# PostgreSQL - Cluster, Alta Disponibilidade e Balanceamento de Carga

0.1

Brasília – DF

Organização
Eduardo Santos

Autoria
Elias Mussi

PostgreSQL - Cluster, Alta Disponibilidade e Balanceamento de Carga.
Brasília, 2013.
48 p. : il.

Inclui Bibliografia.

1. PostgreSQL. 2. Cluster e Grid. 3. Tecnologias da Informação e Comunicação. 4. Alta Disponibilidade.

**“A menos que modifiquemos a nossa maneira de pensar, não seremos capazes de resolver os problemas causados pela forma como nos acostumamos a ver o mundo”.**

Albert Einstein

# Coordenação

Textos compilados por Eduardo Santos
Material contido no Guia de Cluster e Grid do Governo Federal

O material aqui contido é resultado de uma compilação de textos do Guia de Cluster e Grid do Governo Federal. Assim como o texto original, obedece à licença Creative Commons 2.0 Atribuição. A reprodução em parte ou totalmente é autorizada desde que a fonte seja reconhecida, de acordo com as orientações da CC-GNU GPL, cujo conteúdo está disponibilizado no Apêndice A.


**Figura 1:** Creative Commons

O Guia de Cluster original está disponível no Portal do Governo Eletrônico, no endereço http://www.governoeletronico.gov.br/anexos/guia-de-cluster

# Sumário

# Lista de Figuras

# Lista de Tabelas

# Capítulo 1

# Introdução

O documento descrever as tecnologias de cluster e grid utilizadas para a implementação de cluster em bancos de dados corporativos baseados no SGBD PostgreSQL.

## 1.1 O que é o PostgreSQL?

O PostgreSQL é um SGBD (Sistema Gerenciador de Banco de Dados) objeto-relacional de código aberto, com mais de 15 anos de desenvolvimento. É extremamente robusto e confiável, além de ser extremamente flexível e rico em recursos. Ele é considerado objeto-relacional por implementar, além das características de um SGBD relacional, algumas características de orientação a objetos, como herança e tipos de dados personalizados.

## 1.2 Histórico

**1985** Primeiro desenho de um possível sistema de armazenamento (ainda projeto Postgres) na universidade de Berkeley, Califórnia [13]

**1987** Primeira modelagem [14]

**1989** Definição do primeiro sistema de regras [15]
**Fim da fase acadêmica**

**1993** Empresa Illustra transforma parte do código em um produto comercial. Mais tarde a empresa se tornaria Informix, até ser comprada pela IBM

**1993** Desenvolvimento de uma versão paralela pela comunidade (principalmente universidades) e novo nome: **PostgreSQL**

# Capítulo 2

# Cluster e Grid

## 2.1 As Novas Demandas Computacionais

As atividades econômicas que utilizam redes eletrônicas como plataforma tecnológica têm sido denominadas negócios eletrônicos (*e-business*). Essa expressão engloba os diversos tipos de transações comerciais, administrativas e contábeis, que envolvem governo, empresas e consumidores. O comércio eletrônico (*e-commerce*) é a principal atividade dessa nova categoria de negócios.

Os atores institucionais envolvidos nos serviços governamentais são o próprio Governo ($G$), Instituições Externas ($B$, de *business*), e o Cidadão ($C$), que interagem entre si de várias maneiras. Há cinco tipos de relações entre esses atores em aplicações governamentais:

- $B2B$ (*business-to-business*):
  transações entre empresas, exemplos: EDI, portais verticais de negócios;
- $B2C/C2B$ (*business-to-consumer/consumer-to-business*):
  transações entre empresas e consumidores, exemplos: lojas e shoppings virtuais;
- $B2G/G2B$ (*business-to-government/government-to-business*):
  transações envolvendo empresas e governo, exemplos: EDI, portais, compras. Corresponde a ações do Governo que envolvem interação com enti-

dades externas. O exemplo mais concreto deste tipo é a condução de compras, contratações, licitações, etc, via meios eletrônicos;

- *C2C* (*consumer-to-consumer*):
  transações entre consumidores finais (exemplos: sites de leilões, classificados on-line);

- *G2C/C2G* (*government-to-consumer/consumer-to-government*):
  transações envolvendo governo e o cidadão (consumidores finais dos serviços do Governo), exemplos: pagamento de impostos, serviços de comunicação). Corresponde a ações do Governo de prestação (ou recebimento) de informações e serviços ao cidadão via meios eletrônicos. O exemplo mais comum deste tipo é a veiculação de informações em um *website* de um órgão do governo, aberto a todos os interessados;

- *G2G* (*government-to-government*):
  transações entre instituições do governo em qualquer nível ou esfera do Poder. Corresponde a funções que integram ações do Governo horizontalmente, exemplo: no nível Federal, ou dentro do Executivo; ou verticalmente, exemplo: entre o Governo Federal e um Governo Estadual.

Tais aplicações, especialmente as de escala nacional, que costumam tratar de imensas quantidades de dados, que perpassarão inúmeras gerações tecnológicas, são tão carregadas de variáveis e condicionantes que, tipicamente, são descritos e caracterizados como "sistemas complexos":

- têm dimensões gigantescas, tais como milhões de usuários, centenas de funções, etc.;

- têm especificação dinâmica, isto é, se modifica ao longo do tempo, para acomodar novas necessidades, revisão de prioridades, etc.;

- nunca terminam de ser implementados, como conseqüência natural das duas características anteriores.

A ampliação e a melhoria da qualidade dos processos internos e dos serviços prestados refletem-se na necessidade de aumento da capacidade computacional do setor público e, por se tratarem de serviços críticos, possuem como características principais de demandas computacionais:

- alta disponibilidade;
- suporte a milhões de usuários simultâneos;
- alta capacidade de processamento;
- capacidade de trabalhar com bancos de dados da ordem de milhões de registros;
- tolerância a falhas de *hardware* e *software*;
- facilidade de integração e interoperabilidade;
- adoção de padrões abertos de *hardware* e *software*;
- armazenamento massivo da ordem de TeraBytes de dados;

### 2.1.1 Computação sob Demanda

Vários sistemas de governo possuem demandas flutuantes, ou seja, durante um momento convivem com pouca ou nenhuma carga de processamento e em outro momento específico possuem uma elevada carga de processamento.

Um exemplo deste perfil é o sistema de declaração de imposto de renda. Durante um período do ano é ativada a entrada de dados no sistema para a realização de declarações de imposto de renda. Quanto mais se aproxima o final deste período, maior a quantidade de declarações e, conseqüentemente, a capacidade computacional necessária para o funcionamento do sistema. O mesmo ocorre com o SIAPE, só que neste caso a utilização para processamento e entrada de dados no sistema ocorre com maior concentração durante alguns dias do mês.

A arquitetura da computação de grande porte não possui capacidade para que facilmente se aumente ou diminua seu poder computacional, sem esbarrar nas barreiras impostas por seu *hardware* especializado e proprietário, por conta de seu alto custo total de propriedade e a dificuldade de aquisição destes equipamentos. Em função desta relação de dependência, a administração pública é obrigada a adquirir um *hardware* com maior capacidade de processamento para atender a demanda de pico de processamento do sistema. Durante quase toda a sua vida

útil, o equipamento adquirido possuirá capacidade de processamento ociosa, devido às características de demandas flutuantes de processamento desses sistemas governamentais.

Em resumo, a administração alocará na maior parte do tempo recursos financeiros em um equipamento subutilizado, quando este recurso poderia ser utilizado em áreas com demandas mais urgentes.

Para equacionar questões como essas, a administração pública está em busca de alternativas computacionais baseadas em Cluster e Grid que auxiliem na resolução desse desafio de computação sob demanda, minimizando a capacidade de processamento ociosa existente dentro da administração pública, bem como a alocação de recursos financeiros desnecessários.

### 2.1.2 Aproveitamento de Ciclos Ociosos

Estima-se que a administração pública direta possua um parque computacional em torno de 300 mil estações de trabalho, que adotam um padrão de uso comum. Este padrão consiste numa maior utilização destes equipamentos durante os horários de expediente comercial de trabalho. Entretanto, até mesmo durante os horários de maior utilização destes equipamentos existem grandes períodos de ociosidade do uso de processamento dos mesmos. Este imenso recurso computacional ocioso poderia ser amplamente aproveitado através do emprego de paradigmas de computação probabilística e *Grid Computing*. Alguns possíveis usuários desta capacidade de processamento seriam: o governo através do processamento e execução de sistemas internos, e as universidades e centros de pesquisa através de projetos de pesquisa científica [9].

Existem diversos projetos mundiais de aproveitamento de recursos ociosos de processamento que demonstram o poder da computação probabilística. Alguns exemplos são: SETI@home, que posteriormente deu origem ao BOINC[1], uma infra-estrutura aberta de computação em rede; e o distributted.net[2], um projeto de computação distribuída para finalidades genéricas criado em 1997. Nestes

[1] *Berkeley Open Infrastructure for Network Computing*
http://boinc.berkeley.edu/
[2] http://distributted.net

projetos são utilizados os ciclos ociosos de processamento de máquinas interligadas na internet, não dedicadas exclusivamente à rede de processamento e dispersas mundialmente.

Uma lista extensa porém incompleta de projetos de aproveitamento de ciclos de processamento ociosos pode ser encontrada na página:
"http://en.wikipedia.org/wiki/List_of_distributed_computing_projects"

Existem no Brasil diversas atividades de pesquisa em andamento e soluções desenvolvidas em universidades brasileiras que poderiam ser utilizadas para aproveitar a capacidade de processamento ocioso das milhares de estações de trabalho do governo brasileiro. Algumas dessas soluções são: o vCluster[8] e o Ourgrid[6], desenvolvidos respectivamente pela Pontifícia Universidade Católica do Rio Grande do Sul e pela Universidade Federal de Campina Grande.

## 2.2 Dois Paradigmas Computacionais

O modelo atualmente adotado pelas empresas de informática governamentais e por muitas grandes empresas, para resolver o paradigma exposto na sessão anterior, é caracterizado principalmente pelos sistemas heterogêneos de grande porte, com a utilização de *mainframes* e supercomputadores.

A evolução das soluções de Cluster e Grid vem criando uma nova alternativa para os ambientes computacionais de grande porte. A utilização de ambientes clusterizados para computação de alta performance vem crescendo rapidamente e já é quase predominante para as grandes máquinas utilizadas nos dias de hoje.

### 2.2.1 Computação de Grande Porte

A computação de grande porte é definida como sistema de alta capacidade de computação, também conhecida como "Alta plataforma", esta é caracterizada pela utilização de *Mainframes* e supercomputadores.

*Mainframes* são sistemas de computação, dedicados normalmente ao processa-

mento de um volume grande de informações e transações. Os *mainframes* são capazes de oferecer serviços de processamento a milhares de usuários através de milhares de terminais conectados diretamente ou através de uma rede distribuída. São computadores que geralmente ocupam um grande espaço físico e necessitam de ambiente especial para seu funcionamento. Os *mainframes* são capazes de realizar operações em grande velocidade e sobre um volume muito grande de dados. Os *mainframes* nasceram em 1946 e foram constantemente aperfeiçoados. Em 7 de abril de 1964, a IBM apresentou o "System/360", *mainframe* que, na época, foi o maior projeto de uma empresa. Desde então, outras empresas, como a HP e a Burroughs (atual Unisys) lançaram seus modelos de *mainframe*.

Supercomputador é um computador com altíssima velocidade de processamento e grande capacidade de memória, empregado em pesquisas científicas e militares. Este termo é geralmente confundido com cluster, um tipo de supercomputador criado a partir da cooperação de vários computadores convencionais. Os primeiros supercomputadores foram criados na década de 1960 por Seymour Cray. Seymour Cray fundou sua própria empresa, a Cray Research, em 1970 e dominou o mercado da supercomputação durante 25 anos (1965-1990).

A distinção entre supercomputadores e *mainframes* não é clara e direta, mas genericamente são diferenciados pelas tarefas submetidas, os supercomputadores são utilizados na solução de problemas em que o tempo de cálculo é um limite (processamento), enquanto os *mainframes* são utilizados em tarefas que exigem alta disponibilidade, envolvem alta taxa de transferência de dados (internos ou externos ao sistema) e acessos simultâneos (WikiPedia[17]).

A Figura 2.1, representa o problema de escalabilidade e dimensionamento decorrente da utilização da computação de grande porte. A área mais escura reflete uma possível evolução da carga[3] de processamento em um período de tempo.

[3] A palavra carga é utilizada nesta seção para representar o uso de recurso computacional, seja de processamento, rede e armazenamento.

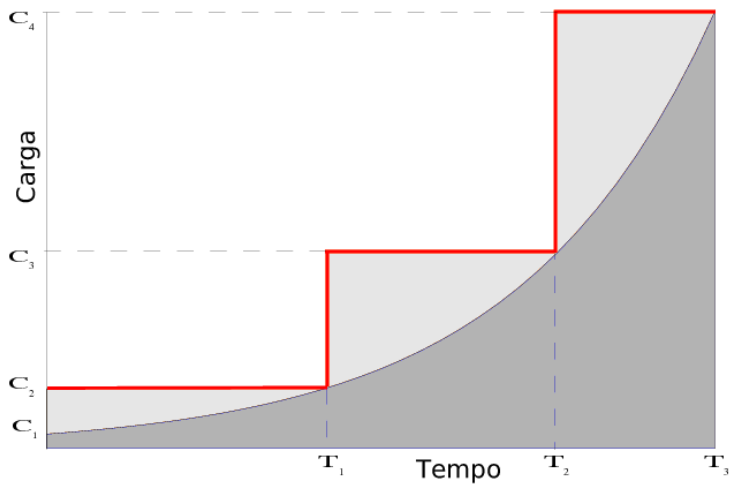


**Figura 2.1:** Evolução da carga de processamento e a utilização da computação de grande porte.

Inicialmente, para responder à uma demanda de carga de processamento $C_1$ é necessário que a Administração adquira uma solução com capacidade de processamento superior à exigência inicial. Essa medida justifica-se pelo já citado alto custo do equipamento envolvido: uma vez que recursos financeiros serão empregados na utilização de máquinas multiprocessadas, é interessante que essas máquinas operem no maior tempo possível. Dessa forma, para satisfazer a demanda de processamento destacada em $C_1$, adquire-se uma solução com capacidade $C_2$, superior, que irá garantir o funcionamento do ambiente até que a exigência de processamento atinja seu limite máximo. Na figura 2.1, a área mais clara representa a capacidade ociosa de processamento do equipamento utilizado em função do tempo.

Quando o limite de processamento do equipamento for alcançado, torna-se necessário a realização de um *upgrade*, que geralmente caracteriza-se pela substituição do equipamento original por um novo, ou pela incorporação de novo *hardware* ao equipamento original. Qualquer uma das alternativas exigirá elevado custo financeiro. Assim, passa-se à utilização de um equipamento com capacidade de processamento $C_3$, para novamente garantir a operacionalização do ambiente por mais um período de tempo ($T1$ até $T2$), inaugurando um ciclo constante de atualizações determinado pela carga de processamento a ser suportada.

No caso da utilização da computação de grande porte, percebe-se que as soluções adquiridas operam a maior parte do tempo com carga de processamento inferior à sua capacidade, devido ao alto custo de *hardware* associado à dificuldade de dimensionamento do ambiente, conforme representado pela área mais clara na Figura 2.1 e normalmente quando atingem a carga máxima, sofrem dificuldades de expansão do *hardware*(capacidade de processamento).

Portanto, em última instância, a Administração aloca recursos financeiros em uma solução cuja capacidade de processamento não será plenamente exigida na fase inicial de alocação de recursos computacionais.

### 2.2.2 Computação Distribuída

Por utilizar componentes físicos comuns em sua arquitetura, um ambiente *cluster* apresenta facilidade de dimensionamento da capacidade de processamento. O ambiente pode ser concebido de acordo com a exigência inicial da carga de processamento do ambiente. À medida que a carga aumenta, novos componentes físicos podem ser facilmente alocados no ambiente para suprir a necessidade de processamento. Como nestes ambientes podem ser empregados *hardwares* mais abertos, ou utilizados pelo mercado, consegue-se realizar um rápido dimensionamento com custo reduzido, maximizando a capacidade de processamento do ambiente.

A Figura 2.2 apresenta o resultado da utilização do ambiente *cluster*. Para efeito de ilustração foram utilizados os mesmos parâmetros da Figura 2.1 (relativa à adoção da computação de grande porte). As linhas em vermelho representam a evolução do dimensionamento da carga de processamento do ambiente *cluster*.

### 2.2.3 Comparação: Grande Porte e Distribuída

Existem similaridades e diferenças entre os ambientes de grande porte e os de computação distribuída. Embora os ambientes de computação distribuída sejam mais recentes e tenham arquitetura computacional mais moderna, alguns especialistas cogitam uma volta ao passado do grande porte.

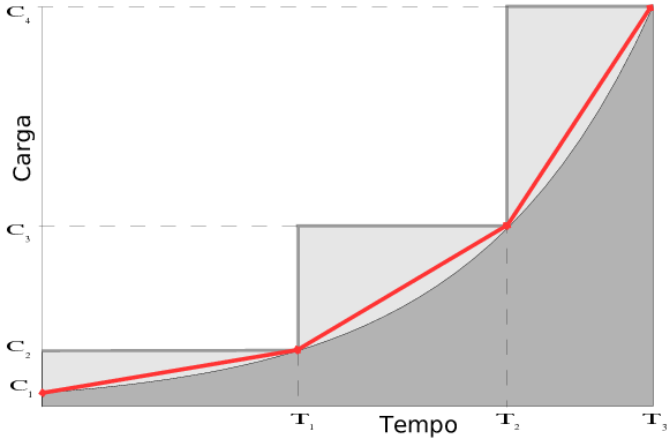


**Figura 2.2:** Evolução da carga de processamento e a utilização da solução de processamento distribuído.

Existem grandes similaridades entre as arquiteturas de computação de grande porte e a computação distribuída. Por exemplo, os dois ambientes precisam de uma estrutura física complexa, no que tange a: segurança e controles de acesso ao ambiente, de refrigeração do ambiente e a organização semelhante do espaço.

Algumas dessas similaridades são:

- Alto Poder de Processamento;
- Alta Disponibilidade;
- Suporte a Milhares de Transações e Usuários simultâneos;
- Contingenciamento de recursos;
- Administração do ambiente operacional;
- Grande Capacidade de Armazenamento.

A tabela 2.1 apresenta as principais diferenças entre as duas abordagens tecnológicas tratadas na seção 2.2.

| **Grande Porte** | ***Cluster* e *Grid*** |
|---|---|
| - Alto custo de implantação;<br>- Dependência de fornecedor único;<br>- Utilização de *hardware* específico;<br>- Alto custo de manutenção;<br>- Dificuldade de redimensionamento do ambiente;<br>- Utilização parcial da capacidade de processamento;<br>- Grande custo total de propriedade;<br>- Tecnologia estabelecida no mercado. | - Baixo custo de implantação;<br>- Independência de fornecedores – facilidade de negociação;<br>- Utilização de *hardware* comum – padrão PC;<br>- Baixo custo de manutenção;<br>- Facilidade de redimensionamento do ambiente;<br>- Maximização da capacidade de processamento;<br>- Baixo custo total de propriedade;<br>- Tecnologia inovadora. |

**Tabela 2.1:** Diferenças entre computação de grande porte e distribuída

### 2.2.4 As Gerações da Computação Distribuída

Durante os últimos 20 anos, a computação distribuída passou por um processo intenso de mudanças e revoluções. Este processo foi marcado por 5 gerações computacionais descritas a seguir:

- Primeira Geração de Computação distribuída:
  A primeira geração, também conhecida como *host-based computting*, é baseada na utilização de terminais burros que servem apenas como meio de visualização de aplicações, *softwares* e dados que encontram-se no computador central. Os recursos computacionais de processamento e armazenamento utilizados nesta geração são exclusivamente do computador que hospeda as aplicações.

- Segunda Geração de Computação distribuída:
  Na segunda geração, passam a ser utilizados computadores clientes com pequena capacidade de processamento, capazes de suportar a emulação de

terminal, entretanto, as aplicações continuam sendo armazenadas e executadas em um servidor remoto.

- Terceira Geração de Computação distribuída:
  A terceira geração é caracterizada pelo utilização do paradigma de cliente e servidor, as aplicações são desenvolvidas para serem parcialmente executadas em um computador cliente, terem uma interface com o usuário e interagirem com servidores de aplicação.

- Quarta Geração de computação distribuída:
  A quarta geração é caracterizada pela utilização de aplicações multicamadas com regras de negócio, interface de usuários e dados separadas entre ambiente de interação do usuário e várias camadas de servidores.

- Quinta geração de computação distribuída:
  A quinta geração, também conhecida como *grid computing*, é caracterizada pela existência e pela utilização por parte do cliente de recursos computacionais alocados em um "pool virtual", de forma que o cliente utiliza capacidade computacional de acordo com a sua necessidade, sem precisar ter maiores detalhes ou controle de onde estão os recursos utilizados.

Da mesma forma que em cada uma destas inovações aconteceu mudanças estruturais nas relações entre as pessoas (usuárias ou desenvolvedores de tecnologias) e os sistemas computacionais, bem como nas concepções de desenvolvimento e aplicação de sistemas informatizados, o mesmo irá ocorrer na adoção de tecnologias em Cluster e Grid. Não existe tecnologia pior ou melhor do ponto de vista global, cada tecnologia possui seu nicho de utilização e aplicação.

Caberá aos gestores dos sistemas realizar as devidas análises para verificar quais procedimentos deverão ser tomados, tanto para a migração ou desenvolvimento de aplicações, quanto para evitar gastos dispendiosos e criar um ambiente propício para a utilização destas tecnologias.

## 2.3 Por que utilizar?

O uso das tecnologias de Cluster e Grid tem aumentado nos últimos 25 anos, principalmente em aplicações de pesquisa, algumas das áreas de maior utilização

destas tecnologias são: pesquisa genética, bioinformática, física, química, engenharia, climatologia, petroquímica, pesquisa espacial e resolução de equações e métodos matemáticos.

Apesar das tecnologias de clusters serem recentes, sua utilização é crescente e já domina a lista das máquinas mais rápidas do mundo. A figura 2.3 mostra a evolução percentual das arquiteturas de sistemas de alto desempenho no mundo, onde os sistemas classificados como Clusters já são responsáveis por $72,80\%$ dos ambientes de supercomputação classificados na lista. Os dados são da organização "top500.org" [16].

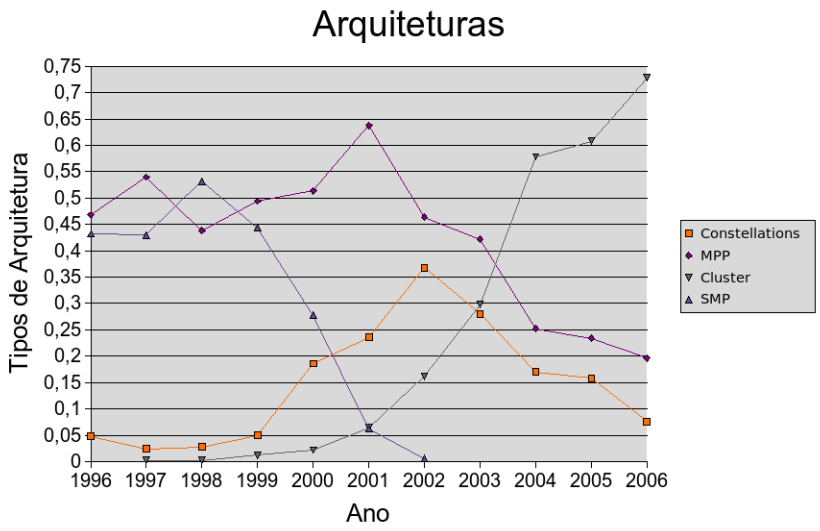


**Figura 2.3:** Evolução da utilização de Arquiteturas de alto desempenho. Fonte Top500.org

Assim como as arquiteturas de cluster vem crescendo, a utilização de software livre (GNU/Linux) também vem crescendo de forma progressiva. A figura 2.4 mostra a evolução de sua utilização nos últimos anos.

O mercado corporativo tem percebido as vantagens e possibilidades de negócios relacionadas a utilização de tecnologias baseadas em Cluster e Grid, sendo observado um crescimento da adoção destas tecnologias nesse mercado. Este fato pode ser analisado pelo investimento para desenvolvimento destas tecnologias realizado pelas maiores empresas de tecnologia do mundo, como Oracle, Google, IBM, Intel, AMD, Sun, Cisco, entre outras, somado ao aumento da demanda por arquiteturas computacionais alternativas. Uma pesquisa realizada no ano de 2004

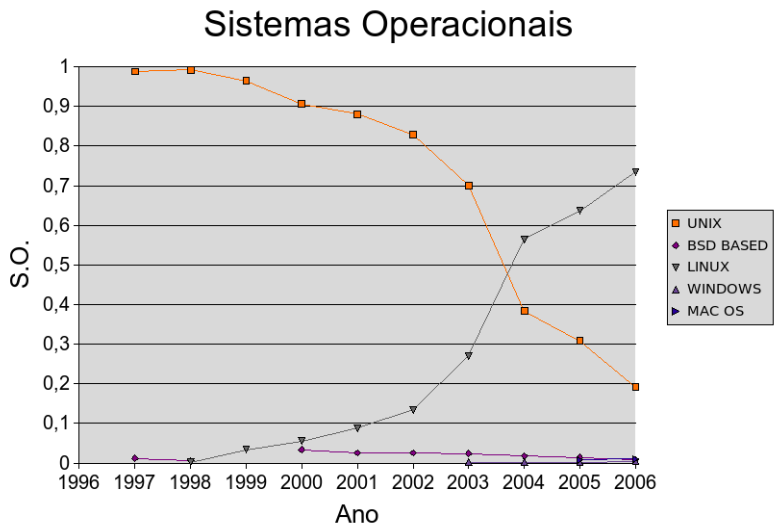


**Figura 2.4:** Evolução da utilização de S.O. Fonte Top500.org

pelo instituto Forrest Research[4] constatou que 37% das grandes empresas do mercado corporativo estão em alguma fase de adoção/desenvolvimento de projetos baseados em tecnologias de Grid Computing.

A organização "Top500" também mantém os dados sobre os segmentos corporativos que utilizam as máquinas de maior capacidade computacional do mundo, a figura ?? mostra a evolução no tempo desses segmentos de utilização.

## 2.4 Vantagens Técnicas de Utilização de Cluster e Grid

A sessão 2.1 apresenta algumas demandas e desafios computacionais e a possibilidade de utilização de tecnologias baseadas em cluster e grid para auxiliar no atendimento destas demandas. Assim observa-se que utilização destas tecnologias possui as seguintes vantagens técnicas:

- Utilização de *hardware* padrão de mercado em sistemas críticos através da transferência do gerenciamento das funções de alta disponibilidade, tolerância à falhas e balanceamento de carga do *hardware* para o software.

[4]Forrester, 2004. http://www.forrester.com/go?docid=34449

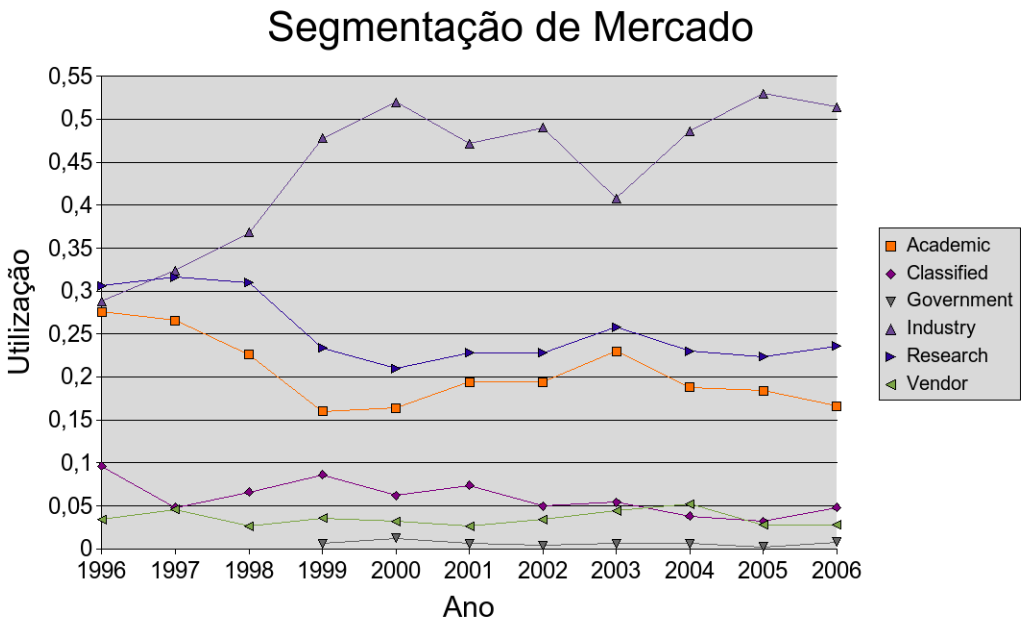


**Figura 2.5:** Evolução da utilização por segmento de mercado. Fonte Top500.org

Diminuindo a necessidade de *hardware* especializado, aumentando a concorrência entre as empresas fornecedoras e propiciando ao governo a independência tecnológica de fornecedores de *hardware*.

- Em geral, as tecnologias de Cluster e Grid possuem como base padrões abertos e interoperáveis. Facilitando a integração de sistemas baseados nestas tecnologias, em oposição a sistemas em computação de grande porte que utilizam, em sua grande parte, tecnologias proprietárias e padrões fechados.

- Disponibilidade de soluções baseadas em software livre que permitem a implementação de sistemas de cluster e grid sem a necessidade de ônus de licenças de software, além de permitir a melhoria, alteração, distribuição e compartilhamento de soluções, segurança, transparência e possibilidade de auditoria plena do sistema.

- Maior Facilidade de aumentar ou diminuir a capacidade computacional de acordo com a demanda existente, utilizando Grids e Clusters computacionais.

- Possibilidade do desenvolvimento de sistemas e serviços que utilizem os conceitos de computação sob demanda, com o objetivo de aproveitar da melhor maneira possível os sistemas e recursos computacionais existentes.

- Possibilidade de realizar o aproveitamento de "ciclos ociosos" de computadores já existentes na infra-estrutura de TI atual.

## 2.5 Cluster de Banco de Dados

A adoção de tecnologias em Cluster e Grid muitas vezes poderá impactar nas relações entre as pessoas (usuários ou desenvolvedores de tecnologias) e os sistemas computacionais, da mesma forma que qualquer outra mudança tecnológica. Os gestores, juntamente com os técnicos, deverão definir qual tecnologia será adotada, quando adotar e sobre qual metodologia/procedimento através de uma análise prévia dos possíveis benefícios obtidos com a mudança tecnológica e os riscos/impactos envolvidos.

A camada de banco de dados é uma das partes críticas da maioria dos sistemas. Uma falha, indisponibilidade ou problemas de integridade na camada de banco de dados pode ser responsável pela indisponibilidade de um sistema inteiro ou até mesmo pela perda de dados que encontravam-se armazenados. Por conta desse motivo, esta camada deve ser avaliada, desenvolvida e implementada com cuidado.

Existem diversos cenários em que as tecnologias de cluster para banco de dados podem ser utilizadas, sendo que as principais podem ser classificadas em:

- Alta Disponibilidade: Nesta categoria encontram-se as tecnologias de replicação e alta disponibilidade. Para replicação normalmente é utilizada alguma solução própria ou específica para o sistema de banco de dados utilizado.
- Paralelização de Consultas SQL: Nesta categoria encontram-se as tecnologias de paralelização de consultas SQL, cujo objetivo é aumentar a velocidade de processamento e execução de consultas sql complexas, particionando-a e distribuindo em um conjunto de servidores.
- Distribuição do banco e balanceamento das requisições: Este tipo de tecnologia é utilizada normalmente em grandes aplicações transacionais, onde é necessário aumentar a performance e disponibilidade.

# Capítulo 3

# Construindo o Cluster de Banco de Dados

Existem muitas formas de se trabalhar com banco de dados de forma a se obter maior performance e/ou para obter outras características desejáveis que não estão disponíveis facilmente nos SGDBs mais conhecidos.

Algumas das áreas de pesquisa e desenvolvimento de bancos de dados mais avançados são apresentadas a seguir.

***Design* de bancos de dados distribuídos**.

O *design* de banco de dados distribuídos responsivo é uma preocupação básica para os sistemas de informação. Em redes com grande largura da banda, latência e processamento local são os fatores mais significativos para consultas e atualização de dados no que se refere ao tempo de resposta do sistema. O processamento paralelo de consultas pode ser usado para minimizar os efeitos, particularmente se for considerado no momento do *design* do sistema de banco de dados.

O *design* de banco de dados distribuído pode ser visto como um problema de otimização que requer soluções que possuem relação com: a fragmentação de dados, a alocação de dados, e a otimização local.

**Processamento de consultas distribuído**.

Em sistemas distribuídos de grande escala, é freqüentemente difícil achar um plano ótimo para consultas distribuídas: sistemas distribuídos podem ficar muito grandes, envolvendo milhares de parques computacionais heterogêneos. Como novos bancos de dados podem ser adicionados/removidos do sistema, fica mais difícil para o "processador de consulta" manter estatísticas precisas das relações participantes armazenadas nos diferentes sites e das operações de consulta relacionadas. Também, como a carga de trabalho dos vários servidores de processamento e a velocidade de transmissão dos links entre eles flutuam durante o tempo de processamento dos trabalhos, há a necessidade de mecanismos de consulta distribuídos, que dinamicamente se adaptem a grandes ambientes distribuídos.

**Controle de Concorrência**.

O Controle de concorrência (CC) permite os usuários acessar um banco de dados distribuído mantendo a impressão que está acessando o banco de dados em um sistema dedicado.

Para isto, são necessários mecanismos de CC que intercalem a execução de um conjunto de transações debaixo de certas regras de consistência, enquanto maximizam a capacidade de execução concorrente do sistema. As duas principais categorias de mecanismos de CC são:

- **Concorrência Otimizada** - Retardo da sincronização das transações até que as operações sejam confirmadas. Conflitos são menos prováveis mas não serão conhecidos até que eles aconteçam, tornando operações de *rollback* mais caras.
- **Pessimista** - As execuções potencialmente concorrentes de transações são sincronizadas no início de seus ciclos execução. Desta forma, fica mais fácil realizar o bloqueio, entretanto os problemas devem ser conhecidos anteriormente para diminuir os custos de *rollbacks*.

**Processamento de Transações**.

Transações distribuídas provêem unidades de execução segura que permitem que várias operações sejam executadas em ”locais” diferentes e provêem a preservação da consistência dos dados de um estado de execução para outro. Um protocolo comum para assegurar cumprimento correto de uma transação distribuída é o de ”execução em duas fases” (*two-phase commit* - 2PC). Enquanto o 2PC é normalmente aplicado para transações que são executadas em um curto período de tempo, ele se torna impraticável para transações distribuídas de grande escala por causa do lock de recursos disponíveis/utilizados concorrentemente. Para isto, existem diferentes propostas, como a de dividir a execução de processos que ocupam muito tempo em sucessões menores de tarefas atômicas e a definição de mecanismos de compensação.

**Replicação de Dados**.

O desafio fundamental na replicação de dados é manter um baixo custo nas atualizações enquanto se assegura a consistência dos parques computacionais do cluster. A dificuldade do problema aumenta significativamente em ambientes de larga escala devido a latência alta e a probabilidade de quedas da rede. As duas principais categorias de técnicas de replicação de banco de dados são:

- **Replicação síncrona** - implica em um protocolo de “commit” atômico ao longo do qual assegura consistência alta às custas de transações mais lenta e a presença de *deadlocks*.
- **Replicação assíncrona** - as atualizações são processadas periodicamente por todos os nós do cluster, permitindo um processo de transação mais eficiente, ao custo de uma baixa garantia da consistência global dos dados.

**Integração de Dados**.

Conflitos diferentes surgem da representação descoordenada de conceitos ao se integrar fontes de dados autônomas e distribuídas. Exemplos destes conflitos são: tipo de dados, estrutura, conflito de nomes, atributos perdidos, e conflitos de generalização. Todos estes conflitos podem ser estaticamente endereçados entre eles, durante integração dos esquemas de dados ou dinamicamente na geração

de visão/consultas. De acordo com a arquitetura de integração, e a estratégia de manipulação de consultas, sistemas de integração de banco de dados podem ser:

- **Sistema de Multi-banco de dados** - coleção de bancos de dados integrados nos quais cada DBMS mantém controle em cima de seus bancos de dados locais, mas coopera com a federação aceitando operações globais.
- **Sistema de mediador** - bancos de dados são integrados, através de um componente de mediação que executa tradução de dados em um modelo de dados canônico comum.
- **Sistemas de Metadados** - consultas são formuladas dinamicamente em cada banco de dados pela interação com um dicionário de metadados global.

Existem vários tipos de "clusters" de banco de dados:

- Banco de dados distribuídos: Nesse tipo de banco de dados os dados são distribuídos em um conjunto de servidores. O acesso ao banco de dados é direcionado ao servidor onde se encontra o dado.
- Banco de dados em alta disponibilidade: Dois ou mais servidores em cluster de alta disponibilidade (MASTER/SLAVE), onde um servidor MASTER é responsável pelo serviço e os servidores SLAVE ficam aguardando a falha do MASTER para assumirem o serviço.
- Banco de dados em alta disponibilidade e distribuídos: É um cluster de banco de dados onde as duas tecnologias anteriores estão presentes, criando um banco de dados escalável e tolerante a falhas.

## 3.1 Banco de Dados Distribuídos

### Definições

1. Segundo C. J. Date [7],
   Um sistema de banco da dados distribuídos consiste em uma coleção de locais, conectados por alguma rede de comunicação e que:

   a. Cada um dos locais é um sistema de banco de dados completo com seus próprios direitos, mas

   b. Os bancos de dados locais trabalham em conjunto para que os usuários que acessam os dados de qualquer outro local da rede possa acessar os dados de forma transparente.

2. Um banco de dados distribuído é um banco de dados que está sob o controle de um sistema de administração de banco de dados central, no qual dispositivos de armazenamento (*storage*) são anexados a um computador. O armazenamento pode ser em vários computadores localizados no mesmo local físico, ou dispersos em uma rede de computadores interconectados.

O banco de dados, pode ser distribuído fisicamente através de múltiplos locais. Um banco de dados distribuído é dividido em várias partes ou fragmentos. Essas partes ou fragmentos do banco de dados distribuído podem ser replicadas, para por exemplo criar ambientes de redundância, RAID, ou mesmo copias para *Data Warehouse*.

Além de replicação e fragmentação em banco de dados distribuídos, existem várias outras tecnologias para *design* de banco de dados distribuídos. Por exemplo, autonomia local, tecnologias de bancos de dados distribuídos síncronos e assíncronos. A implementação destas tecnologias podem e definitivamente dependem das necessidades das áreas de negócios e de a sensibilidade e confidenciabilidade dos dados a serem armazenados no banco de dados.

## 3.2 Replicação de Banco de Dados

Um banco de dados replicado é um sistema de bancos de dados com cópias distribuídas em uma ou mais máquinas. Este tipo de sistema oferece alta disponibilidade e tolerância a falhas, já que não há apenas uma única cópia dos dados, e melhoria de desempenho, posto que as requisições não onerarão apenas uma fonte de dados.

Para se implementar a replicação em bancos de dados podem ser usados dois métodos levando-se em consideração a maneira como é feita a propagação de uma atualização.

Uma primeira aproximação é chamada replicação síncrona, ou seja uma atualização (modificação fruto de uma operação de UPDATE, INSERT ou DELETE, por exemplo) só é consumada se for realizada em todos os nós que compõem o sistema. Isto significa que o cliente terá de esperar até que todas as instâncias do banco de dados sejam modificadas para receber uma confirmação, garantindo a integridade da informação entre os nós.

A outra aproximação é realizar a atualização de maneira assíncrona, ou seja as modificações são difundidas entre os nós após ser efetivada em um servidor e a resposta ser enviada ao cliente. O tempo de resposta, se comparado ao método anterior é menor, porém isto pode gerar inconsistências entre as replicas.

Uma maneira de se contornar estes problemas é restringir as atualizações a um único nó, chamado cópia primária ou MASTER, o que impede que atualizações em um mesmo objeto sejam feitas em duas máquinas diferentes. Todas as operações de modificação no banco serão enviadas para esta máquina que cuidará de propagar as modificações. A contrapartida para este modelo é permitir atualizações em qualquer banco que componha o sistema, não introduzindo uma réplica privilegiada mas requerendo um sistema que resolva conflitos de possíveis inconsistências.

O uso de *middleware* (*software* interface entre os clientes e o sistemas de bancos de dados) se tornou um atrativo, já que permite a construção de sistemas replicados sem a necessidade de modificação do sistema de gerenciamento de banco de dados, nem no banco de dados em si. Em sistemas desta natureza as requisições são enviadas ao *middleware* que se encarrega de propagá-las às réplicas de maneira a prover controle de replicação e balanceamento de carga.

# Capítulo 4

# Cluster PostgreSQL

Nos últimos anos, com o crescimento do PostgreSQL no Mercado Corporativo, a necessidade de possuir algum tipo de ferramenta de replicação que permitisse a utilização em sistemas de grande porte se tornou crítica. Já havia no Mercado ferramentas de replicação como o Slony-I 4.1.1, que permitiam a escalabilidade para sistemas com milhares de consultas simultâneas, mas a performance e a dificuldade de manutenção tornavam difícil a adoção em larga escala.

Um grande passo foi dado no lançamento da versão 8.2 [11], quando foi adicionada a configuração de servidores *Warm Stand By*. A ferramenta possibilitava configurar o arquivamento de transações do PostgreSQL para subir automaticamente em outro servidor, trazendo principalmente:

- Facilidade para recuperação do Banco de Dados em caso de desastre;
- Administração simplificada, facilitando a implementação.

Contudo, o passo definitivo foi dado no lançamento da versão 9.0 [12] com a adição do mecanismo de *Hot Stand By*. Passou a ser possível, utilizando apenas configuração do banco de dados, montar um cluster do tipo Master-Slave em poucos minutos, de maneira relativamente segura e confiável.

Alguns dos principais mecanismos descritos serão abordados no texto a seguir, com descrição da arquitetura e exemplos de utilização.

## 4.1 Alta Disponibilidade[10]

Os servidores de banco de dados podem trabalhar em conjunto para permitir que um outro servidor assuma rapidamente caso ocorra uma falha no servidor primário (alta disponibilidade), ou para permitir que muitos computadores forneçam os mesmos dados (balanceamento de carga). Normalmente é possíve que vários servidores de banco de dados trabalhem em conjunto perfeitamente; servidores web que servem páginas estáticas podem ser combinados facilmente simplesmente balanceando as requisições entre diferentes máquinas. De fato, caso as operações do banco de dados fossem somente leitura o balanceamento poderia ser feito de maneira natural. Contudo, a maor parte dos servidores de banco de dados precisam atender requisições de leitura e escrita com muita frequência, tornando a combinação significativamente mais complexa. Isso acontece porque é necessário propagar os dados escritos em um dos servidores para todos os outros que trabalham em modo somente leitura, de modo que os resultados retornados sejam consistentes.

O problema de sincronização é a principal dificuldade para construir um cluster de banco de dados. Como não há solução simples para eliminar o impacto da operação em todos os casos de uso, há múltiplas soluções desenvolvidas. Cada uma ataca o problema de forma diferente, minimizando o impacto para a carga proposta.

Algumas soluções permitem que somente um dos servidores modifiquem os dados, chamados como servidores de escrita, servidores *Master* ou primários. Servidores que registram as mudanças no *Master* são chamados de *Standby* ou *Slave*. Um servidor *Standby* que não aceita conexões até que seja promovido a *Master* é chamado de *Warm Stand By*, e um que permite a conexão em modo somente leitura é chamado de *Hot Stand By*.

Há ainda as soluções síncronas, significando que uma transação que modifique os dados não é considerada finalizada até que seja propagada a todos os outros servidores. Assim, qualquer falha em um dos servidores não representa perda de dados, além de garantir que qualquer servidor do cluster vai retornar resultados consistentes a qualquer momento. Ao contrário das soluções assíncronas, que permitem algum *delay* entre o instante do commit e a sua propagação aos outros servidores, tornando possível que uma consulta em algum dos servidores re-

torne um resultado ligeiramente atrasado. A comunicação assíncrona é utilizada quando a comunicação síncrona seria lenta demais.

Em qualquer uma das escolhas a performance é o fator mais importante.

### 4.1.1 Replicação Master-Slave

Na replicação Master-Slave há um servidor definido como *Master* ou Mestre, único do cluster onde é possível inserir os dados. Os outros servidores se comportam como *Slave* ou Escravos, permitindo a consulta de informações. O ajuste é útil em sistemas com muita demanda em operações de consulta, sem o risco de envio de informações inconsistentes em qualquer um dos servidores.

#### Hot Stand By

*Hot Stand By* é o termo utilizado para descrever a possibilidade de conectar ao banco para executar operações somente leitura enquanto o servidor está em modo *archive recovery* ou *standby*. Pode ser utilizado tanto para replicar os dados quanto para restaurar um backup do banco para um estado desejado com muita precisão. O termo *Hot Standby* também se refere à possibilidade do servidor de mudar de um estado de recuperação à operação normal enquando os usuários continuam executando consultas e/ou mantém suas conexões abertas.

Os dados em *standby* levam algum tempo para chegar do servidor primário, de forma que há um *delay* mensurável entre ambos. Executar a consulta em ambos os servidores quase ao mesmo tempo pode retornar resultados diferentes. Dizemos então que os dados no servidor *standby* estão eventualmente consistentes com o primário. Uma vez que o registro do commit da transação é executado novamente no servidor *standby*, as mudanças causadas pela transação serão visíveis em qualquer *snapshot* do *standby*.

O servidor *Hot Stand By* permite a execução dos seguintes comandos:

- Consulta - SELECT, COPY TO

- Cursores - DECLARE, FETCH, CLOSE
- Parâmetros - SHOW, SET, RESET
- Controles de gerenciamento de transações
  - BEGIN, END, ABORT, START TRANSACTION
  - SAVEPOINT, RELEASE, ROLLBACK TO SAVEPOINT
  - EXCEPTION blocks and other internal subtransactions
- LOCK TABLE, somente nos seguintes modos: ACCESS SHARE, ROW SHARE or ROW EXCLUSIVE.
- Plans e resources - PREPARE, EXECUTE, DEALLOCATE, DISCARD
- Plugins e extensões - LOAD

Transações iniciadas no *Hot Stand By* não receberão um ID e não podem escrever no WAL (*Write-Ahead Log*). Assim, as seguinte ações produzirão erros:

- Data Manipulation Language (DML) - INSERT, UPDATE, DELETE, COPY FROM, TRUNCATE. Note que não há ações permitidas que resultarão em um gatilho (TRIGGER) sendo executado em estado de recuperação (*recovery*). A restrição se aplica até mesmo a tabelas temporárias porque as linhas da tabela não podem ser lidas ou escritas sem atribuir um ID de transação, o que não é possível atualmente num ambiente de *Hot Stand By*.
- Data Definition Language (DDL) - CREATE, DROP, ALTER, COMMENT. A restrição se aplica até mesmo a tabelas temporárias, pois para permitir tais operações seria necessária atualizar as tabelas de catálogo do sistema.
- SELECT ... FOR SHARE — UPDATE, porque não é possível atribuir travas de linha sem atualizar os respectivos arquivos de dados.
- Regras em execução de SELECT que geram comandos DML.
- LOCK que requisita um modo maior que ROW EXCLUSIVE MODE.
- LOCK na forma padrão resumida, já que ela solicita ACCESS EXCLUSIVE MODE.

- Comandos para gerência de transações que ajustam o estado explicitamente a não somente leitura:
  - BEGIN READ WRITE, START TRANSACTION READ WRITE
  - SET TRANSACTION READ WRITE, SET SESSION CHARACTERISTICS AS TRANSACTION READ WRITE
  - SET transaction_read_only = off
- Comandos para commit em duas fases: PREPARE TRANSACTION, COMMIT PREPARED, ROLLBACK PREPARED porque até mesmo as tranações somente leitura precisam escrever no WAL na fas de preparação (primeira fase do commit de duas fases).
- Atualização de sequências - nextval(), setval()
- LISTEN, UNLISTEN, NOTIFY

No modo de operação normal é permitido a transações somente leitura utilizar os comandos LISTEN, UNLISTEN e NOTIFY, de forma que as sessões *Hot Stand By* operam sob restrições maiores do que sessões somente leitura. É possível que algumas dessas restrições sejam suavizadas em versões futuras do PostgreSQL.

### Slony

Slony[1] é um sistema de replicação " master" para "muitos slaves" que suporta cascateamento e transformação de "slaves" para " master" , possui capacidade para replicar grandes bancos de dados em até doze servidores *slave*. O Slony foi criado para atender a *data centers* e *sítios de backup* onde todos os nós devem estar disponíveis a qualquer momento.

Há alguns modelos distintos de replicação de bancos de dados, é difícil que um modelo se adapte à todas as necessidades. O modelo implementado no Slony-I é chamado de replicação assíncrona usando *triggers* para coletar as atualizações, onde uma origem comum é replicada em múltiplas cópias incluindo cópias cascateadas.

Em algumas situações, Slony não é aconselhável:

. Sistemas com conectividade ruim

. Replicação em nós com certa imprevisibilidade na conexão.

. Sistemas cuja configuração mude de maneira aleatória.

. Sistemas onde *schemas* de bancos de dados podem ser mudados arbitrariamente.

Slony é um sistema de replicação independente de versão do PostgreSQL, permitindo ser iniciado ou parado sem a necessidade de ciclos de dump/reload.

Dentre as coisas que Slony não é:

. Não é um sistema de gerenciamento de rede.

. Não possui nenhuma funcionalidade para detectar falha de nó, nem muda um nó *master* para outro, embora isso possa ser conseguido em conjunção com outras ferramentas.

. Não é um sistema de replicação *multi-master*, mas há planos para transformá-lo em *multi-master* na versão 2, muito embora seja um projeto separado e ainda encontre-se em desenvolvimento.

**Modelos de replicação** Há alguns modelos distintos de replicação de bancos de dados, é difícil que um modelo se adapte à todas as necessidades. O modelo implementado no Slony-I é chamado de replicação assíncrona usando *triggers* para coletar as atualizações, onde uma origem comum é replicada em múltiplas cópias incluindo cópias cascateadas.

Slony não propaga mudanças em *schemas*, nem replica grandes objetos. Slony coleta as atualizações com *triggers*, sendo que apenas tabelas e seqüencias são replicadas.

### 4.1.2 Replicação Multi-Master

Na replicação Muti-Master as transações são enviadas a todos os servidores do Cluster, mantendo a informação sempre atualizada em cada um dos nós. A consequência é a perda de performance na inseração dos dados, uma vez que é necessário esperar a propagação dos dados terminar antes de disponibilizá-los para consulta.

#### Bucardo

O Bucardo é um *daemon* escrito em Perl que espera uma operação de NOTIFY e age quando ela acontece, conectando-se a outros bancos de dados e propagando neles as informações. Toda a informação necessária ao *daemon* é armazenada em um banco de dados interno, incluindo a lista de todos os servidores envolvidos na replicação e como acessá-los, além da lista de tabelas a serem replicadas e como devem ser replicadas.

A replicação dentro do Bucardo segue o seguinte fluxo:

1. Uma mudança é feita em alguma tabela, e a alteração é gravada na tabela bucardo_delta;
2. Uma notificação é enviada ao *daemon* principal do Bucardo, informando que a tabela foi alterada;
3. O *daemon* envia uma notificação para o controlador sobre aquela sincronia e volta a escutar por mudanças;
4. O controlador cria um *kid* para cuidar da replicação ou envia um sinal para algum existente;
5. O *kid* inicia uma nova transação e desabilita os TRIGGERS e Rules para a tabela em questão.
6. Ele recupera uma lista de quais linhas foram alteradas desde a última replicação e compara as duas para saber o que deve ser feito;

7. Se houver um conflito ele será tratado por um mecanismo de gerência de conflitos padrão ou por um customizado, ajustado por tabela;

8. TRIGGERS e Rules são habilitados novamente e a transação recebe um COMMIT;

9. Se a transação falhar são executados os tratamentos de exceção;

10. O filho sinaliza ao controlador que foi finalizado.

## 4.2 Balanceamento de Carga

O Balanceamento de carga divide as execuções entre os vários servidores do cluster.

## 4.3 PGpool

PGpool[4] é um *middleware* para PostgreSQL, distribuído sob licença BSD, que se situa entre os clientes e os servidores de banco de dados provendo alta disponibilidade, replicação e balanceamento de carga. Além destas características em comum com outros sistemas similares, PGpool adicionalmente salva as conexões com os servidores PostgreSQL por ele coordenados (PGpool atualmente trabalha apenas com dois servidores PostgreSQL), reutilizando-as quando uma nova conexão com mesmas propriedades (nome de usuário, banco de dados, protocolo) chega, reduzindo sobrecarga de conexão e aumentando a taxa de transferência de todo o sistema.

Como sistema de replicação de dados, PGpool permite *backup* em tempo real de bancos de dados, enviando as mesmas declarações SQL para ambos os servidores, podendo ser considerado um sistema de replicação síncrona. Apesar disto algumas instruções SQL são dependentes do servidor no qual são executadas como funções aleatórias, OID, XID e *timestamp*, não sendo replicadas com o mesmo valor para ambos servidores.

Se algum problema torna um dos servidores PostgreSQL indisponível, o PGpool tenta continuar o serviço com o servidor ainda ativo, este modo é chamado " modo degenerado". Inconvenientemente, o PGpool não oferece nenhum método para voltar um servidor com problemas de novo no cluster, sendo necessário que os bancos sejam sincronizados novamente. A melhor maneira é desativar o servidor ativo, sincronizar os arquivos do Postgresql via *rsync*, por exemplo, reiniciar os bancos e o PGpool.

O PGpool envia uma query para o " master" que envia para o " slave" antes do " master" completar a query. Isto pode aumentar a performance (desempenho) mas acrescenta o risco de "deadlock". Para balancear performance e risco, PGpool pode operar de duas formas:

1) modo "*restrict*": Neste modo, PGpool espera a conclusão da query no nó *master* para depois envia-la para o secundário. Este é o modo de operação padrão e mais seguro do PGpool.

2) palavra chave /*STRICT*/: Visando performance, o modo restrict pode ser desabilitado através do ajuste da diretiva " PGpool_restrict"na configuração do PGpool. Para inibir "deadlocks", deve-se inserir a /*STRICT*/ no inicio de cada query passível de produzir "deadlock", como por exemplo:

```
/*STRICT*/ LOCK TABLE t1;
```

Caso algum "deadlock" ocorra, não sendo detectado pelo próprio PostgreSQL, PGpool abortará a sessão se um dos nós não responderem por um certo intervalo de tempo configurável.

Para propósitos de balanceamento de carga (configurável pelo parâmetro "load_balance_mode" no arquivo de configuração), as consultas "SELECT"são distribuídas entre o nó *master* e o *slave* de maneira aleatória, para aumentar performance.

É importante notar que mesmo instruções "SELECT"podem introduzir alterações em bancos chamando funções que podem modificá-los. Em casos como este não se deve usar o balancemanto de carga, sendo necessário se usar espaços em branco antes da *query* para que ela não seja distribuída.

Eis uma lista de vantagens no uso do PGpool:

. Não é necessário modificações na aplicação.

. Qualquer linguagem pode ser usada.

. O número de conexões com o PostgreSQL pode ser limitado.

. Tolerância a falhas. Caso ocorra falha em um servidor PostgreSQL, o outro assume automaticamente.

. Replicação.

. Balanceamento de carga, consultas somente leitura podem ser distribuídas entre os servidores.

Desvantagens:

. Sobrecarga. Todos os acessos ao PostgreSQL passam pelo PGpool o que pode reduzir um pouco a performance (de 1 a 15%, de acordo com os desenvolvedores, em testes feitos com pgbench).

. Nem todos protocolos da *libpq* são suportados:

  1 Nenhum método de autenticação exceto "trust"e "clear text password"em modo de replicação.

  2 Em modo não replicado só são aceitos "trust", "clear text password", "crypt"e "md5".

  3 Controle de acesso via pg_hba.conf não é suportado.

. Sem controle de acesso, qualquer um pode acessar PGpool, o que pode ser impedido via *iptables*, por exemplo.

### 4.3.1 PGpool-II

PGpool-II é um projeto que herdou as características do PGpool, mas que suporta múltiplas instâncias do PostgreSQL (128 nós, expansível via recompilação

do código-fonte) e processamento paralelo nos múltiplos nós, o que aumenta muito a performance.

A arquitetura do PGpool-II consiste, em um "System DB" que processa as informações administrativas e operações agregadas, e múltiplos nós onde são armazenados os dados. Novos dados serão incluídos/alterados no nó DB baseado em regras de particionamento pré-definido. Uma regra de particionamento é definida por funções SQL e são armazenadas no "System DB", que é outro servidor PosgreSQL. A arquitetura do PGpool-II é ilustrada na figura 4.1 que se segue.

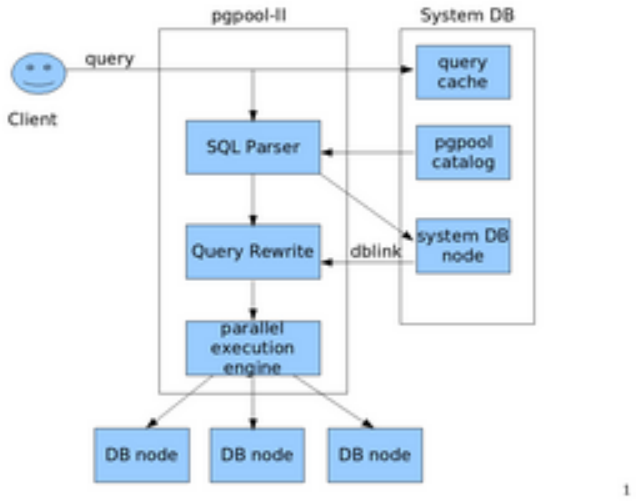


**Figura 4.1:** Arquitetura PG-pool

# Parte I

# Apêndices

# Apêndice A

# Licença CC-GNU GPL


**Figura A.1:** Creative Commons

Licença Pública Geral do GNU (GPL) [General Public License]

Versão 2[1], Junho de 1991 Direitos Autorais Reservados (c) 1989, 1991 Free *Software* Foundation, Inc. 59 Temple Place, Suite [conjunto] 330, Boston, MA [Massachusetts] 02111-1307 USA [Estados Unidos da América]

É permitido a qualquer pessoa copiar e distribuir cópias sem alterações deste documento de licença, sendo vedada, entretanto, qualquer modificação.

### Introdução

As licenças da maioria dos *softwares* são elaboradas para suprimir sua liberdade de compartilhá-los e modificá-los. A Licença Pública Geral do GNU, ao contrário,

[1] Disponível em http://creativecommons.org/licenses/GPL/2.0/legalcode.pt.

visa garantir sua liberdade de compartilhar e modificar *softwares* livres para assegurar que o *software* seja livre para todos os seus usuários. Esta Licença Pública Geral é aplicável à maioria dos *softwares* da Free *Software* Foundation [Fundação do *Software* livre] e a qualquer outro programa cujos autores se comprometerem a usá-la. (Em vez dela, alguns outros *softwares* da Free *Software* Foundation são cobertos pela Licença Pública Geral de Biblioteca do GNU). Você também poderá aplicá-la aos seus programas.

Quando falamos de *Software* Livre, estamos nos referindo à liberdade, não ao preço. Nossas Licenças Públicas Gerais visam garantir que você tenha a liberdade de distribuir cópias de *Software* Livre (e cobrar por isso se desejar), que receba código-fonte ou possa obtê-lo se desejar, que possa modificá-lo ou usar partes dele em novos programas livres; finalmente, que você tenha ciência de que pode fazer tudo isso.

Para proteger seus direitos, necessitamos fazer restrições que proíbem que alguém negue esses direitos a você ou que solicite que você renuncie a eles. Essas restrições se traduzem em determinadas responsabilidades que você deverá assumir, se for distribuir cópias do *software* ou modificá-lo.

Por exemplo, se você distribuir cópias de algum desses programas, tanto gratuitamente como mediante uma taxa, você terá de conceder aos receptores todos os direitos que você possui. Você terá de garantir que, também eles, recebam ou possam obter o código-fonte. E você terá a obrigação de exibir a eles esses termos, para que eles conheçam seus direitos.

Protegemos seus direitos através de dois passos: (1) estabelecendo direitos autorais sobre o *software* e (2) concedendo a você esta licença, que dá permissão legal para copiar, distribuir e/ou modificar o software.

Além disso, para a proteção de cada autor e a nossa, queremos ter certeza de que todos entendam que não há nenhuma garantia para este *Software* Livre. Se o *software* for modificado por alguém e passado adiante, queremos que seus receptores saibam que o que receberam não é o original, de forma que quaisquer problemas introduzidos por terceiros não afetem as reputações dos autores originais.

Finalmente, qualquer programa livre é constantemente ameaçado por patentes de software. Queremos evitar o risco de que redistribuidores de um programa livre

obtenham individualmente licenças sob uma patente, tornando o programa, com efeito, proprietário. Para impedir isso, deixamos claro que qualquer patente deve ser licenciada para o uso livre por parte de qualquer pessoa ou, então, simplesmente não deve ser licenciada.

Os exatos termos e condições para cópia, distribuição e modificação seguem abaixo. TERMOS E CONDIÇÕES PARA CÓPIA, DISTRIBUIÇÃO E MODIFICAÇÃO.

1. Esta Licença se aplica a qualquer programa ou outra obra que contenha um aviso inserido pelo respectivo titular dos direitos autorais, informando que a referida obra pode ser distribuída em conformidade com os termos desta Licença Pública Geral. O termo "Programa", utilizado abaixo, refere-se a qualquer programa ou obra, e o termo "obras baseadas no Programa" significa tanto o Programa, como qualquer obra derivada nos termos da legislação de direitos autorais: isto é, uma obra contendo o Programa ou uma parte dele, tanto de forma idêntica como com modificações, e/ou traduzida para outra linguagem. (Doravante, o termo "modificação" inclui também, sem reservas, a tradução). Cada licenciado, doravante, será denominado "você".

   Outras atividades que não a cópia, distribuição e modificação, não são cobertas por esta Licença; elas estão fora de seu escopo. O ato de executar o Programa não tem restrições e o resultado gerado a partir do Programa encontra-se coberto somente se seu conteúdo constituir uma obra baseada no Programa (independente de ter sido produzida pela execução do Programa). Na verdade, isto dependerá daquilo que o Programa faz.

2. Você poderá fazer cópias idênticas do código-fonte do Programa ao recebê-lo e distribui-las, em qualquer mídia ou meio, desde que publique, de forma ostensiva e adequada, em cada cópia, um aviso de direitos autorais (ou copyright) apropriado e uma notificação sobre a exoneração de garantia; mantenha intactas as informações, avisos ou notificações referentes a esta Licença e à ausência de qualquer garantia; e forneça a quaisquer outros receptores do Programa uma cópia desta Licença junto com o Programa.

   Você poderá cobrar um valor pelo ato físico de transferir uma cópia, e você pode oferecer, se quiser, a proteção de uma garantia em troca de um valor.

3. Você poderá modificar sua cópia ou cópias do Programa ou qualquer parte dele, formando, dessa forma, uma obra baseada no Programa, bem como copiar e distribuir essas modificações ou obra, de acordo com os termos da Cláusula 1 acima, desde que você também atenda a todas as seguintes condições:

   a. Você deve fazer com que os arquivos modificados contenham avisos, em destaque, informando que você modificou os arquivos, bem como a data de qualquer modificação.

   b. Você deve fazer com que qualquer obra que você distribuir ou publicar, que no todo ou em parte contenha o Programa ou seja dele derivada, ou derivada de qualquer parte dele, seja licenciada como um todo sem qualquer custo para todos terceiros nos termos desta licença.

   c. Se o programa modificado normalmente lê comandos interativamente quando executado, você deverá fazer com que ele, ao começar a ser executado para esse uso interativo em sua forma mais simples, imprima ou exiba um aviso incluindo o aviso de direitos autorais (ou copyright) apropriado, além de uma notificação de que não há garantia (ou, então, informando que você oferece garantia) e informando que os usuários poderão redistribuir o programa de acordo com essas condições, esclarecendo ao usuário como visualizar uma cópia desta Licença. (Exceção: se o Programa em si for interativo mas não imprimir normalmente avisos como esses, não é obrigatório que a sua obra baseada no Programa imprima um aviso).

      Essas exigências se aplicam à obra modificada como um todo. Se partes identificáveis dessa obra não forem derivadas do Programa e puderem ser consideradas razoavelmente como obras independentes e separadas por si próprias, nesse caso, esta Licença e seus termos não se aplicarão a essas partes quando você distribui-las como obras separadas. Todavia, quando você distribui-las como parte de um todo que constitui uma obra baseada no Programa, a distribuição deste todo terá de ser realizada em conformidade com esta Licença, cujas permissões para outros licenciados se estenderão à obra por completo e, conseqüentemente, a toda e qualquer parte, independentemente de quem a escreveu.

      Portanto, esta cláusula não tem a intenção de afirmar direitos ou contestar os seus direitos sobre uma obra escrita inteiramente por você;

a intenção é, antes, de exercer o direito de controlar a distribuição de obras derivadas ou obras coletivas baseadas no Programa.

Além do mais, a simples agregação de outra obra que não seja baseada no Programa a ele (ou a uma obra baseada no Programa) em um volume de mídia ou meio de armazenamento ou distribuição, não inclui esta outra obra no âmbito desta Licença.

4. Você poderá copiar e distribuir o Programa (ou uma obra baseada nele, de acordo com a Cláusula 2) em código-objeto ou formato executável de acordo com os termos das Cláusulas 1 e 2 acima, desde que você também tome uma das providências seguintes:

   a. Incluir o código-fonte correspondente completo, passível de leitura pela máquina, o qual terá de ser distribuído de acordo com as Cláusulas 1 e 2 acima, em um meio ou mídia habitualmente usado para intercâmbio de software; ou,

   b. Incluir uma oferta por escrito, válida por pelo menos três anos, para fornecer a qualquer terceiro, por um custo que não seja superior ao seu custo de fisicamente realizar a distribuição da fonte, uma cópia completa passível de leitura pela máquina, do código-fonte correspondente, a ser distribuído de acordo com as Cláusulas 1 e 2 acima, em um meio ou mídia habitualmente usado para intercâmbio de software; ou,

   c. Incluir as informações recebidas por você, quanto à oferta para distribuir o código-fonte correspondente. (Esta alternativa é permitida somente para distribuição não-comercial e apenas se você tiver recebido o programa em código-objeto ou formato executável com essa oferta, de acordo com a letra b, acima).

      O código-fonte de uma obra significa o formato preferencial da obra para que sejam feitas modificações na mesma. Para uma obra executável, o código-fonte completo significa o código-fonte inteiro de todos os módulos que ela contiver, mais quaisquer arquivos de definição de interface associados, além dos *scripts* usados para controlar a compilação e instalação do executável. Entretanto, como uma exceção especial, o código-fonte distribuído não precisa incluir nada que não seja normalmente distribuído (tanto no formato fonte como no binário) com os componentes principais (compilador, kernel e assim por diante) do sistema operacional no qual o executável é executado, a menos que este componente em si acompanhe o executável.

Se a distribuição do executável ou código-objeto for feita mediante a permissão de acesso para copiar, a partir de um local designado, então, a permissão de acesso equivalente para copiar o código-fonte a partir do mesmo local será considerada como distribuição do código-fonte, mesmo que os terceiros não sejam levados a copiar a fonte junto com o código-objeto.

5. Você não poderá copiar, modificar, sublicenciar ou distribuir o Programa, exceto conforme expressamente estabelecido nesta Licença. Qualquer tentativa de, de outro modo, copiar, modificar, sublicenciar ou distribuir o Programa será inválida, e automaticamente rescindirá seus direitos sob esta Licença. Entretanto, terceiros que tiverem recebido cópias ou direitos de você de acordo esta Licença não terão suas licenças rescindidas, enquanto estes terceiros mantiverem o seu pleno cumprimento.

6. Você não é obrigado a aceitar esta Licença, uma vez que você não a assinou. Porém, nada mais concede a você permissão para modificar ou distribuir o Programa ou respectivas obras derivativas. Tais atos são proibidos por lei se você não aceitar esta Licença. Conseqüentemente, ao modificar ou distribuir o Programa (ou qualquer obra baseada no Programa), você estará manifestando sua aceitação desta Licença para fazê-lo, bem como de todos os seus termos e condições para copiar, distribuir ou modificar o Programa ou obras nele baseadas.

7. Cada vez que você redistribuir o Programa (ou obra baseada no Programa), o receptor receberá, automaticamente, uma licença do licenciante original, para copiar, distribuir ou modificar o Programa, sujeito a estes termos e condições. Você não poderá impor quaisquer restrições adicionais ao exercício, pelos receptores, dos direitos concedidos por este instrumento. Você não tem responsabilidade de promover o cumprimento por parte de terceiros desta licença.

8. Se, como resultado de uma sentença judicial ou alegação de violação de patente, ou por qualquer outro motivo (não restrito às questões de patentes), forem impostas a você condições (tanto através de mandado judicial, contrato ou qualquer outra forma) que contradigam as condições desta Licença, você não estará desobrigado quanto às condições desta Licença. Se você não puder atuar como distribuidor de modo a satisfazer simultaneamente suas obrigações sob esta licença e quaisquer outras obrigações

pertinentes, então, como conseqüência, você não poderá distribuir o Programa de nenhuma forma. Por exemplo, se uma licença sob uma patente não permite a redistribuição por parte de todos aqueles que tiverem recebido cópias, direta ou indiretamente de você, sem o pagamento de royalties, então, a única forma de cumprir tanto com esta exigência quanto com esta licença será deixar de distribuir, por completo, o Programa.

Se qualquer parte desta Cláusula for considerada inválida ou não executável, sob qualquer circunstância específica, o restante da cláusula deverá continuar a ser aplicado e a cláusula, como um todo, deverá ser aplicada em outras circunstâncias.

Esta cláusula não tem a finalidade de induzir você a infringir quaisquer patentes ou direitos de propriedade, nem de contestar a validade de quaisquer reivindicações deste tipo; a única finalidade desta cláusula é proteger a integridade do sistema de distribuição do *Software* Livre, o qual é implementado mediante práticas de licenças públicas. Muitas pessoas têm feito generosas contribuições à ampla gama de *software* distribuído através desse sistema, confiando na aplicação consistente deste sistema; cabe ao autor/-doador decidir se deseja distribuir *software* através de qualquer outro sistema e um licenciado não pode impor esta escolha.

Esta cláusula visa deixar absolutamente claro o que se acredita ser uma conseqüência do restante desta Licença.

9. Se a distribuição e/ou uso do Programa for restrito em determinados países, tanto por patentes ou por interfaces protegidas por direito autoral, o titular original dos direitos autorais que colocar o Programa sob esta Licença poderá acrescentar uma limitação geográfica de distribuição explícita excluindo esses países, de modo que a distribuição seja permitida somente nos países ou entre os países que não foram excluídos dessa forma. Nesse caso, esta Licença passa a incorporar a limitação como se esta tivesse sido escrita no corpo desta Licença.

10. A Free *Software* Foundation poderá de tempos em tempos publicar novas versões e/ou versões revisadas da Licença Pública Geral. Essas novas versões serão semelhantes em espírito à presente versão, mas podem diferenciar-se, porém, em detalhe, para tratar de novos problemas ou preocupações.

    Cada versão recebe um número de versão distinto. Se o Programa especificar um número de versão desta Licença que se aplique a ela e a "qualquer

versão posterior", você terá a opção de seguir os termos e condições tanto daquela versão como de qualquer versão posterior publicada pela Free *Software* Foundation. Se o Programa não especificar um número de versão desta Licença, você poderá escolher qualquer versão já publicada pela Free *Software* Foundation.

11. Se você desejar incorporar partes do Programa em outros programas livres cujas condições de distribuição sejam diferentes, escreva ao autor solicitando a respectiva permissão. Para *software* cujos direitos autorais sejam da Free *Software* Foundation, escreva para ela; algumas vezes, abrimos exceções para isso. Nossa decisão será guiada pelos dois objetivos de preservar a condição livre de todos os derivados de nosso *Software* Livre e de promover o compartilhamento e reutilização de software, de modo geral.

**EXCLUSÃO DE GARANTIA**

11. COMO O PROGRAMA É LICENCIADO SEM CUSTO, NÃO HÁ NENHUMA GARANTIA PARA O PROGRAMA, NO LIMITE PERMITIDO PELA LEI APLICÁVEL. EXCETO QUANDO DE OUTRA FORMA ESTABELECIDO POR ESCRITO, OS TITULARES DOS DIREITOS AUTORAIS E/OU OUTRAS PARTES, FORNECEM O PROGRAMA "NO ESTADO EM QUE SE ENCONTRA", SEM NENHUMA GARANTIA DE QUALQUER TIPO, TANTO EXPRESSA COMO IMPLÍCITA, INCLUINDO, DENTRE OUTRAS, AS GARANTIAS IMPLÍCITAS DE COMERCIABILIDADE E ADEQUAÇÃO A UMA FINALIDADE ESPECÍFICA. O RISCO INTEGRAL QUANTO À QUALIDADE E DESEMPENHO DO PROGRAMA É ASSUMIDO POR VOCÊ. CASO O PROGRAMA CONTENHA DEFEITOS, VOCÊ ARCARÁ COM OS CUSTOS DE TODOS OS SERVIÇOS, REPAROS OU CORREÇÕES NECESSÁRIAS.

12. EM NENHUMA CIRCUNSTÂNCIA, A MENOS QUE EXIGIDO PELA LEI APLICÁVEL OU ACORDADO POR ESCRITO, QUALQUER TITULAR DE DIREITOS AUTORAIS OU QUALQUER OUTRA PARTE QUE POSSA MODIFICAR E/OU REDISTRIBUIR O PROGRAMA, CONFORME PERMITIDO ACIMA, SERÁ RESPONSÁVEL PARA COM VOCÊ POR DANOS, INCLUINDO ENTRE OUTROS, QUAISQUER DANOS GERAIS, ESPECIAIS, FORTUITOS OU EMERGENTES, ADVINDOS DO USO OU IMPOSSIBILIDADE DE USO DO PROGRAMA (INCLUINDO, ENTRE OUTROS,

PERDAS DE DADOS OU DADOS SENDO GERADOS DE FORMA IMPRECISA, PERDAS SOFRIDAS POR VOCÊ OU TERCEIROS OU A IMPOSSIBILIDADE DO PROGRAMA DE OPERAR COM QUAISQUER OUTROS PROGRAMAS), MESMO QUE ESSE TITULAR, OU OUTRA PARTE, TENHA SIDO ALERTADA SOBRE A POSSIBILIDADE DE OCORRÊNCIA DESSES DANOS.

**FINAL DOS TERMOS E CONDIÇÕES**

Como Aplicar Estes Termos para Seus Novos Programas.

Se você desenvolver um programa novo e quiser que ele seja da maior utilidade possível para o público, o melhor caminho para obter isto é fazer dele um *Software* Livre, o qual qualquer pessoa pode redistribuir e modificar sob os presentes termos.

Para fazer isto, anexe as notificações seguintes ao programa. É mais seguro anexá-las ao começo de cada arquivo-fonte, de modo a transmitir do modo mais eficiente a exclusão de garantia; e cada arquivo deve ter ao menos a linha de "direitos autorais reservados" e uma indicação de onde a notificação completa se encontra.

¡uma linha para informar o nome do programa e uma breve idéia do que ele faz.¿ Direitos Autorais Reservados (c) ¡nome do autor¿ Este programa é *Software* Livre; você pode redistribuí-lo e/ou modificá-lo sob os termos da Licença Pública Geral GNU conforme publicada pela Free *Software* Foundation; tanto a versão 2 da Licença, como (a seu critério) qualquer versão posterior.

Este programa é distribuído na expectativa de que seja útil, porém, SEM NENHUMA GARANTIA; nem mesmo a garantia implícita de COMERCIABILIDADE OU ADEQUAÇÃO A UMA FINALIDADE ESPECÍFICA. Consulte a Licença Pública Geral do GNU para mais detalhes.

Você deve ter recebido uma cópia da Licença Pública Geral do GNU junto com este programa; se não, escreva para a Free *Software* Founda-

> tion, Inc., no endereço 59 Temple Street, Suite 330, Boston, MA 02111-1307 USA. Inclua também informações sobre como contatar você por correio eletrônico e por meio postal.

Se o programa for interativo, faça com que produza uma pequena notificação como esta, quando for iniciado em um modo interativo:

> Versão 69 do Gnomovision, Direitos Autorais Reservados (c) ano nome do autor. O Gnomovision NÃO POSSUI QUALQUER TIPO DE GARANTIA; para detalhes, digite 'show w'. Este é um *Software* Livre e você é bem-vindo para redistribuí-lo sob certas condições; digite 'show c' para detalhes.

Os comandos hipotéticos 'show w' e 'show c' devem mostrar as partes apropriadas da Licença Pública Geral. Naturalmente, os comandos que você utilizar poderão ter outras denominações que não 'show w' e 'show c'; eles poderão até ser cliques do *mouse* ou itens de um menu – o que for adequado ao seu programa.

Você também pode solicitar a seu empregador (se você for um programador) ou sua instituição acadêmica, se for o caso, para assinar uma "renúncia de direitos autorais" sobre o programa, se necessário. Segue um exemplo; altere os nomes:

> A Yoyodyne Ltda., neste ato, renuncia a todos eventuais direitos autorais sobre o programa 'Gnomovision' (que realiza passagens em compiladores), escrito por James Hacker.
>
> ¡Assinatura de Ty Coon¿ 1º de abril de 1989, Ty Coon, Presidente

Esta Licença Pública Geral não permite a incorporação do seu programa a programas proprietários. Se seu programa é uma biblioteca de sub-rotinas, você poderá considerar ser mais útil permitir a ligação de aplicações proprietárias à sua biblioteca. Se isso é o que você deseja fazer, utilize a Licença Pública Geral de Biblioteca do GNU, ao invés desta Licença.

# Referências Bibliográficas


[1] Introducing slony. http://www.onlamp.com/pub/a/onlamp/2004/11/18/slony.html.

[2] Modifying slony clusters. http://www.onlamp.com/pub/a/onlamp/2005/03/17/slony_changes.html.

[3] Pgcluster. http://pgcluster.projects.postgresql.org/.

[4] Pgpool. http://pgpool.projects.postgresql.org/.

[5] Postgresql. http://www.postgresql.org.

[6] Nazareno Andrade, Walfredo Cirne, Francisco Brasileiro, and Paulo Roisenberg. Ourgrid: An approach to easily assemble grids with equitable resource sharing. *9th Workshop on Job Scheduling Strategies for Parallel Processing*, June 2003.

[7] C. J. Date. *An Introduction to Database Systems*. Addison-Wesley, Reading, MA, 6 edition, 1995.

[8] C. DE ROSE, BLANCO, F. MAILLARD, N. SAIKOSKI, K. NOVAES, R. RICHARD, and O. RICHARD. The virtual cluster: a dynamic environment for exploitation of idle network resources. *14th symposium on Computer Architecture and High-Performance Computing (SBAC-PAD 2002). USA: IEEE Computer Society*, pages p.141 – 148, 2002.

[9] Corinto Meffe, Elias O. P. Mussi, Leonardo Mello, and Rogério Santanna dos Santos. A tecnologia de cluster e grid na resolução de problemas de governo eletrônico. *The 18th International Symposium on Computer Architeture and High Performance Computing*, pages 1–8, Outubro 2006.

[10] PostgreSQL. High availability. http://www.postgresql.org/docs/9.1/static/high-availability.html. Acessado em 16/06/2013.

[11] PostgreSQL. Postgresql 8.2 release notes. http://www.postgresql.org/docs/8.2/static/release-8-2.html. Acessado em 16/06/2013.

[12] PostgreSQL. Postgresql 9.0 release notes. http://www.postgresql.org/docs/9.0/static/release-9-0.html. Acessado em 16/06/2013.

[13] Michael Stonebreaker and Lawrence A. Rowe. The design of postgres, 1985. http://s2k-ftp.cs.berkeley.edu:8000/postgres/papers/ERL-M85-95.pdf Acessado em 22/05/2013.

[14] Michael Stonebreaker and Lawrence A. Rowe. The postgres data-model, 1987. http://s2k-ftp.cs.berkeley.edu:8000/postgres/papers/ERL-M87-13.pdf Acessado em 22/05/2013.

[15] Michael Stonebreaker, Lawrence A. Rowe, and S. Potamianos. Postgres rule system, 1989. http://www.postgresql.org/docs/9.2/interactive/biblio.html#STON87A Acessado em 22/05/2013.

[16] TOP500. Top500.org. http://www.top500.org/. Ultima Visita em 20.04.2006 12:20.

[17] WikiPedia. Wikipedia, the free encyclopedia. http://pt.wikipedia.org/wiki/Mainframes. Ultima Visita em 20.04.2005 12:20.