TUNING - Técnicas de Otimização de Banco de Dados
Um Estudo Comparativo: Mysql e Postgresql

Alessandro Pinto Carneiro, Julinao Lucas Moreira, André Luis Castro de Freitas1

[1]Centro de Ciências Computacionais – Universidade Federal do Rio Grande (FURG)
Caixa Postal 474 – 96201.900 – Rio Grande – RS – Brasil

dmtalcf@furg.br

Trabalho de Graduação

Abstract. *The importance of the information for the decision making in the organizations, the increase data quantity and the time to search for information, they have been demanding concern with the speed access to the databases. Considering in this way Tuning is evidenced as a process of the database systems refine, aiming at to improve his acting. This work shows the concepts of the tuning in PostgreSQL, specifically with the focus in the parameters configuration and the impact in the modification of the same ones. It is made an analysis of the configuration aspects presented and it is drawn a hierarchy of importance of each one in agreement with his purpose.*

Resumo. *A importância da informação para a tomada de decisão nas organizações, o volume cada vez maior de dados e o tempo de busca da informação, têm exigido preocupação com a velocidade de acesso aos bancos de dados. Neste contexto, evidencia-se o Tuning como um processo de refino dos sistemas de banco de dados, objetivando melhorar seu desempenho. Este trabalho apresenta os conceitos de tuning no PostgreSQL, especificamente com o foco na configuração de seus parâmetros e o impacto na modificação dos mesmos. Faz-se uma análise dos aspectos de configuração apresentados e traça-se uma hierarquia de importância de cada um de acordo com o seu propósito.*

Palavras-Chave. *Tuning, SGBD, Postgresql*

1. Introdução

A demanda por Sistemas de Gerenciamento de Bancos de Dados cresce continuamente. Juntamente com essa demanda, cresce também o volume de dados que estes sistemas devem gerenciar e a complexidade de suas aplicações. Neste cenário, realizar operações, de forma eficiente, sobre estas grandes coleções de dados é uma questão fundamental, já que o desempenho de um SGBD (Sistema Gerenciador de Banco de Dados) é medido a partir de sua eficiência diante de consultas e alterações.

Atualmente, encontram-se no mercado diversos sistemas gerenciadores de banco de dados, os quais na maioria dos casos são instalados, configurados e utilizados com todos os seus parâmetros em valores padrões, sem levar em consideração o tipo de aplicação para o qual serão utilizados, o hardware, e até mesmo o sistema operacional. Desta forma nem sempre obtém-se o melhor desempenho do sistema, visto que, diversos parâmetros podem ser considerados e ajustados em um SGBD.

O primeiro caminho para conseguir um desempenho adequado de um sistema de banco de dados é tomar boas decisões durante o projeto desse. Várias considerações

deverão ser feitas durante a fase de projeto, entre elas: o volume esperado de dados em cada relação do sistema e quais consultas serão realizadas com mais freqüência. Mas, percebe-se na maioria dos sistemas é que seu real desempenho só pode ser conseguido após algum tempo de uso, e muitas das considerações que os projetistas haviam feito podem mostrar-se incorretas. Portanto, uma fase subseqüente de ajuste do sistema torna-se necessária, com base em dados reais de seu comportamento, com o objetivo de maximizar o desempenho e a estabilidade. Esta fase é chamada de *database tuning* ou simplesmente *tuning*.

Este trabalho está organizado nas seções 2, 3 e 4. A seção 2 define os conceitos e técnicas utilizados a fim de realizar *tuning* (sintonia) em um SGBD. A seção 3 apresenta as ferramentas utilizadas na realização dos testes propostos. Na seção 4 são apresentados os padrões adotados para testar o desempenho de um SGBD, bem como, otimizações são realizadas. Após procede-se a conclusão e trabalhos futuros.

2. *Tuning*

Segundo Date (2004), "o Sistema Gerenciador de Banco de Dados é o *software* que manipula todos os acessos ao Banco de Dados". O SGBD é um *software* que funciona como uma *interface* entre o usuário e o Banco de Dados, ou seja, todas as solicitações dos usuários, como criação de tabelas, inserção de dados, recuperação de dados, são manipuladas pelo SGBD.

Após um banco de dados ter sido desenvolvido e estar em operação, o uso real das aplicações, das transações, das consultas e das visões revela fatores e áreas de problemas que podem não ter sido considerados durante o projeto físico inicial. As informações de entrada para o projeto físico podem ser revisadas por meio da coleta de estatísticas reais sobre os padrões de uso. A utilização dos recursos, bem como o processamento interno do SGBD pode ser monitorado para revelar gargalos, tais como a disputa pelos mesmos dados ou dispositivos. Os volumes de atividades e os tamanhos dos dados podem ser bem mais estimados.

Portanto, é necessário monitorar e revisar o projeto físico de banco de dados constantemente. Os objetivos da sintonia (ou *tuning*) são os seguintes:

I. fazer com que as aplicações sejam executadas mais rapidamente,
II. diminuir o tempo de resposta de consultas/transações e
III. melhorar o desempenho geral das transações.

A linha divisória entre o projeto físico de um banco de dados e sua sintonia é muito pequena. As mesmas decisões de projeto são revisadas na fase de sintonia, que é um ajuste continuado do projeto. As informações de entrada para o processo de sintonização incluem estatísticas relacionadas a diversos fatores. Em particular um SGBD pode coletar internamente as seguintes estatísticas:

- tamanho de tabelas individuais,
- número de valores distintos em uma coluna,
- número de vezes que uma consulta ou transação em particular é submetida/executada em um intervalo de tempo. Os tempos necessários para as diferentes fases de processamento de consultas.

Essas e outras estatísticas criam um perfil do conteúdo e do uso de banco de dados. Outras informações obtidas a partir do monitoramento das atividades do sistema de banco de dados incluem as seguintes:

- estatísticas de armazenamento: dados a respeito da alocação de armazenamento para espaço de tabelas, espaço de índices e portas de buffer.

- estatísticas de desempenho de entrada/saída: atividade total de leitura/escrita (paginação) do disco.
- estatísticas de processamento de consultas:tempos de execução de consultas, tempos de otimização durante a otimização de consultas.
- estatísticas relacionadas a bloqueios/registro de *log*: taxas de definição de diferentes tipos de bloqueios, taxas de desempenho de transações e registros de *log* de atividades.

Muitas dessas estatísticas acima se referem a transações, ao controle de concorrência e a recuperação de dados. Mas a sintonia de bancos de dados envolve tratar diretamente os seguintes tipos de problema: como evitar excessivas disputas por bloqueios, aumentando, de modo, a concorrência entre as transações, como minimizar a sobrecarga de registrar *logs* e o armazenamento desnecessário de dados, como otimizar o tamanho do buffer e o escalonamento de processos e ,finalmente, como alocar recursos tais como discos, memória e processos para uma utilização mais eficiente.

A maioria desses problemas mencionados pode ser resolvida por meio de ajuste apropriado de parâmetros físicos do SGBD, da alteração das configurações de dispositivos, da alteração de parâmetros do sistema operacional e de outras atividades similares.

2.1 Sintonia de Índices

A escolha inicial de índices pode precisar de uma revisão pelas seguintes razões: certas consultas podem demorar demais para serem executadas por conta da ausência de um índice, certos índices podem, absolutamente, não ser utilizados e certos índices podem estar causando sobrecarga excessiva porque são baseados em um atributo que constantemente sofre alterações.

A maioria dos SGBDs possui um comando ou um meio de rastreamento que pode ser usado para solicitar que o sistema mostre como uma consulta foi executada, quais operações foram realizadas e em qual ordem e quais estruturas de acesso foram utilizadas. Por meio de análise desses planos de execução é possível diagnosticar as causas dos problemas citados. Alguns índices podem ser excluídos e alguns novos podem ser criados com base na análise de sintonia.

O objetivo da sintonia é avaliar dinamicamente os requisitos, os quais variam durante diferentes períodos do mês ou da semana, e reorganizá-los de forma a proporcionar melhor desempenho global. A exclusão e a criação de novos índices são sobrecarga que pode ser justificada em função de melhorias de desempenho. A atualização de uma tabela geralmente é suspensa enquanto um índice estiver sendo excluído ou criado.

2.2 Sintonia de Consultas

O monitoramento e o ajuste de consultas SQL é a atividade que consome grande tempo de administradores de banco de dados por sua complexidade e por representar a maior parte dos acessos realizados no SGBD, sendo que muitas não alcançam o desempenho esperado, devido a terem sido escritas pensando-se no resultado a serem obtidos, e não no melhor caminho para obtê-los. Fatores como falta de experiência em desenvolvimento, baixo nível de conhecimento técnico, prazos de entregas sub-dimensionados e falta de monitoramento individual contribuem para que as consultas sejam ineficientes, o que determina a análise freqüente das consultas.

O otimizador constrói o plano de execução a partir da consulta SQL, tendo uma margem limitada de opções sobre os operadores utilizados. Uma consulta SQL mal escrita leva o otimizador a utilizar um caminho que nem sempre é o mais adequado, o

que gera um plano de execução que normalmente compromete o desempenho. Em ambientes nos quais existem inúmeras consultas construídas dessa forma e que são executadas freqüentemente, a conseqüência acaba sendo drástica, dificultando o uso eficiente da maior parte dos recursos disponíveis e deixando de atender em tempo outros processos críticos cujas execuções são rápidas e necessitam de prioridade.

Desta forma existem algumas técnicas utilizadas pelos administradores de banco de dados a fim de alcançar desempenho. A justificativa para se usar a técnica de reescrita de consultas como uma das primeiras a ser utilizada, é que essa técnica afeta apenas uma consulta específica, não se propagando para outras aplicações que acessam as tabelas envolvidas na instrução SQL (NAVATHE, 2006).

2.3 Critérios Adotados

De uma forma simplificada, existem alguns critérios principais que podem ser adotados para identificar as consultas que devem ser modificadas, são eles:

- monitorar as sessões ativas que estão sendo executadas no banco de dados,
- separar as consultas que estão com execuções demoradas,
- dividi-las em grupos, como: prioridade, freqüência de execução e fraco desempenho,
- implementar os ajustes reescrevendo as consultas que estão com fraco desempenho.

Cada banco de dados fornece suas ferramentas específicas para capturar as consultas citadas nos itens acima, porém a maneira mais eficiente e explícita é o próprio uso feito pelo usuário final.

3. Ferramentas

Nesta seção serão apresentadas as ferramentas/aplicativos utilizados para o desenvolvimento deste artigo, bem com a ferramenta desenvolvida para auxiliar no processo de testes e otimização dos SGBDs.

- O *PostgreSQL* é um Sistema Gerenciador de Banco de Dados Objeto-Relacional (SGBDOR). Devido à sua licença aberta, o *PostgreSQL* pode ser utilizado, modificado e distribuído por qualquer pessoa para qualquer finalidade, seja privada, comercial ou acadêmica, livre de encargos.
- Um servidor Web é um programa de computador responsável por processar solicitações HTTP (*Hyper-Text Transfer Protocol*), o protocolo padrão da Web. Quando utiliza-se um navegador de internet para acessar um site, este faz as solicitações devidas ao servidor Web do site por meio de HTTP e então recebe o conteúdo correspondente. Como servidor Web, o Apache é o mais conhecido e usado. Os motivos incluem sua excelente *performance*, segurança, compatibilidade com diversas plataformas e todos os seus recursos.
- PHP (*Hypertext Preprocesso*r) é uma linguagem de programação de computadores interpretada, livre e muito utilizada para gerar conteúdo dinâmico na World Wide Web. A linguagem PHP é uma poderosa linguagem orientada a objetos.
- A ferramenta *BenchmarkSQL* (SOURCEFORGE, BenchmarkSQL, 2006) foi desenvolvida utilizando a linguagem de programação Java. Utiliza as bibliotecas JDBC para realizar a comunicação com diferentes SGBDs por meio da linguagem de programação SQL.

3.1 BenchmarkSQL

Sua utilização consiste em quatro etapas: Criação das Tabelas no SGBD, Carga das Tabelas, Criação de Índices e Execução do *BenchmarkSQL*. Estas são descritas conforme segue.

Na etapa de criação das tabelas no SGBD são criadas as tabelas que fazem parte do método TPC-C (*Transaction Processing Performance Council*). Para sua utilização é necessário a execução de um arquivo executável que faz parte do software *BenchmarkSQL*, chamado *runSQL.bat* para o sistema operacional Windows ou *runSQL.sh* para o sistema Unix. Para execução desta etapa é necessário a entrada de alguns parâmetros no momento da execução do arquivo, que são o nome do arquivo de configuração do banco de dados e o comando que indica a criação de tabelas.

Para a carga das tabelas no SGBD é necessário executar o arquivo *loadData.bat* para o sistema operacional Windows ou *loadData.sh* para o sistema Unix. Quando executado gera dados aleatórios nas tabelas do método TPC-C.

Após a criação e carga das tabelas passa-se a criação dos índices das tabelas no SGBD. Nesta etapa são criados os índices e as chaves que identificam as tabelas.

Após as três etapas anteriores serem executadas é possível executar o programa *BenchmarkSQL*. Para isto é necessário apenas informar como parâmetro o nome do arquivo de configuração do banco de dados, por meio do comando *runBenchmark postgres.properties*. Após sua execução é apresentada a interface gráfica da aplicação. Nesta interface são apresentadas as propriedades especificadas no arquivo de configuração do SGDB escolhido, neste caso o PostgreSQL. Nesta mesma interface é necessário especificar o número de terminais que serão utilizados no teste, o número de *warehouses* especificado na criação do SGBD de teste e, por último, o tipo de execução do teste, que pode ser por tempo ou por transações por minuto. Após a configuração dos parâmetros para executar o benchmark é necessário a escolha dos percentuais de execução das transações do método TPC-C. Com todas as configurações feitas é possível executar os testes de *performance* no SGBD, por meio da opção *Create Terminals* e *Start Transactions*.

Tendo os terminais sido criados e postos em execução é possível acompanhá-los e então observar todas as transações que estão sendo feitas pelo Benchmark, bem como a média de transações por minuto e o número de transações correntes.

A partir dos arquivos e dos tempos de execução das outras etapas da utilização do benchmark foi possível efetuar as comparações entre os SGBDs PostgreSQL e MySQL, nos sistemas operacionais Windows XP e Linux kernel 2.6.8, sendo também comparada a instalação do sistema Linux com diferentes sistemas de arquivos.

Para facilitar a configuração dos parâmetros dos SGBDs foi desenvolvida uma ferramenta que facilita a leitura e alteração dos parâmetros do arquivo de configuração dos SGBDs. A ferramenta foi desenvolvida em linguagem PHP. A ferramenta é inicializada por meio de um navegador web. Digita-se o endereço de onde está localizado o arquivo principal da ferramenta e a mesma é então inicializada. Para executar os testes foi utilizado um servidor web local. A interface inicial da ferramenta permite que o usuário selecione qual o SGBD que deseja utilizar, PostgreSQL ou MySQL. Após selecionar o banco de dados que deseja utilizar basta avançar para prosseguir para a próxima etapa de configuração dos parâmetros. Esta etapa exibe duas opções de escolha como demonstrado abaixo: a primeira opção permite que se escolha um padrão de configuração previamente salvo e a segunda opção permite que se selecione um arquivo de configuração qualquer. Por exemplo, o usuário pode carregar um arquivo que utilizou em uma outra base de dados, permitindo assim copiar todos os

seus parâmetros para a base de dados atual. Após escolher um padrão de configuração ou abrir um arquivo salvo, seleciona-se a próxima tela de configuração. Nesta tela são exibidos todos os parâmetros alterados a partir da escolha feita na tela anterior. Nessa tela de configuração o usuário pode conferir os parâmetros alterados pelo padrão aplicado, e se desejar pode alterar algum outro parâmetro individualmente. Quando todos os parâmetros da configuração estiverem com os valores desejados, seleciona-se a opção para gerar o código que será, efetivamente, transferido para o arquivo de configuração do SGBD em uso.

4. Padrões e Otimizações

Serão apresentados nesta seção os testes que serviram como base para medir o desempenho do SGBD PostgreSQL em relação ao SGBD MySQL, bem como um comparativo de seus desempenhos em diferentes sistemas operacionais e sistemas de arquivos. A partir destes comparativos foram desenvolvidas configurações de otimização para o SGBD PostgreSQL e realizadas novas medições de desempenho com o objetivo de demonstrar que o processo de otimização pode gerar algumas melhorias significativas quando realizado corretamente.

Partindo do objetivo de gerar testes de desempenho confiáveis e certificados, foi escolhido o método TPC-C, a partir deste método foi desenvolvido um conjunto de testes para serem executados nos diferentes sistemas operacionais e sistemas de arquivos abordados no projeto. Primeiramente foram configurados três sistemas operacionais distintos e estes sistemas estão especificados na Tabela 1.

Tabela 1: Sistemas operacionais utilizados

Sistema Operacional	Sistema de Arquivos
Windows XP Home SP2	NTFS
Linux Kernel 2.6.18	ReiserFS
Linux Kernel 2.6.18	Ext3

Como conjunto de testes para utilização do método TPC-C no software BenchmarkSQL adotou-se as três configurações para transações demonstradas na Tabela 2.

Tabela 2: Conjunto de testes para o método TPC-C e tamanho aproximado da base de dados resultante

Nº de WareHouses	Nº de Terminais	Tamanho Decorrente
1	10	300MB
10	10	1GB
10	100	1GB

4.1 Resultados Obtidos

Seguindo o conjunto de testes especificado, foi utilizado o software *BenchmarkSQL* nos sistemas citados anteriormente, utilizando a configuração padrão dos SGBDs PostgreSQL e MySQL. Os resultados destes testes serão demonstrados a seguir.

Tabela 3: Resultados obtidos com as configurações padrão para execução de 1 warehouses e 10 terminais

WareHouses 1 - Terminais 10				
Sistema Operacional	Configuração	Nº Transações	Média TPm	Duração (min)
Windows - NTFS	PostgreSQL	13078	577,4689178	10
	MySQL	19546	887,6759983	10
Linux - ReiserFS	PostgreSQL	21961	993,6732044	10
	MySQL	17846	796,1059276	10
Linux - Ext3	PostgreSQL	11744	536,5409173	10
	MySQL	14957	673,0266379	10

Pode-se constatar que o SGBD MySQL obteve um desempenho superior ao SGBD PostgreSQL, em duas situações. Estas foram utilizando o sistema operacional Windows com NTFS, e no sistema operacional Linux utilizando Ext3, para o teste de 10 terminais e 1 warehouse que utiliza uma base de dados de aproximadamente 300MB de espaço em disco.

No sistema operacional Linux utilizando ReiserFS foi constatado que o SGBD PostgreSQL apresenta um bom desempenho em sua configuração padrão em comparação com o SGBD MySQL, para este teste que apresenta um número não muito grande de conexões concorrentes.

Tabela 4: Resultados obtidos com as configurações padrão para execução de 10 warehouses e 10 terminais

WareHouses 10 - Terminais 10				
Sistema Operacional	Configuração	Nº Transações	Média TPm	Duração (min)
Windows - NTFS	PostgreSQL	7413	330,4362277	10
	MySQL	13416	592,6841333	10
Linux - ReiserFS	PostgreSQL	10649	476,1026567	10
	MySQL	13187	595,8599729	10
Linux - Ext3	PostgreSQL	6547	294,8913817	10
	MySQL	11154	361,796517	10

Constatou-se que conforme são aumentados os números de conexões concorrentes o SGBD PostgreSQL apresenta uma queda de desempenho inclusive no sistema operacional Linux utilizando ReiserFS, isto devesse ao fato de que a configuração padrão do SGBD PostgreSQL não é otimizada para ambientes OLTP (*On-Line Transaction Processing*), que possuem grande número de conexões concorrentes e consultas simples. Desta forma constatou-se a necessidade de otimizar o SGBD PostgreSQL a fim de melhorar o seu desempenho neste tipo de ambiente.

Desta forma desenvolveu-se duas configurações otimizadas para o arquivo de configuração, com o objetivo de melhorar o desempenho do PostgreSQL quando submetido a testes baseados no padrão TPC-C, ou seja, quando submetido a testes em um ambiente com muitas transações, mas com comandos simples. Baseando-se nas análises realizadas foram alterados alguns parâmetros que se demonstraram mais importantes quando o sistema é submetido a um ambiente de testes OLTP.

- *max_connections* – esse parâmetro especifica o número máximo de conexões concorrentes que fazem acesso ao banco de dados. Apesar dessas conexões concorrentes serem importantes em ambientes OLTP, não modificou-se o valor desse parâmetro nessa primeira otimização. O valor padrão para esse parâmetro é *max_conection* = 100, ou seja, até cem conexões simultâneas.

Há um grande número de parâmetros que podem ser alterados no arquivo *postgresql.conf* a fim de otimizar o SGBD de acordo com a necessidade de cada aplicação. Mas existem alguns poucos parâmetros que a maioria dos administradores de banco de dados alteram a fim de conseguir desempenho em ambientes OLTP. Com base nisso demonstra-se aqui algumas configurações desses parâmetros:

- *shared_buffers* – esse parâmetro é um dos mais importantes para uma boa performance do PostgreSQL, nele definem o número de buffers da memória compartilhada utilizada pelo servidor do banco de dados. O valor padrão para esse parâmetro é *shared_buffers*=32MB. Para servidores com boa disponibilidade de memória (mais de 1GB) é interessante utilizar um valor que fica entre 8% e 20% da memória RAM disponível. Dessa forma esse valor foi ajustado para 64MB.
- *maintenance_work_mem* – o valor padrão para esse parâmetro esta definido como 16MB, isto significa a quantidade de memória que pode ser utilizada em operações de manutenção. Como percebeu-se que o tempo para executar a operação de *Vacuum* estava um pouco acima da media, decidiu-se aumentar esse valor para 32MB. Em grandes bancos de dados que possuem *queries* complexas e altíssima disponibilidade de memória, normalmente eleva-se muito o valor desse parâmetro. Em bancos de dados que possuem *queries* simples, capacidade de memória moderada e grande concentração de usuários concorrentes, não eleva-se tanto esse valor.Também pode ser calculado um valor aproximado utilizando-se a seguinte formula:

work_mem = memória disponível / usuários concorrentes
work_mem = 1000MB/100 usuários
work_mem = 10MB

O arquivo de configuração padrão do PostgreSQL define esse valor com *work_mem* = 1MB. Como se está utilizando testes baseados em acessos concorrentes não eleva-se muito esse valor, e define-se com 2MB.

- *Wal_buffers* – em bancos que possuem muitas transações, deixar esse parâmetro com seu valor padrão pode diminuir o desempenho. Isso deve-se ao fato que bancos transacionais executam muitas transações de escrita que precisam ser recuperada caso ocorra uma falha. Como é esse o parâmetro que define o numero de buffers que podem ser utilizados pelo WAL (*Write Ahead Log*) que garante que registros sejam gravados em LOG para possível recuperação em caso de falha, optou-se por seu valor padrão de 64KB para 128KB.
- *effective_cache_size* – considerado por muitos como o parâmetro que mais adiciona desempenho no banco de dados. É normalmente configurado com tamanho igual a 25% do total de memória RAM disponível. O valor padrão do PostgreSQL vem ajustado com *effectice_cache_size* =128MB. Como possuímos um pouco mais do que 1GB de memória disponível, aumentou-se os o valor desse parâmetro para 256MB.
- *max_connections* e *shared_buffers* influenciam no tamanho do segmento de memória compartilhada do sistema operacional. Em alguns sistemas operacionais esse valor de memória compartilhada ajusta-se automaticamente, como por exemplo, o Windows XP. Já no Linux quando o valor de memória compartilhada é excedido, o servidor PostgreSQL não consegue inicializar, provocando uma falha de segmento. Desta forma o administrador do sistema deve aumentar o valor de memória compartilhada do sistema operacional Linux alterando uma variável do Kernel chamada *shmmax*. O valor que deve ser atribuído a essa variável pode ser calculado da seguinte forma: *Shmmax* final = (*shmmaxdefault* * 1024) + 250kb + (8.2kb * *shared_buffers*) + (14.2kb * *max_connections*)

Como alterou-se valor de *shared_buffers* na otimização, foi preciso aumentar o valor de memória compartilhada no Linux. Definiu-se o valor de *shmmax* = 89302784.

Após análise dos resultados referentes a otimização 1 desenvolveu-se uma nova configuração para o SGBD PostgreSQL, com o objetivo de tentar melhorar seu desempenho e para auxiliar na análise de seu comportamento em um ambiente OLTP quando o número de conexões concorrentes é aumentado.

Foram alterados os seguintes parâmetros para esta otimização:

- *max_connection* – este parâmetro foi alterado de 100 para 120, com o objetivo de elevar a capacidade de conexões concorrentes no SGBD PostgreSQL.
- *shared_buffers* – este parâmetro também foi aumentado, de 64MB para 128MB, aumentar a memória compartilhada, visto que, mais conexões serão permitidas pelo SGBD PostgreSQL.
- *effective_cache_size* – aumentamos este parâmetro de 256MB para 512MB, para permitir que mais memória seja utilizada para *cache* do SGBD.

As tabelas com as configurações otimizadas são mostradas conforme segue:

Tabela 5: Resultados obtidos com as configurações de otimização para execução de 1 warehouses e 10 terminais

WareHouses 1 - Terminais 10				
Sistema Operacional	Configuração	Nº Transações	Média TPm	Duração (min)
Windows - NTFS	PostgreSQL	13078	577,4689178	10
	MySQL	19546	887,6759983	10
	Otimização 1	19485	875,0847805	10
	Otimização 2	21533	960	10
Linux - ReiserFS	PostgreSQL	21961	993,6732044	10
	MySQL	17846	796,1059276	10
	Otimização 1	24159	1087,337227	10
	Otimização 2	25220	1131,795429	10
Linux - Ext3	PostgreSQL	11744	536,5409173	10
	MySQL	14957	673,0266379	10
	Otimização 1	16948	756,4587944	10
	Otimização 2	23954	1039,800706	10

Tabela 6: Resultados obtidos com as configurações de otimização para execução de 10 warehouses e 10 terminais

WareHouses 10 - Terminais 10				
Sistema Operacional	Configuração	Nº Transações	Média TPm	Duração (min)
Windows - NTFS	PostgreSQL	7413	330,4362277	10
	MySQL	13416	592,6841333	10
	Otimização 1	7930	358,3240263	10
	Otimização 2	7568	344,9668244	10
Linux - ReiserFS	PostgreSQL	10649	476,1026567	10
	MySQL	13187	595,8599729	10
	Otimização 1	12187	553,4874576	10
	Otimização 2	12748	573,5600607	10
Linux - Ext3	PostgreSQL	6547	294,8913817	10
	MySQL	11154	361,796517	10
	Otimização 1	7725	348,4984517	10
	Otimização 2	8219	364,3128001	10

Conclusões

Este trabalho teve como objetivo abordar o *tuning*, que é um processo evolutivo de análise e otimização que pode ser realizado nos SGBDs, com a finalidade de melhorar o seu desempenho de acordo com a aplicação que está sendo utilizada. Aqui foi enfatizada a configuração dos parâmetros do SGBD PostgreSQL para otimizá-lo para um ambiente OLTP, visto que, esse ambiente que apresenta inúmeras transações concorrentes executando consultas simples não é favorável a esse SGBD, o qual comporta-se de forma eficiente em base de dados grandes e quando executa funções

complexas. Para facilitar o processo de testes e otimização dos parâmetros do PostgreSQL foi desenvolvida uma ferramenta capaz de automatizar o processo de leitura e modificação desse parâmetros. Percebe-se que ao modificar certos parâmetros do PostgreSQL consegue-se uma melhora significativa de desempenho. Percebe-se também que liberar demasiadamente recursos do sistema para o SGBD nem sempre resulta em aumento de desempenho, muito pelo contrário, às vezes provoca perda de desempenho. Conclui-se que no momento de escolher um SGBD para uma aplicação especifica é necessário saber quais recursos serão necessários.

Trabalhos Futuros

Os testes realizados utilizaram um dos Benchmarks padrões da TPC: o TPC-C o qual mede o desempenho de sistemas em ambientes OLTP. A proposta é realizar testes utilizando os Benchmarks TPC-H e TPCW. O primeiro simula um sistema de suporte à decisão com grandes volumes de dados, sincronizado com bancos de dados de produção on-line. O segundo simula as atividades de um servidor Web dedicado a transações de negócios.

Referências

Date, C. J. *Introdução a Sistemas de Bancos de Dados*. Campus, Rio de Janeiro, 2004.

Elmasri, R. E., & Navathe, S. (2006). *Sistema de Banco de Dados.* Pearson, São Paulo, 2005.

Korth, H. F. *Sistema de Banco de Dados*. Campus, Rio de Janeiro, 2005.

Oltp, W. (2007, Nov). *Online Transaction Processing.* Disponível em http://en.wikipedia.org/wiki/OLTP

Rodrigues, S. (2007, Set). *Conceitos Básicos de BD, SBD e SGBD*. Disponível em http://www.sergiorodrigues.net/aulas/downloads/bd1/bd1_apostila1_conceitosBasicos.pdf

SourceForge. (2006, Dec). *BenchmarkSQL*. Disponível em https://burley2-0.mptest.sf.net/project/showfiles.php?group_id=121036

SourceForge. (2007, Nov). *Database Test Suite*. Disponível em http://osdldbt.sourceforge.net/

SourceForge. (2007, Nov). *The Open Source Database Benchmark*. Disponível em http://osdb.sourceforge.net/