# Soluções para processamento paralelo e distribuído de dados

Ilustração de Oliver Munday

## Soluções Processameto paralelo e distribuído de de dados

Conceitos

- Motivação
- Arquitetura de soluções paralelas
- Paradigmas de programação paralela
- Exemplo práticos de soluções paralelas 'atuais'

# Soluções para processamento paralelo e distribuído de dados

## Conceitos – Motivação

Ilustração de Oliver Munday

## Conceitos - Motivação

### Porque computação paralela e distribuída?

Sistemas de computadores sequenciais cada vez mais velozes

- Velocidade de processador
- Memória
- Comunicação com o mundo externo

# Conceitos - Motivação

## Porque computação paralela e distribuída?

Mas, e os dados massivos ???


# Conceitos - Motivação

## Porque computação paralela e distribuída?

Mas, e os problemas derivados de dados massivos ???

- Cosmologia e Astrofísica;
- Clima global e modelagem do ambiente;
- E os carros autônomos, e as montadoras e locadoras de carros? Como será a digitalização deste negócio com dados granulares em tempo real?
- Detecção de padrões em videos em tempo real ?
- E o atacadista ali perto? Cruzar o histórico de 5 anos de compras de cada um de seus clientes por produto (novas oportunidades de vendas): 5.000 produtos * 100.000 clientes * 1825 dias = 912.500.000.000

# Conceitos - Motivação

## Porque computação paralela e distribuída?

Em resumo:

As aplicações que exigem computadores cada vez mais rápidos estão por toda parte;

Estas aplicações ou requerem um grande poder de computação ou requerem o processamento de grandes quantidades de informação;

# Conceitos - Motivação

## Porque computação paralela e distribuída?

Fatos:

Estrutura de arquivos convencional não é capaz de lidar com dados massivos;

É necessário um paradigma que comporte escalabilidade elástica;

Bancos de dados relacionais utilizam conceitos, que às vezes, não são um requisito para dados massivos;

Um infra estrutura tolerante a falhas e que permita computação paralela é necessária;

# Soluções para processamento paralelo e distribuído de dados

## Conceitos – Arquitetura de soluções paralelas

Ilustração de Oliver Munday

## Conceitos - Arquitetura de soluções paralelas

Características interessantes de soluções paralelas

- Conectividade ⇒ rede de interconexão
- Heterogeneidade ⇒ hardware e software distintos
- Compartilhamento ⇒ utilização de recursos (memória, disco, CPU)
- Imagem do sistema ⇒ como usuário o percebe
- Escalabilidade ⇒ mais estações de trabalho levam a melhor desempenho/eficiência

# Conceitos - Arquitetura de soluções paralelas

## Taxonomia de Flynn

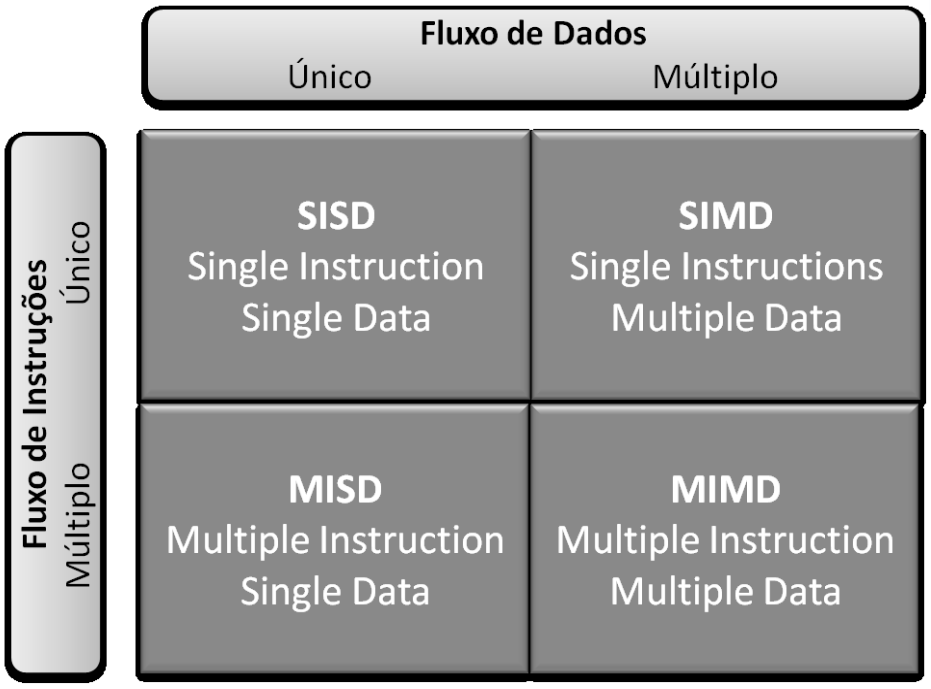
Uma das metodologias mais conhecidas e utilizadas para classificar uma arquitetura de computadores ou conjunto de computadores é a taxonomia de Flynn (1966);

Duas dimensões: instruções e dados;

Cada dimensão assume dois valores distintos: *single* ou *multiple;*

# Conceitos - Arquitetura de soluções paralelas

## Taxonomia de Flynn



# Conceitos - Arquitetura de soluções paralelas

## Taxonomia de Flynn

### SISD – Single Instruction Single Data

Arquitetura dos computadores com um único processador;

Apenas uma instrução é processada a cada momento.

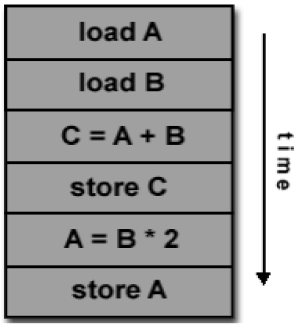
Apenas um fluxo de dados é processado a cada momento.

Exemplos: PCs, workstations e servidores com um único processador.

# Conceitos - Arquitetura de soluções paralelas

## Taxonomia de Flynn

### SISD – Single Instruction Single Data



# Conceitos - Arquitetura de soluções paralelas

## Taxonomia de Flynn

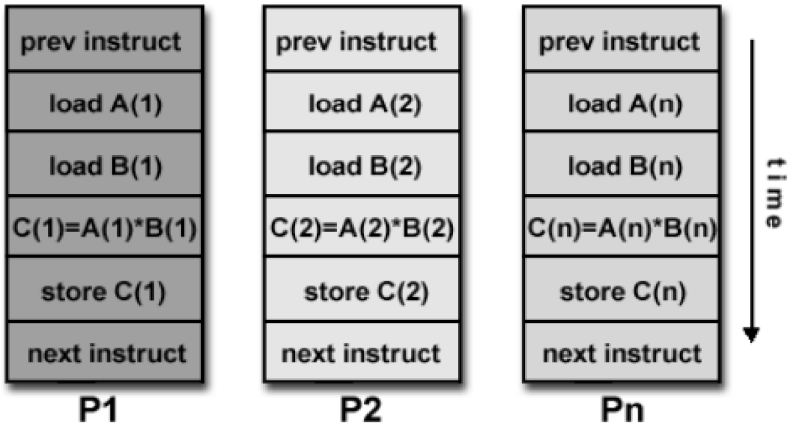
SIMD – Single Instruction Multiple Data

- Tipo de arquitetura paralela desenhada para problemas específicos (alto padrão de regularidade nos dados, ex.:processamento de imagem);
- Todas as unidades de processamento executam a mesma instrução a cada momento;
- Cada unidade de processamento pode operar sobre um fluxo de dados diferente;
- Exemplos: Computadores com unidades de processadores gráficos (GPUs);

# Conceitos - Arquitetura de soluções paralelas

## Taxonomia de Flynn

### SIMD – Single Instruction Multiple Data



# Conceitos - Arquitetura de soluções paralelas

## Taxonomia de Flynn

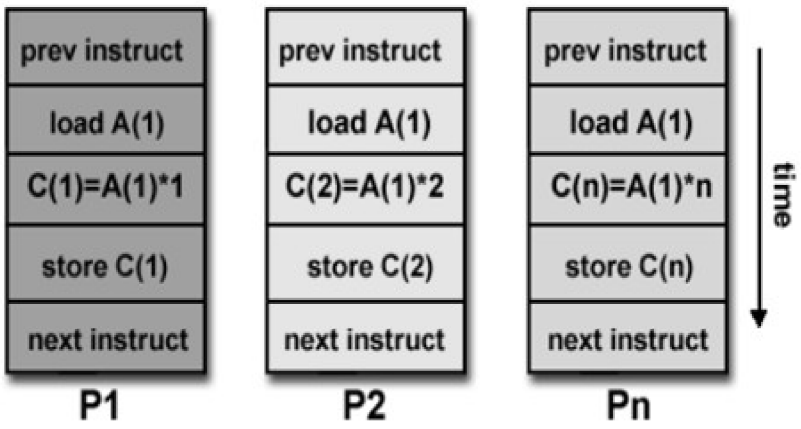
### MISD – Multiple Instruction Single Data

- Arquitetura paralela desenhada para problemas caracterizados por um alto padrão de regularidade funcional (ex.: processamento de sinal);
- Constituída por uma pipeline de unidades de processamento independentes que operam sobre um mesmo fluxo de dados enviando os resultados de uma unidade para a próxima;
- Cada unidade de processamento executa instruções diferentes a cada momento;
- Exemplos:Não existem exemplos práticos

# Conceitos - Arquitetura de soluções paralelas

## Taxonomia de Flynn

### MISD – Multiple Instruction Single Data



# Conceitos - Arquitetura de soluções paralelas

## Taxonomia de Flynn

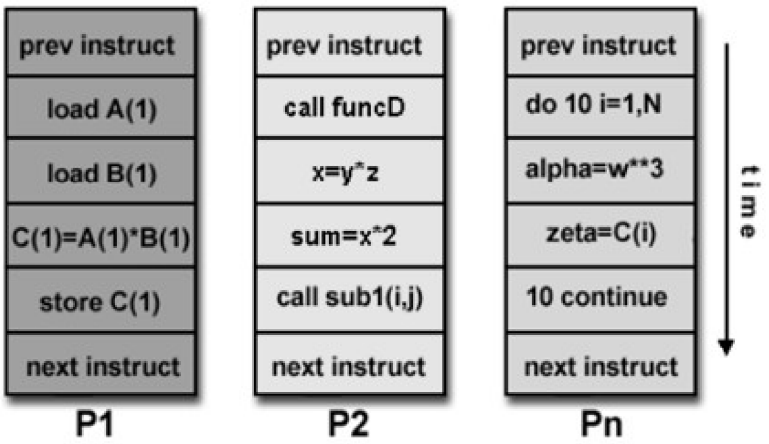
### MIMD – Multiple Instruction Multiple Data

- Arquitetura paralela predominante;
- Cada unidade de processamento executa instruções diferentes a cada momento;
- Cada unidade de processamento pode operar sobre um fluxo de dados diferente;
- Exemplos: alguns supercomputadores, clusters de computadores paralelos em rede, PCs multi-core;

# Conceitos - Arquitetura de soluções paralelas

## Taxonomia de Flynn

### MIMD – Multiple Instruction Multiple Data



# Conceitos - Arquitetura de soluções paralelas

Mas, no fundo, o que se espera com computadores paralelos ?

- Queremos melhorar o “Speedup”....
- Lei de Amdahl: Speedup esperado: $\frac{T(1)}{T(P)}$
- Speedup esperado com número de processadores:

$$\frac{1}{\frac{P}{N}+S}$$

P = fração paralelizável
N = número dos processadores
S = fração sequencial

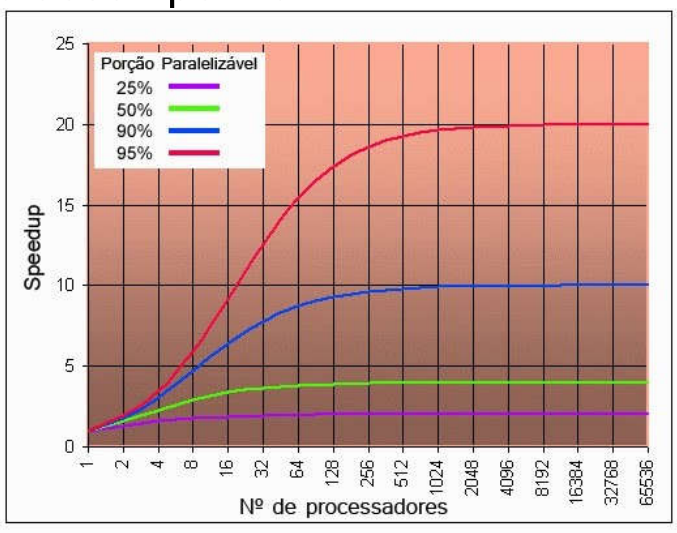
- Exemplo: P = 0.7 S = 0.3 N = 1 Speedup = 1
  P = 0.7 S = 0.3 N = 2 Speedup = 1.53
  P = 0.7 S = 0.3 N = 3 Speedup = 1.87

# Conceitos - Arquitetura de soluções paralelas

Mas, no fundo, o que se espera com computadores paralelos ?

- Speedup : melhorar o desempenho

| | speedup | | |
|---|---|---|---|
| N | P = .50 | P = .90 | P = .99 |
| 10 | 1.82 | 5.26 | 9.17 |
| 100 | 1.98 | 9.17 | 50.25 |
| 1,000 | 1.99 | 9.91 | 90.99 |
| 10,000 | 1.99 | 9.91 | 99.02 |
| 100,000 | 1.99 | 9.99 | 99.90 |




*Fonte:http://www.felipealencar.net/2014/12/programacao-paralela-e-por-que-voce.html*

# Conceitos - Arquitetura de soluções paralelas

## Plataformas MIMD - Multiple Instruction Multiple Data

- Espaço de endereçamento
- Mecanismo de comunicação

## Podem ser agrupadas em quatro grupos

- SMPs (Symmetric MultiProcessors)
- MPPs (Massively Parallel Processors)
- Cluster ou NOWs (Network Of Worstations)
- Grades Computacionais

# Conceitos - Arquitetura de soluções paralelas

## SMPs ou Multiprocessadores

- Espaço de endereçamento
  - Único espaço de endereçamento lógico
  - Mecanismo de hardware (memória centralizada)
- Mecanismo de comunicação
  - Comunicação entre processadores se dá através de um espaço de endereçamento compartilhado
  - Operações de loads(leitura) e stores(escrita) feitos com acesso a memória ;

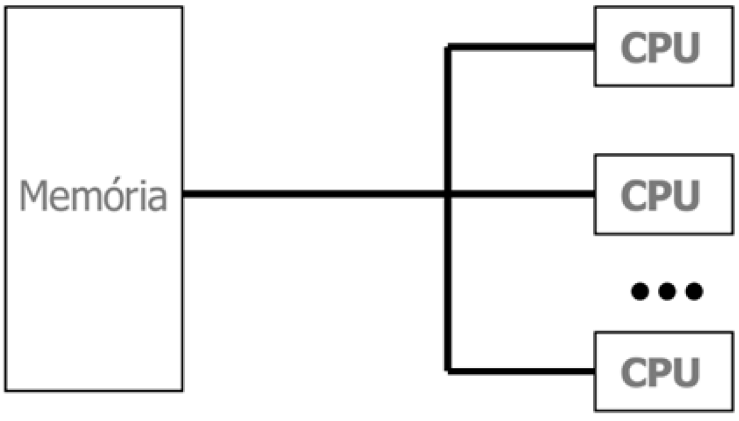
# Conceitos - Arquitetura de soluções paralelas

## SMPs ou Multiprocessadores

- Características
  - Sistema homogêneo
  - Comunicação através da mesma memória
  - Uma única cópia do Sistema Operacional
  - Imagem única do sistema
  - Fortemente acoplados
  - Não escalável

# Conceitos - Arquitetura de soluções paralelas

## SMPs ou Multiprocessadores



# Conceitos - Arquitetura de soluções paralelas

## MPPs (Multicomputadores)

- Espaço de endereçamento
  - Não compartilhado
  - Memória distribuída
- Mecanismo de comunicação
  - Troca de mensagens;
- Unidades de processamento
  - Múltiplos processadores com memória privativa
  - Computadores completos

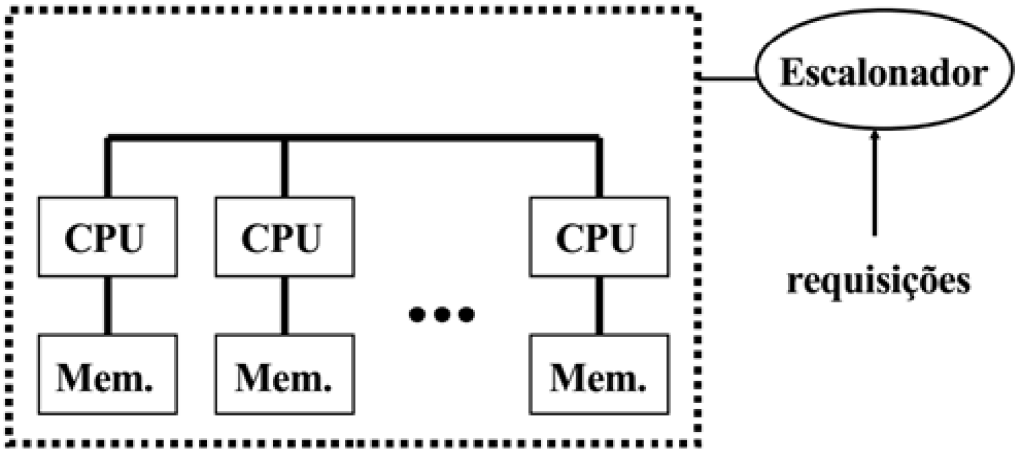
# Conceitos - Arquitetura de soluções paralelas

## MPPs (Multicomputadores)

- Características
  - - Sistema homogêneo ou heterogêneo
  - - Cada nó executa sua própria cópia do Sistema Operacional
  - - Imagem única do sistema
  - - Rede de interconexão: diferentes topologias
  - - Fracamente acoplados
  - - Escaláveis
  - - Aplicações não compartilham recursos: Pode ocorrer que uma aplicação permaneça em estado de espera

# Conceitos - Arquitetura de soluções paralelas

MPPs (Multicomputadores)




# Conceitos - Arquitetura de soluções paralelas

Clusters

- Espaço de endereçamento
  - Não compartilhado
- Mecanismo de comunicação
  - Troca de mensagens;
- Unidades de processamento
  - Conjunto de estações de trabalho ou Pcs (Nós: elementos de processamento = processador + memória)

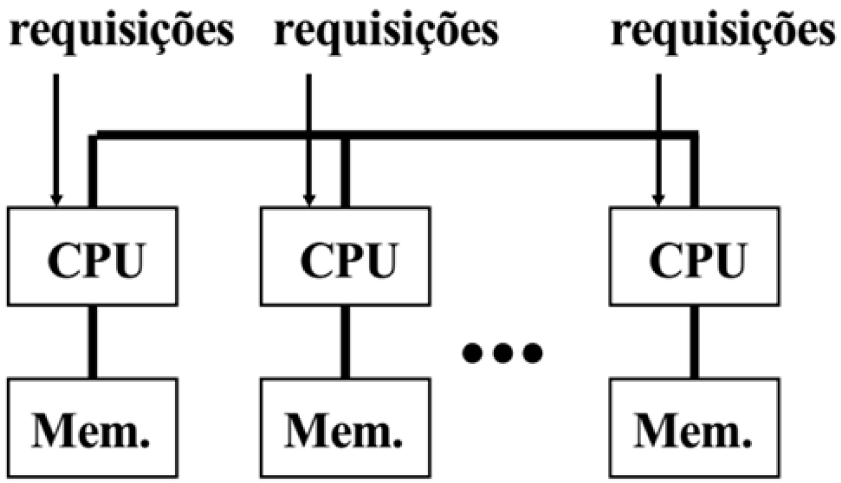
# Conceitos - Arquitetura de soluções paralelas

## Clusters

- Características
  - Sistema homogêneo ou heterogêneo
  - Interconexão: redes locais e tendem a ser mais lentas que no MPP
  - Não existe um escalonador centralizado
  - Cada nó tem seu próprio escalonador local
  - Possibilidade de compor um sistema de alto desempenho e um baixo custo (principalmente quando comparados com MPP).

# Conceitos - Arquitetura de soluções paralelas

## Clusters



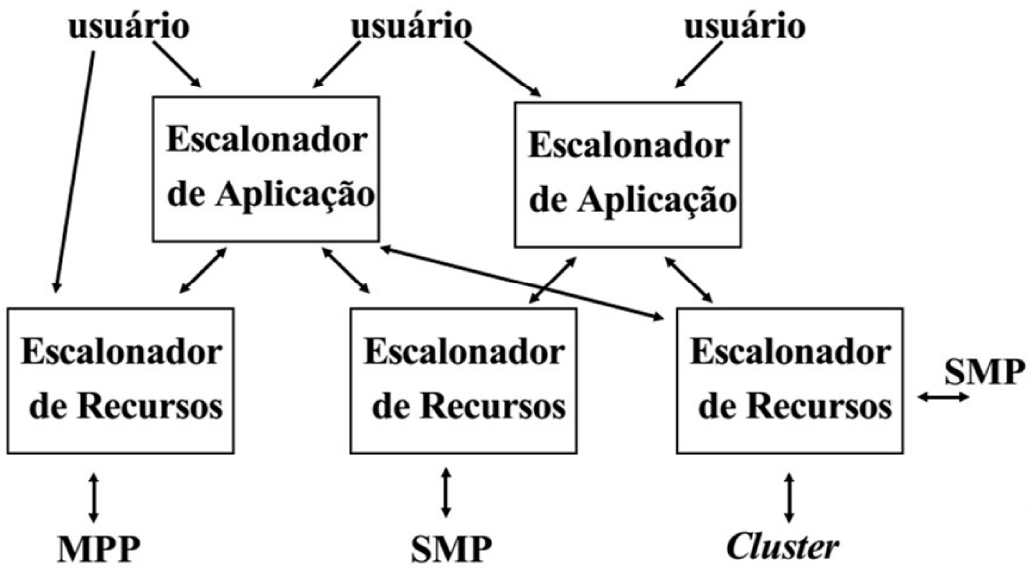
# Conceitos - Arquitetura de soluções paralelas

## Grids

- Utilização de computadores
  - independentes
  - geograficamente distantes
- Não têm uma imagem única do sistema a princípio (em desenvolvimento)
- Pode ser formado por PCs, SMPs, MPPs e clusters

# Conceitos - Arquitetura de soluções paralelas

## Grids



# Soluções para processamento paralelo e distribuído de dados

## Conceitos – Paradigmas de programação paralela

Ilustração de Oliver Munday

# Conceitos – Paradigmas de programação Paralela

## Fatores que limitam o desempenho

**Código Sequencial**: existem partes do código que são inerentemente sequenciais

**Concorrência**: o número de tarefas pode ser escasso e/ou de difícil definição.

**Granularidade**: o número e o tamanho das tarefas é importante porque o tempo que demoram a ser executadas tem de compensar os custos da execução em paralelo (custos de criação, comunicação e sincronização);

**Balanceamento de Carga**: ter os processadores o máximo ocupados (durante toda a execução) é decisivo para o desempenho global do sistema;

# Conceitos – Paradigmas de programação Paralela

## Modelos de programação paralela

**Programação em Memória Partilhada**

- Programação usando processos ou threads
- Comunicação através de memória partilhada.

**Programação em Memória Distribuída**

- Programação usando troca de mensagens.
- Comunicação e sincronização por troca de mensagens.

# Conceitos – Paradigmas de programação Paralela

## Principais Paradigmas de Programação Paralela

A escolha do paradigma para aplicar a um dado problema é determinado pelo:

- Tipo de paralelismo inerente ao problema
- Tipo de recursos computacionais disponíveis

# Conceitos – Paradigmas de programação Paralela

## Principais Paradigmas de Programação Paralela

Diversos problemas aplicáveis

O desenvolvimento de algoritmos paralelos pode ser classificado em diferentes paradigmas;

Cada paradigma representa uma classe de algoritmos (similares):

- Master/Slave
- Single Program Multiple Data (SPMD)
- Data Pipelining
- Divide and Conquer
- Speculative Parallelism

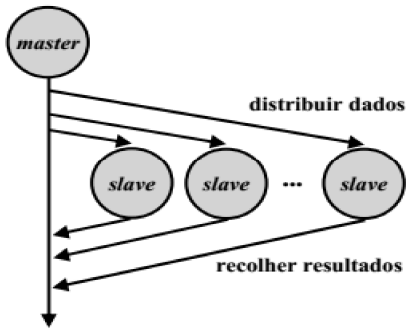
# Conceitos – Paradigmas de programação Paralela

## Master/Slave

Divide a computação em duas entidades distintas:

- **Master**: é o responsável por decompor o problema em tarefas, distribuir as tarefas pelos slaves e recolher os resultados parciais dos slaves de modo a calcular o resultado final

- **Slaves**: responsabilidade trivais e simples: obter uma tarefa do master, processar a tarefa e enviar o resultado de volta para o master.



# Conceitos – Paradigmas de programação Paralela

## Master/Slave

- O balanceamento de carga pode ser estático ou dinâmico:

  Estático: a divisão de tarefas é feita no início da computação;

  Dinâmico: quando o tempo de execução das tarefas é desconhecido no início da computação;

- Existe apenas comunicação entre o master e os slaves: bons desempenhos e um elevado grau de escalabilidade

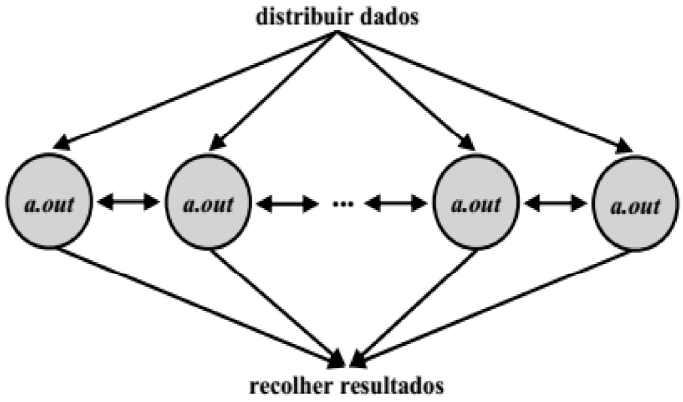
- Controle centralizado no master pode ser um problema quando o número de slaves é elevado;

- Possível aumentar a escalabilidade do paradigma considerando vários masters em que cada um controla um grupo diferente de slaves;

# Conceitos – Paradigmas de programação Paralela

## Single Program Multiple Data (SPMD)

-Consiste em processos que executam o mesmo programa (executável) mas sobre diferentes partes dos dados:



# Conceitos – Paradigmas de programação Paralela

## Single Program Multiple Data (SPMD)

- Dados devem ser bem distribuídos (mesma quantidade e regularidade);

- São lidos individualmente por cada processo ou um dos processos é o responsável por ler todos os dados e depois distribui-los para os demais;

- Os processos comunicam quase sempre com processos vizinhos e apenas esporadicamente existem pontos de sincronização global;

- Consegue bons desempenhos e um elevado grau de escalabilidade;

# Conceitos – Paradigmas de programação Paralela

## Single Program Multiple Data (SPMD)

- Muito sensível a falhas: no caso de falha de um nó  há problemas na resolução da computação global;

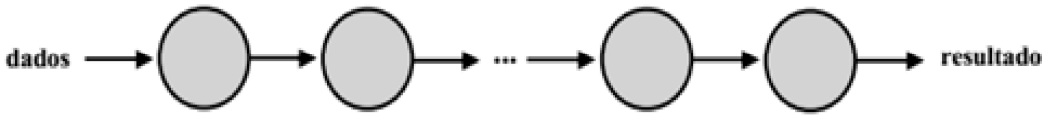
# Conceitos – Paradigmas de programação Paralela

Data Pipelining

- Utiliza uma decomposição funcional do problema em que cada processo executa apenas uma parte do algoritmo completo e total;

- O padrão de comunicação é definido e simples: processos são organizados em sequência (pipeline) e cada processo só troca informação com o processo seguinte;

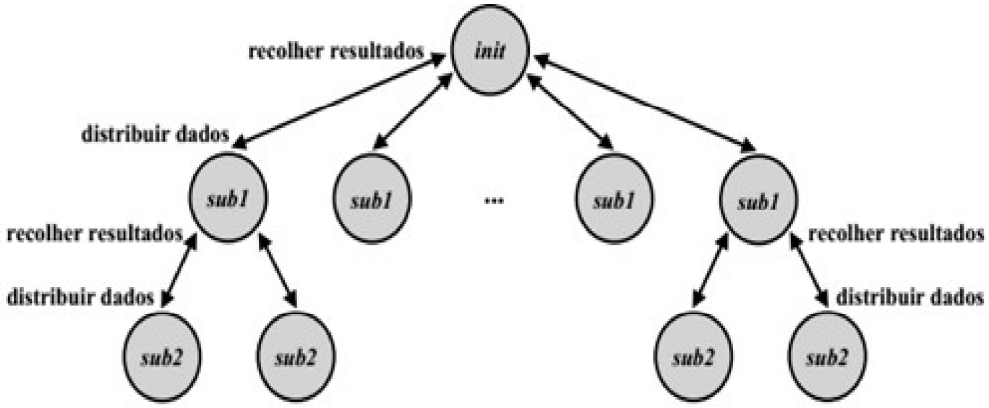
- Dependente muito da capacidade de balancear a carga entre as diferentes etapas da pipeline;




# Conceitos – Paradigmas de programação Paralela

Divide and Conquer

- Utiliza uma divisão recursiva do problema inicial em subproblemas independentes instâncias mais pequenas do problema inicial) cujos resultados são depois combinados para obter o resultado final



# Conceitos – Paradigmas de programação Paralela

## Divide and Conquer

Visão dos processo como uma especie de arvore:

- Os processos nos nós folha processam as subtarefas;

- Os restantes processos são responsáveis por criar as subtarefas e por agregar os seus resultados parciais;

Subtarefas são totalmente independentes, não há qualquer tipo de comunicação durante o processamento das mesmas;

Apenas existe comunicação entre o processo que cria as subtarefas e os processos que as executam;

# Conceitos – Paradigmas de programação Paralela

## Speculative Parallelism

- Utilizado quando as dependências entre os dados são tão complexas que tornam difícil explorar paralelismo usando os paradigmas anteriores;

- Uma aplicação deste paradigma é quando se utiliza simultaneamente diversos algoritmos para resolver um determinado problema e se escolhe aquele que primeiro obtiver uma solução;

# Soluções para processamento paralelo e distribuído de dados

## Conceitos – Exemplo práticos de soluções paralelas 'atuais'

Ilustração de Oliver Munday

## Conceitos - Exemplo práticos de soluções paralelas 'atuais'


### Hadoop

- Talvez seja a solução 'clássica' e mais conhecida atualmente;
- É um framework para processamento paralelo e distribuído;
- Utiliza paradigma de Master\Slave;
- É uma arquitetura do tipo cluster ou NOW;
- Possui vários componentes: HDFS, MapReduce, Yarn, Pig, Hive, Oozie, Ambari....
- Grande alteração da versão 1.0 para a versão 2.0 em diante

# Conceitos - Exemplo práticos de soluções paralelas 'atuais'

## Spark


- Spark também é uma plataforma para processamento distribuído;
- Pode trabalhar em conjunto com o Hadoop  ou não...
- Spark não possui um componente de armazenamento
- Processamento pode ser feito em memória e flexibiliza o pipeline de processamento de dados

# Conceitos - Exemplo práticos de soluções paralelas 'atuais'

## Storm


-É um sistema de computação distribuída em tempo real, por natureza;

- Utilizado em real-time analytics (Processamento de streaming de dados);
- Pode ser usado com muitas linguagens de programação;
- Capaz de processar 1 millhão de linhas por nó, por segundo;
- Integra-se com o Hadoop;

-Não possui suporte para processamento Batch;

# Conceitos - Exemplo práticos de soluções paralelas 'atuais'

## Flink

-É um *engine* de fluxo de dados em real time;

- Trata tanto fluxos em batch quanto em real time(apesar de ser orientado para streamings de dados);

- Possui APIS para Java, Scala, Python e uma API SQL;

- Possui módulos para grafos e apredinzado de máquina;

- Consegue trabalhar com granularidade de registros(linhas de uma tabela -  Hadoop e Spark vão ter dificuldades neste caso);