

PostgreSQL em alta disponibilidade (parte 1): usando sistema de arquivos distribuído GlusterFS

Author: André Felipe Machado<andremachado@techforce.com.br> - Escopo

Este estudo analisará e avaliará tecnologias para alta disponibilidade de servidor de banco de dados PostgreSQL, com ênfase em preservação da escalabilidade de porte e carga e viabilidade de distribuição geográfica.

Sempre que possível, buscará manter compatibilidade com tecnologias de alto desempenho.

- Objetivos e critérios

Simplificação da infraestrutura de alta disponibilidade, eliminação de SPOF e gargalos, mantendo escalabilidade, com redução de custos, menor complexidade de administração e maior flexibilidade para diferentes cenários de carga e porte, mantendo a máxima transparência para sgbds e aplicações.

Estudaremos conceitos que sejam aplicáveis sobre equipamentos desde os mais singelos até o estado da arte.

- Justificativa

A parte mais frágil da infraestrutura, em 2009, é o sistema de armazenamento de dados.

Atualmente, as memórias de massa mais viáveis para grandes volumes de dados ainda são os discos rígidos magnéticos.

Em algumas implantações, visando alto desempenho, partes dos dados ou dos arquivos de trabalho do sgbds podem ser armazenados em ainda caros discos de estado sólido (SSD).

Dos mais simples discos IDE de baixa velocidade até velozes unidades SAS internas aos SAN, todos sofrem com as mazelas dos desgastes mecânicos e taxas de falhas agravadas pela

temperatura e até mesmo ruído sonoro.

São o elo mais fraco da corrente de disponibilidade.

Para se alcançar níveis de disponibilidade maiores, é necessário pensar em locações diversificadas geograficamente para os centros de dados.

- Cenário

Um servidor executando o sgbd Postgresql, acessando arquivos distribuídos transparentemente, montados em modo local em seu ponto de vista. Sem alterações na aplicação ou no sgbd.

Na futura Parte 2 estudaremos a alta disponibilidade, possivelmente com balanceamento de carga, com mais de um sgbd e failover automático de forma transparente à aplicação, sem SPOF.

- Arquitetura do cluster

Estudaremos a abordagem com sistema de arquivos compartilhado, em leitura e escrita, e distribuído.

Essencial ter locks coordenados e conformidade POSIX.

Será usado espelhamento com failover automático.

Será usado balanceamento de carga round-robin, o mais simples dos disponibilizados.

Sendo um sgbd único (e na parte 2 um grupo finito e conhecido), para eliminar os SPOF e gargalos, o espelhamento e failover é feito no cliente do fs (no caso, o host do sgbd).

Toda a infra estrutura física e lógica de rede e armazenamento pode ser redundante e tolerante a falhas (espelhamento, stripping, múltiplas fontes de alimentação, etc), com múltiplas interfaces de rede e múltiplos discos físicos em cada servidor.

Os servidores de filesystem podem estar dispersos geograficamente, limitados em desempenho

pela velocidade de rede.

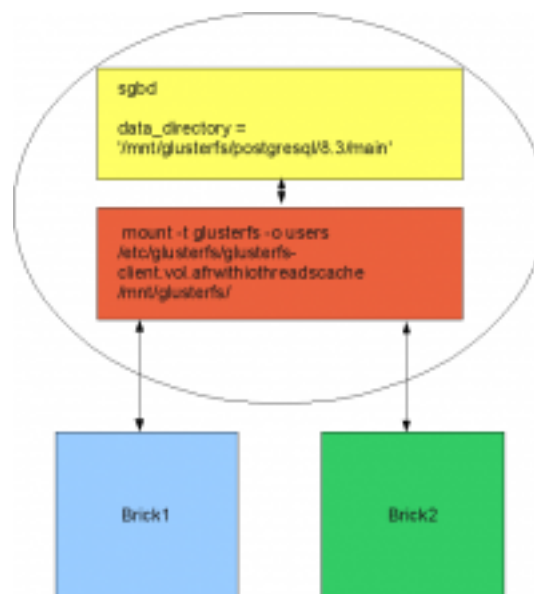
A tecnologia de rede adotada será tcp/ip sobre ethernet, entre as disponibilizadas.

O sistema de arquivos nativo dos servidores será o ext3, adotando journaling de metadados *E* dados em um deles, para avaliações.

Não abordaremos tunings de sistema de arquivos nativos, sgbd ou kernel e rede, adotando os defaults do Debian.

Configurações de cluster para confiabilidade e disponibilidade em primeiro lugar, sem impedir futuro desempenho.

Cuidados com split brain.



- Por que GlusterFS?

Ver artigos anteriores sobre virtualização e cluster em Debian publicado em 2007.

[Virtualização e serviço de arquivos em cluster de alta disponibilidade com Debian Etch, com redundância, espelhamento, replicação, em ambientes de desenvolvimento](#) (parte 1, 2, 3).

[Red Hat cluster suite in Debian GNU / Linux 4.0 Etch.](#)

Full POSIX.

Simplicidade e elegância eficiente e eficaz.

Sem reinventar a roda. Usa atributos extendidos no sistema de arquivos nativo que for escolhido. Não escreve, manda o fs escrever.

Flexibilidade total de configuração para diferentes cenários.

Modular.

Escalável.

Não exige hardware especial.

Pode ser configurado para não ter gargalos nem SPOF.

Pode ser configurado para diversos cenários de confiabilidade, disponibilidade, desempenho, escalabilidade, complexidade, transparência.

Pode usar rede especial de muito alta velocidade.

Baixíssimo consumo de recursos da máquina hospedeira.

Sem modificações de kernel (estabilidade e segurança). Opera em user space.

Tráfego passível de firewalling.

Self healing razoavelmente automatizado.

Vários protocolos de comunicação. Ainda mais possíveis, dependendo da arquitetura de implantação (iSCSI, NFS, SMB...).

- Contras

Maturidade em pequena variedade de cenários.

Mas já há uma boa massa de usuários que aponta obscuros bugs em cenários diversos.

Evolução bastante rápida.

Migração da 1.x para 2.x exige backup, reimplementação, restore. Sem automação.

Na versão 2.x existe possibilidade de solução automatizada de split-brain, mas é perigosa por assumir um brick como referência SEMPRE. Melhor adotar manual a menos que haja um brick com hardware ultra confiável.

- Avaliações comparativas

Partindo do estudo publicado em 2007, ainda hoje conhecemos apenas GlusterFS, GFS, GFS2 e OCFS2 como full POSIX.

A variedade de cenários aplicáveis de GFS e OCFS2 é limitada e mínima flexibilidade.

GFS2 é considerado imaturo pela Red Hat.

Alta complexidade, exigem hardwares especiais para alcançar funcionalidade failover correta.

Exigem recursos de cpu e intenso tráfego, exigindo redes próprias de alta velocidade e boas cpu.

Precisam módulos de kernel, arriscando instabilidade e dificultando updates e upgrades.

Implantação e administração complexa.

Inflexibilidade para variados cenários.

Complexo conseguir sem SPOF. Implantação e administração.

Não escaláveis.

Reinventam a roda, dispendiosamente.

- Prova de conceito

Tudo Debian Lenny 5.0 e Pg 8.3 defaults.

Hardware modesto, obsoleto.

brick1: Pentium III 966, 256 MB ram, discos IDE, partições 10GB, rede 100 Mb/s.

brick2: VM VirtualBox, com 256 MB ram, disco virtual ext3 sobre ext3 com journaling total.

host sgbd: P4 , 1GB ram, disco sata, rede 100 Mb/s

- Resultados

modo nfs-like (porém mais simples) sobre ext3 normal:

12 MB/s.

75 TPS pgbench TPC-B sem reconexão.

modo afr com caches:

5 MB/s (ext3 journal total e virtual)

8 MB/s (ext3 journal normal)

21 TPS pgbench TPC-B sem reconexão

1 TPS pgbench TPC-B com reconexão.

2% cpu no brick1

15% cpu no brick2

rede 100% ocupada

disco (cpu wait states) menor que 8% no brick1 e menor que 35% no brick2 em scp.

disco (cpu wait states) menor que 3% no brick1 e menor que 8% no brick2 em pgbench.

- Potenciais

Simplificação da arquitetura de alta disponibilidade.

As possibilidades de failover, escalabilidade, desempenho, ficam preservadas.

A partir de hardwares obsoletos já é possível ter um cluster.

- Observações

Este estudo foi apresentado durante o [Rantem 8 em Porto Alegre](#) em 2009.

Na própria semana da palestra, foi lançada a versão 2.0 do GlusterFS. Tornou irrelevante a disponibilização de arquivos de configuração, pois houveram profundas alterações, novos plugins disponibilizados e uma grande mudança no protocolo interno de comunicação. O uso de largura de banda foi reduzido quase à metade, reduzindo maciçamente o overhead de comunicação do cluster. Mas não há upgrade direto possível, sendo necessário um minucioso e manual ciclo de backup, redeployment, restore. Isto foi explicado durante a palestra.