Tuning de Banco de Dados: PostgreSQL

Sílvia Lúcia Borowicc
Chapecó – SC - Brasil
silvialucia.borovicc@gmail.com

Resumo. *O estudo apresentado neste artigo teve como principal objetivo implementar ajustes nas configurações do Sistema Gerenciador de banco de dados PostgreSQL visando melhorar seu desempenho. Apresenta breve histórico do PostgreSQL e descreve a forma de instalação, algumas características importantes, parâmetros a configurar bem como a funcionalidade de cada um. Ao final são apresentados os resultados de testes de stress realizados antes e depois dos ajustes.*
Palavras-chave: PostgreSQL, Desempenho, Tuning.

1 Introdução

Este artigo apresenta breve histórico e algumas características importantes do SGDB (Sistema Gerenciador de Banco de Dados) PostgreSQL.

O objetivo do estudo foi pesquisar e aplicar recursos de configuração e ajuste no SGBD para melhorar sua performance e descrever as medidas tomadas para atingir esta melhoria.

O trabalho está estruturado da seguinte forma, na sessão 2 consta um breve histórico e determinadas características do PostgreSQL, na sessão 3 está descrito o processo de instalação do SGBD, na sessão 4 foram relacionados os parâmetros configurados no processo de ajuste e a funcionalidade oferecida por cada um deles, no item 5 apresentam-se recursos que permitem medir o desempenho das consultas SQL e consequentemente auxiliam na sua otimização, no item 6 os relatórios de testes realizados antes e após os ajustes demonstram os resultados do processo, no item 7 referências bibliográficas.

2 Histórico e Características do PostgreSQL

O PostgreSQL é um sistema gerenciador de banco de dados relacional cuja construção se iniciou em 1986, na Universidade Norte-Americana da Califórnia, a Universidade de Berkeley sendo descendente do Ingres. A abertura completa de seu código se deu em 1996. É considerado de alta performance e de fácil administração e utilização (PEREIRA NETO, 2003).

Dentre as características do SGBD este artigo apresenta índices e a implementação de índices hipotéticos, clusterização, Full Text Search, window function e herança.

a. Índices

Usado com o propósito de melhorar o desempenho do banco de dados, no entanto deve ser usado de forma cautelosa pois também adiciona sobrecarga ao banco de dados como um todo.

O PostgreSQL permite usar quatro tipos de índices, cada um com um algoritmo diferente e adequado para diferentes tipos de consultas. São eles: B-tree, Hash, GiST e GIN. Por padrão o comando CREATE INDEX cria B-tree.

Índices hipotéticos, implementados pelo PostgreSQL, são estruturas de simulação de índices. Os índices são criados exclusivamente no tipo catalog sem extensão física e, portanto, não podem ser usados para consultas. O principal benefício é proporcionar um meio para simular quais mudanças nos planos de execução de consulta aconteceriam se os índices hipotéticos fossem de fato criados no banco de dados.

b. Clusterização

Clusterização consiste em instruir o PostgreSQL a agrupar uma tabela com base em um determinado índice. Quando é realizado este agrupamento a tabela é fisicamente reordenada. Atualizações posteriores não serão agrupadas, ou seja, será necessário reaplicar o comando periodicamente. Pode-se usar também as ações CLUSTER ou SET WITHOUT CLUSTER do comando ALTER TABLE para definir qual índice será usado nos processos futuros de clusterização ou ainda remover as especificações de cluster usadas anteriormente.

O comando CLUSTER sem nenhum parâmetro reagrupa todas as tabelas cujo usuário seja dono e que foram previamente agrupadas no banco de dados atual. Ou todas as tabelas caso o comando seja executado por um superusuário.

c. Full Text Search

Fornece a capacidade de identificar documentos em linguagem natural que satisfazem uma consulta e, opcionalmente, classificar os resultados por relevância. O tipo mais comum é buscar todos os documentos que contêm os termos informados na consulta e retornar os resultados em ordem de similaridade com os mesmos.

d. Window Function

Executa cálculos em conjuntos de linhas relacionadas à linha atual da consulta. Os resultados são comparáveis às funções de agregação, mas este processo não causa agrupamento de registros em um único registro de saída pois a função é capaz de acessar mais linhas além da atual resultante da consulta.

e. Herança e Sobrecarga de Funções

O PostgreSQL implementa o conceito de herança permitindo que tabelas mais especializadas herdem características de outras tabelas mais generalizadas em uma mesma classificação de entidades. Na herança de tabelas do PostgreSQL os atributos se repetem fisicamente ao contrário do que se refere às classes em Orientação a Objetos.

Quando há uma inserção em uma tabela especializada os dados são inseridos na tabela generalizada. O mesmo não se aplica quando a inserção é feita na tabela generalizada já que nem todo registro possui especializações. Há possibilidade de manipular os registros sem especialização utilizando ONLY no comando utilizado, como por exemplo: SELECT * FROM ONLY [tabela_generalizada].

No PostgreSQL há a possibilidade de realizar sobrecarga de funções. Funções com mesmo nome apresentam comportamentos diferentes de acordo com a parametrização que é diferente para cada implementação. A sobrecarga de funções aliada à herança é característica importante do polimorfismo em Orientação a Objetos.

3 Instalação do PostgreSQL

Esta instalação foi realizada no sistema operacional Ubuntu Server.

Instalando PostgreSQL via apt-get:

<$ sudo apt-get install postgresql-8.4 postgresql-contrib-8.4>

Configurar a senha para usuário postgres:

<$ sudo su - postgres -c psql>
<postgres=# alter role postgres with password ‘senha’;>
<postgres=# \q> sair

A máquina possui um processador 64 bits com 4 núcleos, 4Gb de RAM e dois discos sendo utilizados um para dados e outro para o sistema operacional, aplicação e o SGBD. Por questões de padronização foram criadas duas tablespaces, uma para dados e outra para índices.

4 Parâmetros a Configurar

Configurações do PostgreSQL podem ser manipuladas de diversas maneiras mas geralmente são realizadas no arquivo postgresql.conf. O arquivo é encontrado em $PGDATA / postgresql.conf.

As linhas do arquivo precedidas de ‘#’ são comentários. Quando uma propriedade estiver presente mais de uma vez no arquivo a última vai vigorar. As alterações neste arquivo só passam a valer após um reload </etc/init.d/postgresql reload > ou restart </etc/init.d/postgresql restart>.

As alterações das configurações conforme descritas a seguir apresentam bons resultados, obviamente em conjunto com boas práticas. Para ver as configurações atuais basta acessar a visão *pg_settings*.

a. listen_addresses

Lista os endereços dos quais será possível obter resposta do PostgreSQL. Por padrão é o host local. Alterar para ouvir todos os endereços: listen_addresses = ‘ * ’. Posteriormente através do arquivo $PGDATA / pg_hba.conf controlam-se os endereços que poderão se conectar.

b. max_connections

Número máximo de conexões permitidas. Esta propriedade impacta em outras propriedades como work_mem, pois alguns recursos de memória serão alocados por cliente e neste caso este parâmetro sugere o uso máximo de memória possível.

O PostgreSQL suporta algumas centenas de ligações, sendo necessárias milhares recomenda-se o uso de um software para pool de conexões como pgpool-II reduzindo sobrecarga de conexão.

c. shared_buffers

Este parâmetro determina quanta memória será dedicada ao PostgreSQL para cache de dados. Este recurso é limitado por um parâmetro do Kernel do sistema operacional, SHMMAX que deve ser alterado quando deseja-se aumentar shared_buffers para mais de 32MB.

Para um sistema com mais de 1GB de RAM ¼ do total é um valor razoável para este parâmetro.

d. effective_cache_size

Utilizado pelo planejador de consultas como estimativa de quanta memória está disponível incluindo-se inclusive a memória dedicada ao banco de dados. Se esta configuração for muito baixa a utilização de índices nas consultas não se dará conforme esperado. Um número razoável, ainda um tanto conservador, é ½ da RAM. Pode-se encontrar uma melhor estimativa observando estatísticas do SO.

e. checkpoint_completion_target

Especifica o objetivo de conclusão do checkpoint, como uma fração do tempo total entre checkpoints. Por default é 0.5. Pode-se aumentar para 0,9 diminuindo a sobrecarga de escrita média.

f. checkpoint_segments

Número máximo de segmentos de arquivo de log entre pontos de verificação automáticos do Write Ahead Log (cada segmento é normalmente 16 megabytes). O padrão é três segmentos. O aumento deste parâmetro pode aumentar a quantidade de tempo necessário para a recuperação de falhas. Valores de 32 (a cada 512 MB) a 256 (a cada 4GB) são comuns atualmente.

g. checkpoint_timeout

Intervalo de tempo entre os checkpoints.

h. autovaccum

Determina a execução automática de recuperação de espaço em disco ocupado com lixo de tuplas mortas. Ativado por padrão e deve permanecer inalterado.

i. default_statistics_target

Define o padrão de coleta de informações de estatística. Valores maiores aumentam o tempo necessário para fazer ANALYZE, mas podem melhorar a qualidade das estimativas do planejador. O padrão é 100.

j. work_mem

Especifica a quantidade de memória para ser usado por operações internas de classificação e tabelas de hash antes de gravar arquivos temporários em disco. O valor padrão é um megabyte (1 MB). É importante lembrar que este valor será usado por cada operação em cada sessão em execução.

k. maintenance_work_mem

Especifica a quantidade máxima de memória a ser utilizada pelas operações de manutenção, como VACUUM, CREATE INDEX e ALTER TABLE ADD FOREIGN KEY. O padrão é 16MB. Normalmente não há muitas dessas operações rodando simultaneamente. Configurações maiores podem melhorar o desempenho em VACUUM e restauração de backups.

Quando autovacuum é executado, essa memória pode ser alocada até tantas vezes o valor definido em autovacuum_max_workers (define o número máximo de processos autovacuum que pode ser executado). Portanto, deve haver cuidado para não aumentar muito o valor deste parâmetro.

l. random_page_cost

Esta configuração sugere ao otimizador quanto tempo vai levar procurar uma página aleatória do disco, como um múltiplo de quanto tempo uma leitura seqüencial leva.

O uso de valores menores irá incentivar o otimizador de consulta a usar varreduras de índice de acesso aleatório.

m. max_locks_per_transaction

Este parâmetro controla o número médio de locks de objetos alocado para cada transação. Será necessário aumentar este valor caso existam clientes que acessam muitas tabelas diferentes em uma mesma transação.

5 SQL

Observar estatísticas quanto aos objetos do banco e utilizar comandos para analisar consultas permitem ao desenvolvedor obter melhor performance alterando seu código e tomando decisões apropriadas para tarefas como a criação de índices.

No PostgreSQL o comando <EXPLAIN ANALIZE [CONSULTA]> permite verificar estatísticas de custo de execução da query em questão. A ferramenta pgAdmin pode ser usada para visualizar de forma gráfica o resultados destes testes verificando o custo da operação.

A utilização de linguagens que permitam executar instruções de maior complexidade, como PL/pgSQL, oferecem possibilidade de diminuir o número de acessos ao disco durante determinadas operações.

6 Testes de ajustes

Foram realizados testes de performance para verificar as mudanças provocadas pelos ajustes realizados nos parâmetros descritos no item 4 cujos resultados são apresentados no subitem 6.1. Os testes foram realizados em uma base de aproximadamente 40GB.

6.1 Consulta de performance: ferramenta de stress pgbench

A ferramenta utilizada é do pacote contrib do PostgreSQL cuja carga se dá pelo comando: <./pgbench -i –s 10 -h localhost -p 5432>. A consulta de teste é realizada pelo comando: < ./pgbench -t 80 -c 80 -U postgres banco_de_dados>.

No quadro abaixo, resultados da consulta antes das alterações dos parâmetros descritos no item 5.

1. starting vacuum...end.
2. transaction type: TPC-B (sort of)
3. scaling factor: 100
4. query mode: simple
5. number of clients: 80
6. number of transactions per client: 80
7. number of transactions actually processed: 6400/6400
8. tps = 34.738978 (including connections establishing)
9. tps = 34.781432 (excluding connections establishing)

No próximo quadro, resultados após as alterações dos parâmetros descritos no item 5.

1. starting vacuum...end.
2. transaction type: TPC-B (sort of)
3. scaling factor: 10
4. query mode: simple
5. number of clients: 80
6. number of transactions per client: 80
7. number of transactions actually processed: 6400/6400
8. tps = 223.619417 (including connections establishing)
9. tps = 225.397033 (excluding connections establishing)

7 Referências Bibliográficas

BROWNE, C; TREAT, R; SMITH, G. **Tuning Your PostgreSQL Server**. Disponível em <http://wiki.postgresql.org/wiki/Tuning_Your_PostgreSQL_Server>. Acesso em 07 Jul. 2011.

PEREIRA NETO, ALVARO. **PostgreSQL: técnicas avançadas: versões open source: soluções para desenvolvedores e administradores de banco de dados**. São Paulo: Érica, 2003. 284 p.

PGBOUNCER. **PgBouncer**. Disponível em <http://wiki.postgresql.org/wiki/ PgBouncer>. Acesso em 07 Jul. 2011.

PGPOOL. **What is pgpool-II?**. Diponível em <http://pgpool.projects.postgresql.org/>. Acesso em 07 Jul. 2011.

POSTGRESQL. **Documentation**. Disponível em <http://www.postgresql.org/docs/>. Acesso em 07 Jul. 2011.