Importância do tempo: Em SD, a noção de tempo é crucial para determinar a ordem dos eventos, auditar processos e identificar eventos simultâneos. Os processos executam-se em máquinas distintas sem partilha de memória, processador ou relógio.

Relógios:

Skew (Distorção): É a diferença do valor do tempo lido de dois relógios diferentes.

Clock drift: É a diferença no valor lido de um relógio e o valor do tempo fornecido por um relógio de referência perfeito por unidade de tempo do relógio de referência.

Drift rate: Taxa que mede o desvio da leitura de um relógio face a uma referência horária perfeita.

Sincronização com UTC: Coordinated Universal Time (UTC) é uma norma que combina o International Atomic Time (IAT) com ajustes ocasionais para alinhamento com o tempo astronômico real.

Relógio global tem como objetivo fornecer o mesmo tempo para todos os intervenientes do sistema.

Sincronização de Relógios: Um computador pode acertar o seu relógio utilizando um recetor para captar sinas UTC/GPS ou em alternativa, um computador pode receber o tempo oficial através de uma linha telefónica ou da rede, a partir de organizações.

Modos de Sincronização:

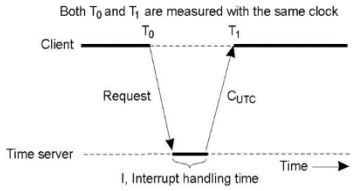
Modo Interno: Os relógios são sincronizados uns com os outros, mas sem referência de uma fonte de tempo externa.

Modo Externo: Os relógios do sistema são sincronizados com uma fonte de tempo externa, de forma a manter os relógios alinhados com o tempo real mundial.

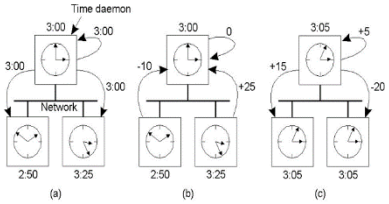
Relógios sincronizados de modo interno não estão necessariamente sincronizados de modo externo, pois eles podem desviar-se coletivamente de uma fonte de tempo externa, mesmo que concordem uns com os outros.

Métodos de Sincronização:

Algoritmo de Cristian: Usa a sincronização cliente-servidor, considerando o tempo de troca de mensagens. Um cliente solicita a hora atual a um servidor. Ao receber a solicitação, o servidor responde com o seu horário atual. O cliente recebe a resposta e estima o tempo de viagem da mensagem para calcular o atraso de rede. O cliente ajusta o seu relógio com base no tempo recebido do servidor e no atraso estimado. Problemas: Ponto único de falha e congestionamento (apenas 1 servidor), ou um servidor em falha ou malicioso pode provar estragos. Formas de melhorar a precisão: Informação horária prestada por um grupo de servidores sincronizados. Cliente faz multicast do pedido para o grupo de servidores e usa a primeira resposta obtida.



Algoritmo de Berkeley: Para sincronização interna, um master coordena a hora com os demais computadores (slaves). O master interage periodicamente com todos os elementos do grupo para saber a diferença relativamente a cada um. O master estima a hora em cada slave. O master calcula a média de todos os valores, incluindo o seu. Em vez de enviar o tempo atualizado aos slaves, o master envia a cada um o valor exato que este deve usar para ajustar o seu relógio.



Network Time Protocol (NTP): Multiplos servidores de tempo espalhados pela Internet. Distribui informação horária na Internet, permitindo sincronização precisa de clientes com UTC. Serviço fiável que resiste a perdas de conectividade. Serviço suportado por uma rede de servidores na internet onde os primários estão ligados diretamente a uma fonte de UTC e os secundários sincronizados com a informação dos primários. Permite sincronizar um elevado número de máquinas.

Modo de sincronização do NTP:

Modo multicast: Usado em redes locais de alta velocidade. Um ou mais servidores enviam periodicamente o tempo para outros servidores. Os servidores acertam o seu relógio assumindo um pequeno delay. Baixa precisão.

Modo "procedure call": Idêntico ao algoritmo Cristian. Os clientes solicitam o tempo de um ou vários servidores, e estes enviam o valor do seu relógio. Quando se pretende melhor precisão ou não é possível o multicast.

Modo simétrico: Maior exatidão, usado em servidores primários ou próximos. Pares de servidores solicitam o tempo um ao outro.

Concorrência: Pode ocorrer localmente (várias threads num processo) ou de forma distribuída (vários processos em máquinas diferentes). Pelo que é necessário garantir exclusão mútua (garante que apenas um processo pode estar na zona crítica a qualquer momento) a nível do SD.

Controlo de concorrência, é a gestão de transações sobre objetos de forma que nenhuma transação tenha efeito/impacto na operação das demais transações.

Tipos de Controlo de Concorrência:

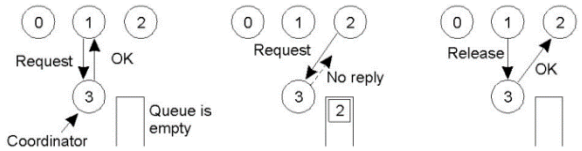
Pessimista: Acesso exclusivo por uma transação durante a sua operação.

Otimista: Assume que não há interferências, avaliando posteriormente e desfazendo se necessário.

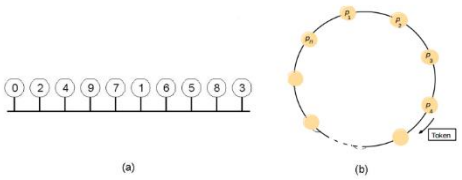
Requisitos desejáveis na exclusão mútua: Safety, nó máximo pode haver 1 processo na zona critica. Liveness, os pedidos para acesso à ZC terão uma resposta. Ordering, se o pedido P1 chegou primeiro que o P2, terá de ser atendido primeiro.
A performance de cada algoritmo é analisada em termos de largura de banda consumida, delay para o cliente e tempo entre um processo sair e outro entrar na ZC.

Algoritmos de Exclusão Mútua:

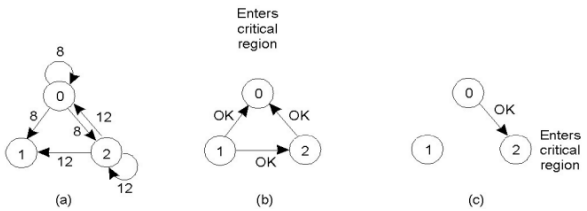
Algoritmo Centralizado: Um processo coordena o acesso à zona crítica, mantendo uma fila de espera com os processos interessados em aceder à zona.



Algoritmo de Coordenação em Anel: Os Processos formam um anel lógico e passam um token, quem se encontra detentor do token tem acesso à zona de exclusão.



Algoritmo Distribuído: Processos comunicam-se via multicast, incluindo timestamps para determinar a ordem de acesso à zona.



Replicação: Num SD espera-se que os serviços tenham alta disponibilidade, o melhor desempenho possível e que o efeito das falhas seja mínimo. A replicação é fundamental para atingir esses objetivos.

Vantagens da Replicação:

Ganho de Desempenho: Melhora tempos de acesso e resposta, distribuindo a carga entre servidores.

Aumento da Disponibilidade: Disponibilidade: Quantidade de tempo em que o serviço está funcional e com tempos de resposta razoáveis. Reduz o impacto das falhas dos servidores, mantendo o serviço acessível mesmo em caso de falha de um servidor. Diferenças entre replicação de servidor e cache: Cache não tem obrigatoriamente conjuntos de objetos completos (ficheiros inteiros). A cache é uma forma de replicação parcial.

Tolerância a Falhas: Garante um comportamento correto mesmo na ocorrência de falhas.
Problemas:

Crash de N servidores. Resolver com replicação N+1 servidores.

N falhas bizantinas (existe resposta, mas com um valor errado). Resolver com replicação 2N+1 servidores. A resposta válida é a da maioria, pelo menos N+1.

Requisitos da Replicação:

Transparência: Os clientes não devem perceber detalhes da replicação.

Consistência: As operações em objetos replicados devem obedecer a critérios de correção.

Consistência sequencial: As operações de todos os processos são vistas numa única ordem sequencial em máquinas diferentes.

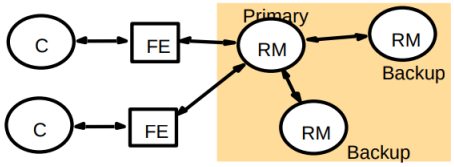
Consistência linear: Exige que todas as operações num sistema distribuídos sejam vistas por todos os processos na mesma ordem, e essa ordem deve corresponder à ordem real em que as operações ocorreram.

Modo Geral de Replicação: Inclui um Replica Manager (RM) que atua sobre réplicas e coordena os outros RMs. Cada RM tem uma réplica de tudo ou parte dos dados.
FrontEnd: Módulo mais exterior do serviço. Torna a replicação transparente para o cliente.
Modelo geral para atendimento do pedido do cliente: 1. FrontEnd envia o pedido para um ou mais RM. 2. Os RMs coordenam-se para executarem a operação de forma consistente. 3. Execução do pedido nos RM. 4. RMs entendem-se relativamente ao resultado e em função disso fazem rollback ou commit. 5. Um ou mais RM respondem ao FE.

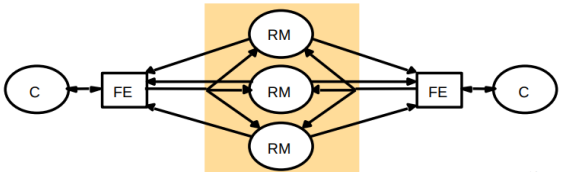
Comunicação em Grupo: A troca de mensagens com os RM é mais eficaz através da comunicação para um grupo (multicast).

Modelos de Replicação para Tolerância a Falhas:

Replicação Passiva: Existe um único RM primário e vários secundários ou de backup. Os FE comunicam apenas com o RM primário. O RM primário executa a operação e envia cópias dos dados atualizados aos RM Backup.
Sequencia de eventos para atender um pedido: 1. FE passa o pedido, com um identificador único, ao RM primário. 2. O primário trata cada pedido pela ordem em que recebe, se já foi executado reenvia a resposta. 3. Primário executa a operação e guarda a resposta. 4. Em caso de update, o primário envia para todos os backups. 5. O primário responde ao FE, que por sua vez responde ao cliente.
Existe consistência linear, o RM primário lineariza os pedidos.
Em caso de falha no RM primário um dos backups é promovido a RM primário.
Tolerância a falhas: Permite sobreviver a N crashes de servidores com N+1 RM. Não tolera falhas bizantinas.
Vantagens: Simplicidade. Desvantagem: Lentidão na sincronização entre o primário e RMs backups.



Replicação Ativa: RMs organizados como um grupo. FE envia o pedido por multicast aos elementos do grupo. Todos os RMs dão uma resposta ao FE. A falha de um RM não tem impacto sobre o serviço.
Sequencia de eventos para atender um pedido: 1. FE faz multicast para o grupo de RM. FE não envia o próximo pedido até receber uma resposta. 2. Todos os RMs recebem a resposta. 3. Cada RM executa o pedido. 4. Os RM enviam a resposta, juntamente com o identificador único ao FE.
O número de respostas que o FE processa depende dos objetivos. Se for tolerar falhas crash, devolve ao cliente a primeira resposta. Se for tolerar falhas bizantinas, é preciso recolher várias repostas e comparar valores, procurando o da maioria e devolver.
Existe consistência sequencial, os FE tratam os pedidos de forma sequencial. Não existe consistência linear porque a ordem pela qual os RMs processam os pedidos pode não ser igual à ordem pela qual os FEs pediram a operação.
Tolerância a falhas: Tolera até N falhas bizantinas com, pelo menos, 2N+1 RM. Cada FE espera até receber N+1 respostas com o mesmo valor e responde isso ao cliente.
Vantagens: Alta disponibilidade, distribuição da carga. Desvantagens: A replicação ativa consome muitos recursos, pois todas as réplicas executam cada pedido. Um método alternativo para reduzir a computação nos backups é a replicação passiva.



Arquitetura Gossip: Serviço de replicação de alta disponibilidade, criado de modo a diminuir o tempo de resposta ao cliente sacrificando um pouco a consistência da informação nos RMs. Os RMs podem ficar temporariamente desligados, quando voltam a ligar-se trocam mensagens com as atualizações (gossip messages).
Os RMs podem ficar algum tempo sem comunicar. Existe a garantia de que cada cliente observa respostas consistentes ao longo do tempo. Há "relaxed consistency" nas réplicas, todos os RMs eventualmente recebem todas as atualizações, que aplicam por ordem.

Sharding: Consiste na distribuição dos dados por várias BDs, os dados são particionados e cada bloco vai para uma BD/servidor dedicado. Partição horizontal. Permite distribuir dados e carga sobre os servidores.
Vantagens: Escalabilidade, redução de latência, fácil gestão de grandes volumes de dados. Desvantagens: Complexidade na implementação, dificuldade na manutenção e gestão de carga.

Tipos de Sharding:

Algorithmic sharding: Adequado para distribuições homogéneas de dados, com acessos igualmente distribuídos. Faz uso de um algoritmo determinístico para decidir como distribuir os dados entre os shards.

Dynamic sharding: Adequado para distribuição de coleções não uniformes de dados. Este método monitora a carga de trabalho e a distribuição dos dados, ajustando os shards conforme necessário.

Sistemas de Ficheiros Distribuidos (SFD): Permitem que aplicações gravem e acedam a ficheiros remotos exatamente do mesmo modo que utilizam para os ficheiros locais. Permite o acesso dos utilizadores aos seus ficheiros desde qualquer computador.
Um Sistema de Ficheiros é responsável por: organização, armazenamento, acesso, gestão de nomes, partilha, proteção/segurança dos ficheiros.

Armazenamento centralizado vs. Armazenamento paralelo: SFDs podem utilizar armazenamento centralizado, com riscos de congestionamento, falhas e tamanho de ficheiro limitado à capacidade de um servidor, ou armazenamento paralelo, onde um ficheiro pode ser distribuído por várias máquinas (Ex: GFS).

Ficheiro: Coleção de dados e atributos. Um FS armazena um grande número de ficheiros. A gestão dos nomes é facilitada com o uso de diretorias.
Diretorias: Ficheiro especial que mapeia um nome em identificadores internos para os ficheiros.

Requisitos de um SFD: Transparência, Controlo de concorrência, Replicação de ficheiros, Abertura e independência face a diferenças de hardware e SOs, Tolerância a falhas, Consistência, Segurança, Eficiência.

Exemplos de SFDs:

NFS: O NFS fornece acesso transparente a arquivos remotos para programas clientes. O relacionamento cliente-servidor é simétrico: cada computador numa rede NFS pode atuar como cliente e como servidor, e os arquivos de cada máquina podem se tornar disponíveis para acesso remoto por outras máquinas. O NFS tem como objetivo ser independente do sistema operativo

AFS: Desenvolvido para suportar partilha de larga escala, minimizando a comunicação cliente/servidor. O principal objetivo é a escalabilidade, alcançada via cache.

Diferenças: NFS é stateless (o servidor não mantém informações sobre as conexões dos clientes), o AFS é stateful (mantém um registo do estado das interações dos clientes). AFS é considerado mais escalável. AFS possui mecanismos mais robustos para garantir consistência dos dados. AFS é mais complexo. NFS é mais simples e fácil de implementar.

Componentes de um SF: Serviço Flat File, Serviço de Diretorias, Módulo Cliente.

Sun NFS: Sistema de Ficheiros Virtual (VFS) – Faz a integração entre o sistema de ficheiros local e o remoto, encaminhando cada pedido para o destino apropriado (NFS Client ou SF local). Permite que aplicações interajam com vários sistemas de ficheiros de maneira uniforme. Facilita operações comuns independentemente da localização ou do tipo de sistema de ficheiros. O servidor é stateless.

Google File System (GFS): SFD escalável pensado para sistemas que fazem uso intensivo de um grande volume de dados. Tolerante a falhas. Alto desempenho para um grande número de clientes.

Chubby: Serviço para armazenamento de ficheiros pequenos. Complementar o GFS.

Network Attached Storage (NAS): NAS são dispositivos especializados que se conectam diretamente a uma rede para fornecer armazenamento de dados centralizado. Permite o acesso fácil e partilha de ficheiros entre múltiplos utilizadores e dispositivos numa rede.

Apache Spark: Spark é um sistema avançado de processamento de dados em larga escala. É uma alternativa ao modelo MapReduce, oferecendo maior velocidade e flexibilidade. Processa dados rapidamente, o MapReduce depende mais do disco. Facilidade de uso. Processamento generalizado. Spark é ideal para tarefas que requerem processamento rápido e eficiente de grandes volumes de dados, oferecendo uma plataforma robusta.

RDD (Resilient Distributed Dataset): RDD é uma coleção distribuída de elementos que podem ser processados em paralelo. Características: Imutabilidade, Tolerâncias a falhas, Eficiência de dados.

Flash Crowd: Fenómeno na internet onde um site ou serviço online recebe um aumento súbito e inesperado de tráfego. Pode ser tão grande que sobrecarrega os servidores do site, levando a problemas de desempenho ou até mesmo de indisponibilidade do serviço.

Consequências: Indisponibilidade do serviço, degradar a experiência do utilizador.
Soluções: Redes de Entrega de Conteúdo (CDNs) – distribuem o tráfego por vários servidores, o que ajuda a balancear a carga. Usar infraestruturas de cloud que pode escalar automaticamente.