

Replicação

Replicação: Definição

Contribui para o desempenho, alta disponibilidade e tolerância a falhas (ex.: cache, redundância de módulos (software + hardware)).

Replicação: Vantagens

- Ganho de desempenho
 - Cache de dados mais perto do cliente;
 - Divisão e distribuição de grandes coleções de dados:
 - Replicação de servidores para distribuição de carga/processamento permite melhores tempos de resposta.
- Aumento da disponibilidade
 - Disponibilidade do serviço é a quantidade de tempo em que o serviço se encontra disponível;
 - Pode ser prejudicada através de falhas nos servidores, serviços e com problemas de rede de comunicação;
 - Diferença entre replicação de servidores e Cache
 - Cache não tem obrigatoriamente ficheiros completos, é uma forma de replicação parcial e está mais próxima do destino
- Tolerância a falhas
 - Não significa correção ou consistência
 - Problemas:
 - crash de **N** servidores
 - resolvido através de **N+1** servidores
 - **N** falhas bizantinas
 - existe resposta, mas com um valor errado
 - resolvido através de **2N+1** servidores
 - a resposta válida é da maioria dos servidores

Replicação: Requisitos

- transparência
 - do ponto de vista do cliente, tudo deve funcionar como se existisse uma única réplica
- consistência
 - quando as operações efetuadas sobre um conjunto de objetos replicados obtém resultados que obedecem a critérios de correção

Replicação: Modelo Geral

- **Replica Manager (RM)**: módulo ou servidor que contém as replicas, que comunicam entre si, cada RM contém uma replica de tudo ou parte dos dados
- **Operações dos Clientes**: somente operações de leitura e de alteração
- **Frontend**: intermediário entre o cliente do serviço e o RM, e torna a replicação transparente para o

cliente

- **Coordenação:** Ordenação no processo dos pedidos
 - **FIFO:** se o frontend processa r e depois r', então RM trata r antes de r'
 - **Casual:** se o pedido (a um RM) para r aconteceu antes do pedido para r', então RM tratará r antes de r'
 - **Total:** se um RM consistente trata r antes de r', então todos os RM consistentes tratarão r antes de r'

Replicação: Comunicação em Grupo

- A troca de mensagens com os RM é mais eficaz através de comunicação em grupo (multicast)
- **Multicast:** envia pedidos para vários sítios ao mesmo tempo

Replicação: Réplicas e consistência

- Critérios de consistência
 - **Consistência sequencial:** não usa referências temporais mas usa uma ordem (sequência)
 - **Consistência linear:**
 - a sequência permite alcançar uma única cópia correta dos objetos
 - a ordem das operações na sequência está de acordo com o tempo real a que efetivamente ocorreram

Replicação: Tolerância a falhas

- Modelos de replicação para tolerância a falhas
 - replicação passiva
 - replicação ativa

Replicação: Replicação passiva

- Existe um único RM primário e vários secundários
- Frontend comunica apenas com o RM primário
- RM primário executa a operação e envia cópias dos dados atualizados aos RM Backup
- Existe consistência linear:
 - lineariza os pedidos
- Em caso de falha no RM primário:
 - um dos RMs de backup é promovido a RM primário e regista-se como primário no serviço de nomes
 - Se continua do ponto em que o sistema se encontrava mantém a consistência linear
- Falhas:
 - tolera **falhas do tipo crash**, N+1 RM
 - não tolera **falhas bizantinas**
- Desvantagem:
 - lentidão entre RM primário e RMs de backup

Replicação: Replicação ativa

- RM têm igual função e estão organizados como um grupo
- Frontend envia o pedido por multicast aos RMs
- Cada RM processa o pedido e responde ao FE
- A falha de um RM não tem impacto sobre o serviço
- Existe consistência sequencial
 - cada FE trata os pedidos de forma sequencial, como um FIFO
- Então existe consistência linear
- Falhas:
 - tolera **falhas do tipo crash**
 - tolera **falhas bizantinas**, com $2N+1$ RM

Replicação: Serviços de alta disponibilidade

- objetivo: proporcionar aos clientes o acesso ao serviço, com melhor tempo de resposta, ou seja, minimizar os tempos de resposta
- exemplos:
 - Arquitetura **gossip**

Replicação: Arquitetura Gossip

- Os RMs podem ficar temporariamente desligados, sofrem updates individualmente e quando se voltam a ligar e trocam mensagens com as atualizações.
- Garantias:
 - cada cliente observa respostas consistentes ao longo do tempo
 - as atualizações são feitas por ordem cronológica
- Sacrifica o nível de consistência entre os RM
- Cada FE tem uma version timestamp com a versão dos últimos dados observados de cada RM
- Cada RM inclui:
 - **Value** - dados
 - **Value timestamp** - tem uma entrada por cada RM
 - **Update log** - registo de operações sobre os dados, mantidas numa lista por 2 motivos:
 - algumas das operações não podem aplicar-se de imediato, só depois do RM ficar em estado atualizado para a operação em causa
 - essa informação pode ser necessária para efeitos de gossip messages
 - **Replica timestamp** - inclui versão dos updates efetivamente aplicados no RM
 - **Operation Table** - histórico para deteção de duplicados

Sharding

- no contexto das BDs distribuídas, consiste na distribuição dos dados por várias BDs
 - os dados são particionados e cada bloco vai para uma BD/servidor dedicado
- É o bloco de dados com a mesma partition key.
- A ideia é evitar que um só servidor seja alvo de muita carga
- Algorithmic sharding (usa um algoritmo para escolher)
 - adequado para distribuições homogêneas de dados, com acessos igualmente distribuídos
- Dynamic sharding (usa um intervalo para escolher)

- adequado para distribuição de coleções não uniformes de dados

Sharding e Replicação

- Sharding sem replicação
 - (+) Facilita a concorrência
 - (-) Custo de validar restrições e cálculos agregados
- Réplica de shards
 - (+) Desempenho das Queries
 - (-) Custo das atualizações, validar restrições e complexidade do controlo de concorrência
- Réplica parcial
 - Só alguns shards são replicados

Ideias e propósitos

Ideias	Propósito
Partitioning (Sharding)	Escalabilidade; Reduzir latência;
Replication	Robustez; Tolerância a falhas; Disponibilidade do serviço;
Caching	Reduzir latência

Sistemas de Ficheiros Distribuídos

- Partilha de informação previamente armazenada
- Módulos de um SF (não distribuído):
 - Directory module
 - File module
 - Access module control
 - File access module
 - Block module
 - Device module

SFD: Requisitos

- Transparência
- Controlo de concorrência
- Replicação de ficheiros
- Abertura e independência face a diferenças de hardware ou Sistemas Operativos
- Tolerância a falhas
- Consistência
- Segurança
- Eficiência

SFD: Existentes (pré 2000)

- Sun Network File System (NFS)
- Andrew File System (AFS)

SFD: Componentes do serviço de ficheiros

- Serviço flat file
 - operações sobre o conteúdo dos ficheiro com um identificador único (UFIDs)
- Serviço de diretorias
 - mapeia nomes de ficheiros em UFIDS
 - cria diretorias e adiciona ficheiros a diretorias
- Módulo cliente
 - Executado em cada computador cliente
 - Guarda informação sobre a localização na rede dos serviços de diretorias e flat file

SFD: Operações do Serviço Flat File

- Read
- Write
- Create
- Delete
- GetAttributes
- SetAttributes
- Controlo de Acesso
 - Os direitos do utilizador são validados com o modo de acesso especificado na operação open
 - Verificação dos direitos de acesso efetua-se no servidor onde cada pedido inclui a identificação do utilizador

SFD: Operações do Serviço de Diretorias

- Lookup
- AddName
- UnName
- GetNames

SFD: NFS: Sun NFS

- Protocolo NFS:
 - Remote Procedure Calls (RPCs) que permite aos clientes trabalhar com ficheiros remotos e que é independente do Sistema Operativo
- Servidor NFS:
 - corre ao nível do Kernel de um computador
- Os pedidos sobre ficheiros remotos são traduzidos pelo módulo cliente em operações do protocolo NFS e passados ao servidor que detém esses ficheiros
- Podem exigir-se credencias de identificação

SFD: NFS: VFS (Sistema de Ficheiros Virtual)

- Faz a integração entre o sistema de ficheiros local e o remoto
- Em NFS os identificadores dos ficheiros são **file handles**
- VFS contém um v-node por ficheiro aberto
- Cliente NFS é integrado no Kernel

SFD: NFS: Controlo de acesso e autenticação

- Servidor stateless
- Servidor tem de validar a identidade do utilizador junto dos atributos de acesso do ficheiro a cada pedido
- Os clientes enviam informação sobre a autenticação do utilizador, a cada pedido, em campos próprios nas RPCs
- Tem um problema de segurança na versão básica, um utilizador podia alterar o uid passado por RPC
- Solução: Utilização de encriptação DES (Data Encryption Standard) da informação de autenticação do utilizador

SFD: NFS: Interface do servidor NFS

- Lookup
- Create
- Remove
- Getattr
- Setattr
- Read
- Write
- Rename
- Link
- Symlink
- Readlink
- Mkdir
- Rmdir
- Readdir
- Stafs

SFD: NFS: Sistema de ficheiros local e remoto acessíveis

- Mount Service: permite montar parte de um file system remoto por um cliente

SFD: NFS: Cache

- No servidor
 - Mantidos em memória
 - Refletir escrita imediatamente no disco?
 - 1- Write-through caching: atualizar memória e disco antes de responder ao cliente
 - 2- Delayed commit/write: escrever apenas na cache em memória e mais tarde é que é realizada a sincronização com o disco

- No cliente
 - As operações tem todas cache para reduzir os pedidos ao servidor e o tráfego na rede
 - O estado de um ficheiro ativo é atualizado com o servidor a cada 3s

SFD: NFS: Considerações sobre desempenho

- Transparência de acesso e localização, tolerância a falhas
- O servidor é stateless permitindo retornar o funcionamento após um crash
- Normalmente não tem desvantagens relativamente aos ficheiros locais, contudo, pode apresentar problemas derivados a:
 - Escala limitada
 - Sensível à latência da rede
 - Desempenho relativamente inferior na operação write se usar write-through no servidor

SFD: Andrew File System (AFS)

- Acesso transparente a ficheiros partilhados remotos com o NFS
- O acesso aos ficheiros faz-se através das primitivas UNIX
- Servidor AFS: armazena ficheiros UNIX, referenciados de modo semelhante ao NFS
- O principal objetivo de AFS é a escalabilidade alcançada via cache
 - Whole-file serving: transmite o conteúdo inteiro de ficheiro e diretorias
 - Whole-file caching: o cliente faz uma cache que permanece após um crash do cliente

SFD: AFS: Funcionamento

- Quando um utilizador abre um ficheiro remoto que não tem uma réplica local, o servidor respetivo é localizado e envia uma cópia.
- A cópia é armazenada no file system do cliente e aberto, sendo devolvido ao utilizador o respetivo file descriptor UNIX.
- As operações de leitura e escrita ocorrem sobre essa réplica.
- Ao fechar um ficheiro se o conteúdo da réplica foi alterado a mesma é enviada ao servidor file system que a contém

SFD: AFS: Processos

- **Vice**: software que corre do lado do servidor
- **Venus**: software que corre do lado do cliente

SFD: AFS: Identificação de System Calls sobre ficheiros remotos

- O Kernel UNIX interceta system calls sobre partilhados e passa-as ao processo Venus

SFD: AFS: Implementação

- Open
- Read
- Write
- Close

SFD: AFS: Consistência

- Callback promise: garantia de Vice para Venus de que notificará a atualização do ficheiro sempre que outro cliente o alterar
 - Armazenadas do lado do cliente, junto à réplica do ficheiro
 - São revistas quando o servidor recebe uma operação close de um ficheiro atualizado

SFD: AFS: Componentes de service interface vice

- Fetch
- Store
- Create
- Remove
- SetLock
- ReleaseLock
- RemoveCallback
- BreakCallback

SFD: AFS: Outros aspetos

- Desvantagens:
 - Requer alterações no Kernel UNIX
 - O host do servidor é dedicado ao serviço AFS
- Vantagens:
 - Réplica read-only: guardadas em servidores separados o que trás maior performance
 - Desempenho
 - Principal objetivo é a escalabilidade
 - Boa performance quando há um número elevado de utilizadores
 - Reduz a carga no servidor

GFS

GFS: SFD pré-GFS (Google File System): Desenvolvimentos posteriores

- Mecanismos de armazenamento:
 - RAID (Redundant Arrays of Inexpensive Disk)
 - LFS (log-structured File Storage): os dados sofrem updates em memória, havendo lugar a commits periódicos para disco

Google

Google: Motor de pesquisa

- A partir de uma pesquisa obter uma lista ordenada de documentos web
- Crawling: localizar e obter conteúdo de documentos web

- GoogleBot: agente que lê documentos, recolhe URL que referem e avança - recursivamente para os documentos apontados por esses endereços
- Indexação: indexar o conteúdo dos documentos (HTML, PDF, DOC,...)
- Ranking: ordenação do resultado pela importância relativamente à pesquisa
 - Algoritmo PageRanking: usa critérios para apurar a importância para ordenação dos resultados da pesquisa. A página é mais importante se tem mais referências

Google: Fornecedor de serviços cloud based

- Software as a service (SaaS)
 - Paradigma onde as aplicações são alojadas na internet, ao contrário do convencional software local
- Platform as a service (PaaS)
 - Disponibilização de APIs para serviços (distribuídos) que permitem a execução de aplicações (Web)
 - Virtualização da plataforma

Google: Infraestrutura - Modelo Físico

- Não recorrer a hardware caro e especialmente poderoso
 - Procurar o uso de PC com a melhor relação custo/qualidade
 - Falhas são inevitáveis em hardware comum, mas a maioria das falhas estão relacionadas com software
- Racks: conjunto de 40 a 80 PC's com ligações redundantes
- Cluster: grupo de 30 ou mais racks com 2 switches de alta largura de banda, onde cada rack é ligada a ambos os switches, por redundância

Google: Requisitos da arquitetura de suporte

- Escalabilidade: capacidade de tratar mais e mais dados e acomodar o crescimento da web, ou seja, capacidade de processar mais pedidos dos utilizadores
- Fiabilidade: disponibilidade dos serviços
- Desempenho: baixa latência na resposta ao utilizador
- Abertura: APIs e modularidade

Google: Infraestrutura - Elementos por camadas

- Comunicação
 - Direta / invocação remota / serialização: **Protocol Buffers**
 - Assíncrona, grande nº de subscritores: Google **publish-subscribe**
- Armazenamento, coordenação e acesso a dados
 - GFS: SFD específico da google
 - Chubby: coordenação e algum armazenamento
 - Bigtable: repositório de larga escala para dados semi-estruturados
- Computação com carácter distribuído
 - MapReduce: paradigma de computação sobre grandes volumes de dados, nomeadamente sobre

Bigtable

- Sawzall: linguagem de alto nível para descrever computação distribuída

Google: Protocol Buffers

- Mecanismo independente da linguagem e plataforma para serialização de dados do tipo RPC
- Serialização de alto desempenho, dados comprimidos
- Mensagens são constituídas por dois campos, nome e tipo, o tipo pode ser primitivo enumerations e nested
- Métodos gerados automaticamente

Google: GFS

- SFD escalável pensado para sistemas que fazem uso intensivo de um grande volume de dados
- Tolerância a falhas
- Alto desempenho para um grande número de utilizadores
- Design do GFS baseia-se na observação do comportamento dos sistemas existentes

Google: GFS Princípios

- Erros com os componentes são comuns
- Ficheiros muito grandes
- Muitas operações de leitura sequencial longa
- Alta largura de banda tem prioridade sobre a baixa latência
- Chunk: blocos de 64MB que são replicados por ≥ 3 chunkservers
- Mutação: escrita que ocorre em todas as réplicas do chunk
- Gestão centralizada: Master - coordena acessos e controla meta-informação
- Sem cache

Google: GFS Análise

- Gestão da consistência numa escrita
 - Em write, append ou delete, o master atribui o base do chunk a uma das réplicas
 - O cliente começa por contactar o master que lhe identifica os chunkservers
 - Chunkservers confirmam receção e o primário determina ordem de aplicação das operações. Se todos os chunkservers confirmam ao primário, este responde com sucesso ao cliente
- Elementos chaves
 - Redundância e gestão ágil das réplicas
 - Desempenho em situações de acesso integral e sequencial
- Desvantagem: mau desempenho em pequenas operações de escrita ou leitura

Google: Chubby

- Serviço para armazenamento de ficheiros pequenos que complementa o GFS
- Serviço de sincronização distribuída
- Serviço de eleição (por exemplo do chunkserver primário)
- Serviço de nomes

Google: Bigtable

- Armazenamento de dados estruturados e semi-estruturados
 - Acesso pelo offset do conteúdo a partir do início do ficheiro
 - Read/write executados com grande otimização
 - Distribuição, grandes volumes
 - dados organizados em tabelas muito grandes
- Google analytics usa
 - Tabela com o sumário de cliques sobre páginas visitadas (análise com MapReduce)
 - Optar por uma BD relacional distribuída teria inferior
 - Bigtable

Google: Arquitetura Bigtable

- Dados divididos em tablets, armazenadas em GFS
- Chubby: coordenação do acesso e distribuição
- Tablets são representadas em SSTables

Google: Sumário de escolhas de design relacionadas com armazenamento e coordenação

Elemento	escolha de design	Base lógica	Troca
GFS	Uso de um chunk grande	Adequa-se ao tamanho do ficheiro no GFS; Eficáz para leitura e anexos sequenciais grandes; Minimiza a quantidade de metadata;	Seria pouco eficiente para acesso aleatório a pequenas partes do ficheiro
	Uso de um master centralizado	O master mantém a visão global que informa decisões de gestão; Implementação simples;	Único ponto de falha
	Controlo e fluxos de dados separados	Acesso a ficheiros de alto desempenho com o mínimo envolvimento do master	Complica a biblioteca do cliente enquanto lida com o master e os chunkservers
	Modelo consistente de relaxamento	Alto desempenho, explorando semânticas das operações GFS	Os dados podem ser inconsistentes
Chubby	Combinação de aspeto e ficheiros abstratos	Multiúso/universal	Tem de perceber e diferenciar a diferença entre facetas
	Leitura e escrita de Whole-file	Muito eficiente para ficheiros pequenos	Inapropriado para ficheiro grandes

	Caching do cliente com consistência rigorosa	Semânticas determinísticas	Sobrecarga de gerir a consistência rigorosa
BigTable	Usa tabela de abstração	Suporte na eficiência de estruturas de dados	Menos expressividade que um BD relacional
	Uso de um master centralizado	Master com uma visão global; Implementação simples;	Único ponto de falha, possível engarrafamento
	Controlo e fluxos de dados separados	Acesso a ficheiros de alto desempenho com o mínimo envolvimento do master	-
	Monitorizar e carregar o balanceamento	Pode ter clientes em paralelo	Sobrecarga associada com gestão de estados globais