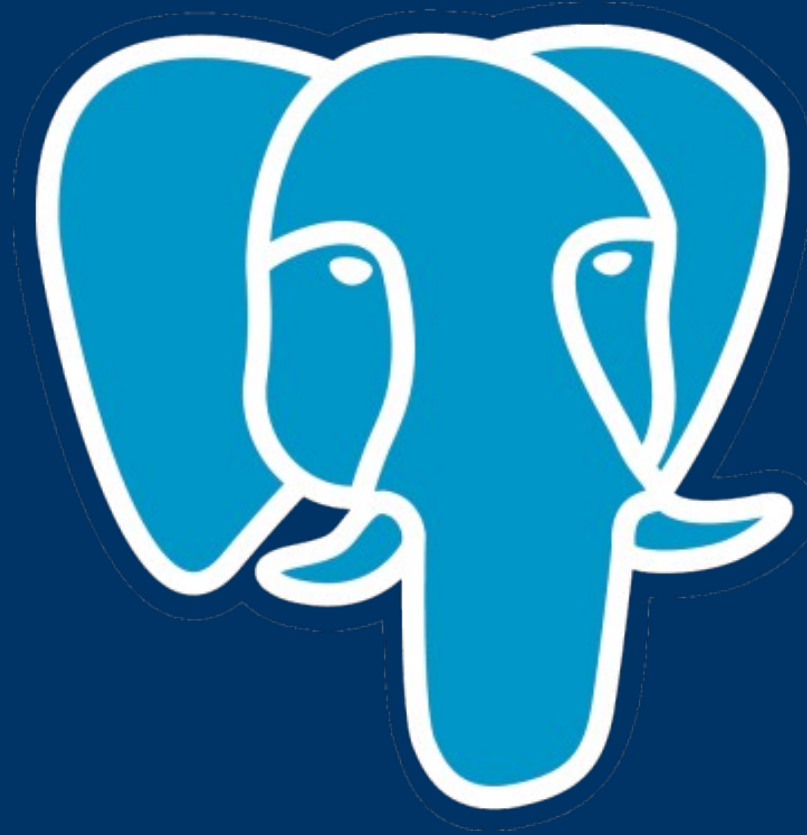


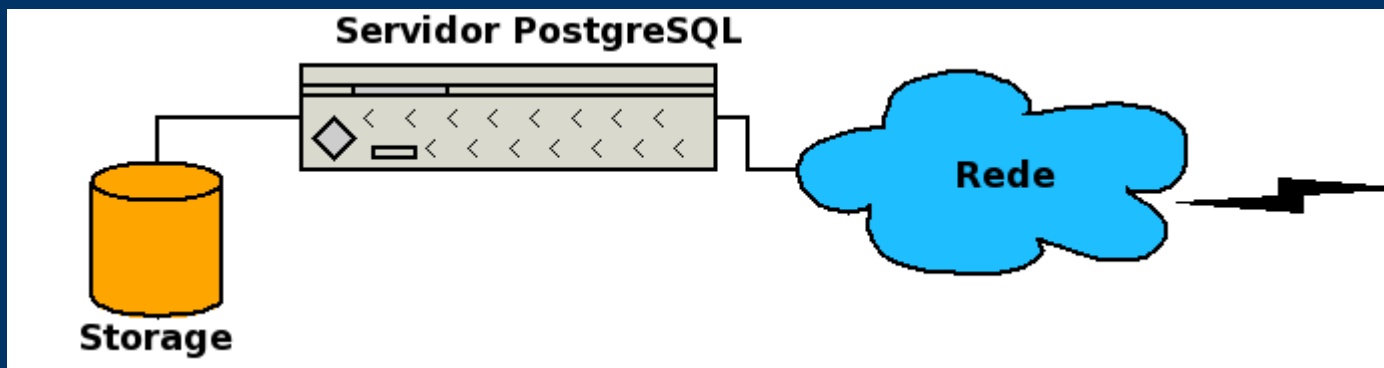
Turbinando o PostgreSQL



Euler Taveira de Oliveira
euler@timbira.com



Ambiente Ideal



- CPU: 32 < 64 bits
- Disco: IDE < SCSI < SAN (FC + iSCSI)
- Memória: unbuffered < registered < ECC
- Rede: ATM < Gigabit < 10-Gigabit

Rede = CPU < RAM < Disco

Sistemas Operacionais

- Linux (POSIX)
- FreeBSD (POSIX)
- M\$ Window\$ (não-POSIX)
- *NIX (POSIX)

Window\$ < *NIX < FreeBSD = Linux

Sistema de Arquivos Linux

XFS

v

EXT3

v

JFS

v

ReiserFS



Layout dos dados

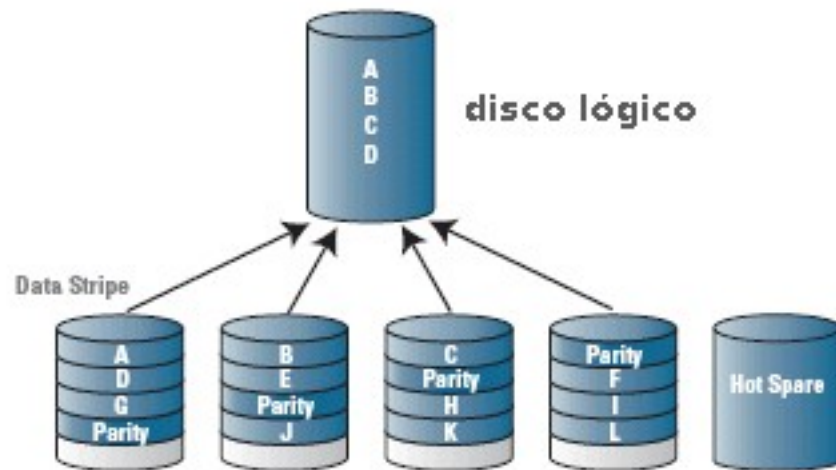
- separar dados do log de transação (pg_xlog)
- separar dados de índices
- separar dados em espaço de tabelas (tablespaces)
- fazer particionamento de tabelas
 - utilizar `constraint_exclusion`



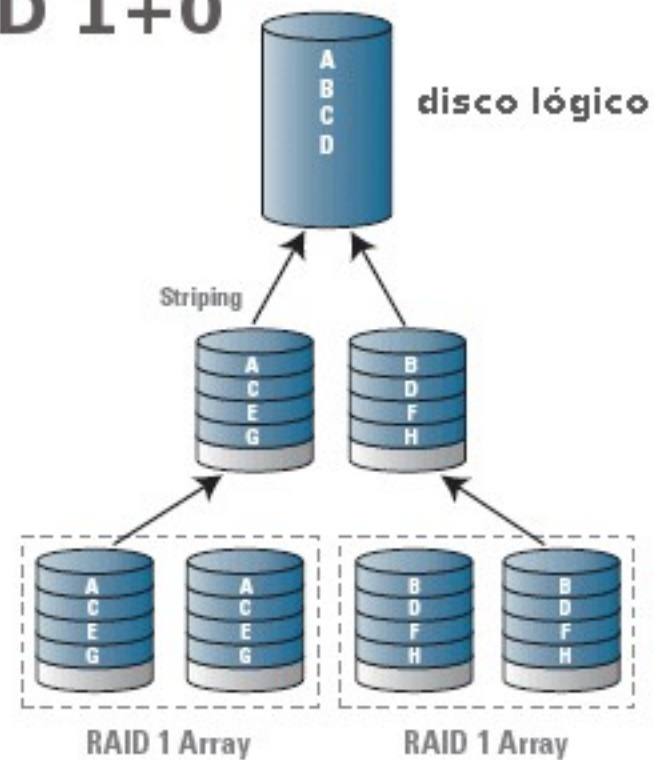
RAID

RAID 5 < RAID 1+0

RAID 5



RAID 1+0



Ferramentas

- `pgbench`
 - distribuído com PostgreSQL (contrib)
 - TPC-B (obsoleto): mede transações por segundo (tps)
 - medir latência de transação (ajuste)
- DBT-2
 - feito pela OSDL Labs
 - TPC-C: mede transações por minuto (tpmC)
 - identificar gargalos de I/O e uso de memória (ajuste fino)

Parâmetros - Conexões

- `listen_addresses`
 - aceitar conexões de onde?
- `max_connections`
 - máximo de conexões simultâneas



Parâmetros - Memória

- `shared_buffers`
 - memória utilizada p/ operações ativas
 - `temp_buffers`
 - utilizado p/ acesso a tabelas temporárias
 - `work_mem`
 - memória utilizada p/ ordenações, agregações, outras
 - `maintenance_work_mem`
 - memória utilizada p/ operações de manutenção (ANALYZE, VACUUM, CREATE INDEX, etc)
 - 50% a 75% do tamanho da maior tabela/índice é um bom número!
-
-

Parâmetros – Disco e WAL

- `max_fsm_pages`
 - máximo de páginas necessárias p/ mapear espaço livre
 - é definido como número de páginas de sofrem UPDATE/DELETE
 - `vacuum_cost_delay`
 - tempo que o processo irá adormecer se limite de custo for atingido
 - `fsync`
 - garante que o SO escreveu o dado no log de escrita prévia (WAL)
 - só desabilite esta opção se você confia no SO, hardware, nobreak
-
-

Parâmetros – Disco e WAL (2)

- wal_buffers
 - páginas de disco alocadas p/ WAL
 - deve ser o tamanho necessário p/ armazenar dados do WAL de uma transação
 - commit_delay
 - tempo entre efetivar registro no WAL e escrever no disco
 - permite efetivar várias transações no mesmo “fsync”
 - checkpoint_segments
 - tamanho da cache em disco p/ operações de escrita
 - checkpoint_timeout
 - tempo entre operações de ponto de controle (checkpoint)
 - em sistemas com grande volume de dados isso pode melhorar a performance
-
-

Parâmetros - Planejador

- `effective_cache_size`
 - número de páginas disponíveis para busca com índices
 - mede possibilidade de um índice ser utilizado x busca sequencial
- `random_page_cost`
 - custo estimado de uma busca não sequencial de página no disco
 - valor menor aumenta chance de uso de índices



Parâmetros - Estatísticas

- stats_start_collector
 - habilita coletor de estatísticas
 - stats_block_level
 - nível de blocos
 - stats_row_level
 - nível de registros
 - Visões
 - pg_statio_* (blocos)
 - pg_stat_* (registros)
-
-

Ilustrando com pgbench

Athon XP 2400+ 2.0 Ghz, 512MB, HD IDE 40GB
Slackware Linux 2.4.29, EXT3

```
starting vacuum...end.  
starting full vacuum...end.  
transaction type: TPC-B (sort of)  
scaling factor: 50  
number of clients: 40  
number of transactions per client: 200  
number of transactions actually processed: 8000/8000  
tps = 40.559690 (including connections establishing)  
tps = 40.611344 (excluding connections establishing)  
starting vacuum...end.  
starting full vacuum...end.  
transaction type: TPC-B (sort of)  
scaling factor: 50  
number of clients: 40  
number of transactions per client: 200  
number of transactions actually processed: 8000/8000  
tps = 43.423502 (including connections establishing)  
tps = 43.476473 (excluding connections establishing)  
starting vacuum...end.  
starting full vacuum...end.  
transaction type: TPC-B (sort of)  
scaling factor: 50  
number of clients: 40  
number of transactions per client: 200  
number of transactions actually processed: 8000/8000  
tps = 43.591294 (including connections establishing)  
tps = 43.652545 (excluding connections establishing)
```

padrão
stats_block_level = on
stats_row_level = on

shared_buffers = 5000
work_mem = 16384

wal_buffers = 64
commit_delay = 1000
checkpoint_segments = 15
checkpoint_timeout = 1800

Manutenção do Banco de Dados

- ANALYZE
 - coleta informações
 - VACUUM
 - recupera espaço e disponibiliza p/ uso
 - VACUUM FULL
 - VACUUM + reorganiza dados
 - autovacuum
 - limpeza automática periodicamente
 - baseia-se em estatísticas
 - executado por período de tempo
 - autovacuum_napstime
-
-

Minhas consultas ainda estão lentas

- verificar a existência de índices
- executar EXPLAIN na consulta
- utilizar comandos preparados
- habilitar/desabilitar métodos do planejador
 - seqscan, bitmapscan, indexscan, nestloop, etc
- aumentar/diminuir parâmetros em tempo de execução



Entendendo EXPLAIN

```
ovg_tst=# EXPLAIN SELECT cd_bem,ds_bem,id_conta,nm_conta,COALESCE(sg_depto, nm_depto) AS nm_depto,  
ovg_tst=# TO_CHAR(dt_aquisicao, 'DD/MM/YYYY') AS dt_aquisicao,vl_aquisicao,  
ovg_tst=# scp.calculaDepreciacao(id_bem,'2006-10-31') AS depreciado  
ovg_tst=# FROM scp.bens a INNER JOIN contas ON con_id_conta = id_conta  
ovg_tst=# LEFT JOIN stp.departamentos ON a.dep_id_depto = id_depto  
ovg_tst=# WHERE a.dep_id_depto = 22 AND id_conta IN (7, 6, 5, 1) AND vl_contabil IS TRUE AND  
ovg_tst=# dt_aquisicao < '2006-10-31'  
ovg_tst=# ORDER BY nm_conta,ds_bem ASC;
```

QUERY PLAN

```
-----  
-----  
Sort (cost=27.70..27.70 rows=1 width=192)  
→ Sort Key: contas.nm_conta, a.ds_bem  
→ Nested Loop Left Join (cost=0.00..27.69 rows=1 width=192)  
  Join Filter: ("outer".dep_id_depto = "inner".id_depto)  
  → Nested Loop (cost=0.00..21.85 rows=1 width=134)  
    → Seq Scan on bens a (cost=0.00..16.00 rows=1 width=86)  
      Filter: ((dep_id_depto = 22) AND (vl_contabil IS TRUE) AND (dt_aquisicao < '2006-10-  
31'::date))  
    → Index Scan using contas_pkey on contas (cost=0.00..5.83 rows=1 width=52)  
      Index Cond: ("outer".con_id_conta = contas.id_conta)  
      Filter: ((id_conta = 7) OR (id_conta = 6) OR (id_conta = 5) OR (id_conta = 1))  
  → Index Scan using departamentos_pkey on departamentos (cost=0.00..5.82 rows=1 width=66)  
    Index Cond: (id_depto = 22)  
  
(12 registros)  
  
ovg_tst=# █
```

Futuro

- melhorar performance do COUNT(*)
 - cache em vários níveis (análise, plano, resultado)
 - melhorar performance do VACUUM
 - implementar buscas sequenciais concorrentes
 - compilar funções
 - utilizar Async I/O (Linux)
 - armazenar dados mais compactados
 - TODO?
-
-

Referências

<http://www.postgresql.org/>
<http://www.postgresql.org.br/>

pgsql-performance@postgresql.org
brasil-usuarios@pgfoundry.org



Vida Longa ao PostgreSQL



Contato

Euler Taveira de Oliveira
euler@timbira.com
<http://www.timbira.com/>
