ACH 2147 — DESENVOLVIMENTO DE SISTEMAS DE INFORMAÇÃO DISTRIBUÍDOS

TIPOS DE SISTEMAS DISTRIBUÍDOS

Tipos de Sistemas Distribuídos

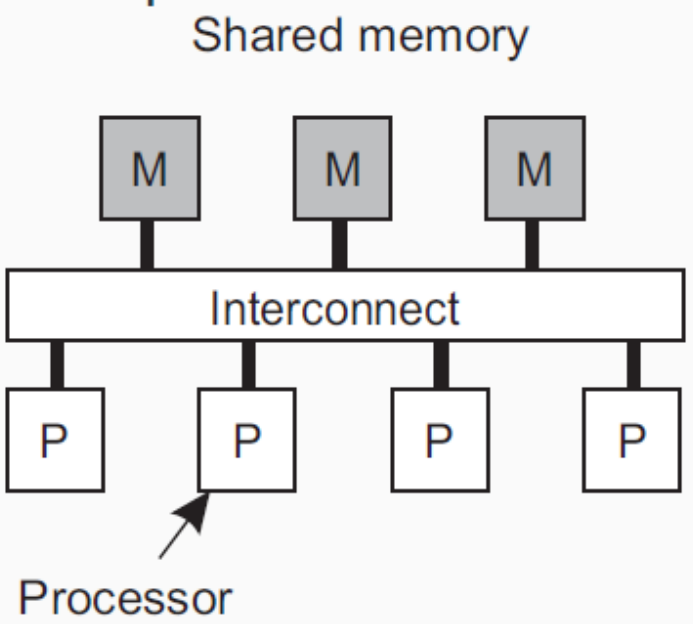
- Sistemas para computação distribuída de alto desempenho
- Sistemas de informação distribuídos
- Sistemas distribuídos para computação ubíqua

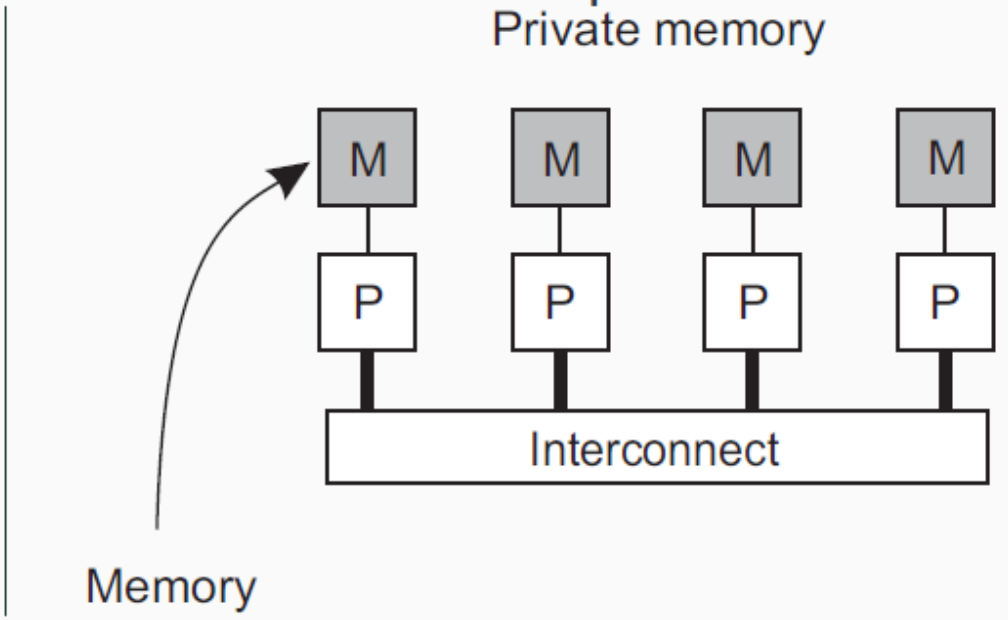
Computação Paralela

Observação

A computação distribuída de alto desempenho foi originada na computação paralela

Multiprocessadores e multicores versus multicomputadores





Sistemas de memória compartilhada

Multiprocessadores são relativamente fáceis de programar se comparados a multicomputadores, mas ainda assim os problemas aparecem quando o número de processadores (ou cores) aumentam. Solução: tentar implementar um modelo de memória compartilhada para multicomputadores.

Exemplo usando técnicas de memória virtual

Mapear todas as páginas da memória principal (de todos os diferentes processadores) em um único espaço de endereçamento virtual. Se o processo no processador *A* referenciar uma página *P* localizada no processador *B*, o SO em *A* lança uma interrupção e recupera *P* de *B*, do mesmo modo que faria se *P* estivesse localizado no disco.

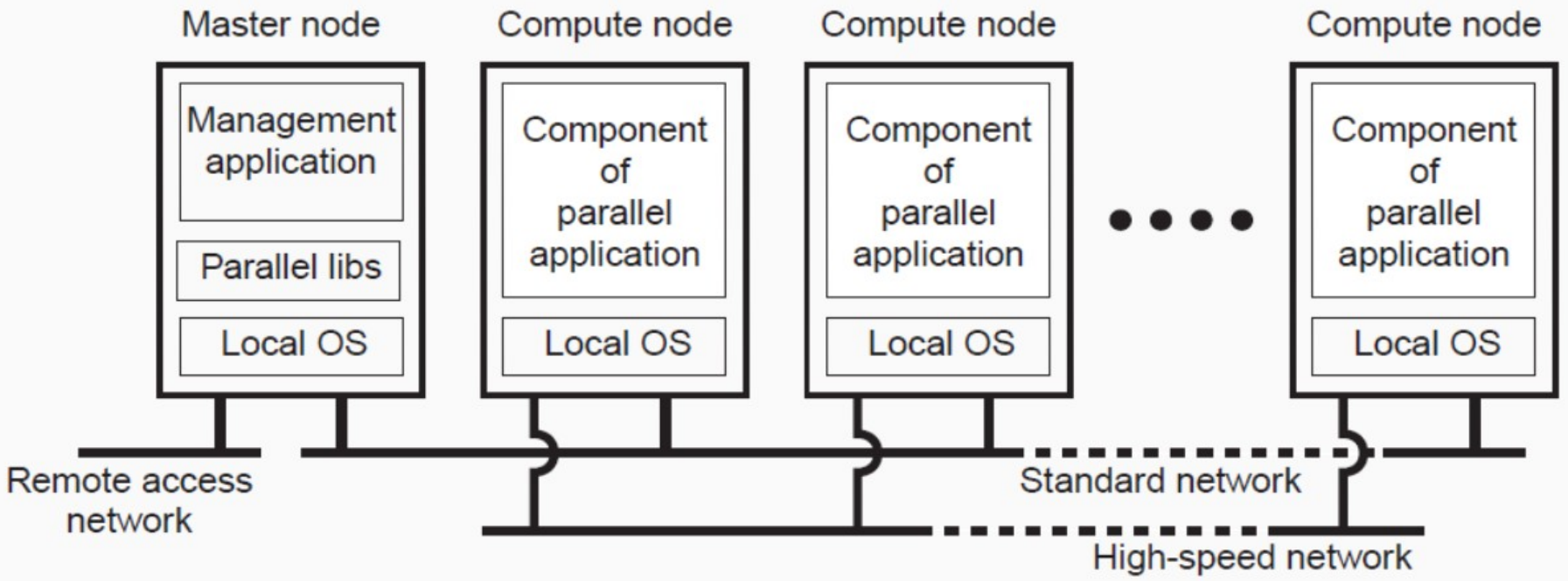
Problema

O desempenho de um sistema de memória compartilhada distribuída nunca poderia competir com o desempenho de multiprocessadores e, por isso, a ideia foi abandonada por hora.

Clusters de computadores

Essencialmente um grupo de computadores de boa qualidade conectados via LAN

- Homogêneo: mesmo SO, hardware quase idêntico
- Um único nó gerenciador



Clusters de computadores

Earth Simulator, #1 2002

Roadrunner, #1 2008 (1,02 petaflops)

Sunway TaihuLight, #1 2017 (93 petaflops)

Fonte: Top500.org
https://www.top500.org/timeline/

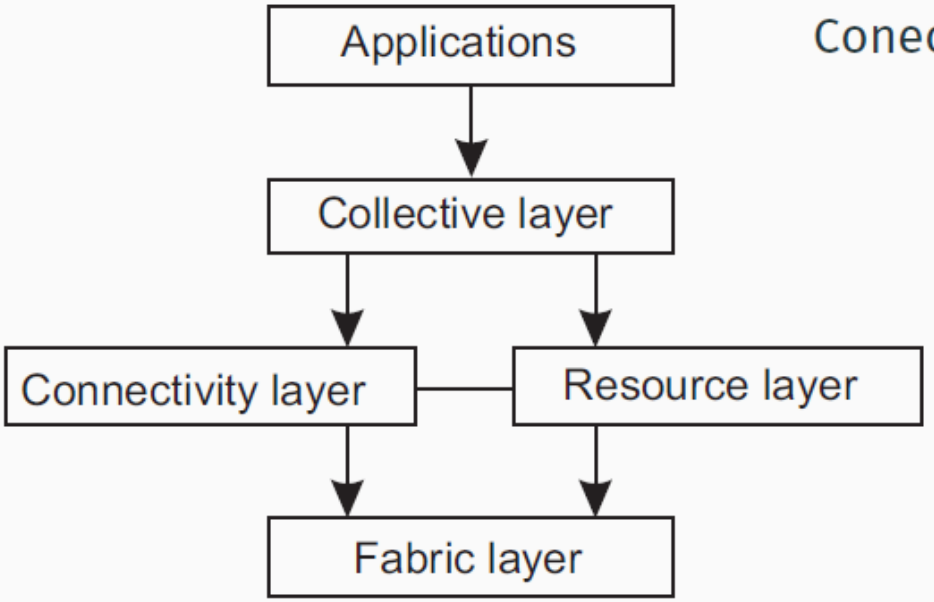
Computação em Grade (Grid Computing)

O próximo passo: vários nós vindos de todos os cantos:

- Heterogêneos
- Espalhados entre diversas organizações
- Normalmente formam uma rede de longa distância (*wide-area network*)

Nota:
Para permitir colaborações, grades normalmente usam *organizações virtuais*. Essencialmente, isso significa que os usuários (ou melhor, seus IDs) são organizados em grupos que possuem autorização para usar alguns recursos.

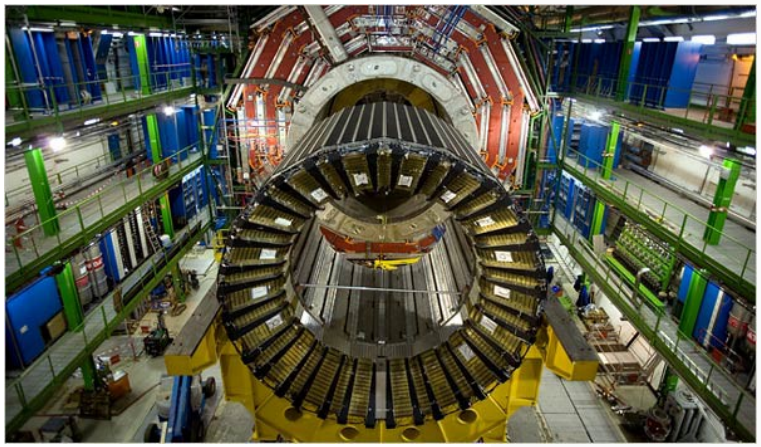
Arquitetura de Computação em Grade



Camadas

Infraestrutura	provê interfaces para os recursos locais
Conectividade	protocolos de comunicação/transação e autenticação
Recurso	gerencia um único recurso, por exemplo criando processos ou lendo dados
Coletiva	realiza acesso à múltiplos recursos: descoberta, escalonamento, replicação
Aplicação	contém a aplicação real da grade em uma única organização

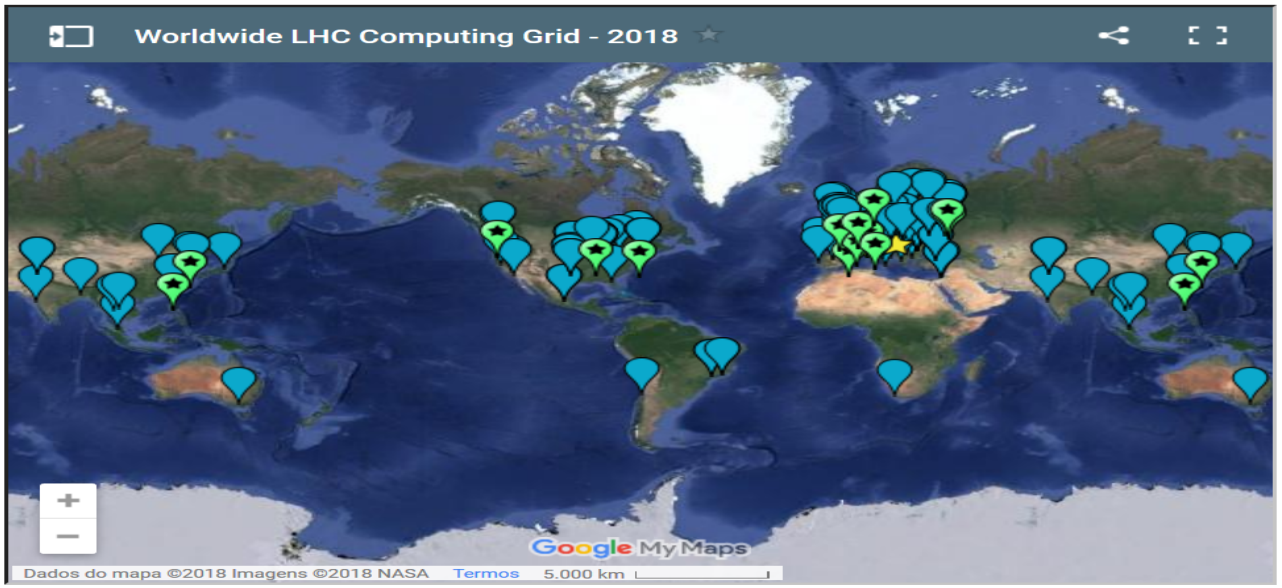
Grade Computacional



$$
\begin{aligned}
1\ \text{PB} &= 1.000.000.000.000.000\ \text{B} \\
&= 1.000^5\ \text{B} \\
&= 10^{15}\ \text{B} \\
&= 1\ \text{milhão de gigabytes} \\
&= 1\ \text{mil terabytes}
\end{aligned}
$$

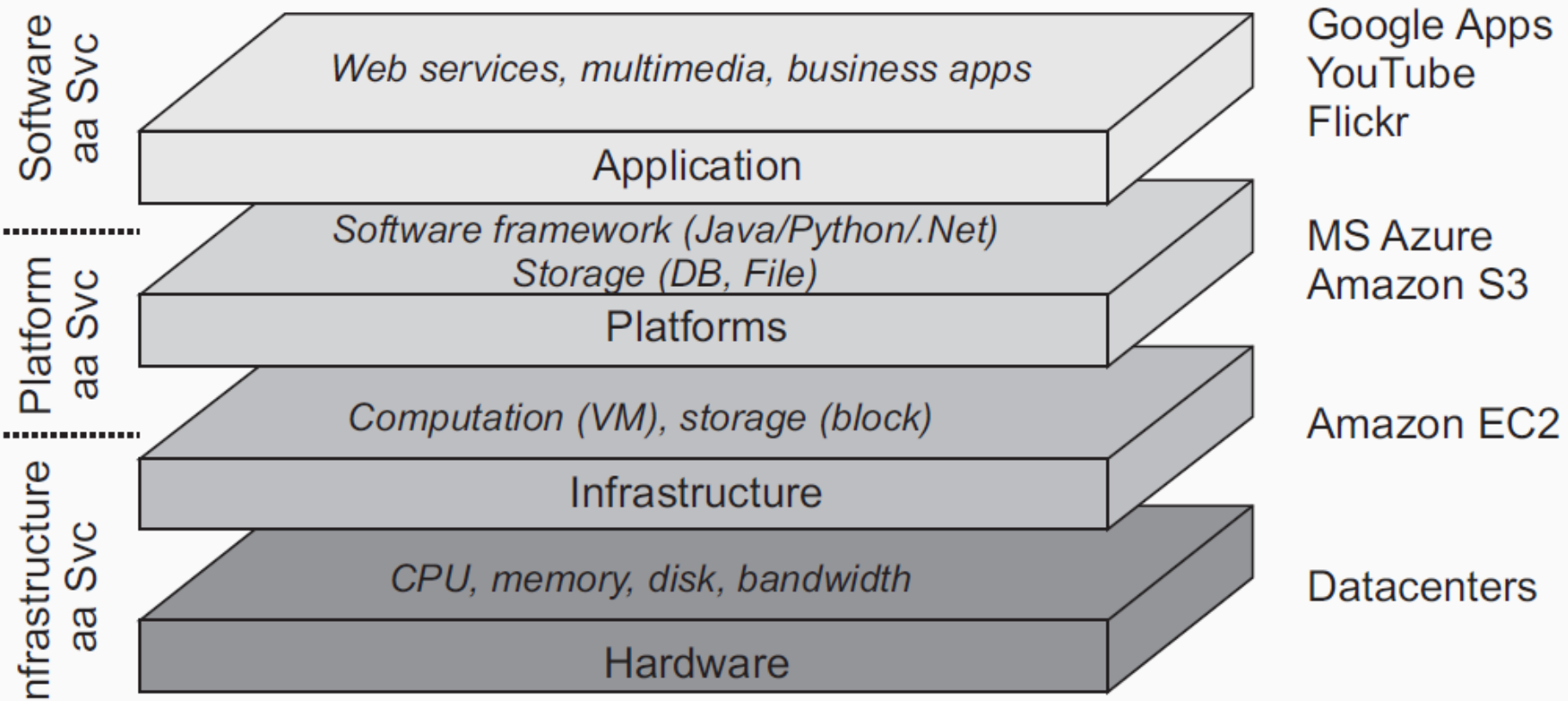
Ou seja, os 15 petabytes que o CERN irá gerar por ano equivalem a 15 milhões de gigabytes. Seriam necessários 1,7 milhão de DVDs *dual-layer* para armazenar tanta informação!

Current WLCG sites



Fonte:Worldwide LHC Computing Grid
http://wlcg.web.cern.ch/

Computação em Nuvem



Computação em Nuvem

Computação em nuvem
Faz uma distinção entre quatro camadas:

Hardware	processadores, roteadores, energia, sistemas de refrigeração
Infraestrutura	Utiliza técnicas de virtualização para alocação e gerenciamento de armazenamento e servidores virtuais
Plataforma	Provê abstrações de alto nível para os serviços da plataforma. Ex: Amazon S3 para armazenamento de arquivos em *buckets*
Aplicação	as aplicações propriamente ditas, tais como as suítes de aplicativos para escritórios.

Computação em Nuvem

Uma razão importante para o sucesso de computação em nuvem é que ela permite que organizações terceirizem sua infraestrutura de TI: hardware e software. A pergunta é: terceirizar é mesmo mais barato?

Abordagem

- Considere *aplicações corporativas*, modeladas como uma coleção de componentes (C_i), cada qual precisando de N_i servidores
- Podemos ver a aplicação como um grafo dirigido, com um vértice representando um componente e um arco $\langle \vec{i,j} \rangle$ representando o fluxo de dados de C_i para C_j.
- Cada arco tem dois pesos associados:
  - $T_{i,j}$, o número de transações por unidade de tempo que causam o fluxo de dados de C_i para C_j
  - $S_{i,j}$, a quantidade de dados total associada a $T_{i,j}$

Computação em Nuvem

Plano de migração

Encontre para cada componente C_i, quantos dos n_i dentre seus N_i servidores deveriam migrar, tal que a economia no orçamento menos os cursos de comunicação via Internet sejam maximais.

Economia no orçamento

- B_C: economias com a migração de um componente computacionalmente intensivo
- M_C número total de componentes computacionalmente intensivos
- B_S: economias com a migração de um componente intensivo em armazenamento
- M_S número total de componentes intensivos em armazenamento

A economia total, obviamente, é: $B_C \times M_C + B_S \times M_S$.

Sistemas Distribuídos

Uma quantidade enorme de sistemas distribuídos em uso hoje em dia são formas de sistemas de informação tradicionais, *integrando* sistemas legados. Exemplo: sistemas de processamento de transações.

```
BEGIN_TRANSACTION(server, transaction)
READ(transaction, file-1, data)
WRITE(transaction, file-2, data)
newData := MODIFIED(data)
IF WRONG(newData) THEN
    ABORT_TRANSACTION(transaction)
ELSE
    WRITE(transaction, file-2, newData)
    END_TRANSACTION(transaction)
END IF
```

Nota:
Transações formam uma operação atômica.

Transações

Uma transação é um conjunto de operações sobre o estado de um objeto (banco de dados, composição de objetos, etc.) que satisfazem as seguintes propriedades (ACID):

Atomicidade ou todas as operações são bem sucedidas, ou todas falham. Quando uma transação falha, o estado do objeto permanecerá inalterado.

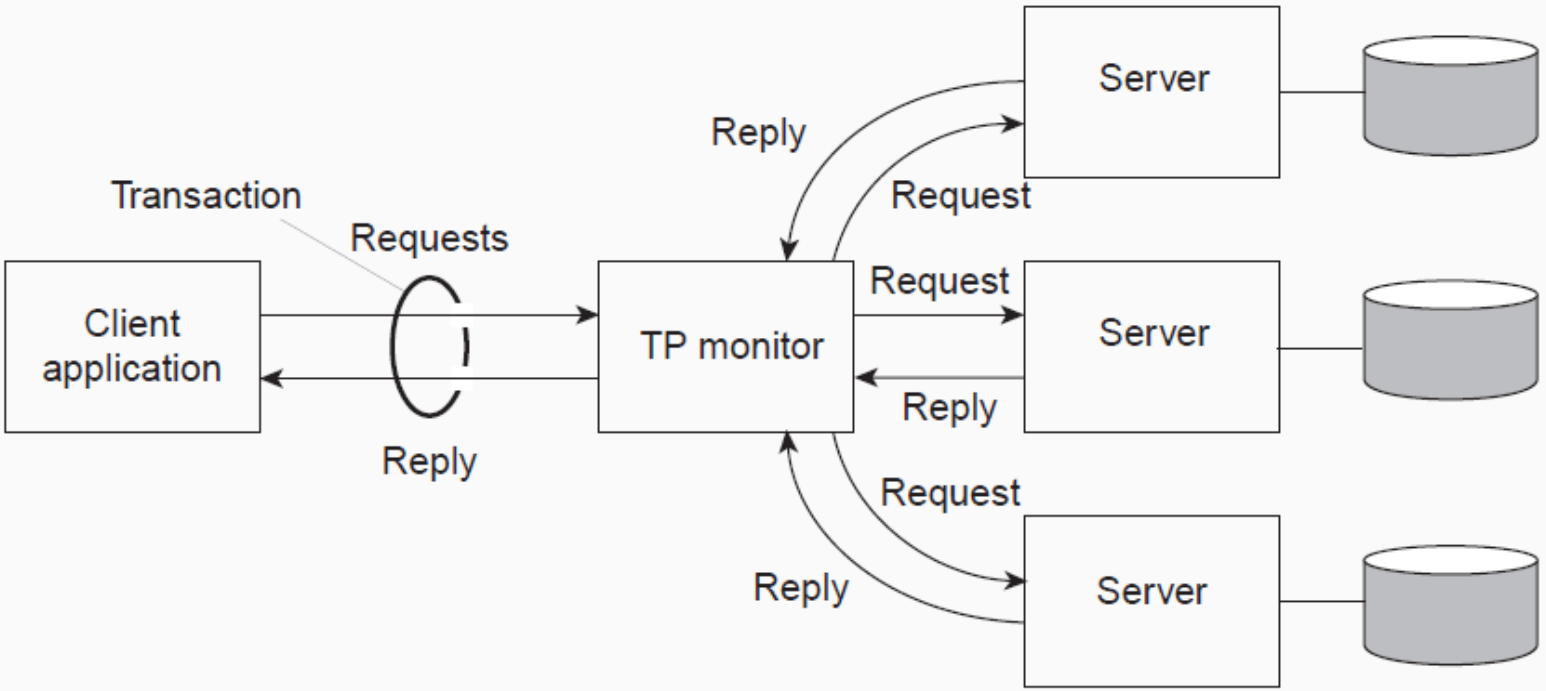
Consistência uma transação estabelece um estado de transição válido. Isto não exclui a existência de estados intermediários inválidos durante sua execução.

Isolamento transações concorrentes não interferem entre si. Para uma transação *T* é como se as outras transações ocorressem ou *antes* de *T*, ou *depois* de *T*.

Durabilidade Após o término de uma transação, seus efeitos são permanentes: mudanças de estado sobrevivem a falhas.

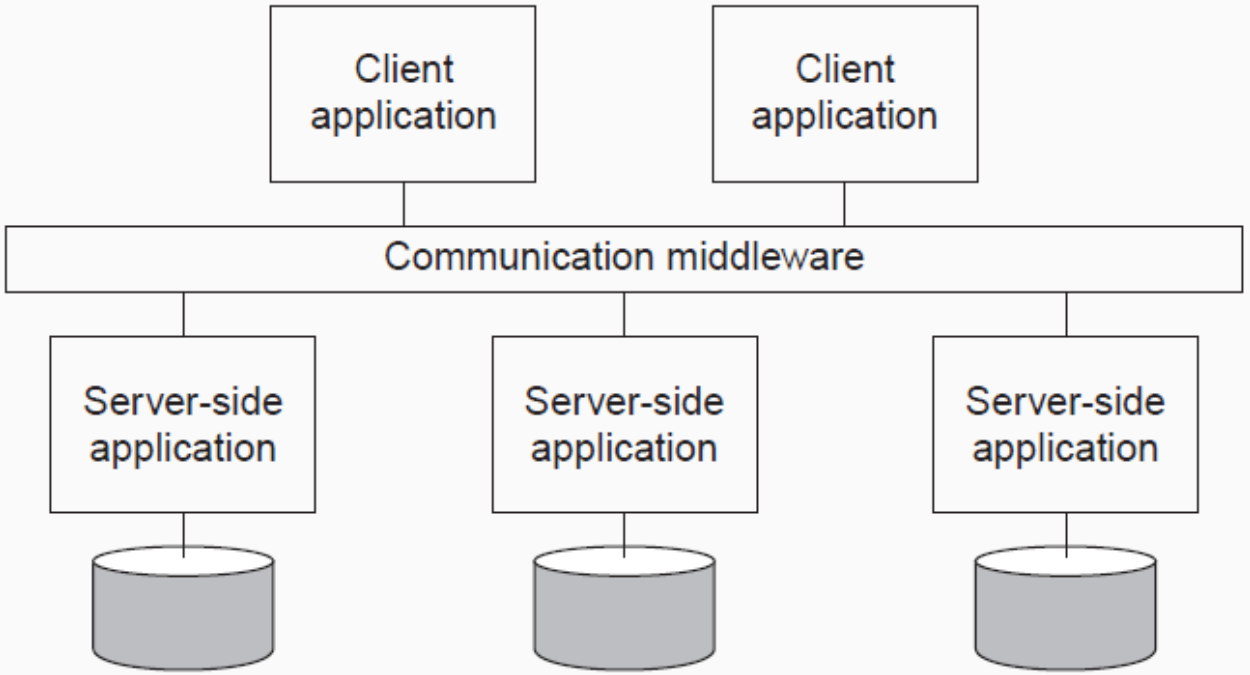
Monitor de Transações

Em muitos casos, o conjunto de dados envolvidos em uma transação está distribuído em vários servidores. Um TP Monitor é responsável por coordenar a execução de uma transação.



Integração de Aplicações

Um TP Monitor não basta, também são necessários mecanismos para a comunicação direta entre aplicações.



- Chamada de Procedimento Remoto (RPC)
- Middleware Orientado a Mensagens (MOM)

Sistemas Ubíquos

Tendência em sistemas distribuídos; nós são pequenos, móveis e normalmente embutidos em um sistema muito maior.

Alguns requisitos:

- Mudança contextual: o sistema é parte de um ambiente onde mudanças devem ser rapidamente levadas em consideração
- Composição ad hoc: cada nó pode ser usado em diferentes maneiras, por diferentes usuários. Deve ser facilmente configurável.
- Compartilhar é o padrão: nós vão e veem, fornecendo serviços e informação compartilháveis. Pede simplicidade.

Nota:
Ubiquidade e transparência de distribuição formam um bom par?

Redes de Sensores

Características

Os nós aos quais os sensores estão presos são:

- Muitos (10s–1000s)
- Simples (pouca capacidade de memória/computação/comunicação)
- Normalmente necessitam de uma bateria

Redes de Sensores