decisivos para garantir em última análise a confiabilidade da TPC no mercado. Por este motivo o processo de certificação de um auditor é rigoroso, assim como é o processo de revisão de um novo resultado. A especificação não reconhece um Auditor que tenha qualquer relação financeira com um testador além daquela reconhecida para o trabalho de revisão.

A especificação TPC-E lista os requisitos gerais de auditoria que devem ser observados para garantir que um resultado esteja aderente à especificação. Além dos requisitos gerais, podem haver detalhes adicionais particulares para cada processo de auditoria. A opinião do Auditor acerca de um resultado se dá na forma escrita, através de uma Carta de Atestação que acompanha a publicação.

O processo de auditoria pode ser dividido em seções: Banco de Dados; Transações; SUT, Driver e Rede; EGen; Regras de Execução e Métricas; Testes ACID (Atomicidade, Consistência, Integridade e Durabilidade); Preços; e Relatório Final, chamado FDR (*Full Disclosure Report*). Alguns destes itens são detalhados no Capítulo 6 desta dissertação, onde se apresenta uma validação do DBT-5, na qual foi utilizada parte do processo de auditoria; os itens não comentados são descritos aqui:

- Auditoria do EGen. Consiste em verificar se o EGen utilizado adere à especificação. Como exemplos de verificação pode-se citar: verificar a versão utilizada; verificar que o CE (*Customer Emulator*) foi implementado usando-se o EGenDriverCE; verificar se o MEE (*Market-Exchange Emulator*) foi implementado usando-se o EGenDriverMEE; verificar se o teste usa o EGenTxnHarness; verificar se os códigos providos pelo TPC não foram alterados.

- Auditoria dos testes ACID. Os Testes ACID consistem em demonstrar que o SGBD ado-

tado é capaz de garantir as propriedades requeridas. Normalmente tais testes são *scripts* que precisam ser executados numa versão reduzida do banco de dados, no qual os resultados são apurados e revistos pelo Auditor. O Auditor confere também se a implementação dos testes ACID é aderente aos requisitos da especificação.

- Auditoria de Preços. As regras para auditoria de preços constam em um documento chamado TPC *Pricing* [pri07]. A listagem de preços é item obrigatório do FDR. Dado os requisitos de preço serem comuns a todos os benchmarks da TPC, resolveu-se criar um documento apenas para este fim ao qual as especificações fazem menção.

- Auditoria do FDR. Após a realização do teste, o avaliador deve confeccionar o FDR para publicação. O Auditor revisará o relatório para constatar que suas cláusulas atendem os requisitos da especificação, e que correspondem a verdade dos fatos encontrados durante a auditoria dos testes.

O FDR é um item obrigatório para uma publicação TPC-E. É composto por vários arquivos: um relatório em formato PDF, sumário executivo em formato PDF e XML, arquivos de suporte (fontes, *scripts*, listagens, etc). O relatório PDF é composto por cláusulas no qual são mostrados os requisitos e os resultados do teste. A Carta de Atestação da Auditoria é parte integrante e obrigatória do FDR. Uma vez finalizado e auditado, o FDR é enviado ao TPC para arquivo e eventual inspeção dos membros. Os membros, através de uma lista de email fechada e específica são informados de cada resultado publicado que é enviado ao TPC e têm acesso imediato ao FDR e suas partes.

O FDR eventualmente pode ser "desafiado"por algum membro do TPC que alegue inconsistências ou falta de aderência à especificação. Há um procedimento rigoroso para tal atividade, no qual podem ser incluídos materiais que suportem a defesa/ataque do FDR dentre outras etapas.

CAPÍTULO 4

Arquitetura e Implementação

Time is a great teacher, but unfortunately it kills all its pupils
— (Hector Berlioz)

O DBT-5 foi escrito em sua maior parte em C++ e PL/pgSQL. Esta última é uma linguagem procedural do PostgreSQL. A PL/pgSQL foi usada para implementar as funções (também conhecidas por procedimentos armazenados ou *stored procedures* em outro SGBD) que definem as transações de banco de dados conforme a especificação TPC-E.

C++ foi utilizado para codificar os servidores, os emuladores, o testador de transações e o programa que carrega o banco; O DBT-5 possui mais de 14.400 linhas de código fonte sem contar a parte do código provida pelo TPC. A Figura 4.1 mostra os módulos do DBT-5. Os demais componentes foram codificados em linguagem script.

Atualmente só o Linux é suporta como plataforma de teste, porém pode ser facilmente portado para outros sistemas operacionais baseado em UNIX. O DBT-5 pode ser livremente obtido da internet a partir deste sítio web: https://svn.sourceforge.net/svnroot/osdldbt/trunk/dbt5. É necessário ter instalado um cliente subversion para efetuar o *download*.

Cinco módulos principais são responsáveis pela execução do DBT-5. São estes:

- **Test Controller** - Este módulo controla os demais componentes durante o teste, sendo a principal interface para interação com a ferramenta. Ainda que os demais módulos possam ser inicializados ou controlados individualmente, a utilização do Test Controller simplifica e automatiza os testes. Neste tipo de ferramenta, rotinas automatizadas diminuem a ocorrência de erros de *cockpit*.

- **Customer Emulator** - É uma peça central do sistema de geração de carga. Sendo responsável pela emulação dos clientes, suas atividades incluem requisição de operações, geração de entrada para transações, envio de dados e recebimento de resposta do servidor. Este módulo também executa a medição e a armazenagem dos tempos de resposta das transações do cliente.

- **Market Exchange Emulator** - Elemento do sistema de geração de carga. É responsável

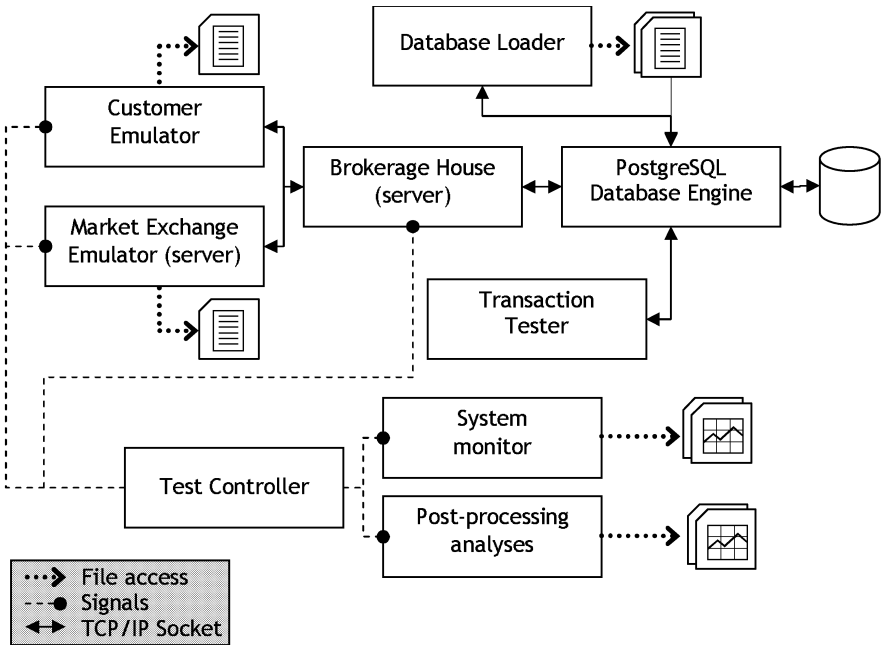

Figura 4.1 Arquitetura do DBT-5

pela emulação do mercado de ações. Suas principais atividades são: recebimento das requisições de operação da corretora, execução das operações de compra/venda, envio dos dados e recebimento de resposta da corretora. Este módulo também executa a medição e a armazenagem dos tempos de resposta das transações do mercado.

- **Brokerage House** - Este componente representa a corretora de ações. O Brokerage House recebe requisições dos emuladores, comunica-se com o gerenciador de banco de dados e envia resposta de volta aos emuladores.

- **Database loader** - O carregador do banco de dados é parte integrante do EGen (para mais informações sobre o EGen vide Capítulo 3). O carregador do DBT-5 é uma extensão do carregador do EGen. Com este carregador é possível popular um banco PostgreSQL sem necessidade de geração de arquivos intermediários, o que torna o processo de carga mais rápido.

Os módulos são descritos a seguir com mais detalhes. A comunicação entre os módulos também é descrita em uma seção posterior.

4.1 Test Controller

O Test Controller é um script bash responsável por iniciar o DBT-5. Suas principais opções são: quantidade de clientes, duração do teste, quantidade de usuários e semente para geração de números aleatórios. Este módulo executa o teste em quatro estágios.

1. Inicialização do servidor da Corretora
2. Inicialização do servidor do Mercado
3. Inicialização do emulador de Cliente
4. Processamento dos resultados do teste

Uma das limitações do Test Controller é não operar em um ambiente de teste com múltiplas máquinas. O script pode inicializar os módulos somente quando estes encontram-se em uma mesma máquina. Entretanto, os módulos por si podem trabalhar em máquinas diferentes, e o teste pode ser executado em um ambiente com múltiplas máquinas sem a mediação do Test Controller.

4.2 Customer Emulator

A funcionalidade do Customer Emulator (CE) é provida por dois submódulos. As funções do submódulo principal são providas pelo EGenDriverCE, não podendo ser alteradas, e as do submódulo periférico são programas por cada implementação particular do TPC-E. No DBT-5, o submódulo periférico é responsável pela coleta de estatísticas de tempos de resposta das transações e pela comunicação entre as partes envolvidas no teste (*System Under Test* - SUT e *Driver*).

Tanto o CE quanto o MEE (visto no próximo item) coletam os tempos de resposta das transações em um arquivo que será processado posteriormente para gerar as métricas e o sumário do teste.

O EGenDriverCE é responsável pelas funções que independem da plataforma de teste, tais como decidir qual a próxima transação a ser executada e a geração de dados para as transações.

A Tabela 4.1 mostra as opções de linha de comando do executável DriverCustomerMain, que implementa o CE. As opções são passadas via flag para o executável, não há interface gráfica disponível. Em termos de implementação, o CE é um arquivo executável, implementado como um servidor C++ *multithread* que escuta uma determinada porta TCP definida pelo usuário.

Tabela 4.1 Opções de linha de comando do Customer Emulator

Opção	Valor Default	Descrição
-e		Atalho para arquivos de entrada EGen
-c	1000	Número de clientes configurados
-a	1000	Número de clientes ativos
-h		Endereço de rede do Brokerage House
-b	30000	Porta de escuta TCP do Brokerage House
-o		Diretório de saída
-f	500	Fator de escalonamento
-d	300	Número de dias populados no banco
-t		Duração do teste (em segundos)
-s	1000	Quantidade de μs entre criação de threads
-u		Quantidade de usuários
-p		Quantidade de μs entre envio de transações
-g		Semente para o gerador de números aleatórios

Tabela 4.2 Opções de linha de comando do Market Emulator

Opção	Valor Default	Descrição
-s		Localização do arquivo Security.txt
-c	1000	Número de clientes configurados
-a	1000	Número de clientes ativos
-l	30010	Porta do socket TCP de escuta a ser usada
-h		Endereço de rede do Brokerage House
-b	30000	Porta de escuta TCP do Brokerage House
-o		Diretório de saída

4.3 Market Exchange Emulator

O Market Exchange Emulator (MEE) segue um linha de implementação similar ao CE. Assim, também pode ser dividido em dois submódulos, um provido pelo EGen, EGenDriverMEE, e outro implementado em particular.

É importante salientar que diferentemente do CE, o MEE não atua somente como consumidor, mas também como servidor. Por representar o Mercado de Ações, o MEE deve estar preparado para receber operações da corretora. Em termos de implementação, o MEE é um arquivo executável, implementado como um servidor C++ *multithread* que escuta uma determinada porta TCP definida pelo usuário.

A Tabela 4.2 mostra as opções de linha de comando do executável DriverMarketMain. As opções são passadas via flag para o executável, não há interface gráfica disponível.

4.4 Brokerage House

O Brokerage House (BH) é composto das seguintes partes: conector *Driver*-SUT, EGenTxnHarness, *frames* e a interface com o SGBD. O EGenTxnHarness é provido pelo TPC e faz chamadas aos frames implementados pelo testador e os *frames* são as transações TPC-E. A interface com o banco de dados é usada na comunicação com o SGBD. No DBT-5, como foi utilizado PostgreSQL como *backend*, a API C++ libpqxx foi escolhida para implementar a comunicação com o banco. O BH também é implementado como um servidor C++ *multithread* que escuta uma porta TCP especificada pelo usuário.

A Tabela 4.3 mostra as opções de linha de comando do executável BrokerageHouseMain.

Tabela 4.3 Opções de linha de comando do Brokerage House

Opção	Valor Default	Descrição
-s	localhost	Servidor de banco de dados
-d	dbt5	Nome do banco de dados
-o	.	Diretório de saída
-p	5432	Porta do PostgreSQL
-l	30000	Porta do socket TCP de escuta a ser usada

As opções são passadas via flag para o executável, não há interface gráfica disponível.

4.5 Database Loader

O Database Loader gera e carrega os dados no banco de dados de teste. Este componente gera a quantidade correta de linhas baseadas em regras definidas na especificação TPC-E. O EGenLoader é parte do EGen, foi arquitetado para gerar todos os dados necessários para um teste e ainda permitir que o testador personalize a função de carga do banco de dados. Desta forma, o EGenLoader pode ser estendido para popular diferentes SGBD.

O EGen fornece uma classe template C++ chamada CBaseLoader que pode ser usada para extender o carregador padrão. A funcionalidade de geração de dados não pode alterada haja visto que independe da função de carga. No DBT-5, uma classe derivada do CBaseLoader foi criada para suportar a carga direta de uma banco de dados PostgreSQL. O carregador de arquivos-texto que acompanha o EGen pode ainda ser utilizado caso se queira popular um banco via arquivos-texto.

A Tabela 4.4 mostra as opções de linha de comando do executável EGenLoader. As opções são passadas via flag para o executável, não há interface gráfica disponível.

4.6 System Monitor e Post-processing Analyzer

O System Monitor é responsável pela coleta de dados de desempenho dos recursos do sistema, incluindo discos, memória e processadores, durante o teste. Os dados coletados são armazenados em formato texto para serem processados após o teste, gerando gráficos e relatórios. Os gráficos são úteis para analisar o desempenho do sistema. Eles ajudam o testador a entender o comportamento do sistema quando está sob ação da carga de trabalho, a achar gargalos e

Tabela 4.4 Opções de linha de comando do Database Loader

Opção	Valor Default	Descrição
-b	1	Valor ordinal inicial do cliente
-c	1000	Número de clientes (para esta instância)
-t	1000	Número total de clientes no banco
-f	500	Fator de escalonamento
-w	300	Número de dias para popular o banco
-i		Diretório para os arquivos de entrada
-m	OVERWRITE	Modo de saída dos arquivos flat
-o		Diretório dos arquivos de saída
-s	localhost	Servidor de banco de dados
-d	dbt5	Nome do banco de dados
-p	5432	Porta do PostgreSQL
-x	-x	Gera todas as tabelas
-xf		Gera apenas tabelas de tamanho fixo
-xd		Gera apenas tabelas que escalonam/crescem
-xs		Gera apenas tabelas de escalonam
-xg		Gera apenas tabelas que crescem

a ajustar o sistema. Este módulo foi escrito em linguagem script e usa a ferramenta gnuplot para gerar os gráficos; outras ferramentas são utilizadas para coleta de dados de desempenho do sistema: vmstat, iostat e OProfile.

O OProfile é usado para traçar um perfil estatístico de freqüência e duração das chamadas do sistema, com intuito de otimizar o sistema. O OProfile trabalha com o *kernel* Linux e também com aplicações de usuário e pode gerar as seguintes informações: um sumário de símbolos que inclui bibliotecas, um grafo com chamadas do sistema e código anotado. Estas informações são úteis para determinar o que o sistema está executando e que caminho o código está tomando durante a sua execução.

O Post-processing Analyzer processa os arquivos-log gerados pelos emuladores durante o teste. Ele executa cálculos estatísticos baseado nos tempos de resposta das transações e gera gráficos que descrevem o comportamento destas. Um gráfico de vazão, visto na Figura 4.2, também gerado pelo DBT-5, serve como principal instrumento para visualização da métrica TRTPS. Este módulo foi escrito em perl e utiliza o gnuplot para gerar os gráficos.

4.7 Comunicação

Sockets TCP/IP são usados como meios de comunicação entre o servidor e os emuladores, e entre o servidor e o SGBD. Os arquivos binários criados para cada um destes componentes não precisam estar na mesma máquina, pois a comunicação pode ser mantida pela rede. Desta forma, pode-se criar diferentes cenários de testes onde os componentes do sistema são distribuídos em diferentes máquinas ou agrupados em uma mesma máquina.

A especificação TPC-E define uma série de interfaces que obrigatoriamente são usadas para derivar classes que governam a comunicação entre cada emulador e o servidor. O TPC disponibiliza as seguintes classes-interface: CCESUTInterface, CMEESUTInterface and CDMSUTInterface. No DBT-5, CCESUT, CMEESUT e CDMSUT, respectivamente, foram as classes geradas a partir destas interfaces. A CCESUT é utilizada pelo Emulador de Cliente, a CMEESUT pelo Emulador de Mercado e a CDMSUT pela transação Data-Maintenance.

Em comparação com o TPCC-UVa [LP06], que é uma ferramenta de software que implementa o benchmark TPC-C, nossa arquitetura tem um diferencial importante. O TPCC-UVa utiliza memória compartilhada para comunicação entre os módulos. Embora seja eficiente, esta abordagem limita a distribuição dos componentes, pois os módulos devem estar localizados na mesma máquina.

A abordagem cliente-servidor do DBT-5 usando *sockets* TCP/IP permite a criação de um ambiente de testes no qual os módulos podem residir em máquinas diferentes. Esta abordagem oferece pelo menos duas vantagens: mitiga a interferência da instrumentação da carga de traba-

```
                              Response Time (s)
 Transaction                %    Average :    90th %        Total        Rollbacks      %
-----------------       -----  ---------------------  -----------  ---------------  -----
Trade Order             10.10      0.072 :     0.057         7490               74   1.00
Trade Result            10.08      0.151 :     0.113         7416                0   0.00
Trade Lookup             8.10      0.655 :     0.696         6006                0   0.00
Trade Update             2.15      0.405 :     0.417         1594                0   0.00
Trade Status            18.94      0.137 :     0.132        14045                0   0.00
Customer Position       12.84      0.492 :     0.479         9522                0   0.00
Broker Volume            4.95      0.026 :     0.016         3670                0   0.00
Security Detail         13.84      0.117 :     0.084        10263                0   0.00
Market Feed              1.00      0.269 :     0.262          741                0   0.00
Market Watch            17.94      0.116 :     0.154        13304                0   0.00
Data Maintenance          ---      0.143 :     0.226           59                0   0.00

2.06 trade-result transactions per second (TRTPS)
40.0 minute duration
0 total unknown errors
5 second(s) ramping up
```

Figura 4.2 Principal Relatório do DBT-5: TRTPS

lho no SUT e melhora a caracterização de um ambiente típico de aplicação de banco de dados, devido a presença da rede.

4.8 Acesso Não-Uniforme

Além de ser uma ferramenta para avaliação de desempenho, o DBT-5 pode ser usado para estudar alguns aspectos do benchmark TPC-E *per se*, como foi mostrado no Capítulo 5 em relação a outros benchmarks da TPC.

O TPC-E preconiza o acesso não-uniforme as tuplas, i.e., dentro de uma relação, algumas tuplas são mais referenciadas do que outras. Por exemplo, na tabela CUSTOMER é esperado que alguns clientes sejam mais exercitados do que outros, para que seja possível modelar o comportamento de uma aplicação OLTP real.

Para verificar tal propriedade, foram coletadas um milhão de ocorrências do campo de identificação do cliente, C_ID da tabela CUSTOMER. Estes números foram criados pelo gerador de números aleatórios não-uniforme que acompanha o EGen. Na Figura 4.3 é mostrada a distribuição de freqüência de C_ID; a não-uniformidade de acesso é visível, pois alguns C_IDs são mais acessados do que outros.

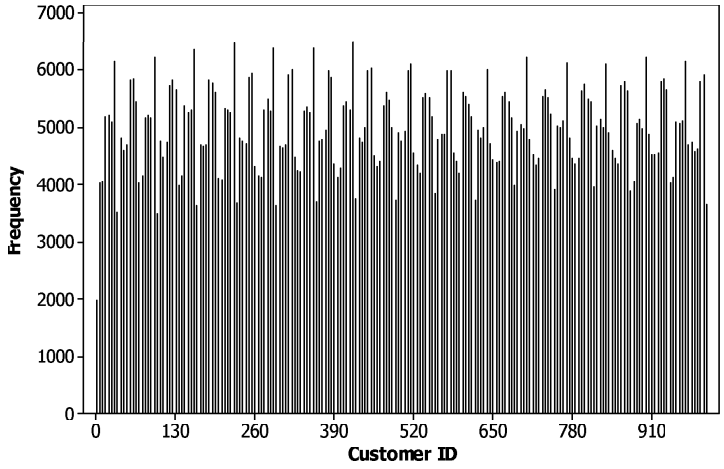

Figura 4.3 Freqüência de Distribuição de C-ID, idenitificador do cliente

CAPÍTULO 5

Trabalhos Relacionados

When the only tool you have is a hammer, every problem begins to resemble a nail

— (Abraham Maslow)

Tendo como base o levantamento bibliográfico realizado, o DBT-5 [dNWM07] é a primeira ferramenta de código aberto a implementar o benchmark TPC-E. Pode-se argumentar que a causa principal da inexistência de outras ferramentas é o pouco tempo que se passou desde sua introdução no mercado. É fato, porém, que os benchmarks da TPC são objetos constantes de estudos científicos, dada as características importantes que estes possuem, tais como: confiabilidade, maturidade e aceitação como padrão pela indústria. O trabalho que mais se assemelha ao DBT-5 é o TPCC-Uva[LP06], ferramenta criada na Universidade da Valladolid baseado no benchmark TPC-C.

O TPCC-UVa é uma implementação não oficial e não auditada do benchmark TPC-C. O propósito do TPCC-Uva é ser utilizado como um benchmark de pesquisa para a comunidade científica. Escrito em C e, assim como o DBT-5, também utilizando o SGBD PostgreSQL como *backend*, tem como plataforma principal o Linux. Porém, pode ser facilmente portado para plataformas correlatas. Seu código-fonte pode ser obtido livremente a partir do sítio web do projeto [Lla08].

A implementação do TPCC-Uva inclui todas as características descritas na especificação TPC-C, exceto suporte para comparações preço/desempenho. A razão para isso é que o TPCC-UVa tem como objetivo ser usado para medições de desempenho em ambiente acadêmico somente.

A arquitetura do TPCC-Uva é apresentada na Figura 5.1. A implementação do TPCC-UVa é composta de cinco módulos que colaboram para executar todas as atividades necessárias para e avaliação de desempenho do sistema em teste. Estes módulos são:

- Benchmark controller. Este módulo interage com o usuário, carregando o banco e permitindo a execução de diferentes experimentos.

- Remote Terminal Emulator (RTE). Para cada terminal ativo há um processo RTE correspondente durante a execução do benchmark. O RTE simula a atividade de um terminal

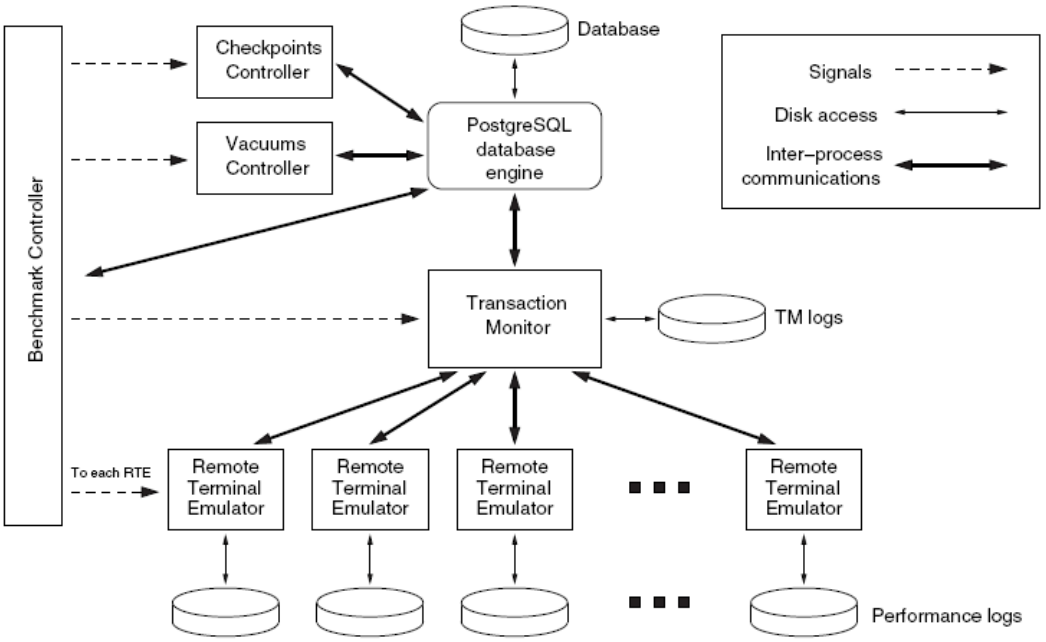

Figura 5.1 Arquitetura do TPCC-UVa

remoto, de acordo com as especificações do TPC-C

- Transaction Monitor. Este módulo recebe as requisições dos RTEs e executa as consultas na camada do SGBD.
- Checkpoints Controller. Este módulo executa *checkpoints* periodicamente no SGBD e registra os tempos de início e fim de cada *checkpoint*.
- Vacuum Controller. Este módulo evita a degradação produzida pelo fluxo contínuo de operações pelo SGBD através de chamadas freqüentes ao programa de vacuum do PostgreSQL.

Em comparação com o DBT-5, a arquitetura e implementação do TPCC-UVa apresentam algumas desvantagens. O TPCC-UVa usa memória compartilhada para comunicação intra-módulo. Apesar de habilitar uma comunicação rápida e eficiente, concentra obrigatoriamente todos os módulos em um única máquina hospedeira.

A abordagem cliente/servidor do DBT-5 cria um ambiente de teste em rede, via sockets TCP/IP, que proporciona a hospedagem dos módulos em diferentes máquinas. Os seguintes benefícios podem ser listados pela inclusão deste ambiente de rede no teste: mitigação da interferência da instrumentação da carga de trabalho no sistema em teste, promovendo assim resultados mais confiáveis e a caracterização típica de um ambiente de aplicação de banco de dados devido a presença da rede.

Em [BCD+00]; o benchmark TPC-W[TPC99c] é caracterizado e implementado em Java como uma coleção de servlets; este estudo tem sua continuação em [CRML01] onde outros achados do estudo e caracterização do TPC-W é apresentado. O TPC-W é atualmente um benchmark obsoleto.

O TPC-W tem como objetivo medir o desempenho de ambientes web transacionais, que servem páginas web de conteúdo estático e dinâmico, processamento transacional on-line e suporte à decisão. O TPC-W modela um livraria online que implementa as seguintes atividades: busca, navegação, carrinho de compra, compra segura, registro de clientes e atualizações administrativas. A linguagem Java foi escolhida pelos autores pois oferece portabilidade e servlets. O estudo foi realizado nas arquiteturas PowerPC e Sparc ISA. Embora a especificação TPC-W imponha uma série de requisitos que uma implementação deve atender (isto é verdade para todos os benchmarks TPC), há liberdade de escolha em relação a implementação. No estudo citado os autores usaram servlets Java para implementar a lógica de aplicação.

Em [CRU07], um benchmark de máquina virtual baseado no TPC-C é apresentado. Usa-se a tecnologia de virtualização para realizar benchmark, testes e análises. A idéia consiste de usar computadores de grande capacidade computacional para atuarem como hospedeiros de diversas máquinas virtuais. O sistema de benchmark criado visa testar o monitor de processamento de transações (TPM). O TPM é um middleware projetado para gerenciar a execução de um transação. O objetivo é executar transações que satisfaçam as propriedades ACID (Atomicidade, Consistência, Integridade e Durabilidade). Para descobrir se um novo TPM é robusto e rápido o suficiente para um ambiente de produção, testes de benchmark antes de sua adoção são essenciais. Neste contexto o trabalho propõe um sistema de análise e benchmarking para TPM baseado no TPC-C. Para testar e avaliar um software tão sofisticado um ambiente de hardware e software precisaria ser montado, a solução baseada em virtualização visa mitigar esta complexidade. Os autores, assim, tentaram desenvolver o ambiente virtual que melhor caracterizasse o ambiente real através de testes com diversos ambientes e configurações.

A suíte de benchmarks da antiga OSDL (Open Source Development Labs), hoje Linux Foundation (www.linux-foundation.org), são baseados nos benchmarks da TPC. São eles o OSDL-DBT-2 [Lab08], OSDL-DBT-3 [Lab08] and OSDL-DBT-4 [Lab08]. Estas cargas de trabalho possuem um esqueleto similar no qual procuram automatizar a execução dos testes, geração de relatórios de desempenho e criação do banco de dados. Embora sejam baseados nos respectivos benchmarks da TPC, seus resultados não podem ser comparados com estes, pois não foram auditados pela TPC.

- O OSDL-DBT-2 é baseado no TPC-C. Foi desenvolvido para ser executado com SAP DB, mas atualmente pode ser executado com outros SGBD, como PostgreSQL e MySQL.

- O kit de teste OSDL-DBT-3 é baseado no TPC-H e foi desenvolvido originalmente para

o SAP DB mas também pode ser executado com o PostgreSQL. A ferramenta de geração de dados é baseado no TPC-H DBGEN e o gerador de consultas é baseado no TPC-H QGEN.

- O OSDL-DBT-4 é baseado no TPC-App v1.1 e foi originalmente desenvolvido para Axi2 e a plataforma Geronimo.

Em [BKL06], o TPC-W foi utilizado para demonstrar as ferramentas propostas para teste de aplicações de banco de dados. As ferramentas são HTDGen, HTTrace e HTPar. HTDGen é um gerador de bancos de dados de teste que considera o esquema do banco e as consultas que a aplicação irá executar. A ferramenta é capaz de "entender"comandos SQL embutidos na aplicação de banco de dados e extrair dados significativos para testar a aplicação; o HTTrace é responsável pela execução do teste. Dado um banco de teste já criado, uma seqüência de requisições é gerada para o teste. O HTTrace é uma ferramenta de captura e *replay* capaz de executar testes de regressão *black box*; O HTPar é responsável pela execução de testes paralelos. O HTPar é uma extensão do HTTrace, de forma que testes possam sem executados concorrentemente em uma ou mais máquinas. O TPC-W foi utilizado no estudo de caso que o trabalho. O processo consistiu de: extração pelo HTDGen de algumas declarações SQL do TPC-W para geração do banco; e execução de testes com o HTTrace e HTPar.

Em [VT06] é criado uma versão reduzida do TPC-H. Dado que o processo de *benchmarking* pode ser complicado, custoso e demorado (principalmente quando se usa simulação) é importante reduzir o tempo necessário para executar benchmarks. Com este objetivo, os autores mostram que das 22 consultas originais do TPC-H um subconjunto formado por seis (6) transações é muito representativo e que reduz o tempo de execução do benchmark original em aproximadamente 60%.

Um dos principais passos para o sucesso no procedimento de criar um subconjunto de um benchmark é a categorização ou agrupamento de benchmarks que possuam características similares. Considerando cada uma das 22 consultas do TPC-H como um benchmark individual, o trabalho de agrupamento consiste em categorizar as 22 consultas em grupos, de acordo com uma métrica de similaridade escolhida. Usa-se uma técnica de agrupamento estatístico chamada K-means para determinar os agrupamentos. Após a etapa de agrupamento, escolhe-se um benchmark dentro de cada grupo que melhor represente aquele grupo. No procedimentos de escolha, leva-se em consideração o benchmark que tenha o menor de tempo de execução possível e que seja representativo do grupo.

Os benchmarks da TPC são bastante utilizados, tanto pela indústria quanto pela academia. Há uma miríade de estudos científicos que utilizam os benchmarks da TPC quando necessitam de um modelo que caracterize sistemas transacionais, sistemas de suporte à decisão e sistemas web com o objetivo de suportar uma nova técnica, metodologia ou ferramenta.

Pode-se citar [FGP03], no qual o TPC-W é utilizado para avaliar otimizações em servidor web multiprocessados; em [FGP04] os mesmos autores do trabalho anterior realizam um estudo de desempenho de memória em sistemas multiprocessados utilizando o TPC-W. [WA01] utiliza o TPC-C para mostrar que a alocação dinâmica de páginas de memória melhora a taxa de acerto na memória local em multiprocessadores CC-NUMA. Em [MSAHB05] o TPC-C é utilizado para demonstrar os benefícios de um nova política de priorização preemptiva de *locks* em cargas de trabalho transacionais. Este estudo também faz uma análise estatística detalhada de *locks* do TPC-C.

[VAT+04] propõe uma metodologia para avaliação de desempenho de memória em sistemas multiprocessados, neste contexto utiliza o TPC-C num servidor multiprocessado para demonstrar a eficácia da metodologia proposta. Em [FS07], Fedorova e Seltzer descrevem um novo algoritmo de sincronização de sistema operacional que melhora o desempenho do TPC-C e outras cargas de trabalho em sistemas chip multiprocessados (CMP). Em [DLM08] é apresentada uma abordagem para classificar os parâmetros de configuração de um banco de dados. Por este método, os parâmetros que apresentam os melhores efeitos no desempenho de uma determinada carga são classificados nas primeiras colocações. Os autores utilizam o TPC-H junto com o PostgreSQL para demonstrar a eficácia deste novo método.

Há ainda trabalhos que focam no estudo do benchmark em si, tentando caracterizar algum aspecto ou comportamento de interesse: [BLSZ04] caracteriza o comportamento de acesso de dados do TPC-C; [KK00] caracteriza a E/S em fase do TPC-H; [CDL01] caracteriza as consultas do TPC-H em um processador AMD; [ZSF+04] sintetiza o comportamento de I/O das consultas do TPC-H. Em [ZZS+02] o TPC-H é caracterizado estatisticamente em um ambiente de cluster Linux considerando a interação entre o engenho do banco de dados e o sistema operacional. [HSY01] apresenta um trabalho detalhado e completo sobre o comportamento lógico de E/S em cargas de trabalho reais e de alguns benchmarks da TPC, TPC-C e TPC-D.

CAPÍTULO 6

Validação

Nas palavras do sábio há favor, mas ao tolo os seus lábios devoram.
—ECLESIASTES DE SALOMÃO

De acordo com a especificação TPC-E, um resultado oficial TPC-E deve ser revisado por um auditor independente certificado pela TPC. A organização TPC certifica um auditor por meio de um processo que avalia sua habilidade de verificar quão aderente um resultado se encontra em relação à especificação. Mais detalhes sobre o processo de certificação pode ser encontrado no documento TPC chamado TPC Policies, que pode ser livremente obtidos do site do TPC (www.tpc.org).

Embora o DBT-5 não permita a publicação oficiais de resultados TPC-E, pode-se aproveitar os requisitos de auditoria presentes na especificação para validar o trabalho. A intenção não é certificar um resultado em particular, mas coletar evidências suficientes que suportem a assertiva de que o DBT-5 é baseado na especificação TPC-E.

A validação apresentada aqui se divide em quatro partes: validação do banco de dados, validação das transações, validação do *driver* e validação das regras de execução. Parte da validação é baseada nos resultados e parte no próprio DBT-5. Uma validação formal pela TPC não é requerida pois o objetivo deste trabalho é criar uma implementação não comercial do benchmark TPC-E.

6.1 Verificação de Requerimento - Banco de Dados

O esquema do banco de dados, os requisitos de tipos de dados, a estrutura requerida para cada tabela e as restrições impostas a cada coluna foram validados.

- O banco de dados foi carregado por dados gerados pelo EGenLoader. O EGenLoader foi estendido para permitir uma carga direta de um banco de dados PostgreSQL, o que é permitido pela especificação.

- A cardinalidade das tabelas criadas pelo DBT-5 no banco de dados inicialmente populado atendeu os requisitos da especificação. Um banco de 5000 clientes foi criado e

populado e as cardinalidades de cada tabela foram definidas de acordo com o requerido pela especificação.

- Todas as chaves-primárias e todos as restrições do tipo *check* foram mantidas pelo banco de dados. Cada tabela tem definida apenas uma chave-primária; as colunas que fazem parte da chave-primária não podem receber valores nulos.
- As 9 tabelas no conjunto Cliente tinham todos os atributos especificados, como descrito na especificação TPC-E, cláusula 2.2.4 [TPC07].
- As 11 tabelas no conjunto Mercado tinham todos os atributos especificados, como descrito na especificação TPC-E, cláusula 2.2.6 [TPC07].
- As 4 tabelas no conjunto Dimensão tinham todos os atributos especificados, como descrito na especificação TPC-E, cláusula 2.2.7 [TPC07].
- Os tipos de dados usados para implementar os atributos atendem os requerimentos da especificação. Foram utilizados tipos de dados nativos do PostgreSQL para string de caracteres, valores numéricos inteiros e fracionários (com ou sem sinal) e data.
- Para facilitar a notação, foram utilizados meta-tipos que por sua vez são implementados através dos tipos de dados nativos do PostgreSQL.
- A tipo de dado LOB (*Large Object*) na tabela NewsItem foi implementada pela inclusão específica de um atributo do tipo `text` nesta tabela.

6.2 Verificação de Requerimento - Transações

A implementação das transações atendeu os requisitos da especificação. O código-fonte e os scripts usados para implementar e carregar as transações podem ser obtidos deste sítio web: https://svn.sourceforge.net/svnroot/osdldbt/trunk/dbt5.

- A implementação de cada transação é aderente aos seus respectivos parâmetros de entrada, saída, diagrama de transação do banco de dados, e requerimentos de implementação.
- Os *frames* foram implementados sem deliberadamente evitar referências a elementos de dados estáticos ou pouco usados no banco de dados.
- Os *frames* não trocam dados diferentes daqueles especificados como entrada e saída para cada *frame*, e que são utilizados para comunicação com o EGenTxnHarness.

- A implementação de cada *frame* é funcionalmente equivalente ao pseudo-código associado a cada *frame*. Pode-se provar tal assertiva pela execução de cada transação individual, comparando-se em seguida a saída deste com a gerada pela execução manual do pseudo-código dadas as mesmas entradas. Os dados de entrada foram gerados pelos seguintes componentes do EGenDriver: EGenDriverCE, que disponibiliza a geração de entrada de dados do Cliente; EGenDriverMEE, que disponibiliza a entrada de dados do Mercado; e EGenDriverDM, que disponibiliza a entrada de dados para a transação Data-Maintenance. Um banco com 5000 clientes foi criado e carregado para esta verificação. Todas as saídas estavam corretas quando comparadas com aquelas geradas pelo pseudo-código. O resultado desta verificação pode ser encontrado no Apêndice A.

6.3 Verificação de Requerimento - *Driver*

A implementação do *Driver* utiliza obrigatoriamente componentes de software providos pelo TPC, são estes o EGenTxnHarness, o EGenDriverCE, o EGenDriverMEE e o EGenDriverDM. Estes componentes não podem ser modificados e servem para garantir que as funcionalidades essenciais do TPC-E sejam preservadas, independentemente da implementação.

- O DBT-5 usou o EGenTxnHarness na confecção do *Driver*. O EGenTxnHarness foi utilizado para invocar os *frames*, fornecendo (recebendo) os parâmetros de entrada (saída) destes.
- O Emulador de Cliente utilizou o EGenDriverCE para geração de dados de entrada das transações e para manter o *mix* de transações durante a execução do teste.
- O Emulador de Mercado utilizou o EGenDriverMEE na geração de dados provenientes do mercado.
- A transação Data-Maintenance utilizou o EGenDriverDM na sua geração de dados.

6.4 Verificação de Requerimento - Regras de Execução

Obrigatoriamente o *mix* de transações de um teste deve atender os requisitos da especificação. Pode-se demonstrar pela execução de um teste que os percentuais apresentados no final do teste atende tais requerimentos.

Para este teste, foi usada uma máquina com 1 Intel Xeon CPU 3.00GHz, 3 GB de memória RAM, sistema operacional Linux Gentoo 2006.1 com o kernel 2.6.19-r5, e o SGBD

Tabela 6.1 Validação do DBT-5 pelas regras de execução do TPC-E

Transação	% mix Espec.	% mix obtido	90°% TR Espec	90°% TR obtido
Trade-Order	10.1	10,1	2s	0,05s
Trade-Result	10	10,08	2s	0,11s
Trade-Lookup	8	8,1	3s	0,69s
Trade-Update	2	2,15	3s	0,41s
Trade-Status	19	18,94	1s	0,13s
Customer Position	13	12,84	3s	0,47s
Broker Volume	4.9	4,95	3s	0,01s
Security Detail	14	13,84	3s	0,08s
Market Feed	1	1	2s	0,26s
Market Watch	18	17,94	3s	0,15s

PostgreSQL 8.2.1. Após exexcutar o DBT-5 em um banco de dados carregado com dados referentes a 1000 clientes, o Test Controller coletou todas as saídas em um diretório numerado (que é automaticamente criado e assinalado para cada teste). O relatório principal é mostrado na Figura 4.2 que mostra o tempo de resposta médio, o número de transações executadas, o número de transações desfeitas (*roll back*) e o nonagésimo percentil cara cada transação.

Aproximadamente 1% das transações Trade-Order obrigatoriamente são desfeitas por determinação da especificação. O *mix* de transações é mostrado na coluna %. Pela revisão da tabela 3.2 e comparando as colunas equivalentes na Figura 4.2, pode-se concluir que o teste encontra-se dentro da faixa requerida de valores para cada transação em relação ao % *mix* e ao nonagésimo percentil[1].

Para melhor visualização, a comparativo dos números da especificação versus os número obtidos é mostrado na Tabela 6.1. O símbolo TR que aparece na Tabela significa Tempo de Resposta.

[1] 90% dos tempos de resposta encontram-se igual ou abaixo do valor requerido

CAPÍTULO 7

Estudos de Caso

Vencer a si próprio é a maior das vitórias.

—PLATÃO

Este capítulo apresenta os resultados de dois experimentos executados com o DBT-5. O primeiro estuda empiricamente o desempenho de alguns sistemas de arquivos executando sob os escalonadores de E/S disponíveis no kernel linux 2.6. O segundo experimento gera um *trace* da atividade de E/S da carga de trabalho e o executa em um simulador de disco.

7.1 Desempenho de Sistemas de Arquivos

Neste experimento foi utilizado um Pentium D 3.40GHz, executando Gentoo 2007 i686, kernel 2.6.23 em uma configuração com disco único. Uma breve introdução sobre os sistemas de arquivos (Ext2, Ext3 e ReiserFS) e escalonadores de I/O (*Deadline*, *Anticipatory*, no-op e CFQ) utilizados nos testes é apresentada no Capítulo 2. Estudos similares foram conduzidos usando cargas de trabalho de data warehousing e servidor de correio [WHRP06], e servidor web e servidor de arquivos [PH04].

A Figura 7.1 mostra o comparativo de desempenho entre os três sistemas de arquivos e os quatro escalonadores de E/S. Três testes foram executados para cada tupla sistema de arquivos, escalonador. As linhas no topo da figura representam a vazão média das três execuções por escalonador.

A linha tracejada representa Ext3, a linha contínua representa Ext2 e a linha pontilhada representa ReiserFS. O melhor desempenho foi alcançado pela tupla Ext3, Deadline. ReiserFS obteve o pior desempenho dentre os sistemas de arquivos testados com o DBT-5. Como já comentado, o ReiserFS lida bem com pequenos arquivos, porém não trata de forma eficiente arquivos grandes, que é o caso nestes testes.

O banco de dados de teste possuia 2,5 GB de tamanho, 83 % dos dados estavam contidos em 10 arquivos de mais de 200 MB de tamanho médio. Ext3 superou Ext2 em 8% e ReiserFS em 18% para todos os escalonadores. Um teste realizado pela Oracle, usando uma carga de dados OLTP, apresentou resultados similares para Ext2 e Ext3 [Ora04]. Considerando somente os escalonadores, os resultados foram mais próximos; Deadline está somente 2% acima do

Tabela 7.1 Tempo de Resposta Médio - Escalonadores de E/S

Escalonador	TR(todos)	TR(EXT3)
CFQ	0.573	0.522
Deadline	0.580	0.537
no-op	0.612	0.563
AS	0.618	0.566

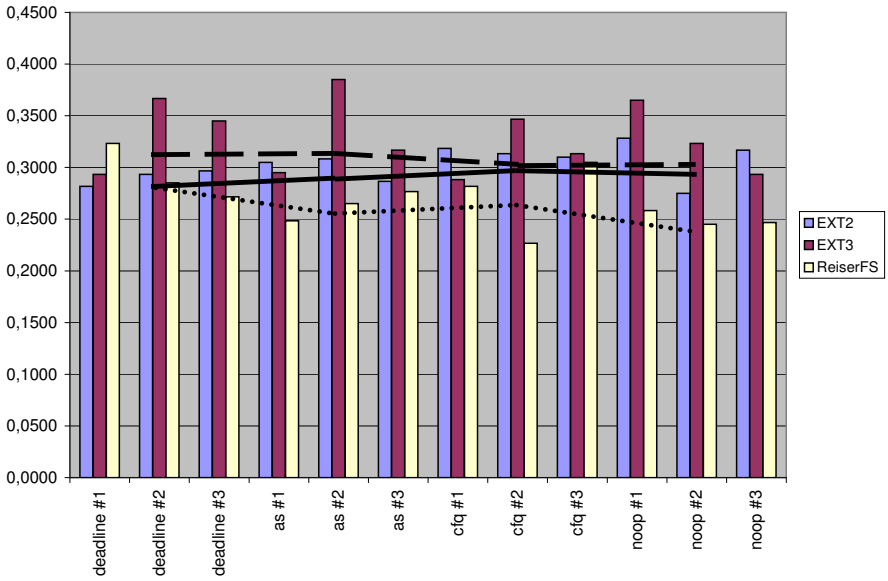

Figura 7.1 Vazão. Escalonadores de E/S e Sistemas de Arquivos - Eixo Y: *Trade-Result Transactions Per Second* (TRTPS)

CFQ, 3% acima de AS e 4% acima de no-op.

As Figuras 7.2-7.7 mostram o Tempo de Resposta (TR) em segundos por transação, escalonador de E/S e sistema de arquivos.

Para uma melhor visualização, os resultados foram divididos em transações leves e pesadas em relação aos TRs obtidos nos testes, onde

- Transações leves são aquelas que tiveram TR < 0,35s, a saber, Market-Watch, Trade-Result, Security-Detail, Data-Maintenance, Market-Feed, Trade-Order e Broker-Volume,
- Transações pesadas são aqueles que apresentaram TR >= 0,35s, a saber, Trade-Lookup, Customer-Position, Trade-Update e Trade-Status.

Ext3 obteve melhor desempenho que o Ext2 também em termos de TR. O TR médio dentre

os três testes, sendo consideradas todas as transações, em relação aos escalonadores de E/S, como mostrado na Tabela 7.1, evidencia um resultado diferente da vazão, possivelmente porque o resultado da vazão leva em consideração apenas a transação Trade-Result.

O resultado relativo ao Ext3 corrobora com o apresentado em [PH04], no qual testes com um único disco e uma única CPU foram realizados com diferentes cargas de trabalho.

7.2 *Tracing*

Neste experimento foram utilizados três componentes de software: uma carga de trabalho de banco de dados (DBT-5), um *tracer* (blktrace) e um simulador de disco (DiskSim). O blktrace[Axb] foi escolhido para ser o *tracer* de E/S de baixo nível devido a sua integração com o kernel Linux desde a versão 2.6.17.

Antes do blktrace foi testado o lltrace e o pacote lttng [Des06]. O lttrace funcionava apenas com o kernel 2.4 e não estava sendo mais mantido, enquanto que o lttng rodava no kernel 2.6 mas, apesar de muito esforço, não foi possível torná-lo operacional em nosso ambiente. Foi tentado diagnosticar ainda o problema com o criador do lttng, mas ainda assim não foi possível progredir com este pacote.

A ferramenta blktrace é executada em nível de bloco, é capaz de capturar um amplo espectro de operações e comandos de E/S dos processos sendo executados no sistema para um determinado dispositivo de E/S. O formato de saída do blktrace teve de ser ajustado para se tornar compatível com o formato de entrada do DiskSim. O DiskSim[GWP98] é um simulador de disco configurável e versátil que pode ser utilizado no lugar de um subsistema de E/S real dada sua eficiência.

Os três componentes foram utilizados num paradigma produtor-consumidor: a carga de trabalho alimentava o *tracer*, que por sua vez alimentava o simulador de disco. O teste consistia de executar o DBT-5 em um sistema de E/S real enquanto a atividade de E/S era capturada com o auxílio do processo blktrace. Durante a execução do DBT-5, o blktrace capturou mais de 40.000 operações de E/S. Um quarto componente de software utilizado foi o SGBD PostgreSQL. O *tracer* foi programado para capturar somente a atividade de disco referente ao processo do SGBD, desprezando todas as outras operações.

Uma vez que a operação de *tracing* foi concluída, o DiskSim foi configurado para utilizar como entrada o arquivo gerado pelo *tracer*. Em seguida, o DiskSim foi executado para simular o atendimento das operações de E/S e ao final gerar o tempo de resposta das requisições. O gráfico de dispersão da Figura 7.8 apresenta os tempos de resposta das requisições de E/S como observado pelo DiskSim durante a simulação.

Este experimento demonstra como um pesquisador pode utilizar o DBT-5 para estudar o

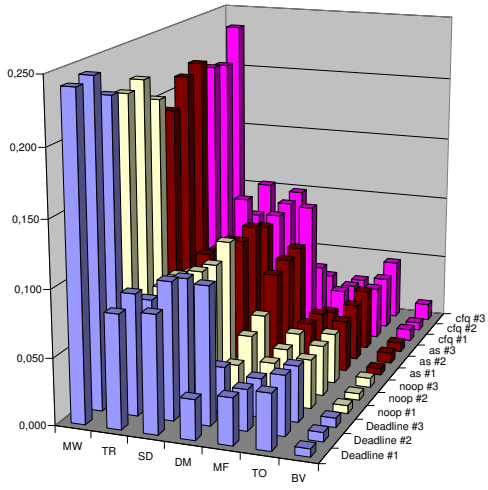

Figura 7.2 EXT2 - Tempo de Resposta Transações Leves

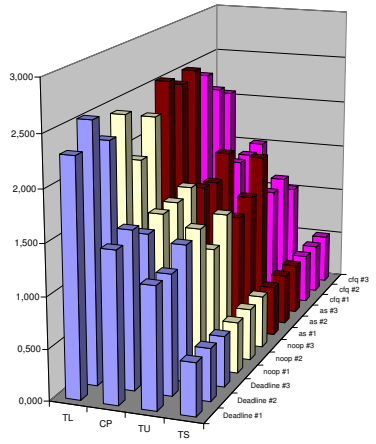

Figura 7.3 EXT2 - Tempo de Resposta Transações Pesadas

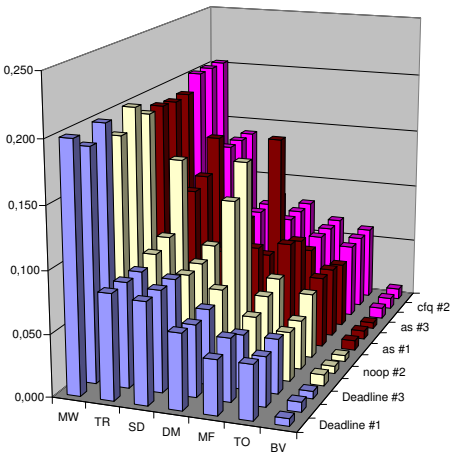

Figura 7.4 EXT3 - Tempo de Resposta Transações Leves

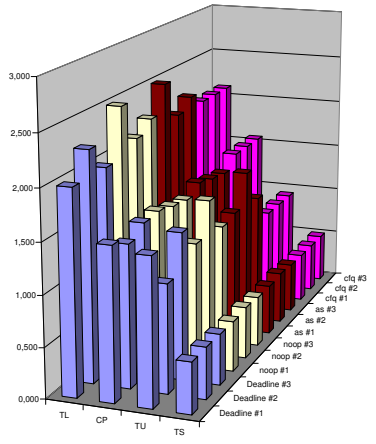

Figura 7.5 EXT3 - Tempo de Resposta Transações Pesadas

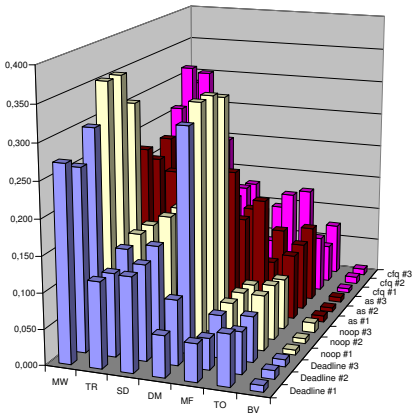

Figura 7.6 ReiserFS - Tempo de Resposta Transações Leves

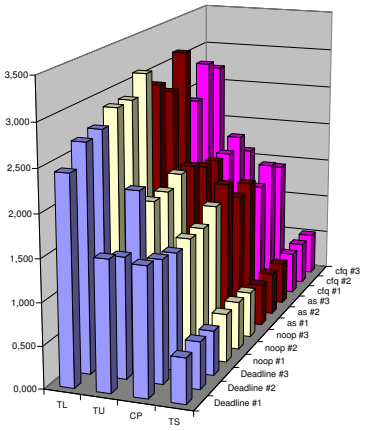

Figura 7.7 ReiserFS - Tempo de Resposta Transações Pesadas

comportamento de E/S dos sistemas de processamento de transação online.

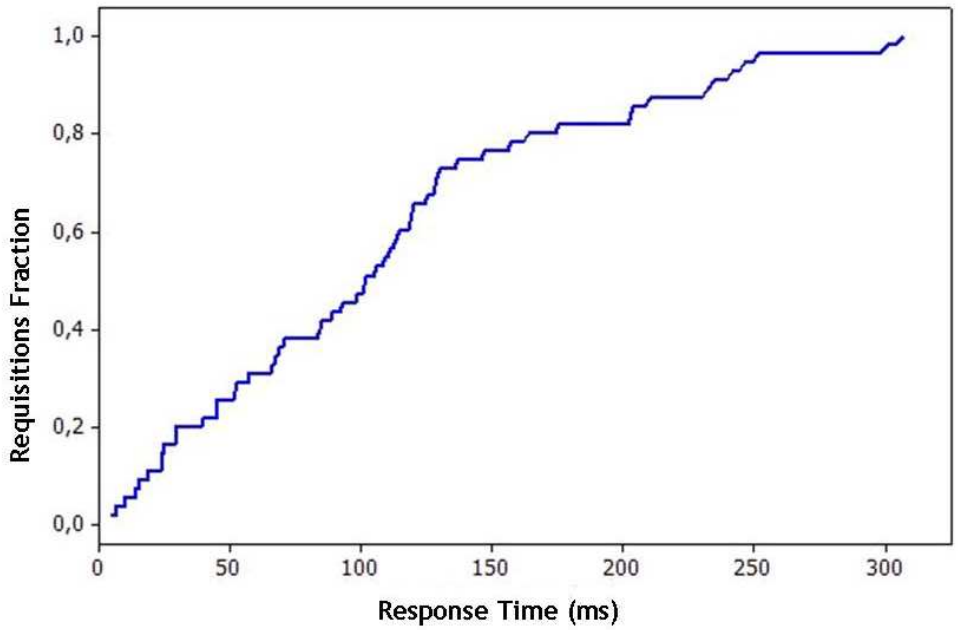

Figura 7.8 Requisições de E/S

CAPÍTULO 8

Conclusão

No mercado competitivo atual, as empresas não podem dispender recursos em sistemas computacionais incapazes de suster de maneira satisfatória seus negócios. Desta forma, escalabilidade e desempenho assumem papéis importantes na decisão de investimento em sistemas computacionais. Neste processo decisório, os administradores necessitam de instrumentos que reportem o desempenho dos sistemas de forma que comparações racionais possam ser feitas entre diferentes sistemas.

Neste contexto, de avaliação de desempenho de sistemas, vários fatores precisam ser considerados para que as conclusões sejam acertadas, tais como, onde serão executadas as avaliações, se na própria empresa ou se serão confiadas a terceiros; e a carga de trabalho ou benchmark que será empregado na avaliação. Embora o ideal seria testar o(s) possível(eis) sistema(s) a ser(em) adquirido(s) dentro do ambiente da empresa, executando aplicações reais de produção, há casos em que não é viável fazê-lo, dados os recursos e o tempo investidos na realização dos experimentoss. Tal atividade não é trivial e pode requerer pessoal especializado que as empresas não possuem ou não estão interessadas em contratar.

Os fornecedores dos sistemas observando esta demanda por parte das empresas, criava e publicava seus próprios números, mostrando-se sempre superior aos concorrentes. Em vez de ajudar o mercado no processo decisório, tal atividade, que ficou conhecida como guerra de benchmarks, gerou desconfiança e descrédito pois não era possível identificar de forma justa e clara os pontos positivos e negativos de cada fornecedor.

Na tentativa de resolver o problema, os fornecedores se uniram e criaram organizações neutras que fossem responsáveis tanto pela confecção dos benchmarks quanto pela geração de políticas que garantissem o uso adequado dos resultados produzidos pelos benchmarks. O TPC e o SPEC são exemplos de tais organizações. Dentre os benchmarks recentes criados pela TPC pode-se citar o TPC-DS e o TPC-E.

O TPC-E é um dos benchmarks do TPC que modelam uma carga de trabalho de processamento de transação *online*. O TPC-E simula um sistema de corretagem de ações com múltiplos tipos de transações e foi projetado para balancear o uso de disco e de processamento.

Os benchmarks também são ferramentas importantes para os próprios fornecedores de sistemas, pois podem ser utilizados na engenharia de novos produtos ou para diagnosticar pontos negativos dos produtos atuais. Os fornecedores (ex: Oracle, IBM, Sun) executam estes bench-

marks internamente como parte do processo de qualidade de seus produtos. Os fornecedores de sistemas de código aberto também necessitam de tais ferramentas, mas existe a dificuldade de encontrar implementações abertas de benchmarks padrões de indústria.

Neste trabalho foi apresentado o DBT-5, uma implementação de código aberto do benchmark TPC-E, que tem como objetivo colocar à disposição da academia e da comunidade de código aberto um ambiente para realização de testes de desempenho de sistemas computacionais, para promover melhorias ou inovações nestes sistemas. Por prover uma carga de trabalho transacional centrada no SGBD, o DBT-5 é particularmente interessante para avaliação de SGBD. O DBT-5 suporta apenas o SGBD PostgreSQL, porém foi projetado para ser extendido para outros SGBD.

Embora o alvo da carga de trabalho seja o SGBD, é importante salientar que todos os outros sistemas são também avaliados, e, portanto, podem ser caracterizados através da ferramenta. Dada a crescente disparidade de velocidade entre o processador e o sistema de E/S, este último se torna um componente crítico para melhora global de desempenho e vem sendo avaliado por diversos trabalhos científicos atuais que utilizam benchmarks na avaliação, como foi mostrado no Capítulo 5 - Trabalhos Relacionados.

A arquitetura do DBT-5 visa permitir a alocação dos seus componentes em diferentes máquinas. Diferentemente de outra solução proposta (TPCC-Uva, baseado no TPC-C, vide Capítulo 5 - Trabalhos Relacionados) a abordagem do DBT-5 permite distribuir os emuladores de forma a não sobrecarregar a máquina-alvo. A implementação do DBT-5 permite automatizar a realização de experimentos e a coleta e geração de resultados.

A ferramenta foi aplicada num estudo de desempenho dos *schedulers* de E/S do Linux executando os sistemas de arquivos EXT2, EXT3 e ReiserFS. O estudo mostrou que para a carga de trabalho proposta o *scheduler* Deadline apresentou melhor desempenho junto com EXT3. Este resultado corrobora com outros estudos encontrados na literatura, como apresentado no Capítulo 7.

No segundo estudo de caso foi mostrado como o DBT-5 pode ser aplicado para analisar o comportamento de E/S dos sistemas OLTP (que o TPC-E e DBT-5 representam). Foram utilizadas ferramentas de captura de E/S e um simulador de disco para realizar um replay da atividade coletada.

Como trabalhos futuros, pretende-se extender o DBT-5 para contemplar outros SGBD (ex: MySQL, Firebird). Isto aumentará a aplicabilidade da ferramenta e atenderá a uma grupo importante de usuários de outros SGBD. Em relação ao PostgreSQL, ao qual a ferramenta já oferece suporte, pretende-se criar versões dos procedimentos armazenados em outras linguagens procedurais, como PL/Tcl e PL/Perl, e também em *Embedded SQL in C* (ECPG).

APÊNDICE A

Tabelas de Validação

A entrada, saída e semente de número aleatório utilizados são mostrado nesta seção. Cada transação é dividida em frames, que vão de 1 a 6. Os frames representam uma forma de separar a transação lógica em unidades de execução menores, que podemos chamar de procedimentos. Cada procedimento requer uma entrada ou conjunto de entradas (Campos de Entrada/Valores) e apresenta um resultado (Campos de Saída/Valores) após o processamento. A semente é utilizada para gerar valores de entrada e foi escolhida aleatoriamente.

Trade-Result: Semente = 160710521

Frame 1 entrada:

Campo	Valor
trade id	1454535

Frame 1 saída:

Campo	Valor
acct id	3341
charge	4.5
hs qty	0
is lifo	0
symbol	ATL
trade is cash	1
trade qty	400
type id	TMS
type is market	1
type is sell	1
type name	Market-Sell

Frame 2 entrada:

Campo	Valor
acct id	3341
hs qty	0
is lifo	0
symbol	ATL
trade id	1454535
trade price	23.09
trade qty	400
type is sell	1

Frame 2 saída:

Campo	Valor
broker id	10
buy Valor	0
cust id	335
sell Valor	0
tax status	1
trade dts	2007-12-1 12:18:23

Frame 4 entrada:

Campo	Valor
cust id	335
symbol	ATL
trade qty	400
type id	TMS

Frame 4 saída:

Campo	Valor
comm rate	0.32
s name	COMMON of Atalanta Sosnoff Capital Corporate

Frame 5 entrada:

Campo	Valor
broker id	10
comm amount	29.56
st completed id	CMPT
trade dts	2007-12-1 12:18:23
trade id	1454535
trade price	23.09

Frame 5 has no saída data Campos.

Frame 6 entrada:

Campo	Valor
acct id	3341
due date	2007-12-3 12:18:23
s name	COMMON of Atalanta Sosnoff Capital Corporate
se amount	9201.94
trade dts	2007-12-1 12:18:23
trade id	1454535
trade is cash	1
trade qty	400
type name	Market-Sell

Frame 6 saída:

Campo	Valor
acct bal	3.08497e+06

Customer-Position: Semente = 325203098

Frame 1 entrada:

Campo	Valor
cust id	0
tax id	379DR2844FG514

Frame 1 saída:

Campo	Valor
cust id	137
acct len	6
acct id[0]	1365
cash bal[0]	2.56905e+06
asset total[0]	-28353
c st id	ACTV
c l name	Baar
c f name	Joel
c m name	T
c gndr	M
c tier	2
c dob[0]	1959-6-11 0:0:0
c ad id	2641
c ctry 1	011
c area 1	334
c local 1	0658259
c ext 1	
c ctry 2	011
c area 2	709
c local 2	4436133
c ext 2	
c ctry 3	011
c area 3	406
c local 3	9424656
c ext 3	
c email 1	JBaar@rr.com
c email 2	JBaar@msn.com

Frame 2 entrada:

Campo	Valor
cust id	1362

Frame 2 saída:

Campo	Valor
hist len	26
trade id[0]	290486
symbol[0]	MACRPRC
qty[0]	200
trade status[0]	Completed
hist dts[0]	2005-1-7 16:56:45

Trade-Order: Semente = 160710521

Frame 1 entrada:

Campo	Valor
acct id	3341

Frame 1 saída:

Campo	Valor
acct name	John Hance Emergency Expenses
broker name	Arthur O. Devan
cust f name	John
cust id	335
cust l name	Hance
cust tier	2
tax id	276FV7250IH330
tax status	1

Frame 3 entrada:

Campo	Valor
acct id	3341
cust id	335
cust tier	2
is lifo	0
issue	
st pending id	PNDG
st submitted id	SBMT
tax status	1
trade qty	400
trade type id	TMS
type is margin	0
co name	
requested price	23.44
symbol	ATL

Frame 3 saída:

Campo	Valor
co name	Atalanta Sosnoff Capital Corporate
requested price	23.02
symbol	ATL
buy Valor	0
charge amount	4.5
comm rate	0.32
cust assets	0
market price	23.02
s name	COMMON of Atalanta Sosnoff Capital Corporate
sell Valor	0
status id	SBMT
tax amount	0
type is market	1
type is sell	1

Frame 4 entrada:

Campo	Valor
acct id	3341
charge amount	4.5
comm amount	29.47
exec name	John Hance
is cash	1
is lifo	0
requested price	23.02
status id	SBMT
symbol	ATL
trade qty	400
trade type id	TMS
type is market	1

Frame 4 saída:

Campo	Valor
trade id	1454535

Broker-Volume: Semente = 1974393655

Frame 1 entrada:

Campo	Valor
broker list[1..10]	Terry R. Vowell, Arthur O. Devan, Maudie U. Shear, Imelda T. Jadlowiec, Donna I. Frie, Alphonso T. Hilgert, Christopher N. Simley, Ida T. Chu, Melissa V. Attard, Eugene T. Belzer
sector name	Transportation

Frame 1 saída:

Campo	Valor
list len	0
broker name[0]	
volume[0]	0

Trade-Lookup: Semente = 1600753948

Frame 1 entrada:

Campo	Valor
max trades	20
Trades[1..6]	374646,376566,490995,457683,48659,47034
Trades[7..12]	456583,260068,126293,261864,425819,6177
Trades[13..18]	126214,254106,106455,545935,516024,130618
Trades[19,20]	260865,393147

Frame 1 saída:

Campo	Valor
num found	20
bid price[0]	24.05
exec name[0]	Andy Mirra
is cash[0]	1
is market[0]	1
trade price[0]	24.01
settlement amount[0]	-9633.39
settlement cash due date[0]	2005-1-6 12:31:10
settlement cash type[0]	Cash Account
cash transaction amount[0]	-9633.39
cash transaction dts[0]	2005-1-4 12:31:10
cash transaction name[0]	Market-Buy 400 shares of COMMON of Mikohn Gaming Corporation
trade history dts[0][0]	2005-1-4 12:31:8
trade history status id[0][0]	SBMT
trade history dts[0][1]	2005-1-4 12:31:10
trade history status id[0][1]	CMPT
trade history dts[0][2]	0-0-0 0:0:0
trade history status id[0][2]	

Trade-Status: Semente = 352098087

Frame 1 entrada:

Campo	Valor
acct id	9812

Frame 1 saída:

Campo	Valor
acct id	9812
cust l name	Mangram
cust f name	Derrick
broker name	Arthur O. Devan
charge[0]	5
exec name[0]	Derrick Mangram
ex name[0]	Pacific Exchange
s name[0]	COMMON of Badger Paper Mills, Inc.
status name[0]	Completed
symbol[0]	BPMI
trade dts[0]	2005/1/7 15:47:57
trade id[0]	282086
trade qty[0]	200
type name[0]	Limit-Buy

Trade-Update: Semente = 328877277

Frame 3 entrada:

Campo	Valor
max trades	20
max updates	5
Trades[1..5]	490275,351795,73485,522968,287115
Trades[6..10]	130868,131061,40865,497631,97218
Trades[11..15]	163475,174124,391031,128676,97244
Trades[16..20]	261016,253949,130811,56241,60103

Frame 3 saída:

Campo	Valor
num found	20
num updated	1
bid price[0]	22.59
exec name[0]	Dong X nito
is cash[0]	1
is market[0]	1
trade price[0]	22.58
settlement amount[0]	-2272.44
settlement cash due date[0]	2005-1-8 12:25:11
settlement cash type[0]	Cash Account
cash transaction amount[0]	-2272.44
cash transaction dts[0]	2005-1-6 12:25:11
cash transaction name[0]	Market-Buy 100 shares of COMMON of Cabot Corporation
trade history dts[0][0]	2005-1-6 12:25:10
trade history status id[0][0]	SBMT
trade history dts[0][1]	2005-1-6 12:25:11
trade history status id[0][1]	CMPT
trade history dts[0][2]	0-0-0 0:0:0
trade history status id[0][2]	

Security-Detail: Semente = 1119871906

Frame 1 entrada:

Campo	Valor
access lob flag	0
start day	2003-3-19 0:0:0
max rows to return	20
symbol	SEMI

Frame 1 saída:

Campo	Valor
fin len	20
day len	20
news len	2
cp co name[0]	Alliance Fiber Optic Products, Inc.
cp in name[0]	Semiconductors
fin[0].year	2000
fin[0].qtr	1
fin[0].start date	2000-1-1 0:0:0
fin[0].rev	1.09906e+10
fin[0].net earn	6.7849e+08
fin[0].basic eps	0.88
fin[0].dilut eps	0.8
fin[0].margin	0.06
fin[0].invent	1.96017e+09
fin[0].assets	5.31968e+10
fin[0].liab	9.99247e+09
fin[0].out basic	7.72332e+08
fin[0].out dilut	8.49565e+08
day[0].date	2003-3-19 0:0:0
day[0].close	20.79
day[0].high	21.83
day[0].low	19.13
day[0].vol	8423
news[0].item	
news[0].dts	2007-10-11 18:28:11
news[0].src	Fegley
news[0].auth	Johnson
news[0].headline	Trabert Lagerman Ballek Feltes Dubberly Selkirk Chisam Stary Longley Upton Thill Trabert
news[0].summary	Lagerman Ballek Feltes Dubberly Selkirk Chisam Stary Longley Upton Thill Finger Mcadory Collom Hickle Craiger Consalvo Bonet Pohlmann Winburn Hines Karella Vicent Spera Kotera Heisinger Richens Gilday Shierling Pola Gainer Furbish Roache Milbradt
last price	20.83
last open	20.83
last vol	0
s name	COMMON of All American Semiconductor, Inc.

Frame 1 saída:

Campo	Valor
co name	All American Semiconductor, Inc.
sp rate	AC
ceo name	Orville Sundin
co desc	The principal activity of the company is to distribute electronic components manufactured by others. the company mainly distributes semiconductor comp
open date	1841-12-10 0:0:0
co st id	ACTV
co ad line1	8144 Hayden Park
co ad line2	
co ad town	Atlanta
co ad div	GA
co ad zip	30396
co ad cty	USA
num out	1656013928
start date	1929-6-24 0:0:0
ex date	1973-9-29 0:0:0
pe ratio	102.49
s52 wk high	25.09
s52 wk high date	2007-10-23 0:0:0
s52 wk low	24.39
s52 wk low date	2007-2-22 0:0:0
divid	0
yield	0
ex ad div	TX
ex ad cty	USA
ex ad line1	17442 Tallowood Upper
ex ad line2	
ex ad town	Lubbock
ex ad zip	79405
ex close	1600
ex desc	Simulates the NASDAQ
ex name	NASDAQ (National Association of Security Dealers and Quotations)
ex num symb	1749
ex open	930

Market-Watch: Semente = 1888197414

Frame 1 entrada:

Campo	Valor
acct id	0
cust id	351
ending co id	0
industry name	
starting co id	0

Frame 1 saída:

Campo	Valor
status	0
pct change	1.65905

Data-Maintenance: Semente = 1579447068

The Data-Maintenance has as only saída the status Campo which is equal to zero in all cases.

Frame 1 entrada:

Campo	Valor
add flag	0
c id	73
co id	0
day of month	0
symbol	
table name	ACCOUNT PERMISSION
tx id name	

Frame 1 entrada:

Campo	Valor
add flag	0
c id	783
co id	0
day of month	0
symbol	
table name	ADDRESS
tx id name	

Frame 1 entrada:

Campo	Valor
add flag	0
c id	0
co id	174
day of month	0
symbol	
table name	COMPANY
tx id name	

Frame 1 entrada:

Campo	Valor
add flag	0
c id	68
co id	0
day of month	0
symbol	
table name	CUSTOMER
tx id name	

Frame 1 entrada:

Campo	Valor
add flag	0
c id	198
co id	0
day of month	0
symbol	
table name	CUSTOMER TAXRATE
tx id name	

Frame 1 entrada:

Campo	Valor
add flag	0
c id	0
co id	0
day of month	20
symbol	AFREZ
table name	DAILY MARKET
tx id name	

Frame 1 entrada:

Campo	Valor
add flag	0
c id	0
co id	0
day of month	0
symbol	
table name	EXCHANGE
tx id name	

Frame 1 entrada:

Campo	Valor
add flag	0
c id	0
co id	486
day of month	0
symbol	
table name	FINANCIAL
tx id name	

Frame 1 entrada:

Campo	Valor
add flag	0
c id	0
co id	250
day of month	0
symbol	
table name	NEWS ITEM
tx id name	

Frame 1 entrada:

Campo	Valor
add flag	0
c id	0
co id	0
day of month	0
symbol	BYBI
table name	SECURITY
tx id name	

Frame 1 entrada:

Campo	Valor
add flag	0
c id	0
co id	0
day of month	0
symbol	
table name	TAX RATE
tx id name	MN2

Frame 1 entrada:

Campo	Valor
add flag	0
c id	445
co id	0
day of month	0
symbol	
table name	WATCH ITEM
tx id name	

Trade-Cleanup: Semente = 413124175

Frame 1 entrada:

Campo	Valor
st canceled id	CNCL
st pending id	PNDG
st submitted id	SBMT
trade id	581761

Frame 1 saída:

Campo	Valor
status	0

Referências Bibliográficas

[A+85] Anon et al. A measure of transaction processing power. Technical report, Tandem Computers, April 1985.

[APH73] A. Alan, B. Pritsker, and Nicholas R. Hurst. Gasp iv: A combined continuous - discrete fortran-based simulation language. In *SIMULATION*, volume 21, pages 65–70, 1973.

[Axb] Jens Axboe. Block io tracing. http://www.kernel.org/git/?p=linux /kernel/git/axboe/blktrace.git.

[BCD+00] T. Bezenek, H. Cain, R. Dickson, T. Heil, M. Martin, C. McCurdy, R. Rajwar, E. Weglarz, C. Zilles, and M. Lipasti. Characterizing a java implementation of tpcw. *Third Workshop On Computer Architecture Evaluation Using Commercial Workloads*, January 2000.

[BDT83] D. Bitton, D. J. DeWitt, and C. Turbyfil. Benchmarking database systems: A systematic approach. *Proceedings of the Very Large Database Conference*, October 1983.

[BGdMT06] Gunter Bolch, Stefan Greiner, Hermann de Meer, and Kishor S. Trivedi. *Queueing Networks and Markov Chains*. Wiley-Interscience, 2006.

[BKL06] Carsten Binnig, Donald Kossmann, and Eric Lo. Testing database applications. *SIGMOD*, June 2006.

[BLSZ04] R. Bonilla-Lucas, A. Sachedina, and C. Zuzarte. Characterization of the data access behavior for tpc-c traces. *IEEE International Symposium on Performance Analysis of Systems and Software*, pages 115–122, 2004.

[BP97] A. Bernstein and E. Newcomer P. *Principles of Transaction Processing for the Systems Professional*. Morgan Kaufmann, 1997.

[CDL01] M. Clark, A. Durg, and K. Lienenbrugger. Characterization of tpc-h queries on amd athlon/sup tm/ microprocessors. *Workshop on Workload Characterization, 2001. WWC-4. 2001 IEEE International*, pages 26–35, December 2001.

[CL99] Christos G. Cassandras and Stéphane Lafortune. *Introduction to Discrete Event Systems*. Kluwer Academic Publishers, 1999.

[CRML01] Harold W. Cain, Ravi Rajwar, Morris Marden, and Mikko H. Lipasti. An architectural evaluation of java tpc-w. *High-Performance Computer Architecture*, pages 229–240, 2001.

[CRU07] Maria Luisa Catalan, Dennis A. Ludeña R., and Hidenori Umeno. Vm-based benchmark and analysis system for testing online transaction processing. *Proceedings of the Second International Conference on Innovative Computing, Informatio and Control*, May 2007.

[Des06] Mathieu Desnoyers. *The LTTng usertrace package*. http://ltt.polymtl.ca/svn/ltt-usertrace/README,, 2006.

[DHP+93] F. Dicesare, G. Harhalakis, J. M. Proth, M. Silva, and F. B. Vernadat. *Practice of Petri Nets in Manufacturing*. Chapman Hall, 1993.

[DLM08] B.K. Debnath, D.J. Lilja, and M.F. Mokbel. Sard: A statistical approach for ranking database tuning parameters. *IEEE 24th International Conference on Data Engineering Workshop*, pages 11–18, April 2008.

[dNWM07] Rilson Oscar do Nascimento, Mark Wong, and Paulo Romero Martins Maciel. Dbt-5: A fair usage open-source tpc-e implementation for performance evaluation of computer systems. In *Anais do Workshop em Desempenho de Sistemas Computacionais e de Comunicação*, July 2007.

[FGP03] P. Foglia, R. Giorgi, and C.A. Prete. Evaluating optimizations for multiprocessors e-commerce server running tpc-w workload. *Proceedings of the 34th Annual Hawaii International Conference on System Sciences, 2001.*, February 2003.

[FGP04] P. Foglia, R. Giorgi, and C.A. Prete. Simulation study of memory performance of smp multiprocessors running a tpc-w workload. *IEEE Proceedings -Computers and Digital Techniques*, pages 93–109, March 2004.

[FM03] Paul J. Fortier and Howard E. Michel. *Computer Systems Performance Evaluation and Prediction*. Digital press - Elsevier Science, 2003.

[FS07] A. Fedorova and M. and Seltzer. Improving performance isolation on chip multiprocessors via an operating system scheduler. *16th International Conference on Parallel Architecture and Compilation Techniques*, pages 25–38, September 2007.

[Gra93] Jim Gray, editor. *The Benchmark Handbook for Database and Transaction Systems (2nd Edition)*. Morgan Kaufmann, 1993.

[GREC91] John Gustafson, Diane Rover, Stephen Elbert, and Michael Carter. The design of a scalable, fixed-time computer benchmark. *Journal of Parallel and Distributed Computing*, 11(8):388–401, August 1991.

[GS95] J.L. Gustafson and Q.O. Snell. Hint: A new way to measure computer performance. *Proceedings of the Twenty-Eighth Hawaii International Conference on System Sciences*, 2:392–401, January 1995.

[GWP98] G. R. Ganger, B. L. Worthington, and Y. N. Patt. The disksim simulation environment. Technical report, Dept. of Electrical Engineering and Computer Science, University of Michigan, February 1998.

[Her01] Ulrich Herzog. Formal methods for performance evaluation. *Lectures on formal methods and performance analysis*, 2001.

[HP07] John L. Hennessy and David A. Patterson. *Computer Architecture - A Quantitative Approach*. Morgan Kaufmann, fourth edition, 2007.

[HSY01] Windsor W. Hsu, Alan Jay Smith, and Honesty C. Young. I/o reference behavior of production database workloads and the tpc benchmarks; an analysis at the logical level. *ACM Transactions on Database Systems (TODS)*, March 2001.

[Jai91] Raj Jain. *The Art of Computer Systems Performance Analysis Techniques For Experimental Design Measurements Simulation And Modeling*. Wiley Computer Publishing, 1991.

[JEL+06] Lizy Kurian John, Lieven Eeckhout, David J. Lilja, Joshua J. Yi, Thomas M. Conte, Paul D. Bryan, Brad Calder, Timothy Sherwood, Greg Hamerly, Erez Perelman, Eun Jung Kim, Ki Hwan Yum, Chita R. Das, Brinkley Sprunt, Alex Mericas, Rumi Zahir, Kishore Menezes, and Susith Fernando. *Performance Evaluation and Benchmarking*. CRC Press, 2006.

[KK00] M.A. Kandaswamy and R.L. Knighten. I/o phase characterization of tpc-h query operations. *Computer Performance and Dependability Symposium, 2000. IPDS 2000. Proceedings. IEEE International*, pages 81–90, March 2000.

[KVM68] P. J. Kiviat, R. Villanueva, and H. M. and Markowitz. The simscript ii programming language. Technical report, RAND CORP SANTA MONICA CA, October 1968.

[Lab08] OSDL (Open Source Development Labs). Database test suite, July 2008. http://sourceforge.net/projects/osdldbt.

[LD03] S. T. Leutenegger and D. Dias. A modeling study of the tpc-c benchmark. In *ACM SIGMOD*, 2003.

[LGKK93] Charles Levine, Jim Gray, Steve Kiss, and Walt Kohler. The evolution of tpc-benchmarks: Why tpc-a and tpc-b are obsolete. Technical Report SFSC Technical Report 93.1, San Francisco Systems Center, September 1993.

[Lil00] David J. Lilja. *Measuring Computer Performance A practitioner's guide*. Cambridge University Press, 2000.

[Lla08] Diego R. Llanos. Tpcc-uva web site, July 2008. http://www.infor.uva.es/ diego/tpcc-uva.html.

[LP06] D. R. Llanos and B. Palop. Tpcc-uva: An open-source tpc-c implementation for parallel and distributed systems. In *IEEE 6th International Workshop on Performance Modeling, Evaluation, and Optimization of Parallel and Distributed Systems*, 2006.

[MLC96] P. Maciel, R. Lins, and P. Cunha. *Introducção às Redes de Petri e Aplicações*. X Escola de Computação - Campinas - SP, July 1996.

[MSAHB05] D.T. McWherter, B. Schroeder, A. Ailamaki, and M. Harchol-Balter. Improving preemptive prioritization via statistical characterization of oltp locking. *Proceedings. 21st International Conference on Data Engineering, 2005. ICDE 2005.*, pages 446–457, April 2005.

[Ora04] Oracle. Linux filesystem performance comparison for oltp with ext2, ext3, raw, and ocfs on direct-attached disks using oracle 9i release 2. Oracle white paper, Oracle, January 2004.

[osd01] Osdb. the open source database benchmark. http://osdb.sourceforge.net/, 2001.

[PdNS07] Carlos Eduardo Santos Pires, Rilson Oscar do Nascimento, and Ana Carolina Salgado. Comparativo de desempenho entre bancos de dados de código aberto. *Escola Regional de Banco de Dados*, 2007.

[Pet62] C. A. Petri. *Kommunikation mit Automaten*. PhD thesis, University of Bonn, Germany, 1962.

[Pet81] J. L. Peterson. *Petri Nets an Introduction*. Prentice-Hall, 1981.

[PH04] S. L. Pratt and D. A. Heger. Workload dependent performance evaluation of the linux 2.6 i/o schedulers. *Proceedings of the Linux Symposium*, 2004.

[plp08] Pl/pgsql - sql procedural language, July 2008. http://www.postgresql.org/docs/8.3/static/plpgsql.html.

[PNW07] Meikel Poess, Raghunath Othayoth Nambiar, and David Walrath. Why you should run tpc-ds: A workload analysis. In *VLDB*. ACM, September 2007.

[pos08] Postgresql, July 2008. http://www.postgresql.org/.

[Pri86] A. Alan Pritsker. *Introduction to Simulation and SLAM II*. John Wiley Sons, Inc, 1986.

[pri07] *TPC Pricing 1.3.0*. December 2007.

[Rob08] L. Robert. I/o schedulers. Technical report, The Linux Journal, Kernel Korner, http://www.linuxjournal.com/article/6931, January 2008.

[Sch90] Thomas J. Schriber. *Simulation Using GPSS*. Krieger Publishing Co, 1990.

[spe08] Standard performance evaluation cooperative (spec) http://www.spec.org, July 2008.

[TOB92] C. Turbyfill, C Orji, and D. Bitton. *AS3AP: An ANSI SQL Standard Scalable and Portable Benchmark for Relational Database Systems. The Benchmark Handbook for Database and Transaction Processing*. Morgan Kaufmann, 1992.

[TOB93] Carolyn Turbyfill, Cyril Orji, and Dina Bitton. *AS3AP: An ANSI SQL Standard Scalable and Portable Benchmark for Relational Database Systems*, chapter 4, pages 167–206. Morgan Kaufmann, 1993.

[TPC89] TPC www.tpc.org. *TPC Benchmark(tm) A Standard Specification*, November 1989.

[TPC90] TPC www.tpc.org. *TPC Benchmark(tm) B Standard Specification*, August 1990.

[TPC92] TPC www.tpc.org. *TPC Benchmark(tm) C Standard Specification*, August 1992.

[TPC95] TPC www.tpc.org. *TPC Benchmark(tm) D Standard Specification*, August 1995.

[TPC99a] TPC www.tpc.org. *TPC Benchmark(tm) H Standard Specification*, August 1999.

[TPC99b] TPC www.tpc.org. *TPC Benchmark(tm) R Standard Specification*, August 1999.

[TPC99c] TPC www.tpc.org. *TPC Benchmark(tm) W Standard Specification*, August 1999.

[TPC05] TPC www.tpc.org. *TPC Benchmark(tm) App Standard Specification*, August 2005.

[TPC07] TPC www.tpc.org. *TPC Benchmark(tm) E Standard Specification*, August 2007.

[TPC08] TPC www.tpc.org. *TPC Benchmark(tm) DS Draft Specification*, August 2008.

[tpp08] Transactioin processing performance council (tpc) http://www.tpc.org, July 2008.

[VAT+04] D. Villa, J. Acosta, P.J. Teller, B. Olszewski, and T. and Morgan. A framework for profiling multiprocessor memory performance. *ICPADS 2004. Proceedings. Tenth International Conference on Parallel and Distributed Systems.*, pages 530–538, July 2004.

[VT06] Hans Vandierendonck and Pedro Trancoso. Building and validating a reduced tpc-h benchmark. *IEEE International Symposium on Modeling, Analysis, and Simulation*, 2006.

[WA01] K.M. Wilson and B.B. Aglietti. Dynamic page placement to improve locality in cc-numa multiprocessors for tpc-c. *ACM/IEEE 2001 Conference on Supercomputing,*, November 2001.

[WHRP06] P. W. Y. Wong, R. Hendrickson, H. Rizvi, and S. Pratt. Performance evaluation of linux file systems for data warehousing workloads. *Proceedings of the First International Conference on Scalable Information Systems*, 2006.

[Wik08a] the free Encyclopedia Wikipedia. ext2, July 2008. http://en.wikipedia.org/wiki/Ext2.

[Wik08b] the free Encyclopedia Wikipedia. ext3, July 2008. http://en.wikipedia.org/wiki/Ext3.

[ZSF+04] Jianyong Zhang, A. Sivasubramaniam, H. Franke, N.; Gautam, Yanyong Zhang;, and S. Nagar. Synthesizing representative i/o workloads for tpc. *10th International Symposium on High Performance Computer Architecture, 2004. HPCA-10. Proceedings.*, February 2004.

[ZZS+02] Yanyong Zhangy, Jianyong Zhangy, Anand Sivasubramaniamy, Chun Liuy, and Hubertus Frankez. Characterizing tpc-h on a clustered database engine from the os perspective. *Eighth International Symposium on High Performance Computer Architecture*, February 2002.

Este volume foi tipografado em LaTeX na classe UFPEThesis (www.cin.ufpe.br/~paguso/ufpethesis).