**Sistema Distribuído**: Conjunto de computadores independentes entre si que se apresenta a seus usuários como um sistema único e coerente. Utiliza troca de mensagens para se comunicar.

“Na prática” existe um SD quando um computador desconhecido impede a realização do seu trabalho. Compostos de uma ou mais camadas de middleware.

**Vantagens**:

- **Economia**: melhor relação custo/desempenho.

- **Eficiência**: maior poder total de computação.

- **Distribuição** Inerente: máquinas espacialmente / geograficamente separadas.

- **Confiabilidade**: se uma máquina falhar o sistema não para.

- **Crescimento incremental**: poder computacional pode ser adicionado posteriormente.

**Middleware**: Situado logicamente entre a camada de aplicação e os SOs.

Características: Ocultar a distribuição geográfica, ocultar heterogeneidade.

Metas de um SD:

**1 – Acesso a recursos:** facilitar o acesso a recursos (sw e hw), de maneira eficiente e controlada. (Segurança de acesso é um desafio.)

**2 – Transparência de Distribuição:** Ocultar o fato de que os recursos estão

distribuídos em uma rede. Transparência de distribuição de: acesso, falha, replicação, localização, migração, concorrência e relocação.

**3 – Abertura**: oferece serviços de acordo com regras padronizadas que descrevem a sintaxe e a semântica desses serviços. Interoperabilidade / Portabilidade.

**4 – Escalabilidade:** Possibilidade de ser expandido em número de recursos e de usuários, mantendo a facilidade de gerenciamento técnico e administrativo.

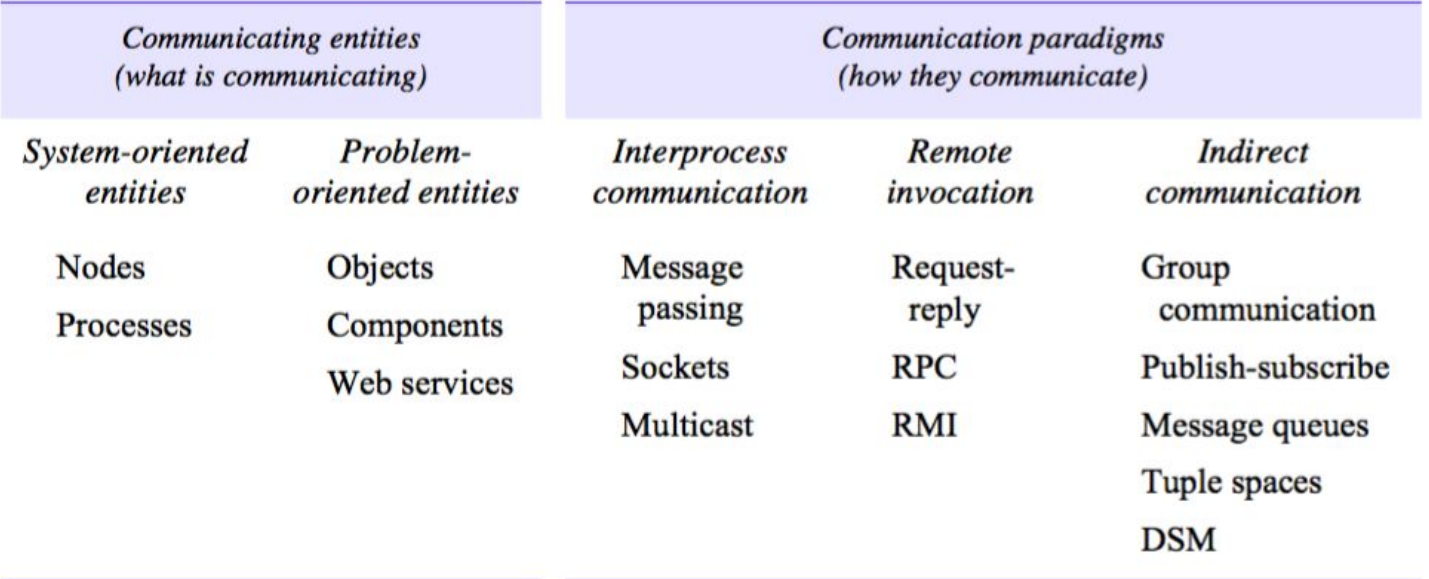
**Modelos de SDs:**

**1. Modelos Físicos:** Consideram os tipos de computadores e dispositivos que constituem um sistema e sua conectividade, sem detalhes de tecnologias específicas.

**2. Modelos Arquiteturais:** Descrevem um sistema em termos de tarefas computacionais e de comunicação realizadas por seus elementos computacionais. (Ex.: Cliente-servidor e ponto-a-ponto).

Núcleo dos elementos arquiteturais;

Padrões arquiteturais compostos;

**3. Modelos Fundamentais:** Deve conter todos os ingredientes essenciais que precisam ser considerados para o entendimento do comportamento de um SD. Dois fatores significativos afetam a interação entre entidades em Sds:

**- Modelos de Interação:**

**- Desempenho de comunicação:**

- **Latência**: tempo de delay para o primeiro bit chegar ao destino.

- **Largura de banda**: quantidade total de informações que se pode transmitir.

- **Jitter**: atraso e distorção na ordem de entrega dos dados/pacotes.

**- Noção de tempo único no sistema:**

- **Clock drift rate:** refere-se à taxa na qual um relógio do computador se desvia de um relógio de referência perfeito.

- Modelos de Falhas:

- Modelos de Segurança.

Sistemas de computação distribuídos:

**- Cluster:** tipo de sistema de processamento paralelo que consiste de uma coleção de computadores independentes interconectados através de uma rede, trabalhando cooperativamente como um único e integrado recurso computacional. Pode ser composto de máquinas específicas costruídas para serem nó do cluster ou por máquinas convencionais, com comunicação realizada via rede. Os nós podem ser dedicados ou não ao cluster. O hw deles pode ser diferente porem é essencial o uso do mesmo SO, para diminuir complexidade de configuração e manutenção.

Middleware: Sistema que permite o controle do cluster em si e, portanto, está intimamente ligado ao sistema operacional. É o middleware que lida, por exemplo, com as bibliotecas que fazem toda a comunicação do cluster.

**- Alto desempenho:** direcionado a aplicações exisgentes no que diz respeito a processamento.

- **Alta disponibilidade:** manter a aplicação em funcionamento.

**- Balanceamento de carga:** tarefas de processamento tem distribuição o mais uniforme possível entre os nós.

Beowulf: Não é middleware, é um padrão de clustering disponibilizado pela NASA. Front-end (nó mestre, pode existir mais de um) o restante é back-end. Comunicação pode ser via ethernet. Sem necessidade de hw específico. SO open source. Nós com dedicação exclusiva ao cluster. Uso de biblioteca para comunicação entre os nós.

**- Grid: C**oleção de recursos computacionais e de comunicação utilizados para execução de aplicações. Usuário vê o grid como uma entidade única. Compostos por recursos heterogêneos, reunindo desde clusters e supercomputadores até desktops e dispositivos móveis.

➔ Elementos

◆ Nó mestre (eventualmente replicado)

◆ Nós de execução

◆ Nós de submissão

➔ Interligação

◆ Rede pública de larga escala: Internet!

Possui caracteristica de pervasividade. Armazenamento de dados remoto. Distribuição geográfica dos nós.

Middleware para gerenciamento do Grid: Gerencia todo o funcionamento do grid, desde a descoberta, alocação e reserva de recursos, até controle de acesso, detecção de falhas, manutenção de registros de utilização, etc.

**- Cloud:** “Computação em Nuvem é um modelo que permite o acesso de forma onipresente, conveniente e sob demanda a um conjunto de recursos computacionais compartilhados e configuráveis (por exemplo redes, servidores, armazenamento, aplicações e serviços), que podem ser facilmente disponibilizados e liberados com o mínimo esforço de gerenciamento ou interação com o provedor de serviço.”

**Características:**

- Serviço sob demanda.

- Acesso via rede

- Polling de recursos

- Elasticidade

- Controle/medição de serviços

**Camadas:**

- SaaS: software as a service

- PaaS: platform as a service

- IaaS: infra as a service

**Modelos de Implantação**:

- Privada

- Pública

- Comunitária

- Híbrida

Infra: baseadas em data center, uso de técnicas de virtualização.

Desafios: Gargalo de conexão, privacidade de dados, confiabilidade do serviço e padronização.

**- Fog:** A abordagem Fog Computing deve ser entendida como uma redefinição escalar do paradigma de Cloud computing. Está relacionada com as seguinte tecnologias:

❑ IoT : Coisas

❑ IoE : Pessoas, Dados, Processos e Coisas

❑ Redes de sensores: de ambiente e bio-médicos

❑ Redes de alto desempenho: SDN

❑ Abordagens de armazenamendo de alto desempenho;

❑ Paradigmas de orientação a contexto: volume de dados e segurança.

Fog Computing é um novo paradigma computacional?

❑ Não: se considerarmos com um ambiente de pré-processamento para as Clouds;

❑ SIM: se considerarmos a multidisciplinaridade envolvida nos ambientes de IoT e IoE.

**Sistemas de Arquivos Distribuidos (SAD):** Permitem que processos armazenem e recuperem arquivos remotamente, por meio de uma rede de interconexão, exatamente como fariam em dispositivos locais.

**Conceitos básicos:**

**- Semântica de compartilhamento:**

Semântica UNIX: toda operação é instantaneamente visível para todos os processos;

Semântica de sessão: nenhuma alteração é visível para outros processos até o arquivo ser fechado;

Arquivos imutáveis: nenhuma alteração é possível;

Transações: todas as alterações acontecem atomicamente.

- **Bloqueios em arquivos:**

Locks, monitores, semáforos.

**Transparências:**

- Acesso

- Localização

- Mobilidade

- Desempenho

- Escalabilidade

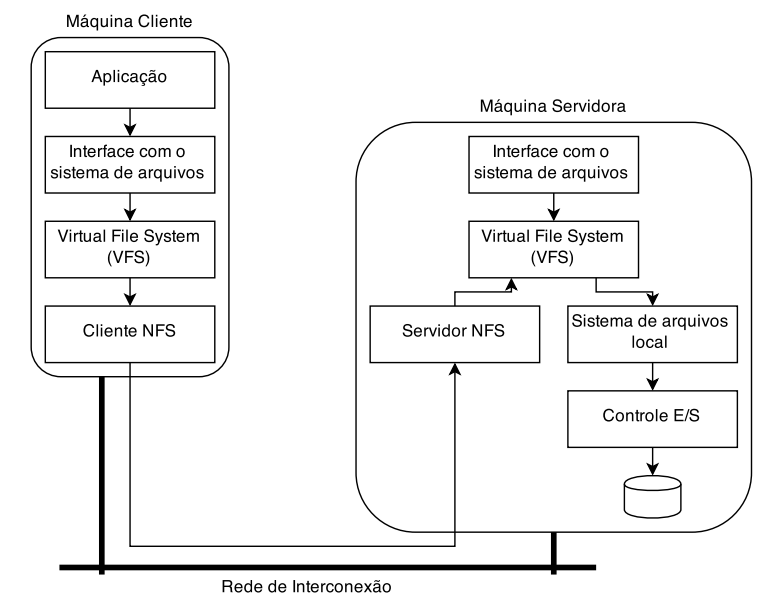
**Arquitetura:**

Cliente-servidor

Baseado em cluster

Simétricos

**NFS (Network File System):** Protocolo de acesso.

 - Comunicação cliente/servidor via RPC

- Espaço de nomes e montagem

- Caching no lado cliente

- Prós:

↑ Simplificação dos protocolos de consistência

↑ Possibilita a utilização em diferentes aplicações

- Contras:

↓ Centralização de carga

↓ Não é "excelente" para nenhuma aplicação em específico

**Sistemas de Arquivos Paralelos (SAP):**

Arquitetura baseada em cluster.

Projetado para paralelismo

Acesso concorrente de muitos processos

Projetado para alto desempenho

Operação sobre redes de interconexão de alta velocidade

Otimizações de E/S para máximo desempenho.

**Servidor de dados:**

**Servidor de metadados** (centralizado(lustre)/distribuido(orangefs)):

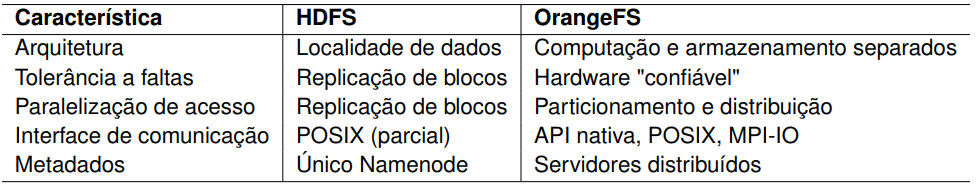
**Cliente(nodo de computação):** interação com o SAP, uso de APIs.

File Striping: Particionamento de um arquivo em diversos servidores, visando ganho de desempenho no acesso paralelo ao arquivo. (ruim para arquivos muito pequenos).

**- Horizontal (ou round robin):** faixas distribuídas entre todos os servidores de dados (máximo de paralelismo)

**- Vertical:** todas as faixas de um arquivo em um mesmo servidor de dados (sem paralelismo)

**- Bidirecional:** híbrido dos anteriores; faixas são distribuídas entre um subconjunto de servidores de dados.



**Relógios em SDs:**

- Tempo é importante em SDs:

- Controle de concorrência com ordenação por timestamp;

- Manter a consistência de dados distribuídos.

- Entretanto, controlar o tempo em um SD é díficil...

Relógios em computadores: Dispositivos eletrônicos que contam as oscilações que ocorrem em um cristal.

- Frequência de oscilação bem definida.

- Contagem e armazenamento do resultado em um registrador contador.

- Podem ser programados para gerar interrupções em intervalos regulares para que, por exemplo, possa ser implementada a fração de tempo de execução.

Para sistemas distribuídos, não existe um tempo global absoluto ao qual possamos recorrer.

**Sincronização:**

O que queremos:

Ordenar os eventos que ocorrem em um SD, utilizar **timestamps** = hora + data;

Como sincronizar os relógios dos processos de um SD?

Poderíamos setar todos para o mesmo horário, porém divergem ao longo do tempo mesmo que correções sejam aplicadas.

Cada relógio possui um **drift rate.**

**Clock Skew:**

Diferença entre a leitura simultânea de dois relógios.

Exemplo: Skew num instante t qualquer:

– Clock 1 = 10:00:00.005

– Clock 2 = 9:00:58.000

– Skew = 2.005

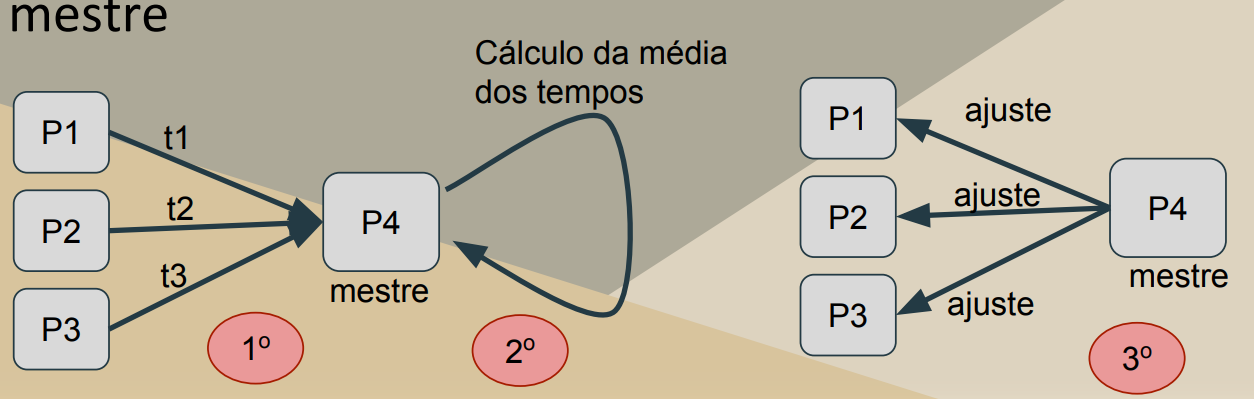
**UTC:** Tempo Universal Coordenado (usado como referência).

**Sincronização de Relógios Físicos:**

Dois tipos de sincronização:

**Interna:** são conhecidos os limites da taxa de deriva de relógios, atraso máximo de transmissão de mensagens e o tempo que leva para executar cada etapa de um processo. **T = t + (Tmax - Tmin) / 2.**

**Algoritmo de Berkeley (Sistemas assíncronos): RTT = 10**



**Externa:** sincronização via autoridade externa.

Para um limite de sincronização D > 0 e para uma fonte S de tempo UTC, a seguinte relação é verdadeira:

|S(t) – Ci (t)| < D para i = 1,2,...N onde

t: tempo físico/real

S: tempo dado por um servidor UTC

C: clock do i-ésimo computador de um SD

D: limite de sincronização

*(Os relógios Ci são precisos dentro do limite D.)*

**Método de Cristian (SD assíncrono):**

Uso de um servidor de tempo, conectado a um dispositivo que recebe sinais de uma fonte UTC, para sincronizar computadores externamente.

Ao receber uma requisição, o processo servidor S fornece o tempo, de acordo com seu relógio.

Cliente seta o clock em t + RTT/2. (**RTT - Round Trip Time (envio+recebimento)**)

Precisão: ±(RTT/2 – Tmin)

Quanto mais o RTT se aproxima Tmin, MAIOR será a precisão

Para obter um RTT mínimo pode-se fazer diversos pedidos ao servidor

– Problemas do algoritmo de Cristian

● Uso de apenas um servidor: se o servidor morre impossível de sincronizar.

● Solução: utilizar vários servidores de tempo, clientes fazem multicast do pedido e guardam a primeira resposta.

● Outros problemas: Servidores intrusos podem difundir um horário falso.

Dois tipos de Sistemas Distrbuídos:

Síncronos

Assíncronos

**NTP (Network Time Protocol):**

– Método de Christian e alg. de Berkeley são utilizados principalmente em intranets enquanto que NTP é um serviço para Internet.

– Permite aos clientes sincronizarem com uma fonte UTC.

– Utilizável em sistemas assíncronos e sincronização externa.

– É confiável: servidores redundantes e caminhos redundantes entre servidores.

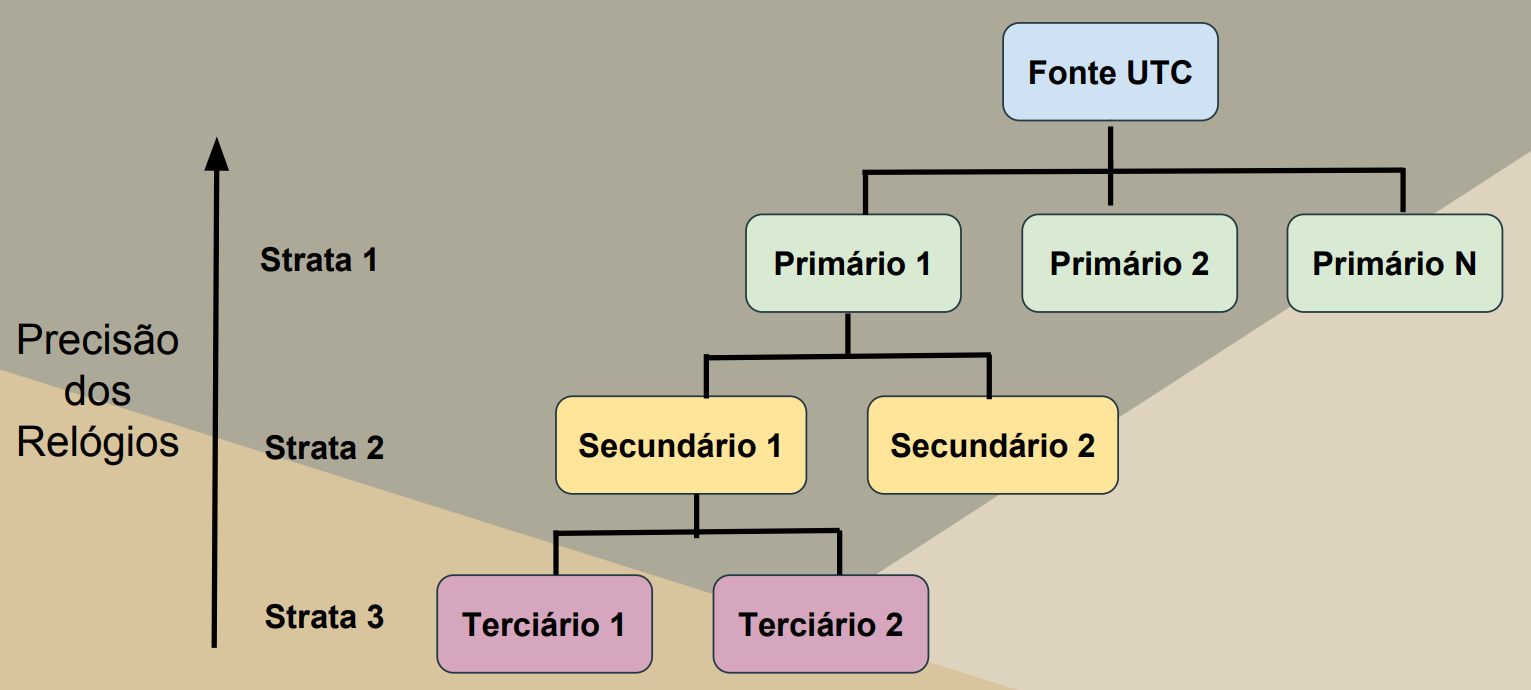
– Atende um grande número de clientes.

– É seguro: técnicas de autenticação para verificar se os dados provém de uma origem conhecida.

Funciona em camadas ou strtatum:

Cada camada contém vários servidores.

Subrede de sincronização pode se reconfigurar quando um servidor cai ou ocorre uma falha.



Sincronização:

multicast: envio periódico do tempo para sincronização, precisão relatiamente baixa

solicitação: quando precisao maior que o multicast necessária, servidor responde a pedidos.

Simétrico: usado para sincronização entre servidores de tempo (baixo stratum), que requer altíssima precisão. servidores trocam mensagens e mantém registros dos valores medidos/ajustados ao longo do tempo (associação) para melhorar cada vez mais a precisão.

