**O que é um Sistema Distribuído?**

"Um sistema distribuído é um conjunto de computadores independentes entre si que se apresenta a seus usuários como um sistema único e coerente."

"Sistemas onde componentes de hardware e / ou software localizados em computadores em rede comunicam-se e coordenam suas ações através somente de troca de mensagens."

"Coleção de entidades independentes que colaboram para resolver um problema que não poderia ser resolvido individualmente."

“Você sabe que existe um sistema distribuído quando a falha de um computador que você nunca ouviu falar não impede que você faça o seu trabalho.”

**Vantagens de um Sistema Distribuído:**

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

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

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

**Confiabilidade:** se uma máquina falhar, o sistema como um todo pode ainda continuar executando.

**Crescimento incremental:** poder computacional pode ser adicionado posteriormente e / ou a qualquer momento.

Sistemas Computacionais Distribuídos (SDs) são implementados, em sua maioria, através de uma ou mais camadas de software denominadas **Middlewares**.

**Middleware:**

Situado logicamente entre uma camada de nível mais alto, composta de usuários e aplicações, e uma camada subjacente, que consiste em sistemas operacionais (SOs) e facilidades básicas de comunicação.

**Características dos Middlewares:**

**Ocultar a distribuição:** A aplicação é executada em diferentes máquinas distribuídas geograficamente.

**Ocultar heterogeneidade:** Diferentes sistemas operacionais e protocolos de comunicação.

**Metas de um Sistema Distribuído (SD):**

**Meta I - Acesso a recursos:**

Um SD deve fornecer fácil acesso a seus recursos: Deve facilitar aos usuários, e às aplicações, o acesso a recursos remotos (hardware e software) e o seu compartilhamento de maneira controlada e eficiente. Desafio: Garantir segurança de acesso.

**Meta II - Transparência da Distribuição:**

Deve ocultar o fato de que os recursos estão distribuídos em uma rede. Implica em transparência de distribuição de:

* **Acesso:** Oculta diferenças na representação de dados e no modo de acesso a um recurso.
* **Localização:** Oculta o lugar em que um recurso está localizado.
* **Migração:** Oculta que um recurso pode ser movido para outra localização.
* **Relocação:** Oculta que um recurso pode ser movido para outra localização enquanto em uso.
* **Replicação:**Oculta que um recurso é replicado.
* **Concorrência:** Oculta que um recurso pode ser compartilhado por vários usuários concorrentes.
* **Falha:** Oculta a falha e a recuperação de um recurso.

**Meta III - Abertura**

Um SD aberto oferece serviços de acordo com regras padronizadas que descrevem a sintaxe e a semântica desses serviços. Os serviços em um SD devem ser especificados por meio de Interfaces.

**Interfaces:** -> Linguagem de Descrição de Interfaces - IDL -> Capturam apenas a sintaxe de serviços . Funções disponíveis, parâmetros e valores de retorno

**Interoperabilidade:**Caracteriza se dois ou mais sistemas / componentes diferentes devem ou podem coexistir e trabalhar em conjunto no mesmo SD.

**Portabilidade:**Define se uma ou mais aplicações desenvolvidas para um SD específico podem ser executadas, sem modificações, em um outro SD que tenha as mesmas interfaces desenvolvidas.

**Meta IV - Escalabilidade**

Um SD deve ter a possibilidade de ser expandido em número de recursos e de usuários, mantendo a facilidade de gerenciamento técnico e administrativo.

**Escalabilidade - Desafios e Soluções**

**Centralização de:**

* Servidores **Solução:** Replicação.
* Dados **Solução:** Distribuição (DNS).
* Algoritmos **Solução:** Algoritmos Distribuídos.

**Expansão geográfica**:

* Retardo para propagação das informações:
  + Replicação e posicionamento de servidores em posições estratégicas.
* Não confiabilidade na rede de conexão:
  + Uso de algoritmos com mecanismos de reconhecimento / confirmações e retransmissões.

**Gerenciamento Administrativo:** Políticas de utilização de recursos e pagamentos. **Gerenciamento e Segurança** Soluções: envolvem aspectos / questões financeiras, leis locais, entre outros fatores.

**Modelos de Sistemas Distribuidos:**

Um Sistema Distribuído pode ser estudado / discutido / descrito sob três importantes e complementares formas:

**1. Modelos Físicos**

**2. Modelos Arquiteturais**

**3. Modelos Fundamentais**

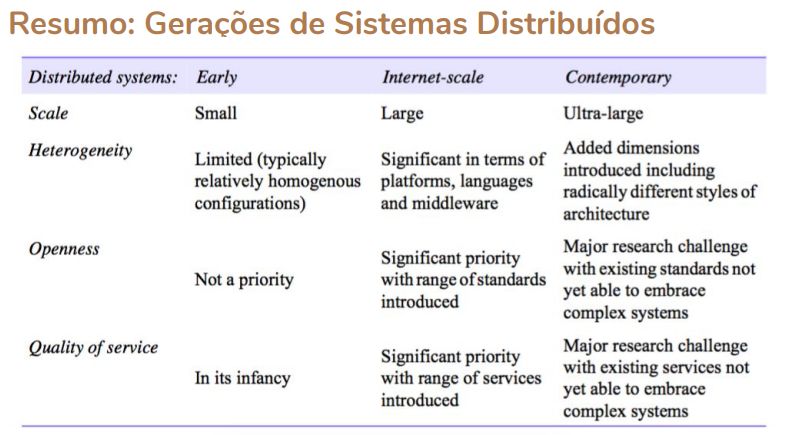
**Modelos Físicos:** Consideram os tipos de computadores e dispositivos que constituem um sistema e sua conectividade, sem detalhes de tecnologias específicas.

**Modelo Físico de Base:** “Aquele em que componentes de hardware ou software localizados em computadores em rede se comunicam e coordenam suas ações apenas passando mensagens.”

Este conceito leva a um modelo físico mínimo de um sistema distribuído como um conjunto extensível de nós de computador interligados por uma rede de transmissão de dados para a passagem de mensagens.

Além deste modelo de base, podemos identificar três gerações de sistemas distribuídos: . Primeiros Sistemas Distribuídos (final década 70 início dos anos 80); . Sistemas Distribuídos na escala da Internet (início do anos 90); . Sistemas Distribuídos recentes / emergentes.

**Sistemas distribuídos de ultra-grande escala (ULS)** sistemas complexos consistindo em uma série de subsistemas que são sistemas computacionais autônomos e que se reúnem para executar uma determinada tarefa ou tarefas.



**Modelos Arquiteturais:** Descrevem um sistema em termos de tarefas computacionais e de comunicação realizadas por seus elementos computacionais. Os elementos computacionais são computadores individuais ou agregados deles suportados por interconexões de rede adequadas. Cliente-servidor e ponto-a-ponto são duas das formas mais utilizadas de modelo arquitetural para sistemas distribuídos.

A arquitetura de um sistema distribuído se refere à sua estrutura em termos de componentes especificamente separados e suas inter-relações.

**Objetivo geral:** Garantir que a estrutura atenda às presentes e prováveis futuras demandas de SDs.

**Análise em três etapas:**

**Núcleo dos elementos arquiteturais** subjacentes que sustentam sistemas modernos distribuídos, destacando a diversidade de abordagens que agora existem;

**Padrões arquiteturais compostos** que podem ser usados isoladamente ou, mais comumente, em combinação, no desenvolvimento de soluções de SDs mais sofisticados;

**Plataformas de middleware** que estão disponíveis para suportar os vários estilos de programação que emergem dos estilos arquitetônicos acima.

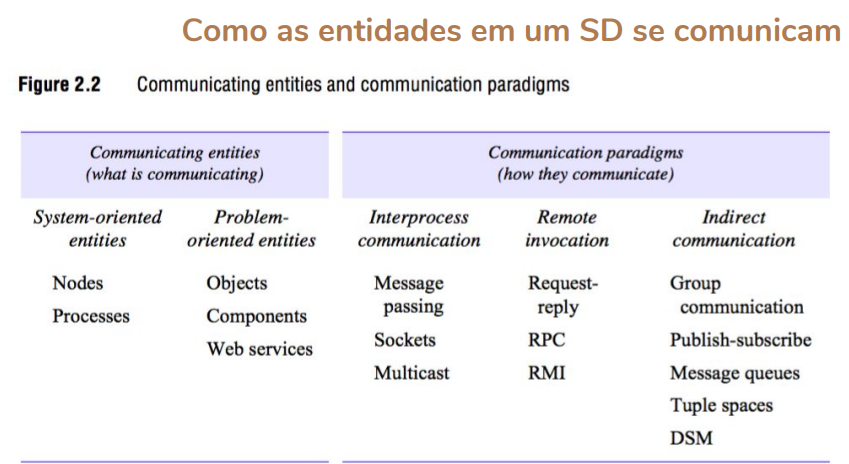
**Para entender os elementos fundamentais de um SD**, é necessário considerar quatro questões chaves:

Quais são as entidades que estão se comunicando no SD?

* Processos ?
* Threads?
* Nenhuma abstração (sistemas mais simples, como redes de sensores, onde SOs usados não oferecem suporte para estas abstrações).
* Objetos?
* Componentes?
* Web Services?

Como elas se comunicam, ou melhor, qual paradigma de comunicação é usado?

* Comunicação inter-processo; .
* Invocação remota de procedimentos; .
* Comunicação Indireta: comunicação em grupo, publish-subscribe, filas de mensagens, espaço de tuplas e memória compartilhada distribuída.



Quais papéis e responsabilidades estas entidades possuem em toda a arquitetura distribuída?

Em um sistema distribuído, processos - ou mesmo objetos, componentes ou serviços, incluindo serviços da web - interagem uns com os outros para realizar uma atividade útil, por exemplo, para oferecer suporte a uma sessão de bate-papo.

Ao fazê-lo, os processos assumem funções determinadas (papéis), e esses papéis são fundamentais para estabelecer a arquitetura geral a ser adotada em um SD. Dois estilos arquiteturais, decorrentes do papel dos processos individuais, podem ser descritos: **Cliente-servidor e Peer-to-peer**

Como estas entidades são mapeadas para a infra estrutura distribuída física?

A questão de onde colocar um determinado cliente ou servidor em termos de máquinas e processos dentro das máquinas é uma importante questão de projeto de SD. O mapeamento precisa ter em conta:

* padrões de comunicação entre entidades,
* a confiabilidade de máquinas e o carregamento atual,
* a qualidade da comunicação entre diferentes máquinas, etc.

O mapeamento deve ser determinado com um forte conhecimento da aplicação, e existem poucas orientações universais para obter uma solução ótima. Estratégias de mapeamento podem alterar significativamente as características de um determinado projeto de SD. Algumas estratégias são:

* Mapeamento de serviços para múltiplos servidores.
* Caching
* Código Móvel
* Agentes Móveis

**Modelos Fundamentais:** Abordam uma perspectiva abstrata para descrever soluções para problemas enfrentados pela maioria dos sistemas distribuídos. Pode-se organizá-las em:

**Modelos de Interação:** SDs são compostos por inúmeras entidades computacionais (processos, threads, componentes, etc) que se comunicam (interagem) também de forma complexa. Dois fatores significativos afetam a interação entre entidades em um SDs:

* **Desempenho de comunicação:**
* **Latência:** O tempo necessário para o primeiro bit, de uma série de bits transmitidos através de uma rede, alcançar seu destino.
* **Largura de banda:** Quantidade total de informações que podem ser transmitidas sobre ele em um determinado momento. Quando um grande número de canais de comunicação estão usando a mesma rede, eles têm que compartilhar a largura de banda disponível.
* **Jitter:** Variação no tempo necessário para entregar uma série de mensagens. O Jitter é relevante para dados multimídia. Por exemplo, se amostras consecutivas de dados de áudio forem reproduzidas com diferentes intervalos de tempo, o som será gravemente distorcido.
* **Noção de tempo único no sistema:**
  + **Clock drift rate** Relógios de computadores derivam do tempo perfeito e, mais importante, suas taxas de deriva diferem umas das outras.

Clock drift rate refere-se à taxa na qual um relógio do computador se desvia de um relógio de referência perfeito. Mesmo que os relógios em todos os computadores de um sistema distribuído estejam configurados ao mesmo tempo inicialmente, seus relógios irão eventualmente variar de forma bastante significativa, a menos que correções sejam aplicadas.

* Há várias formas de corrigir o tempo em um SD. Uso do tempo lido de um GPS. Uso de variações dos modelos de Interação: .
* Sistemas Distribuídos Sincronos; .
* Sistemas Distrbuídos Assíncronos.

**Modelos de Falhas:**

Modelos para prevenir e superar falhas

Modelos para garantir segurança dos dados.

**Modelos de Segurança**.

**Sistemas de computação distribuı́dos .**

**Computação em cluster**

* Os supercomputadores tradicionais foram construídos por um pequeno número de fabricantes, com um alto orçamento destinado ao projeto. . Muitas universidades não podem arcar com os custos de um supercomputador, então o uso de clusters se torna um alternativa interessante. . Com o uso de hardware mais barato e disponível no mercado, sistemas com desempenho similar aos supercomputadores podem ser construídos.

**Conceito:** Um cluster é um 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.

**Características de um cluster:** Rede mais rápida e próxima do que uma rede local; Protocolos de comunicação de baixa latência; Conexão mais frouxa que um SMP (Symmetric multiprocessing).

**Exemplos de Aplicações que exigem uso de um cluster** . Sistemas meteorológicos . Ferramentas de mapeamento genético . Simuladores geotérmicos . Programas de renderização de imagens tridimensionais, entre outros.

**Vantagens de uso de clusters:** Relação custo-benefício: -> Pode-se obter resultados tão bons quanto ou até superiores que um servidor sofisticado a partir de máquinas mais simples e mais baratas.

Relação custo-benefício: -> É possível aumentar a capacidade de um cluster com a adição de nós ou remover máquinas para reparos sem interromper a aplicação. -> Relativa facilidade de customização para o perfeito atendimento da aplicação. Há opções de softwares para cluster disponíveis livremente, o que facilita o uso desta solução

**Desvantagens:** A facilidade de expansão do cluster possui alguns efeitos colaterais: a quantidade de máquinas pode aumentar tanto que a manutenção se torna mais trabalhosa, o espaço físico pode ficar impróprio, etc.

A tecnologia de comunicação utilizada pode não oferecer a velocidade de transferência de dados ou o tempo de resposta necessário, dependendo da aplicação. . Cluster tem como base uma rede local, logo, não se pode acrescentar máquinas que estejam muito distantes geograficamente.

**Tipos de Clusters** .

**Clusters de Alto Desempenho**:

Direcionados para aplicações bastante exigentes no que diz respeito ao processamento. **Exemplos:** Sistemas utilizados em pesquisas científicas. **Objetivo:** Fornecer resultados satisfatórios em tempo hábil, mesmo que haja centenas de milhares de gigaflops envolvidos com a tarefa.

As máquinas são configuradas especialmente para oferecer o maior desempenho possível. . Estes tipos de clusters também tem algumas funcionalidades para balanceamento de carga, já que eles tentam espalhar os processos por máquinas diferentes para obter maior desempenho.

**Clusters de Alta Disponibilidade:**

**Objetivo:** Manter a aplicação em funcionamento. **Exemplos:** Sistemas metroviários de grandes cidades, sistemas bancários, sistemas web de varejo.

Para atender a esta exigência, os clusters de alta disponibilidade podem contar com diversos recursos:

-> Ferramentas de monitoramento que identificam nós defeituosos ou falhas na conexão.

-> Replicação (redundância) de sistemas e computadores para substituição imediata de máquinas com problemas.

-> Uso de geradores para garantir o funcionamento em caso de queda de energia ou problemas relacionados.

**Clusters para Balanceamento de Carga:**

As tarefas de processamento são distribuídas o mais uniformemente possível entre os nós. Os nós devem ser monitorados constantemente para verificar, por exemplo, qual máquina está executando a menor quantidade de tarefas e direcionar uma nova requisição para esta. **Exemplos:** Sites de varejo

->Aumento instantâneo de rajadas de requisições;

**Combinação de Tipos de Clusters:**

Uma aplicação computacional não precisa estar presa a apenas um tipo de cluster. Conforme a necessidade, pode-se combinar características de tipos diferentes no intuito de atender plenamente à aplicação. Exemplo: site de varejo pode utilizar um cluster de alta disponibilidade para garantir que suas vendas possam ser realizadas 24 horas por dia e, ao mesmo tempo, aplicar balanceamento de carga para suportar um expressivo aumento eventual no número de pedidos causados por uma promoção.

**Estrutura Usual de Clusters**

**Máquinas**: Uso de máquinas específicas construidas para serem utilizadas como nós de um cluster. Uso de máquinas (PCs) convencionais.

**Comunicação entre os Nodos de um Cluster**: Realizada a partir de uma tecnologia de rede local. Os padrões Ethernet (Gigabit Ethernet, Fast Ethernet, e outras) são bastante utilizados justamente por serem mais comuns e, portanto, melhor suportados e menos custosos. Outras opções viáveis, entre elas, o Myrinet e o InfiniBand, ambos com características bastante apropriadas para clustering.Outras Redes de Interconexão : Ethernet 10 Gbps . Quadrics QSNet . Mellanox Infiniband (10 Gbps) . SCI (Dolphin – MPI – 12s latência) . ATM . Digital Memory Channel . FDDI

**Tipo de Nodos: dedicados x não dedicados:**

**Dedicados**: O nó é utilizado somente para este fim, fazendo com que dispositivos como teclados e monitores sejam dispensáveis, se, por algum motivo, for necessário acessar uma máquina em particular, pode-se fazê-lo via terminal, a partir do nó principal, por exemplo.

**Não dedicados:** Cada computador que faz parte do cluster não trabalha exclusivamente nele. Uso de máquinas convencionais (PCs).

**Sistema Operacional:** Os nós de um cluster não precisam ser exatamente iguais no que diz respeito ao hardware, mas é essencial que todos os computadores utilizam o mesmo sistema operacional

A homogeneidade do SO é importante para:

-> Diminuir a complexidade de configuração e manutenção do sistema.

->Garantir que os procedimentos rotineiros ao cluster, como monitoramento, distribuição de tarefas e controle de recursos sejam executados de maneira uniforme.

Na formação de um cluster computacional, a configuração do software do ambiente é um elemento diferencial para seu desempenho e deve incluir um sistema de imagem única (SSI – Single System Image) do cluster. Um SSI pode ser implementado pelo sistema operacional (SO) ou por middleware.

Clusters apoiados por SOs suportam a execução eficiente de aplicações paralelas. O SO deve suportar:

->Escalonamento de programas paralelos.

->Identificação de recursos do sistema (processadores, memória, redes) e oferecer acesso à eles.

Um exemplo de SSI implementado em SO é o UnixWare NonStop Clusters, uma extensão do SO UnixWare. Utiliza o protocolo IP para interconexão, removendo a necessidade de hardware proprietário. O kernel foi modificado por extensões modulares para viabilizar:

->visão ampla do sistema de arquivos

->compartilhamento transparente do espaço de paginação

->comunicação de alta performance

->transparência na migração de processos e

->balanceamento de carga.

**Exemplos:** . Mosix . OpenMosix . OpenSSI . Kerrighed. Solaris MC . Unixware . Rocks

**Middleware de um Cluster:** 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 - uma delas é o padrão MPI (Message Passing Interface).

Deve oferecer uma interface para: Configuração, Manutenção, Otimização e Monitoramento do Cluster. Por padrão, ele é instalado em um nó controlador (ou nó mestre).

SSI implementado em middleware incluem: .

->Gerenciador de arquivos (ex: PVM)

->Gerenciamento de tarefas

->Sistemas de escalonamento (CODINE)

->Clusters baseados em máquinas virtuais Java (ex: JESSICA)

O **CODINE** foi desenvolvido a partir da criação do Distributed Queuing System (DQS). . Consiste em um recurso de gerenciamento que objetiva otimizar a utilização de software e hardware em uma rede heterogênea. . Possui interface gráfica e simplifica a administração e configuração de tarefas. . O sistema engloba 4 tipos de daemons: master, escalonador, de comunicação, e execução.

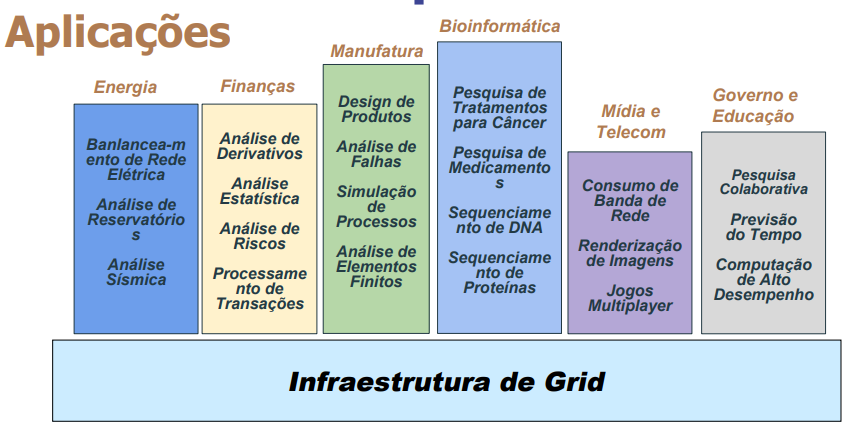
**Bibliotecas de Comunicação para um Cluster** As facilidades tradicionais são suportadas (mas são custosas devido ao protocolo de processamento): Soquetes (TCP/IP), Pipes, etc. Protocolos mais leves são mais adequados (Comunicação no nível de usuário): Active Messages (AM) (Berkeley); Fast Messages (Illinois); U-net (Cornell); XTP (Virginia); Virtual Interface Architecture (VIA)

**Ambientes de Programação para Clusters**

Threads . POSIX Threads . Java Threads . MPI http://www-unix.mcs.anl.gov/mpi/ . PVM http://www.epm.ornl.gov/pvm . Software DSMs (SHMEM da Cray/SGI)

* **Computação em grade**

**Definição:** Um grid computacional é uma coleção de recursos computacionais e de comunicação utilizados para execução de aplicações. . Usuário vê o grid como uma entidade única.

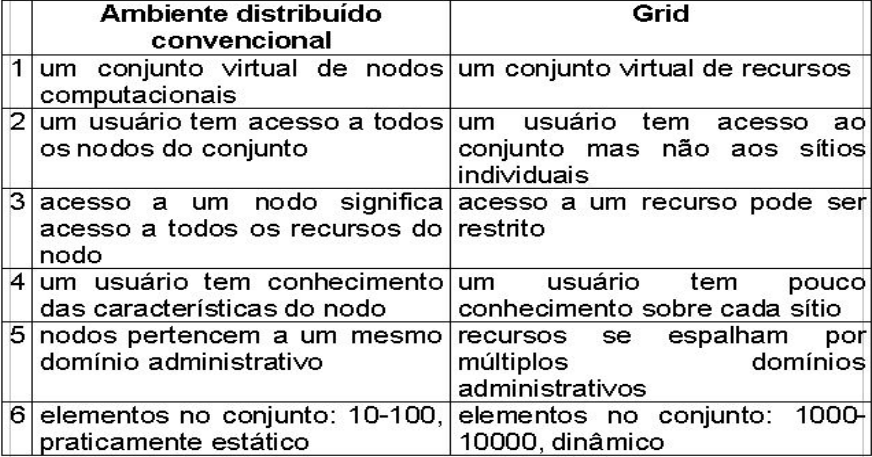


**Composição:** Grids são compostos por recursos heterogêneos, reunindo desde clusters e supercomputadores até desktops e dispositivos móveis

**Elementos:**

* Nó mestre (eventualmente replicado)
* Nós de execução
* Nós de submissão

**Interligação:** Rede pública de larga escala: Internet!



**Características**

* **Pervasividade:**

->O grid está disponível em todo lugar; basta plugar e usar (assim como a rede elétrica).

->Visão uniforme dos recursos computacionais.

->O usuário executa uma aplicação no grid, sem saber quais serão os nós envolvidos.

->Armazenamento remoto de dados.

->Dados do usuário podem ser armazenados de modo transparente e confiável no grid.

-> Abordagem alternativa para solução de problemas / execução de aplicações.

->O uso de recursos remotos é a regra, e não a exceção. .

->Computação distribuída de larga escala:

->É comum que os nós estejam distribuídos em diferentes países

* **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. . Fundamentado em tecnologias já existentes e amplamente disponíveis:

->Comunicação: protocolos da Internet / Web

->Software: tecnologia de Web Services

**Produtos:**

**Globus Toolkit** . Baseado nos seguintes padrões e tecnologias: .. Open Grid Services Architecture (OGSA) .. Web Services (SOAP, WSDL, Web Service Resource Framework, WS-Management) .. Job Submission Description Language (JSDL) .. Distributed Resource Management Application API (DRMAA), Grid Security Infrastructure (GSI)

**Oracle Grid Engine**

**Xmlqstat**

**Exemplos:**

Iniciativas Mundiais ■ Large Hadron Collider (LHC) Computing Grid

Folding@Home

Network for Earthquake Engineering Simulation (NEES)

Cancer Biomedical Informatics Grid (caBIG)

World Community Grid

* **Computação em nuvem**

**Definição:** “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:**

* Usuário solicita os recursos que deseja ..
* Obtenção simplificada de recursos, sem interação humana .

**Acesso via rede:**

* Recursos podem ser acessados remotamente usando diversos tipos de dispositivo .

**Pooling de recursos:**

* Recursos são compartilhados entre diversos usuários e alocados com base na demanda

**Elasticidade:**

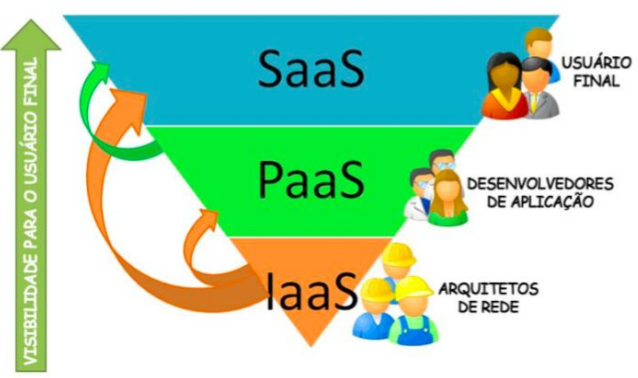
* A quantidade de recursos alocados para um usuário pode variar em função da demanda. .

**Medição de Serviços:**

* O provedor deve monitorar a quantidade de recursos utilizada pelos clientes da nuvem, para fins de controle e tarifação.

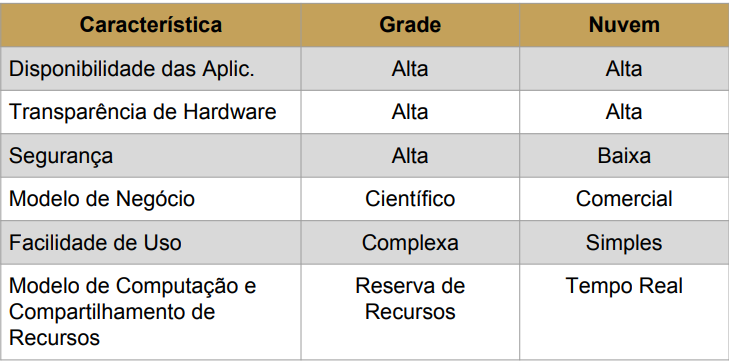
**Camadas:**

* **SaaS (Software as a Service)**
  + Provê acesso a softwares aplicativos
  + Acesso via navegador ou programa cliente
* **PaaS (Platform as a Service)**
  + Consumidor tem disponível uma plataforma computacional gerenciada por um S.O.
  + Gerencia softwares disponibilizados .
* **IaaS (Infrastructure as a Service)**
  + Disponibiliza infraestrutura de hardware
  + Cliente gerencia S.O. e aplicações



**Modelos de Implantação**

* **Nuvem Privada:** de uso exclusivo das unidades de negócio de uma corporação
  + Nuvens operadas exclusivamente para uma única organização.
  + Gerenciamento da rede é feito pela própria organização ou por terceiros
    - A infraestrutura utilizada pertence ao usuário
    - Responsável pelo controle sobre a implementação das aplicações na nuvem
* **Nuvem Pública:** disponível para uso do público em geral.
  + A nuvem é disponibilizada para o público em geral ou para grandes grupos industriais.
  + A nuvem é implementada por um prestador de serviço, que deve ser capaz de garantir o desempenho e a segurança do mesmo.
* **Nuvem Comunitária:** acessível a usuários de organizações que possuem interesses comuns.
  + Infraestrutura de nuvem é compartilhada por várias organizações e suporta uma comunidade específica que partilha das mesmas preocupações, missão, requisito de segurança, política e considerações de conformidade
  + Gerenciado pelas organizações ou por terceiro, localmente ou remotamente
* **Nuvem Híbrida:** composição de infraestruturas de nuvem de diferentes tipos, reunidas para obter um melhor aproveitamento de recursos e permitir portabilidade de dados e aplicações.
  + Infraestrutura composta por dois ou mais modelos de implementação
  + Cada nuvem permanece como uma entidade única
  + Unidas pelo uso de tecnologia proprietária ou padronizada garantindo a portabilidade de dados e aplicações.
* **Infraestrutura:**
  + Baseadas em Datacenters.
  + Uso extensivo da técnica de virtualização.
* **Exemplos:**
  + **Amazon Elastic Compute Cloud (EC2)** . Permite alocação de recursos sob demanda (por hora), reservadas (anual ou por 3 anos) ou pontual (quando houver ociosidade) . Instâncias de VMs com várias capacidades . Grande variedade de sistemas e aplicações . Presente fisicamente em oito locais . Política de preços baseada na alocação de recursos, taxa de transferência de dados, etc
  + **Google Cloud Platform** . Cliente pode usar o Google App Engine para desenvolver suas aplicações e hospedar na infraestrutura de nuvem do Google . Integrado com contas de usuários do Google . Cliente pode alugar VMs com diferentes configurações de recursos . Tarifação por VM, aplicação, tráfego, uso de endereço IP, requisição HTTP, etc.
  + **Windows Azure**
* **Desafios: .** 
  + **Gargalo na conexão**
  + **Privacidade e direito sobre os dados**
  + **Confiabilidade do serviço**
  + **Padronização**



* **Computação em névoa**
  + **Motivação:**

Estudos recentes de grandes corporações de TI, como por exemplo, Cisco [1] e IBM [2], indicam que diariamente são gerados 2.5 exabytes de bytes de dados por dia. Estima-se que em 2020 esse número atinja a casa dos 40 yottabytes, o que significa aproximadamente 5.200 gigabytes para cada pessoa na terra.

Esta grande quantidade de informação deverá ter uma parcela majoritariamente originada da Internet das Coisas (Internet of Things - IoT).

**Definição:** A Internet das Coisas (Internet of Things - IoT) é uma abordagem tecnológica que visa conectar dispositivos eletrônicos utilizados no dia-a-dia, tais como os como aparelhos eletrodomésticos, eletroportáteis, máquinas industriais, meios de transporte à Internet. IoT é um sistema onde dispositivos são interligados computacionalmente, onde máquinas mecânicas e digitais, objetos, animais e / ou pessoas possuem identificadores únicos e a capacidade de disponibilizar / transferir dados através de uma rede.



**A abordagem Fog Computing deve ser entendida como uma redefinição escalar do paradigma de Cloud computing.**

A abordagem está relacionada com algumas tecnologias:

❑ IoT : Coisas

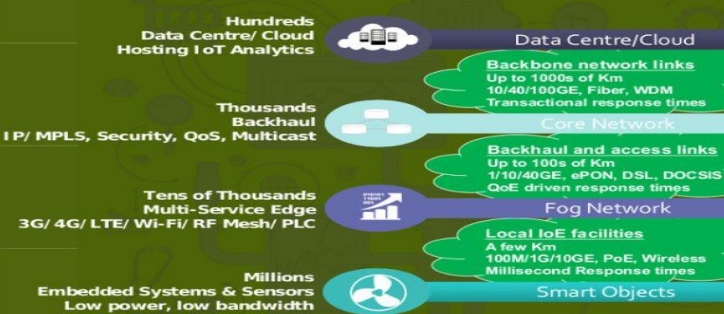
❑ IoE : Pessoas, Dados, Processos e Coisas

❑ Redes de sensores: de ambiente e bio-médicos

❑ Redes de alto desempenho: SDN

❑ Abordagens de armazenamento de alto desempenho;

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



**Sistemas Distribuidos Pervasivos**

* **Redes de Sensores Sem Fio**
* **Redes Veiculares**
* **Smart Cities**
* **Internet Of Things (IOT)**

**1. As propriedades e os problemas de design dos sistemas distribuídos podem ser capturados e discutidos através do uso de modelos descritivos. Cada tipo de modelo destina-se a fornecer uma descrição abstrata, simplificada mas consistente, de um aspecto relevante do design do sistema distribuído. Cite e explique os três modelos propostos no livro base da disciplina.**

Modelo Físico é o modo mais explícito de descrever um sistema; ele captura a composição de hardware em termos de computador (e outros dispositivos) e suas redes de interconexão.

Modelos arquiteturais descrevem um sistema em termos computacionais e tarefas de comunicação realizadas pelos seus elementos computacionais; os elementos computacionais sendo computadores individuais ou agregados deles suportados por interconexões de rede.

Os modelos fundamentais adotam uma perspectiva abstrata para examinar aspectos de um sistema distribuído. Três aspectos importantes são:

Modelos de interação, que considera a estrutura e o sequenciamento da comunicação entre os elementos do sistema; Modelos de falha, que consideram as maneiras pelas quais um sistema pode parar de operar corretamente e; Modelos de segurança, que consideram como o sistema é protegido contra tentativas de interferir com seu correto funcionamento ou roubar seus dados.

**2. Quais são as ameaças e dificuldades para sistemas distribuídos relacionadas com os modos de uso amplamente variados, ambientes de sistemas amplamente variados, problemas internos e ameaças externas?**

R:

Modos de uso amplamente variáveis: os componentes dos sistemas estão sujeitos a variações na carga de trabalho - por exemplo, algumas páginas da Web são acessadas vários milhões de vezes por dia. Algumas partes de um sistema podem estar desconectadas ou mal conectadas- por exemplo, quando computadores móveis são incluídos em um sistema. Alguns aplicativos têm requisitos especiais para alta largura de banda de comunicação e baixa latência - por exemplo, aplicativos de multimídia.

Ampla gama de ambientes de sistema: um sistema distribuído deve acomodar hardware heterogêneo, sistemas operacionais e redes. A internet pode diferir amplamente no desempenho - as redes sem fio operam a uma fração da velocidade das redes com fio. Sistemas de escalas muito diferentes, variando de dezenas de computadores a milhões de computadores, devem ser suportados.

Problemas internos: relógios não sincronizados, atualizações de dados conflitantes e muitos modos de falha de hardware e software que envolvem os componentes individuais do sistema.

Ameaças externas: ataques à integridade e sigilo de dados, ataques de negação de serviço.

**3. Descreva um exemplo de sistema distribuídos de sistemas, semelhante ao descrito no texto do capítulo 2 (página 40).**

**4. Para entender os blocos de construção fundamentais de um sistema distribuído, é necessário considerar quatro questões-chave. Quais são elas? Cite e explique cada uma delas.**

**5. Segundo o autor, qual a diferença entre as seguintes entidades computacionais: Objetos, Componentes e Web Services.**

**6. Diferencie: protocolos requisição-resposta, chamadas de procedimento remoto e invocação de métodos remotos.**

**7. Descreva as técnicas de comunicação indireta.**

**8. Quais as características de um modelo cliente-servidor e um modelo peer-to-peer?**

**9. Liste os tipos de recursos locais que são vulneráveis ​​a um ataque por um programa não confiável que é baixado de um site remoto e executado em um computador local. página 50**

**10. Dê exemplos de aplicações onde o uso do código móvel é benéfico. página 50**

**11. Caracterize sistemas distribuídos síncronos e assíncronos.**

**1. Pesquise na Web um exemplo real de cluster formado por componentes COTS (Commercial Off-The-Shelf). Pode ser algum cluster usado em um laboratório de universidade ou de empresas. Tente identificar os elementos computacionais deste cluster: máquinas usadas, tecnologia de comunicação usada entre as máquinas do cluster e middleware / software usados.**

**2. Na formação de um cluster computacional, a configuração de software do ambiente é um**

**elemento diferencial para seu desempenho e deve incluir um sistema de imagem única (SSI –**

**Single System Image) do cluster. Um SSI pode ser implementado pelo SO ou por middleware.**

**Busque um exemplo de SSI implementados em SO e em middleware.**

**3. Diferencie Grids computacionais e clusters.**

**4. Pesquise dois grids computacionais e liste duas principais características.**

**5. Escolha um exemplo de sistema distribuído em nuvem e relacione as características dele**

**com as cinco características fundamentais deste modelo de computação distribuída.**

**6. A Internet das Coisas (IoT – Internet Of Things) é responsável pela criação do conceito de**

**Computação em Névoa? Justifique sua resposta.**

**7. Já existe uma casa na Califórnia na qual diversos objetos e processos estão ligados e**

**conversando entre si. Por exemplo: a porta da garagem está conectada ao GPS do carro e se**

**abre automaticamente quando o motorista se aproxima. A medida em que fecha a garagem,**

**ela já manda um sinal para as luzes da casa se acenderem e para a cafeteira começar a**

**preparar o café. Enquanto a cafeteira prepara o café, o sistema de som detecta mudanças no**

**ambiente e percebe que o morador da casa chegou, sintonizando nas rádios favoritas, ao**

**mesmo tempo que o termostato começa a se ajustar às preferências do morador que chegou.**

**A conexão entre a porta da garagem e a cafeteira é um exemplo de Internet of Things, mas o**

**sistema de som que é iniciado porque as luzes foram acesas é um processo inteligente e que**

**faz parte da Internet of Everything (IoE). É uma ação que incorpora quatro pilares:**

**• Pessoa: quando a casa percebe que seu morador está chegando.**

**• Informação: o GPS do carro forneceu a localização do morador e, a partir disso, a**

**garagem começou a se preparar para abrir sua porta.**

**• Coisas: cafeteiras, portas da garagem, sistemas de som, luzes, termostato: todos estão**

**ligados e trocando informações entre si.**

**• Processos: o acionamento das luzes quando a porta da garagem se fecha, sistemas de**

**som que iniciam quando percebem movimentações no ambiente.**

**Pesquise e cite um outro exemplo de uso da computação em névoa neste contexto de IoT**

**e IoE.**

**1 questão V/F de 1 ponto**

**E a V/F era sobre as 4 metas**

**Meta I - Acesso a recursos:**

Um SD deve fornecer fácil acesso a seus recursos: Deve facilitar aos usuários, e às aplicações, o acesso a recursos remotos (hardware e software) e o seu compartilhamento de maneira controlada e eficiente. Desafio: Garantir segurança de acesso.

**Meta II - Transparência da Distribuição:**

Deve ocultar o fato de que os recursos estão distribuídos em uma rede. Implica em transparência de distribuição de:

* **Acesso:** Oculta diferenças na representação de dados e no modo de acesso a um recurso.
* **Localização:** Oculta o lugar em que um recurso está localizado.
* **Migração:** Oculta que um recurso pode ser movido para outra localização.
* **Relocação:** Oculta que um recurso pode ser movido para outra localização enquanto em uso.
* **Replicação:**Oculta que um recurso é replicado.
* **Concorrência:** Oculta que um recurso pode ser compartilhado por vários usuários concorrentes.
* **Falha:** Oculta a falha e a recuperação de um recurso.

**Meta III - Abertura**

Um SD aberto oferece serviços de acordo com regras padronizadas que descrevem a sintaxe e a semântica desses serviços. Os serviços em um SD devem ser especificados por meio de Interfaces.

**Interfaces:** -> Linguagem de Descrição de Interfaces - IDL -> Capturam apenas a sintaxe de serviços . Funções disponíveis, parâmetros e valores de retorno

**Interoperabilidade:**Caracteriza se dois ou mais sistemas / componentes diferentes devem ou podem coexistir e trabalhar em conjunto no mesmo SD.

**Portabilidade:**Define se uma ou mais aplicações desenvolvidas para um SD específico podem ser executadas, sem modificações, em um outro SD que tenha as mesmas interfaces desenvolvidas.

**Meta IV - Escalabilidade**

Um SD deve ter a possibilidade de ser expandido em número de recursos e de usuários, mantendo a facilidade de gerenciamento técnico e administrativo.

**Descentralização**

**Expansão geográfica**:

* Retardo para propagação das informações:
* Não confiabilidade na rede de conexão:

**Gerenciamento Administrativo:** Políticas de utilização de recursos e pagamentos. **Gerenciamento e Segurança** Soluções: envolvem aspectos / questões financeiras, leis locais, entre outros fatores.

**1 de "errada anula certa" de 2 pontos**

**vantagens de um sistema distribuído**

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

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

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

**Confiabilidade:** se uma máquina falhar, o sistema como um todo pode ainda continuar executando.

**Crescimento incremental:** poder computacional pode ser adicionado posteriormente e / ou a qualquer momento.

**1 de 4 pontos pedindo pra descrever os tipos de sistemas distribuídos e três ou mais características de cada.**

**Cluster:** Um cluster é um 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.

**Características:** Rede mais rápida e próxima do que uma rede local; Protocolos de comunicação de baixa latência; Conexão mais frouxa que um SMP (Symmetric multiprocessing).

**Nuvem:**

**Serviço sob demanda:**

* Usuário solicita os recursos que deseja ..
* Obtenção simplificada de recursos, sem interação humana .

**Acesso via rede:**

* Recursos podem ser acessados remotamente usando diversos tipos de dispositivo .

**Pooling de recursos:**

* Recursos são compartilhados entre diversos usuários e alocados com base na demanda

**Elasticidade:**

* A quantidade de recursos alocados para um usuário pode variar em função da demanda. .

**Medição de Serviços:**

* O provedor deve monitorar a quantidade de recursos utilizada pelos clientes da nuvem, para fins de controle e tarifação.

**Névoa:**

**A abordagem Fog Computing deve ser entendida como uma redefinição escalar do paradigma de Cloud computing.**

**Grid:**

Definição: Um grid computacional é uma coleção de recursos computacionais e de comunicação utilizados para execução de aplicações. . Usuário vê o grid como uma entidade única.

**Características**

**Pervasividade:**

->O grid está disponível em todo lugar; basta plugar e usar (assim como a rede elétrica).

->Visão uniforme dos recursos computacionais.

->O usuário executa uma aplicação no grid, sem saber quais serão os nós envolvidos.

->Armazenamento remoto de dados.

->Dados do usuário podem ser armazenados de modo transparente e confiável no grid.

-> Abordagem alternativa para solução de problemas / execução de aplicações.

->O uso de recursos remotos é a regra, e não a exceção. .

->Computação distribuída de larga escala:

->É comum que os nós estejam distribuídos em diferentes países

**1 de 1 ponto sobre quais as duas formas de escrita de arquivos no sistema Cliente-Servidor**

**1 de 2 pontos sobre como funciona o Google File System**

**Explicar como é feita a escrita em arquivo no gfs (mutação)**

O GFS cluster é um sistema distribuído com um servidor master com informações sobre os arquivos e chunkservers que armazenam os arquivos em si.

Os chunksevers armazenam os chunks. Cada chunkserver é implementado em uma plataforma Linux e os arquivos são salvos como arquivos Linux comuns, logo fica a cargo do sistema Linux as operações de sistemas de arquivos na máquina (como leitura e escrita em disco) possibilitando que o chunk server abstraia estas tarefas.

**Gravação**:

1 - Aplicação faz um pedido de escrita.

2 - O Cliente GFS traduz o pedido (file name, data) para (file name, chunk index) e manda ao master.

3 - O mestre responde com o (chunk handle, primary and secondary replica locations), localizações das réplicas

4 - O cliente envia dados de escrita para todas as localidades. Os dados são armazenados em buffers internos dos chunkservers.

5 - O cliente manda o comando de escrita a réplica primária.

6 - Esta réplica determina a ordem em série das instâncias dos dados armazenados no buffer e escreve nesta ordem no chunk.

7 - A primária manda a ordem aos secundários, requisitando a escrita.

8 - As secundárias respondem à primária.

9 - A primária responde ao cliente.

No caso de falha da escrita em um dos chunkservers o cliente recebe uma notificação e tenta novamente.

**Leitura:**

1 - Aplicação informa (file name, byte range) para o Cliente GFS.

2 - Cliente GFS traduz o pedido da aplicação para (file name, chunk index).

3 - Master responde para o Cliente GFS com (chunk handle, replica locations).

4 - Cliente GFS manda para os Chunk Servers (chunk handle, byte range).

5 - Chunk Servers respondem com o dado do arquivo.

6 - Enfim, o Cliente GFS manda para aplicação o dado solicitado.