Algoritmos Paralelos

Demanda por Velocidade Computational

Demanda contínua por maior rapidez computational das máquinas que as atualmente disponíveis.

As áreas que exigem maior rapidez computational incluem modelagem numérica e simulação de problemas de engenharia.

Computação dever ser completeda em um tempo "razoável".

Problemas Desafiadores

Um problem desafiador é aquele que não pode ser resolvido em um tempo razoável com os computadores atuais.

Claro, um tempo de execução de 5 anos nunca é razoável.

Exemplos

- Modelar grandes estruturas de DNA
- Previsão de tempo
- Modelar movimentos de corpos astronômicos.

Previsão do tempo

Atmosfera é modeleda dividindo-se em células 3-d. Calculos em cada célula é repetida várias vezes para modelar a passagem do tempo.

Exemplo

A atmosfera inteira é dividida em partes de 1 km × 1km x 1 km para uma altura de 10 km (10 células) - cerca de 5×10^8 células. Suponha que cada cálculo requer 200 flops. Em um passo são necessários 10^{11} flops.

Previsão do tempo

Para prever o tempo em 10 dias usando intervalos de 10 min, um computador operarando em 100 Mflops (10^8 flops/s) gastaria 10^7 segundos ou cerca de 100 dias.

Para realizar o cálculo em 10 minutos precisaria de um computador operando em 1.7 Tflops (1.7×10^{12} flops/sec).

Movimentos de Corpos Celestes

Cada corpo é atraído por outro pela força gravitacional.

O movimento de cada corpo é previsto através do cálculo da força total em cada corpo. Com N corpos, $N-1$ forças calculadas em cada corpo, ou aprox. N^2 cálculos.

Após determinar uma posição dos corpos, repetir os cálculos.

Uma galáxia pode ter 10^{11} astros. Se cada cálculo fosse feito em 1μs (situação otimista), isto demoraria 10^9 anos para uma iteração usando o N^2 algoritmo e quase 1 ano para uma iteração usando um algoritmo aproximado Nlg N

Computação Paralela

Usar mais de um computador, ou um computador com mais de um processador, para resolver um problema.

Motivos

Computação mais rápida - idéia simples - *n* computadores operando simultaneamente produz o resultado *n* vezes mais rápido - Não é sempre *n* vezes mais rápido.

Problemas: Tolerância a falhas, mais memória disponível, ...

Background

Computadores Paralelos: computador com mais de um processador e sua programação - programação paralela - têm por volta de 40 anos.

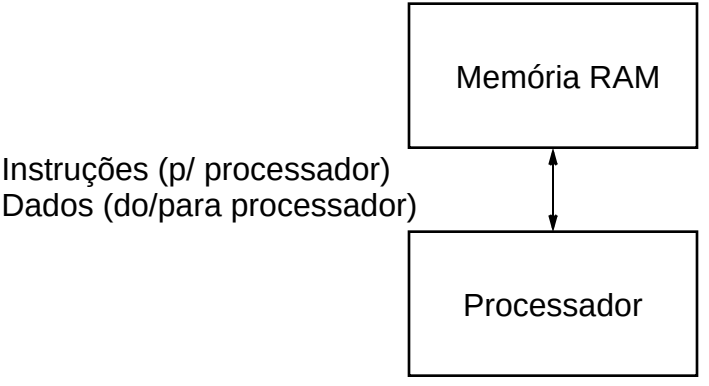
Gill escreveu em 1958:

"... Não há nada de novo na idéia de programação paralela, exceto suas aplicações. O autor não pode acreditar que haverá qualquer difculdade insuperável em extendê-la para os computadores. Não se espere que as técnicas de programação sejam produzidas numa noite. Muitos experimentos deverão ser realizados. Além disso, as técnicas que são mais usadas na programaçao de hoje só foram vencidas a um custo considerável de vários anos. O advento da programação paralela pode fazer algo para reviver o espírito pioneiro da programação, o qual parece atualmente estar degenerado em uma ocupação estúpida de rotina ..."

Gill, S. (1958), "Parallel Programming," *The Computer Journal*, vol. 1, April, pp. 2-10.

Computador Convencional

Consiste de um processador executando um programa armazenado da memória:



Cada localização da memória tem o seu *endereço*. Endereços vão de 0 até $2^n - 1$ onde há n bits (digitos binários) no endereço.

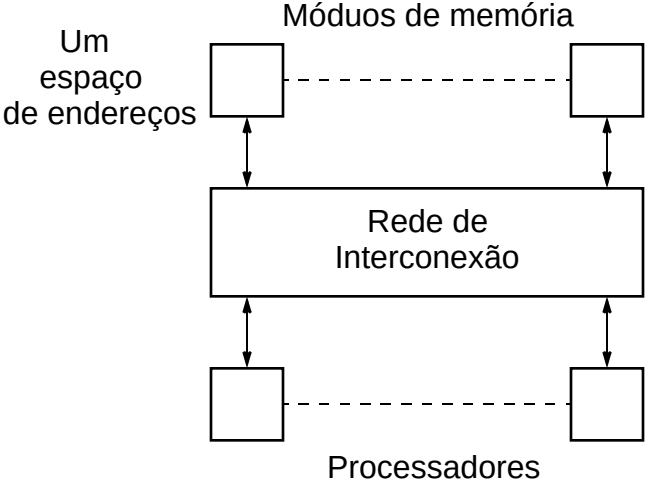
Tipos de Computadores Paralelos

Há dois tipos principais:

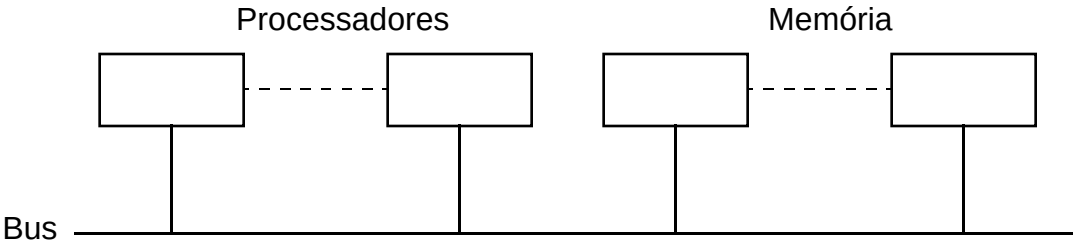
- Multiprocessador de Memória Compartilhada
- Multicomputador de Memória Distribuída

Multiprocessador de Memória Compartilhada

Forma natural de extender o modelo convencional - ter vários processadores conectaddos a vários módulos de memória, tal que cada processador accesse qualquer módulo de memória - *configuração de memória compartilhada*:



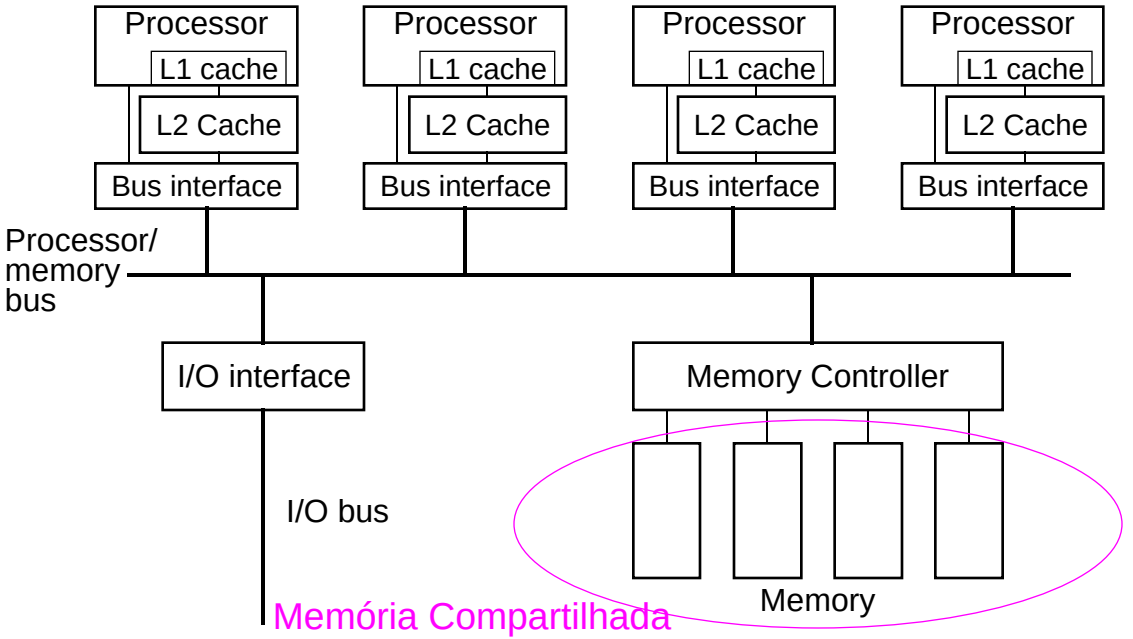
Visão simplificada de um multiprocessador de memória compartilhada



Exemplos:

- Dual Pentium
- Sparcs Sun

Multiprocessador Quad Pentium



Multiprocessador de Memória Compartilhada

Cada processador acessa qualquer localização de memória.

Há um *único espaço de endereço*, i.e. cada local de memória é dado por um único endereço em um só intervalo de endereços.

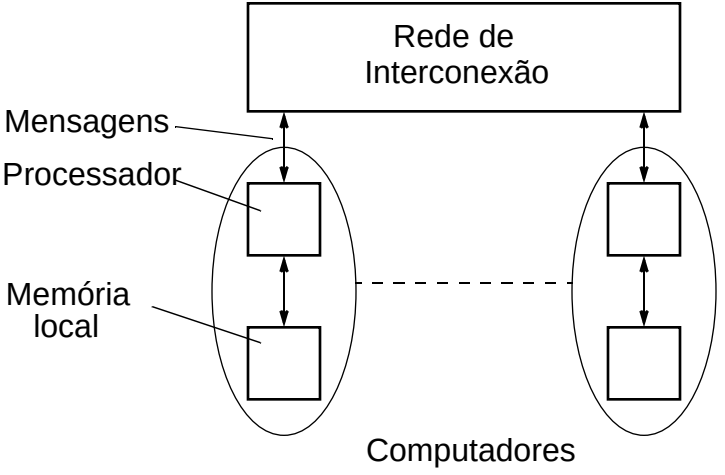
Geralmente, sua programação é mais conveniente embora o acesso a dados compartilhados deva ser controlado pelo programador (usando seções críticas etc.)

Alternativas para Programar MMC:

- Threads (Pthreads, Java, ..) na qual o programador decompõe o programa em sequencias paralelas individuais, cada uma sendo uma thread, e sendo capaz de acessar variáveis declaradas fora das threads.
- Uma linguagem de programação sequencial com diretivas de compilação para declarar variáveis compartilhadas e especificar paralelismo: Ex. OpenMP - industry standard
- Linguagem de programação seqüencial c/ biblioteca user-level para declarar e acessar variáveis compartilhadas.
- Linguagem de programação paralela c/ sintaxe de paralelismo, no qual o compilador cria o código executável apropriado para para cada processador (Incomum)
- Linguagem de programação sequencial que pede um compilador paralelo para converter o executável para código paralelo. (Incomum)

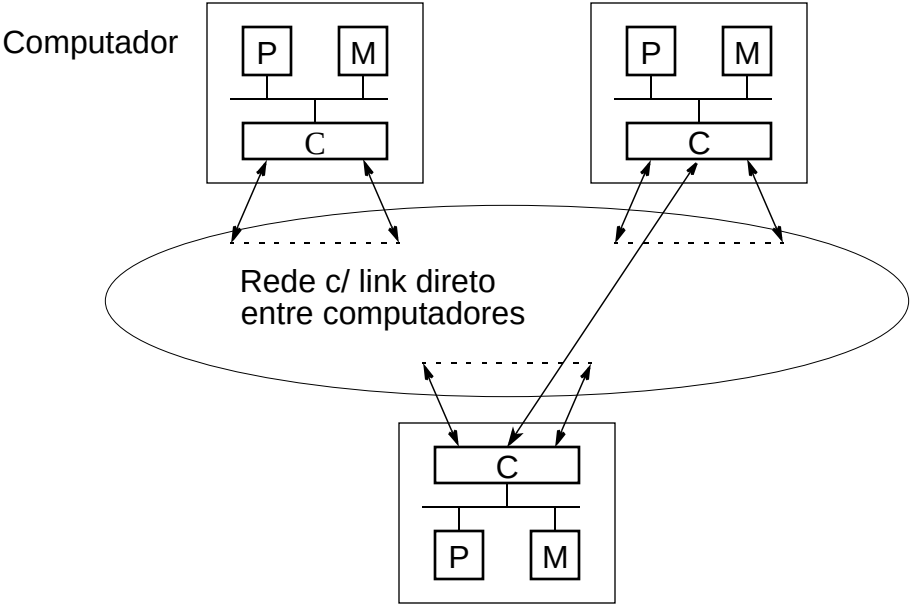
Multicomputador com Troca de Mensagem

Computador completo conectado por uma rede de interconexão:



Multicomputador com Troca de Mensagem e Rede Estática

Computadores conectados por links diretos:

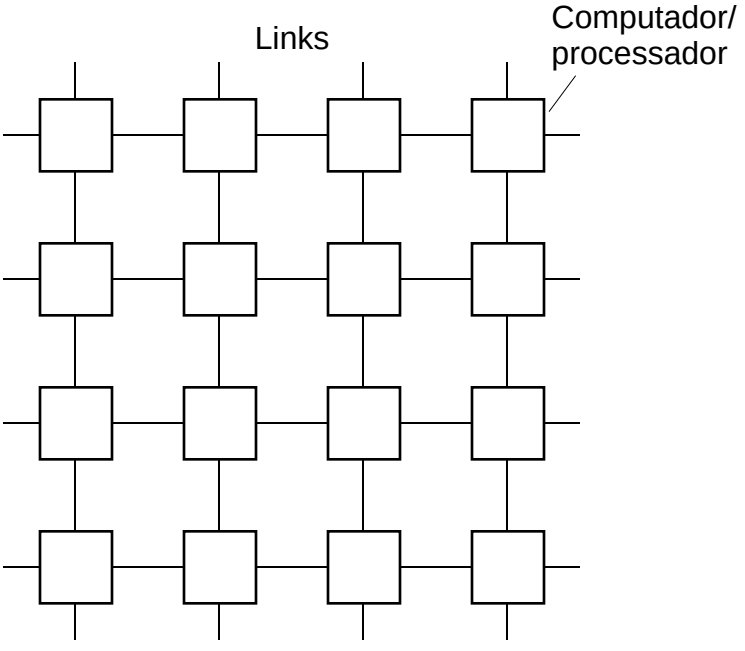


Redes com Links Estáticos

Várias:

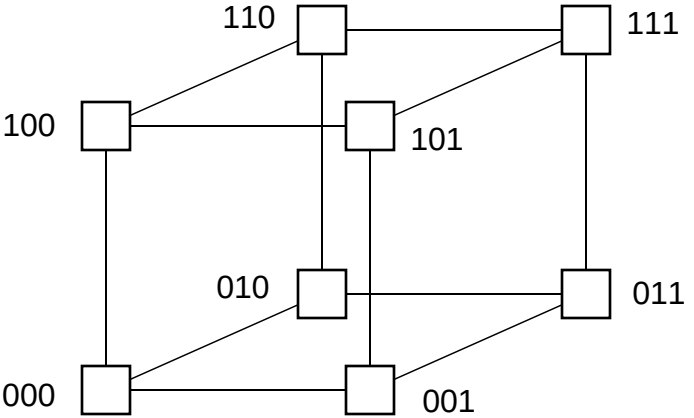
- Anel
- Árvore
- grades 2-D e 3-D
- Hipercubo

Grande Bi-dimensional (2-D)

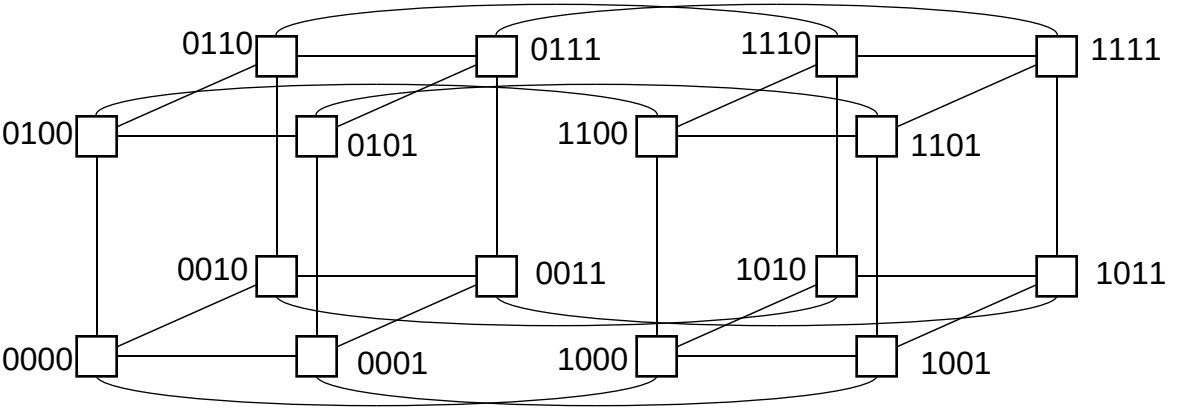


Tri-dimensional - usada em algums sistemas de alto desempenho.

Hipercubo Tri-dimensional



Hipercubo de Dimensão 4 (4-D)



Hipercubos foram populares nos anos 80

Computadores em Rede como Plataforma Multicomputador

As *redes de estações* (NOWs) tornaram-se uma alternativa atraente aos caros supercomputadores e aos computadores paralelos para computação de alto-desempenho nos anos 90.

Vários Projetos

- Projeto Berkely NOW

Vantagens:

- Workstations de alta performance e PCs amplamente disponíveis e com baixo custo.
- O mais recente processador pode ser incorporado ao ao sistema assim que tornar-se disponível.
- Software existente pode ser usado ou modificado.

Clusters Beowulf*

Grupo de computadores interconectaddos em “commodity” para produzir alto desempenho com baixo custo.

Geralmente usando interconexão - Fast-Ethernet e Linux.

* Lendas sobre o nome Beowulf existem várias.

Interconexão do Cluster

- Originalmente fast-Ethernet em clusters de baixo custo
- Gigabit Ethernet - fácil upgrade

Usando switches Ethernet para conectar computadores

Mais especializadas/Maior Performance

- Myrinet - 2.4 Gbits/sec - desvantagem: vendedor único
- cLan
- SCI (Scalable Coherent Interface)
- QsNet

Veja bibliografia sobre Beowulf para detalhes.

Programação Paralela por Troca de Mensagens
Softwares para Clusters

Parallel Virtual Machine (PVM) - desenvolvido no fim dos anos 80.

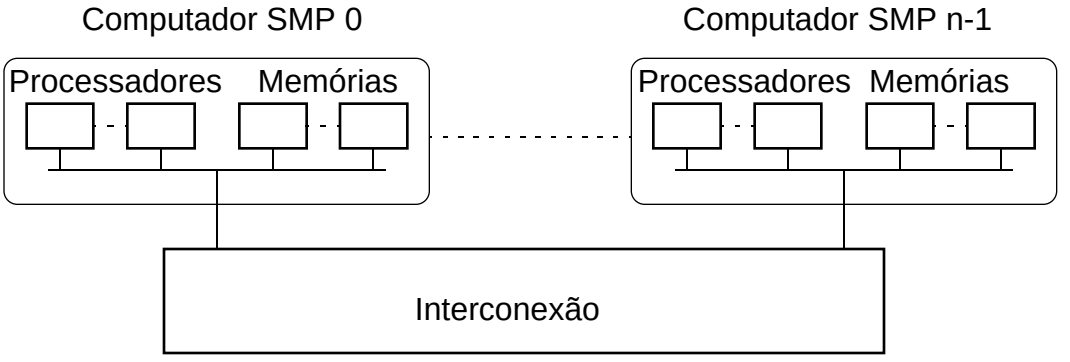
Foi muito popular.

Message-Passing Interface (MPI) - Padrão definido nos anos 90.

Ambos fornecem bibliotecas a nível de usuários para troca de mensagens em linguagens tipo C, C++, Fortran ... .

Cluster SMP

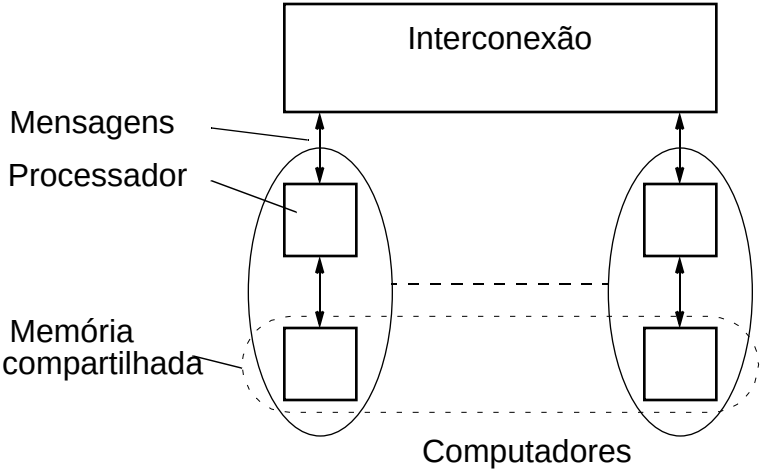
Cluster de compudadores de memória compartilhada (Multiprocessadores Simétricos)



Memória Compartilhada Distribuída

Faz a memória do cluster ver com se ela fosse uma memória única com um único espaço de endereçamento.

Para usar as técnicas de programação c/ memória compartilhada.



Classificação de Flynn

Flynn (1966) criou uma classificação para computadores baseda no fluxo de instruções e no fluxo de dados:

Single instruction stream-single data stream (SISD)

No computador convencional um fluxo único de instruções é gerado do programa. As instruções operam em um único fluxo de dados. Flynn chamou este computador convencional de computador s*ingle instruction stream-single data stream* (SISD).

Multiple Instruction Stream-Multiple Data Stream (MIMD)

Multiprocessador de propósito geral - cada processador tem um programa separado e um fluxo de instruções é gerado de cada programa para cada processador. Cada instrução opera sobre dados diferentes.

Tanto sistemas de memória compartilhada quanto de troca de vistos até agora são do tipo MIMD.

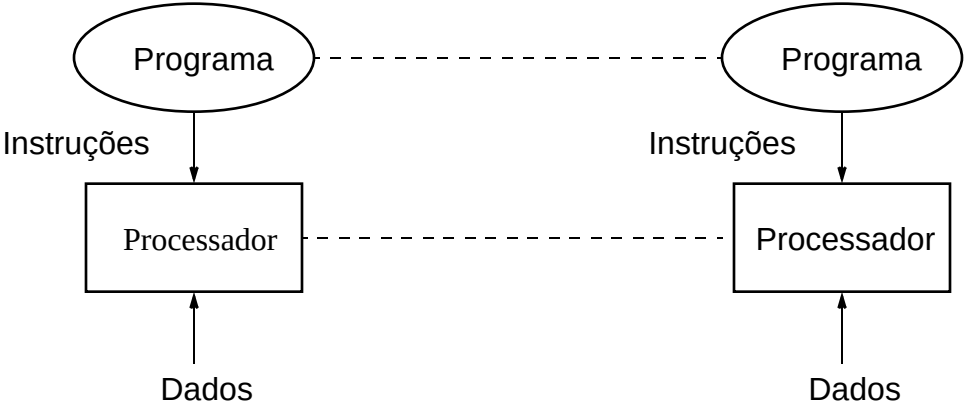
Single Instruction Stream-Multiple Data Stream (SIMD)

Computador projetado para ter um único fluxo de instrução vindo de um único programa, mas com multiplos fluxos de dados. As instruções do programa são disctibuídas para mais de um processador. Cada processador executa a mesma instrução sincronamente, mas usando dados diferentes.

Há várias aplicações importantantes que operam sobre vetores contendo dados (processamento vetorial).

Multiple Program Multiple Data (MPMD)

Similar ao sistema MIMD que vimos, onde cada processador terá seu próprio programa para executar:



Single Program Multiple Data (SPMD)

Um código único é escrito e cada processador executarár sua cópia individual do programa, embora independentemente e de de forma assíncrona.

O programa pode ser construído de forma que partes do mesmo sejam executadas por certos computadores e não por outros dependendo da identidade do computador.

Fator de Speedup

$$S(n) = \frac{\text{Tempo de execução usando um único processador}}{\text{Tempo de execução usando multiprocessador c/ n processadores}} = \frac{t_s}{t_p}$$

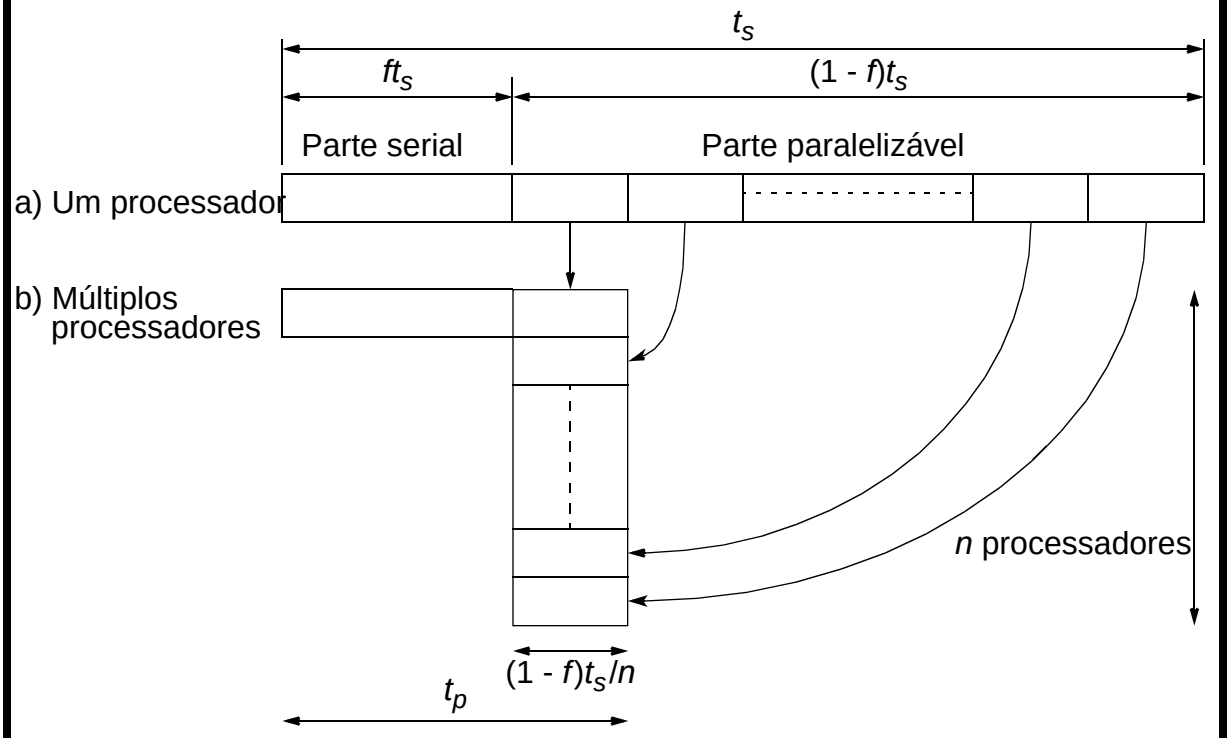
onde t_s é o tempo de execução em um processador e t_p é o tempo de execução no multiprocessador. $S(n)$ dá o ganho usando um multiprocessador. Algoritmos para a implementação paralela podem ser (e geralmente são) diferentes.

Speedup poder ser calculado em passos computacionais:

$$S(n) = \frac{\text{Número de passos usando um processador}}{\text{Número de passos paralelos usando n processadores}}$$

Speedup Máximo é (geralmente) n com n processadores *(speedup* linear).

Speedup Máximo - lei de Amdahl

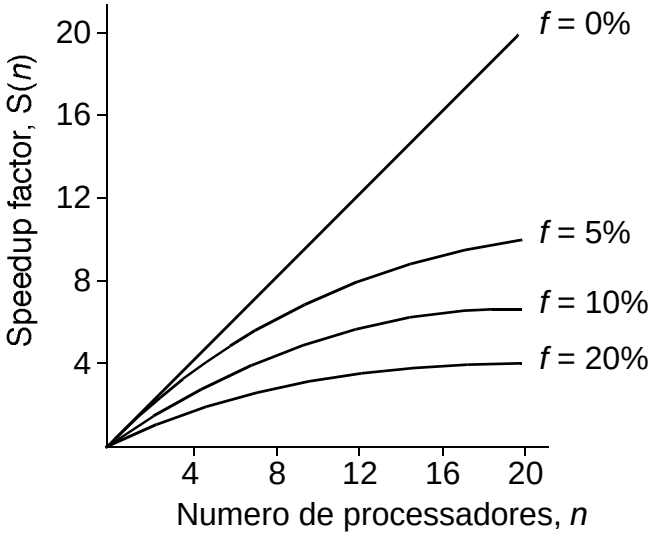


O Speedup é dado por:

$$S(n) = \frac{t_s}{ft_s + (1-f)t_s/n} = \frac{n}{1 + (n-1)f}$$

Esta equação é conhecida como lei de *Amdahl*

Speedup X número de processadores



Mesmo c/ infinitos processadores, o speedup máximo é limitado por 1/*f*.

Ex.: Com apenas 5% de computação sendo serial, o speedup máximo é 20, independente do número de processadores.

Final