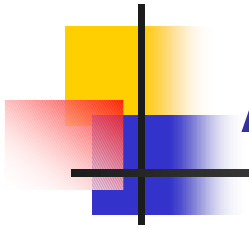
Algoritmos Paralelos usando CGM/MPI: Uma Introdução

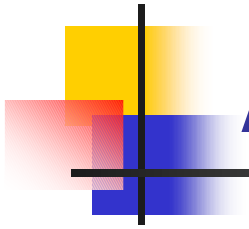
Edson Norberto Cáceres e Siang Wun Song –
DCT/UFMS e DCC/IME/USP

DCC-IME-USP - Aula 02



Aula 2 - Objetivos

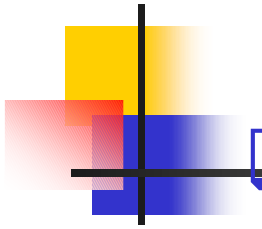
- Introduzir o conceito de um sistema de computação paralela e distribuída
- Descrever os principais modelos de computação paralela
- Introduzir o conceito de avaliação de um algoritmo paralelo



Aula 2 - Objetivos

- Apresentar os principais modelos de computação Realística (BSP/CGM)
- Analisar a eficiências dos algoritmos nos respectivos modelos
- Efetuar a comparação dos modelos

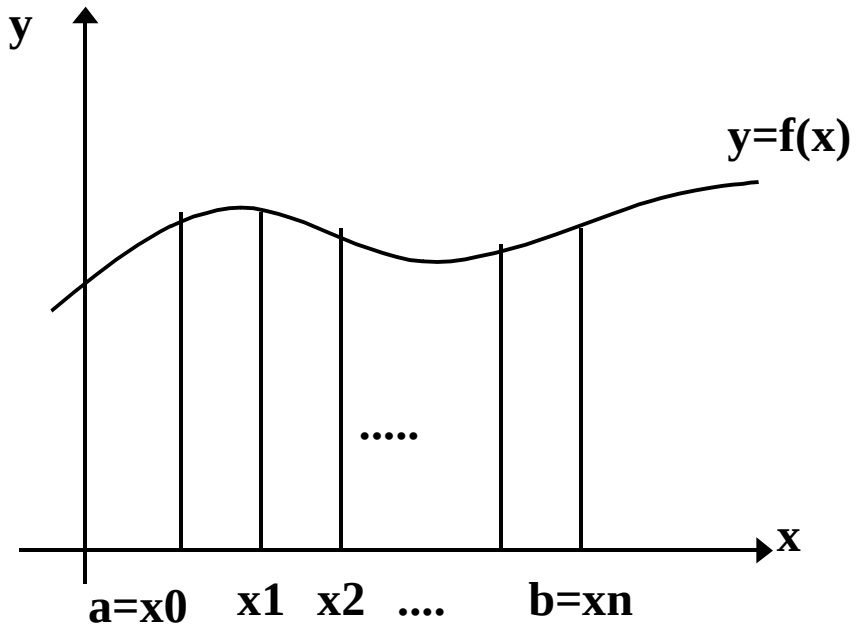
Algoritmos CGM / MPI



- Introdução
  - Sistemas de computação paralela e distribuída
  - Algoritmos paralelos e complexidade
  - Modelos de computação paralela e distribuída
    - Modelo de memória compartilhada
    - Modelo de rede
  - Modelos Realísticos
    - Comparação

Algoritmos CGM / MPI

- Área sob a curva y = f(x)



Algoritmos CGM / MPI

$$\int_a^b f(x)dx$$

$$\frac{1}{2}h[f(x_{i-1}) + f(x_i)]$$

$$[f(x_0)/2 + f(x_n\}/2 + f(x_1) + \cdots + f(x_n)]h$$

- Processador p_i calcula um conjunto de $f(x_i)$
- Algum processador calcula o Resultado Final

Sistemas de Computação Paralela

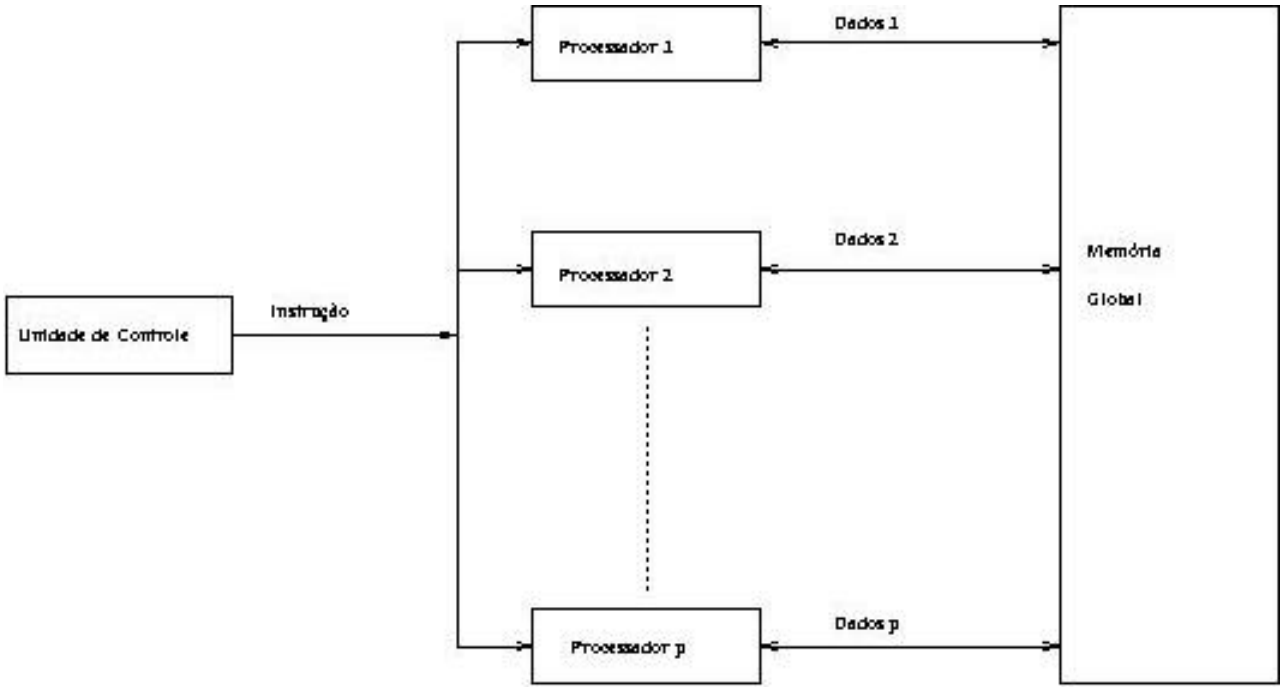
- Um sistema de computação paralela e distribuída é uma coleção de processadores interconectados de maneira a permitir a coordenação de suas atividade e a troca de dados.
- Os processadores trabalham simultaneamente, de forma coordenada para resolver um problema.
- Atuam onde a computação seqüencial não consegue obter solução em tempos razoáveis.

Sistemas de Computação Paralela

- Os sistemas de computação paralela podem ser classificados de acordo com suas características de arquitetura e modos de operação.
- Classificação de Flynn (1966):
  - SISD (Single Instruction stream, Single data stream).
  - MISD (Multiple Instruction stream, Single data stream).
  - SIMD (Single Instruction stream, Multiple data stream).
  - MIMD (Multiple Instruction stream, Multiple data stream).

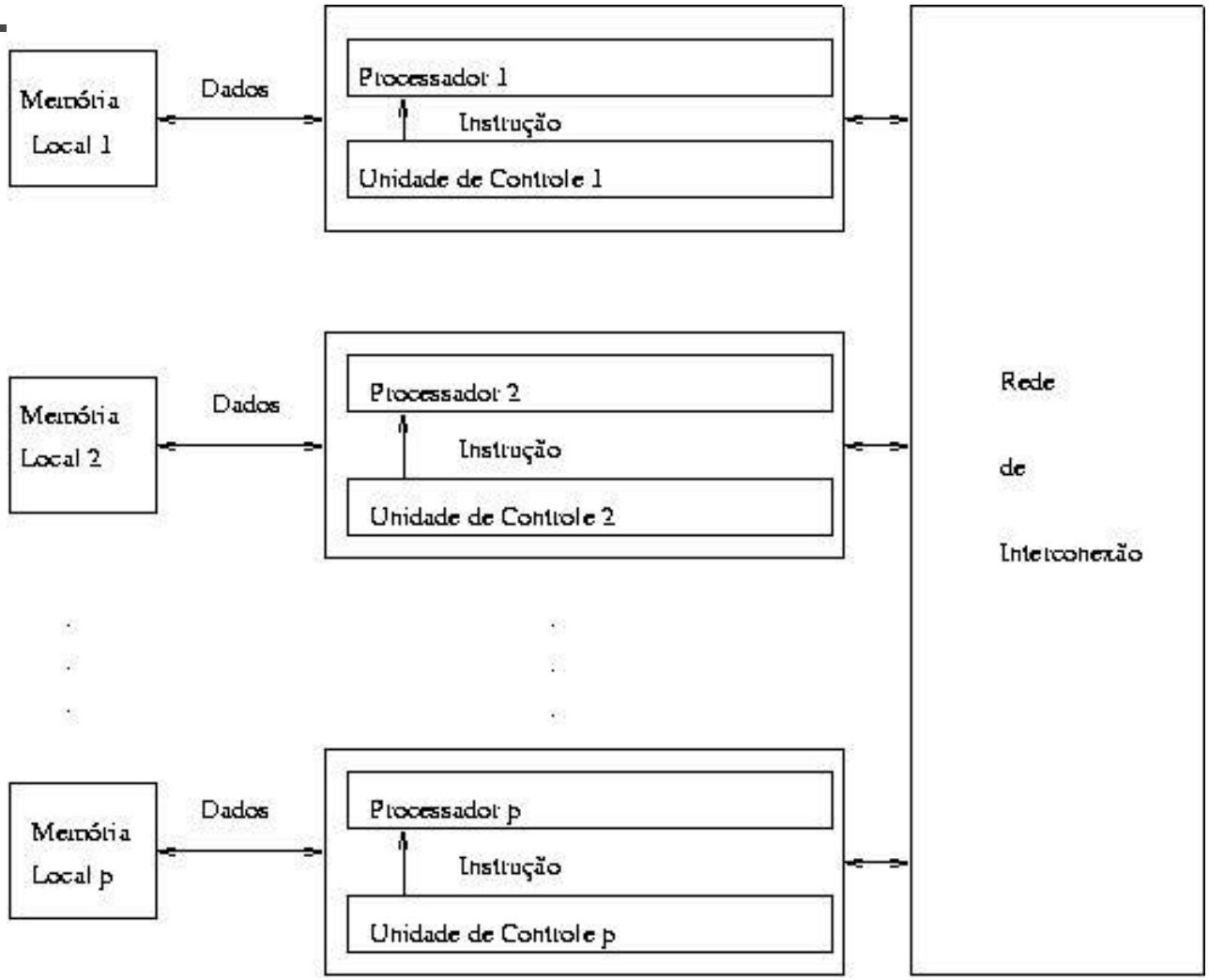
Sistemas de Computação Paralela

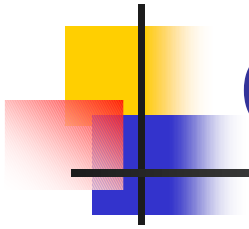
- O modelo SIMD



Sistemas de Computação Paralela

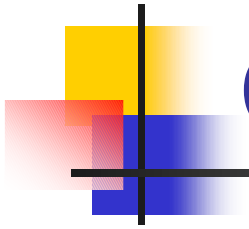
- O modelo MIMD





Outras Classificações

- **Dispersão dos Processadores**
  - **Geográfica**
    - Redes de Computadores
    - Sistemas Distribuídos
  - **Confinados**
    - Máquinas Paralelas
  - **Estrutura de Interconexão**
    - Topologia



Outras Classificações

- **Sincronismo**
  - **Síncrono**
    - **Único relógio global**
  - **Assíncrono**
    - **Não existe um relógio global**

Sistemas de Computação Paralela

- Processadores: ativos ou inativos
  - Ativos: executando alguma tarefa para colaborar na solução de um problema
- Acesso do Processador i aos dados do Processador j – Comunicação
  - Arquitetura da Rede de Interconexão

Sistemas de Computação Paralela

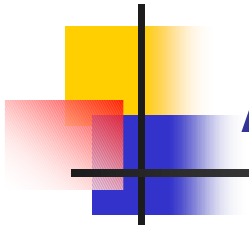
- Existem diferentes tipos de arquiteturas paralelas
- Para cada arquitetura paralela
  - Modelos distintos de desenvolvimento de algoritmos paralelos
- Modelos de computação paralela e distribuída

Modelos de Computação Paralela

- Não existe um modelo que seja amplamente aceito
- Eficiência de um algoritmo depende de um conjunto de fatores da arquitetura
  - Concorrência
  - Alocação e escalonamento de processos
  - Comunicação
  - Sincronização

Modelos de Computação Paralela

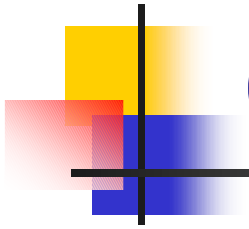
- Memória compartilhada
  - PRAM
- Rede
  - Anel
  - Hipercubo
- Realístico
  - BSP
  - CGM



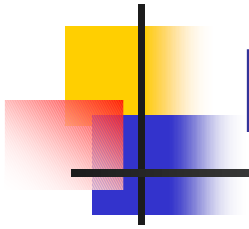
Algoritmos Paralelos

- Muito trabalho na área de ambientes de programação e sistemas
- Modelos teóricos
- Pouca atenção para metodologias de projeto de algoritmos para máquinas paralelas “reais” – redes de workstations - portabilidade

Algoritmos e Computadores Paralelos



- Duas arquiteturas básicas
  - Memória distribuída
  - Memória compartilhada
- Abordagens
  - Data-parallel languages (HPF)
  - Troca de mensagens
  - Threads

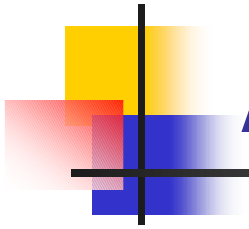


Projeto de Algoritmos

- Decomposição do domínio
  - Paralelismo nos dados
  - SPMD – Single Program Multiple data
- Decomposição funcional
  - Paralelismo de tarefas
  - MPMD – Multiple Program Multiple data
  - Cliente-servidor

Análise de Algoritmos

- ***A* é um algoritmo paralelo que resolve *P***
- **Speedup**
  - $S_p(n) = t^*(n) / T_p(n)$
  - $T^*(n)$ – complexidade seqüencial
  - $T_p(n)$ – complexidade paralela com p processadores



Algoritmos Paralelos

Para *x* em *X* em paralelo faça.

Instrução 1;

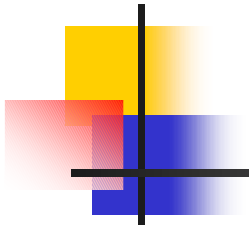
Instrução 2;

.

.

Instrução k;

- **Síncrono – a instrução *i* só é iniciada após todos os processadores finalizarem a instrução *i*-1.**

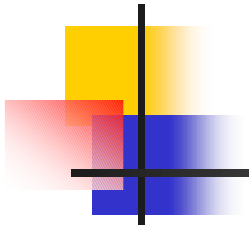


Modelos de Computação

- **Objetivo do projeto de um algoritmo paralelo: obter um desempenho superior com relação a versão seqüencial.**
- **Fatores a serem considerados: balanceamento de carga, minimização de comunicação e sobreposição de comunicação e computação.**

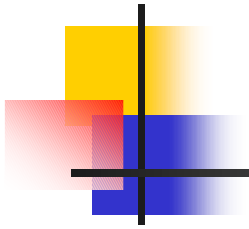
Modelos de Computação

- **Balanceamento de carga: dividir equitativamente o trabalho entre os processadores.**
- **Tempo de execução total: tempo de computação, tempo ocioso e tempo de comunicação.**
  - **Tempo de comunicação: latência (preparação dos pacotes) e largura de banda (velocidade real de transmissão).**



Modelos de Computação

- **Seqüencial:** acesso a qualquer posição da memória em tempo constante.
- **Paralelo:** depende da existência de uma memória global (compartilhada), eficiência da rede de interconexão, e latência e largura da banda (no caso de não haver uma memória global).

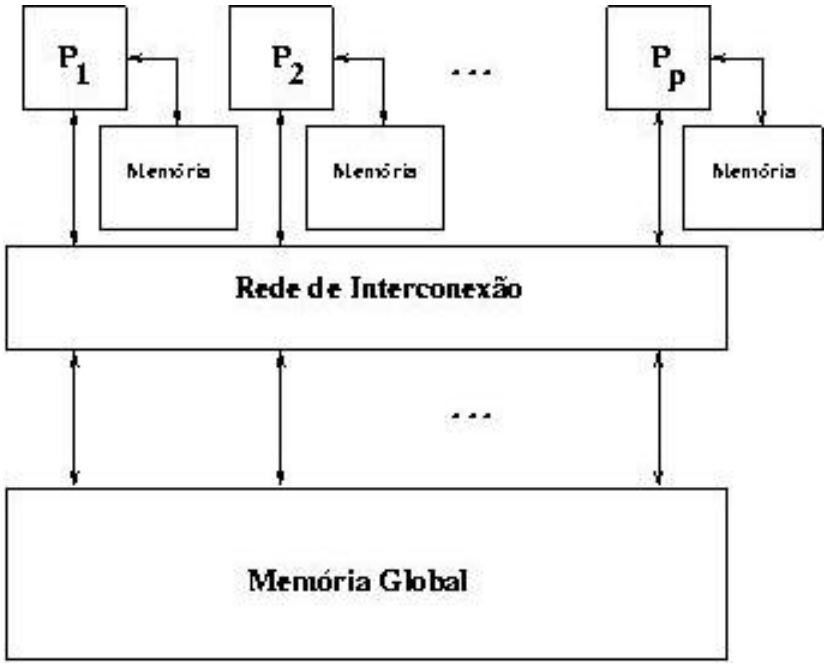


Modelos de Computação

- **Modelo algorítmico: além dos aspectos levantados, deve ter**
  - **Simplicidade**
  - **Implementabilidade**
- **Modelo de memória compartilhada**
- **Modelo de rede**
- **Modelo Realístico**

Modelo de Memória Compartilhada

- **Consiste em um conjunto de processadores idênticos p_i, $1 \leq i \leq p$, cada um com sua memória local. Toda comunicação é feita através de uma memória global.**



Modelo de Memória Compartilhada

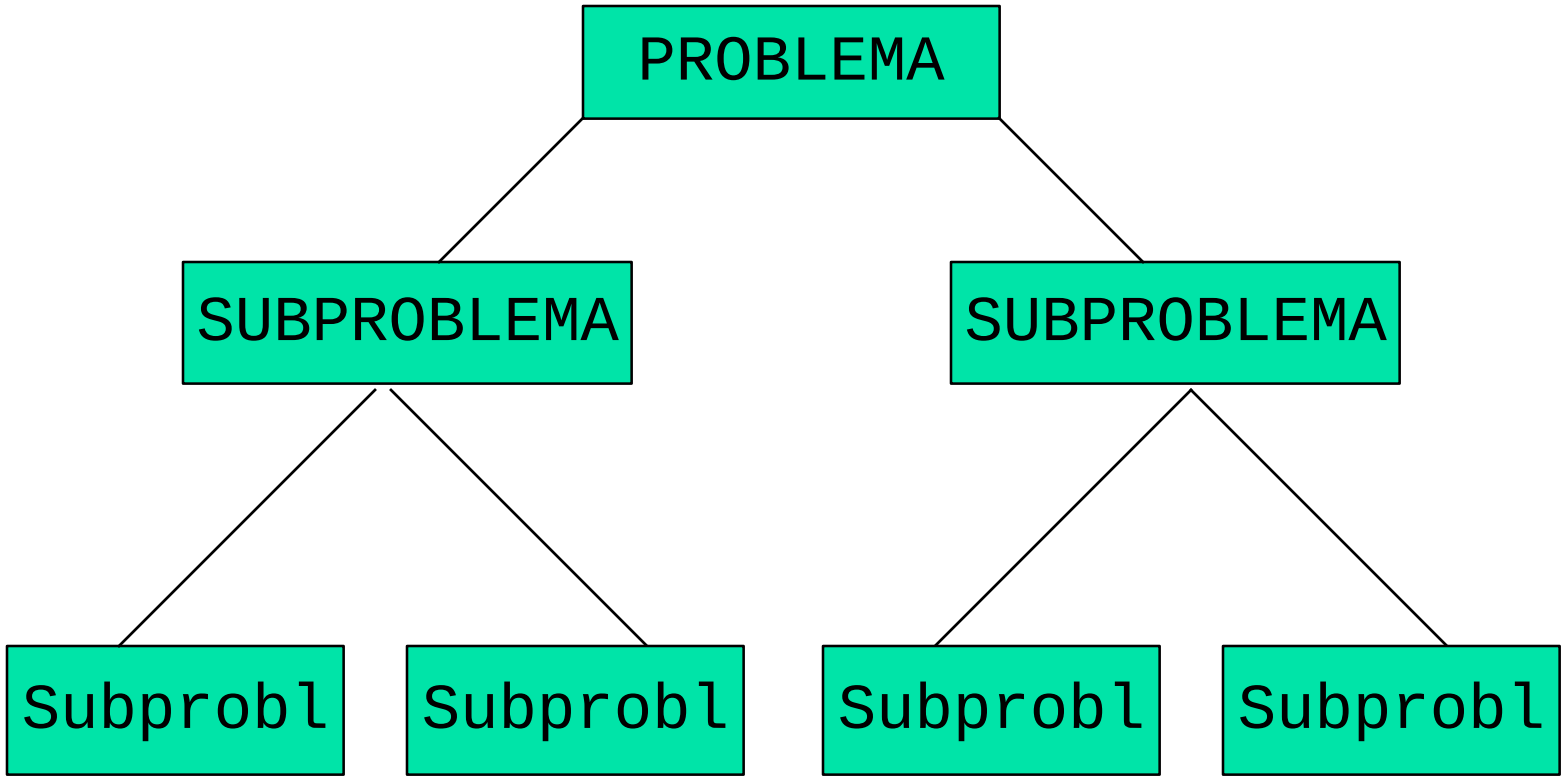
- Modos de operação: síncrono e assíncrono
- O modelo síncrono de memória compartilhada SIMD/MIMD é conhecido como PRAM (*Parallel Random access Machine*):
  - EREW (*exclusive Read, exclusive Write*)
  - CREW (*Concurrent Read, exclusive Write*)
  - CRCW (*Concurrent Read, Concurrent Write*):
    - Escrita comum
    - Escrita arbitrária
    - Escrita prioritária

Algoritmo da Soma - PRAM

- Entrada: um vetor a=[v(1), v(2),...,V(n)]
- Saída: s=v(1)+v(2)+...+V(n)
- Memória compartilhada
- N processadores

- Global read(x, Y): move da compartilhada para local
- Global write(u, V): move da local para compartilhada

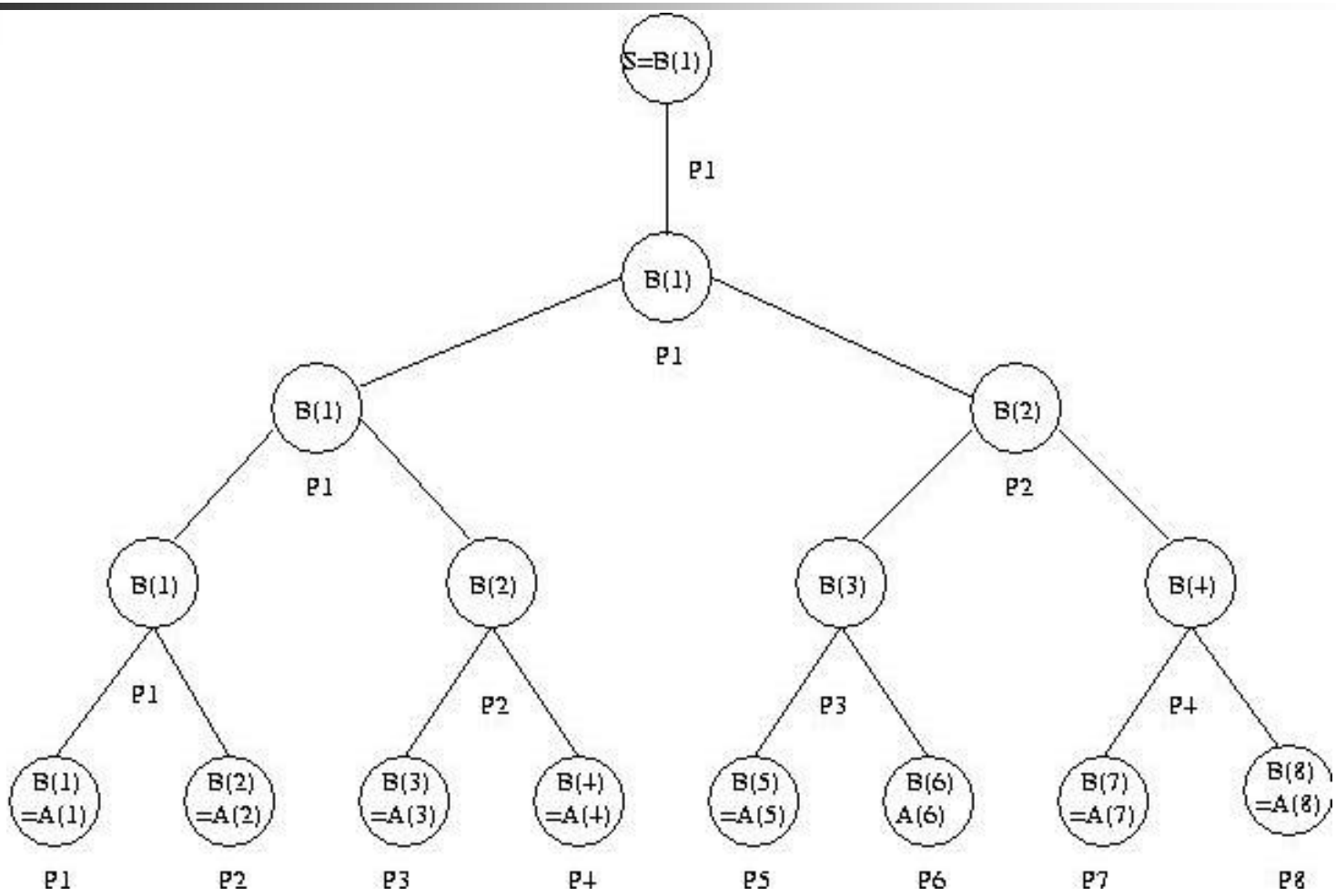
Árvore Binária Balanceada



Soma no Modelo PRAM

1) Global read($a(i)$, a)
2) Global write(a, $b(i)$)
3) Para $h = 1$ a log n faça
   a) Se ($i <= n / 2^h$) então
      i. Global read($b(2i-1)$, x)
      ii. Global read($b(2i)$, y)
      iii. $Z := x + y$
      iv. Global write(z, $b(i)$)
   b) Se $i = 1$ então global write(z, S)

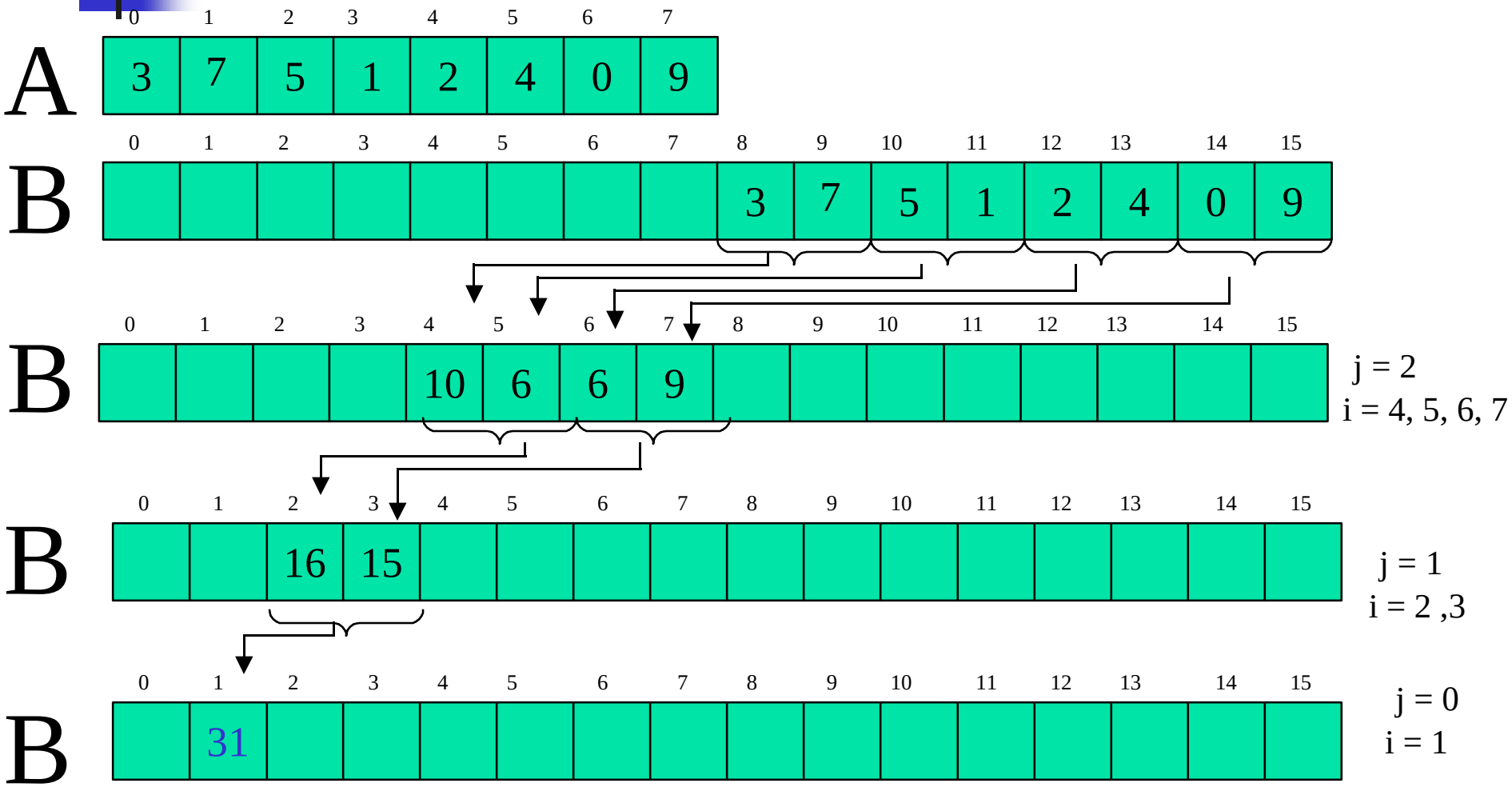
Algoritmo da Soma - PRAM

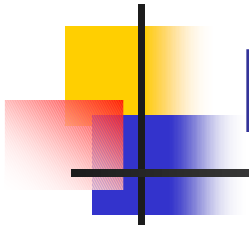


Algoritmo da Soma - PRAM

```
/* Passo 1: vetor A é copiado para 2ª metade de B */
PARA 0 <= i <= n-1 FAÇA EM PARALELO
    B[n+i] := a[i];
/* Passo 2: laço seqüencial, para cada nível da árvore */
PARA j = log2n - 1 ATÉ 0 FAÇA
  /*loop paralelo alocando um processador para cada subproblema deste nível*/
    PARA 2^j <= i <= 2^(j+1)-1 FAÇA EM PARALELO
        B[i] := soma(b[2i], b[2i+1]);
```

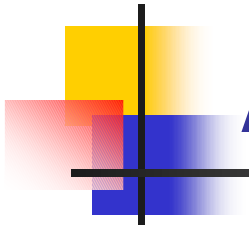
Algoritmo da Soma - PRAM





Eficiência no Modelo PRAM

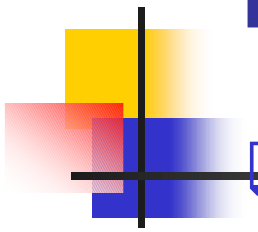
- Um algoritmo é **eficiente** no modelo PRAM se ele for executado em tempo **polilogarítmico** ($o(log^k n)$, k em N) com um número **polinomial** (n^p p em N) de processadores.
- A **comunicação** não é levada em conta.



Algoritmo da Soma - PRAM

- **Submodelo: EREW**
- **Tempo: $o(log_2 n)$**
- **Processadores: n**
- **Custo (tempo x processadores): $o(n log_2 n)$**

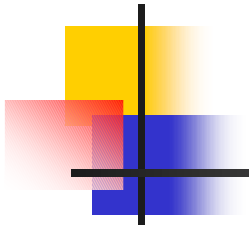
Modelo de Rede



- **Modelo de rede:** também chamado de **modelo de memória distribuída**.
- **Comunicação:** incorpora a topologia da **rede de interconexão** (conjunto de canais que ligam os processadores).
- **Modelo de rede:** consiste de um conjunto de processadores onde a **memória está distribuída** entre os processadores e nenhuma memória global está disponível. A **comunicação** é feita através da **troca de mensagens**.

Modelo de Rede

- O processador *i* não possui acesso à posição da memória local (identificada por um endereço) do processador *j*.
- Os processadores podem estar ativos ou inativos.
- Um processador ativo executa uma instrução (pode ser diferente das dos demais processadores ativos) – MIMD.

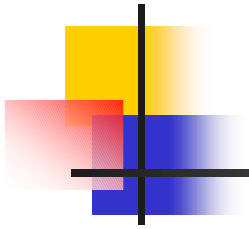


Modelo de Rede

- Abordagem de troca de mensagens: a divisão dos dados e das tarefas, além do gerenciamento das comunicações entre eles, é de responsabilidade do programador.
- Roteamento: processo de entregar cada mensagem da sua fonte ao seu destino.

Modelo de Rede

- **send(*x*,*i*): envia uma cópia de *X* ao processador p_i e passa a executar a próxima instrução (não bloqueante).**
- **receive(*y*,*j*): suspende a execução do seu programa até que os dados do processador P_j sejam recebidos. O processador armazena os dados e continua a execução do programa (bloquenate).**

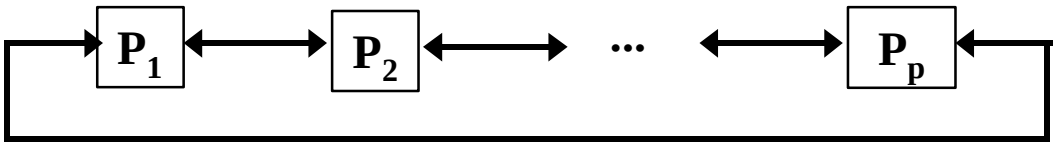


Modelo de Rede

- Lista (array) linear
- Anel
- Mesh
- Hipercubo

Lista (array) Linear e Anel

- **Lista linear: *p* processadores, p_i esta conectado a p_{i-1} e p_{i+1}, sempre que existirem.**
- **Anel: p_1 e p_p estão conectados.**



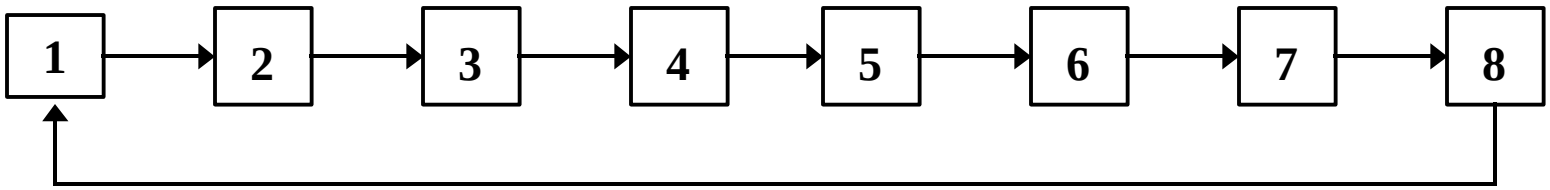
Soma de um Vetor no Anel

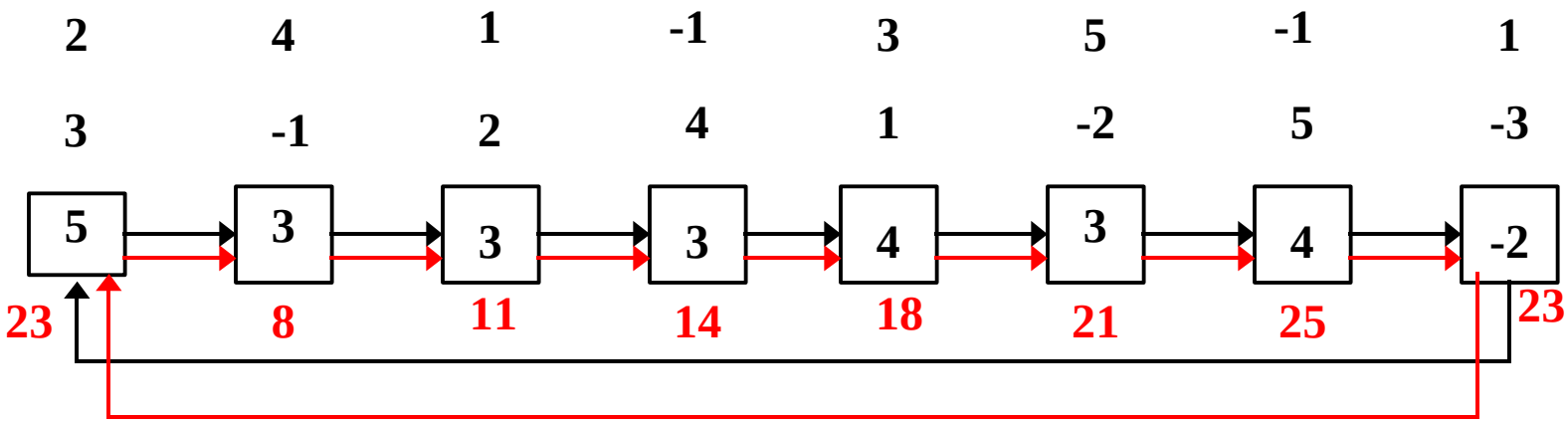
Entrada: O número do processador *i*; O número *p* de processadores; O *i*-ésimo sub-vetor *B* = *a*((*i*-1)*r* + 1: *ir*) de tamanho *r*, onde *r* = *n*/*p*.

Saída: processador p_i calcula o vetor $s=s_1+s_2+...+S_i$ e passa o resultado para a direita. Quando o algoritmo termina, P_1 terá a soma *S*.

1) Z := b[1] + ... + b[r];
2) Se i = 1 então S := 0;
   Caso contrário receive(s, esquerda);
3) S := s+z;
4) send(s, direita);
5) Se i = 1 então receive(s, esquerda);

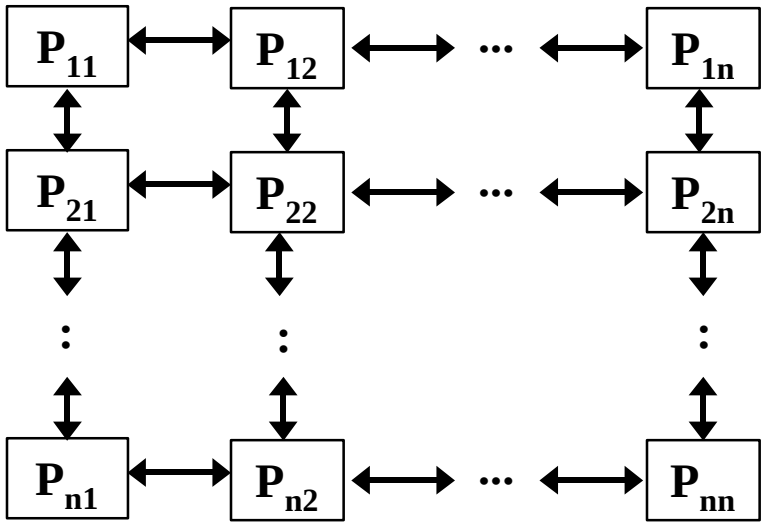
Soma de um Vetor no Anel



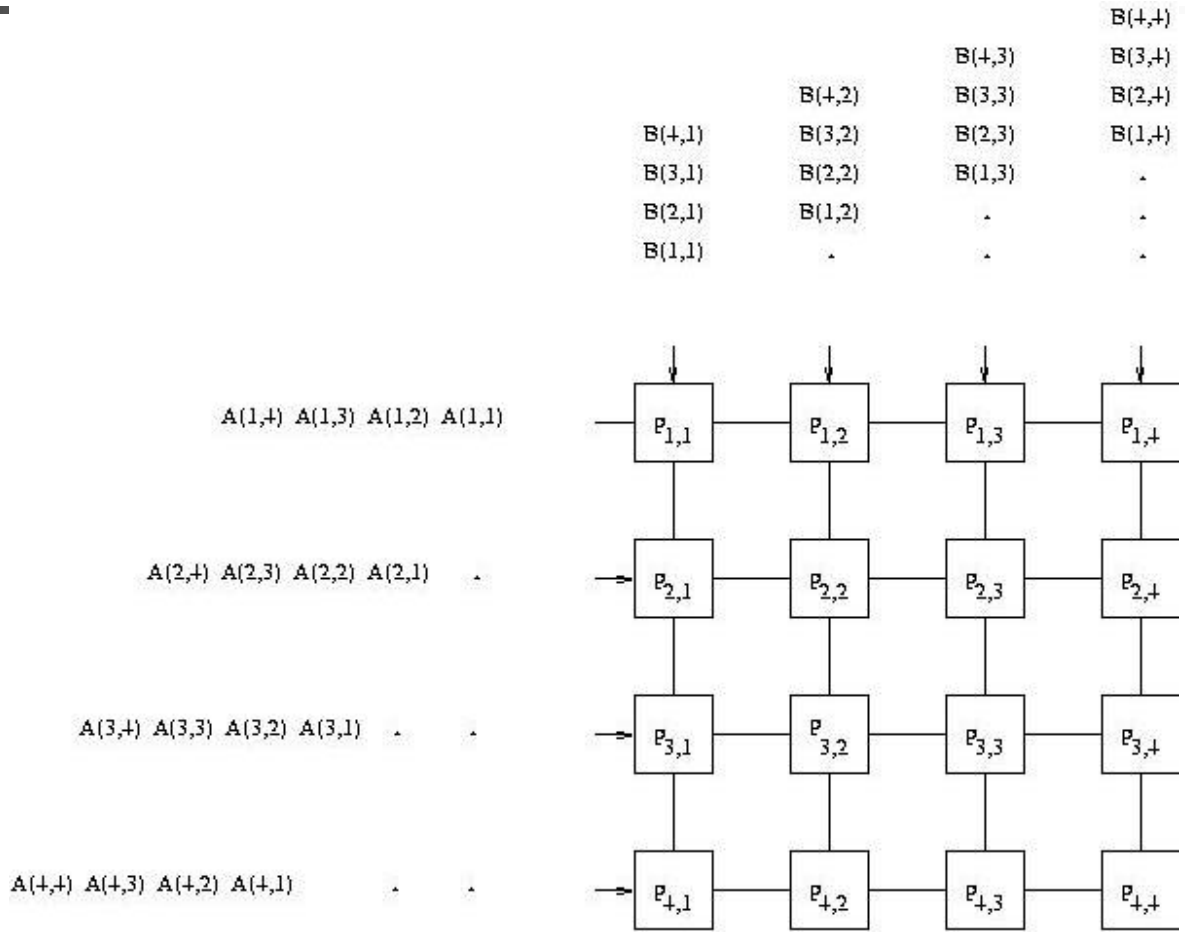


Mesh

- O mesh é uma versão bi-dimensional do vetor linear, $p = m^2$ processadores mxm. $P_{i,j}$ está conectado aos processadores $P_{i+-1,j}$ e $P_{i,j+-1}$ se eles existirem.



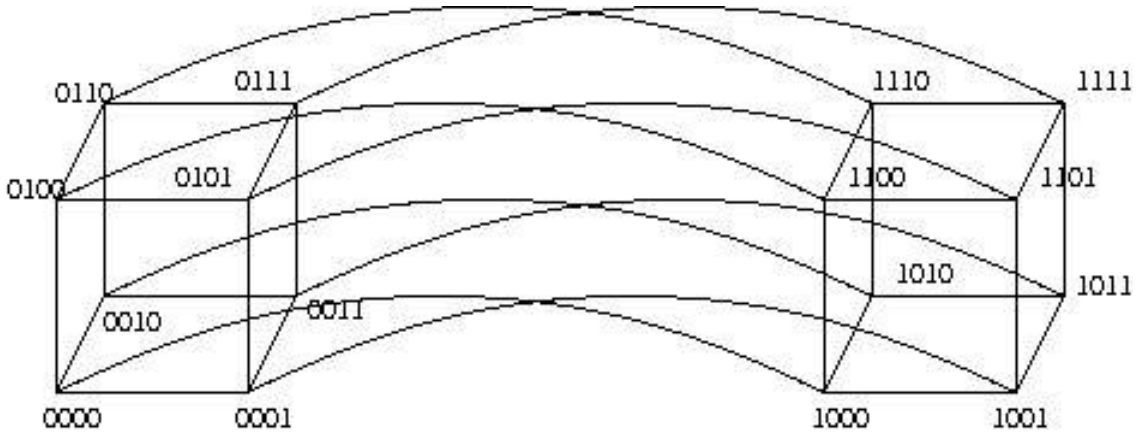
Multiplicação no Mesh



Hipercubo

- $P = 2^d$ processadores interconectados por um cubo booleano.
- $I = i_{d-1} i_{d-2} \ldots I_0$, $i_j = 0,1$ e $0 <= i <= p-1$.
- P_i esta conectado a $p_i^{(j)}$.
- $I^{(j)} = i_{d-1} \ldots \hat{i}_j \ldots I_0$ e $\hat{i}_j = 1 - i_j$, $0 <= j <= d-1$.
- Dois processadores estão conectados se a representação binária de seus índices diferem em apenas um bit.

Hipercubo


0110
0111
1110
1111
0100
0101
1100
1101
1010
1011
0010
0011
0000
0001
1000
1001

Soma no Hipercubo - Síncrono

Entrada: um vetor A de $n = 2^d$ elementos tal que a(i) está armazenado na memória local de p_i, $0 <= i <= n-1$.

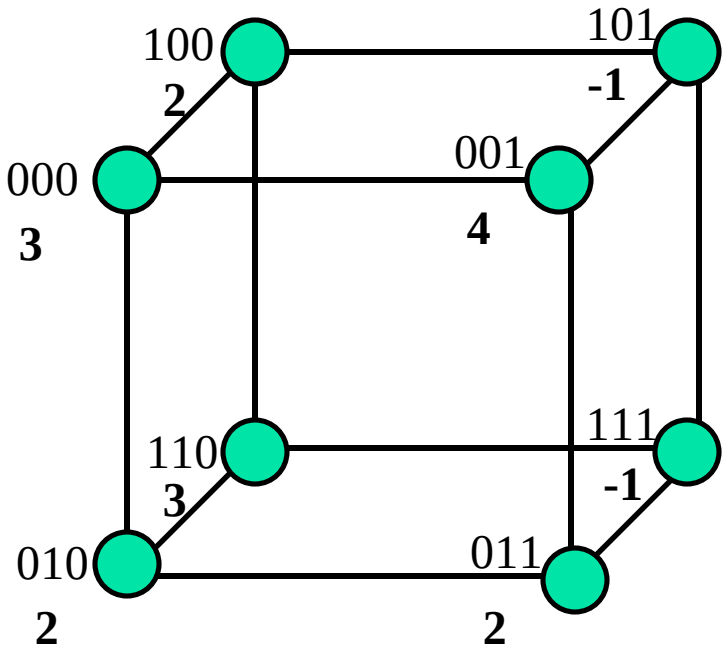
Saída: $S = A(0)+...+A(n-1)$ em p_0.

1) **Para i := d-1 até 0 faça.**

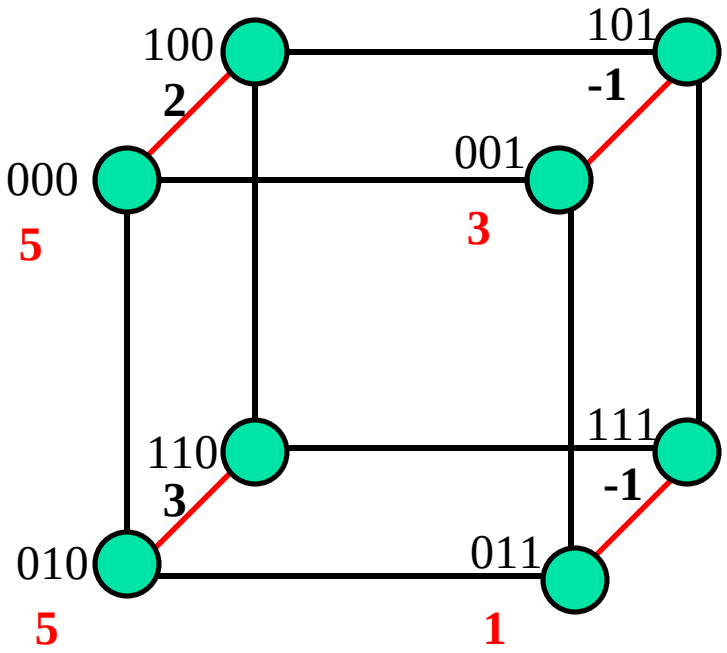
   Se $0 <= i <= 2^l$ então.

   $A[i] := a[i] + a[i^{(l)}]$;

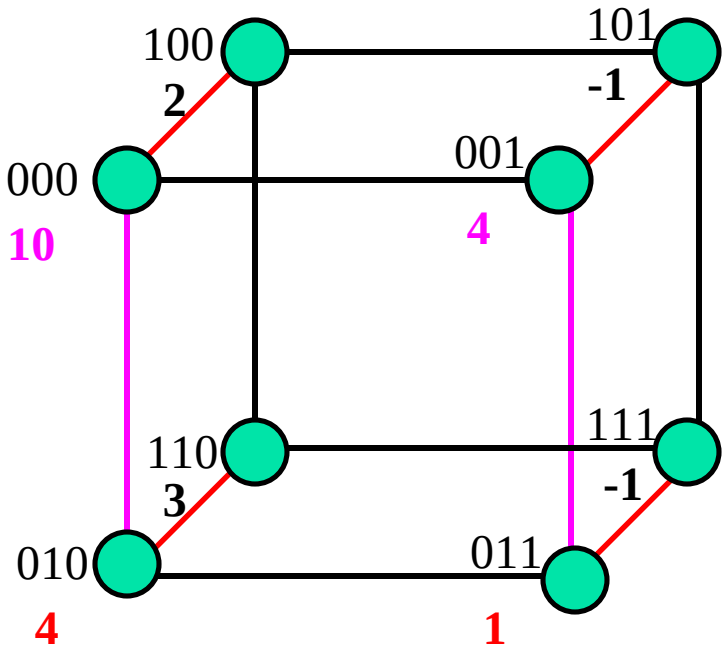
Hipercubo



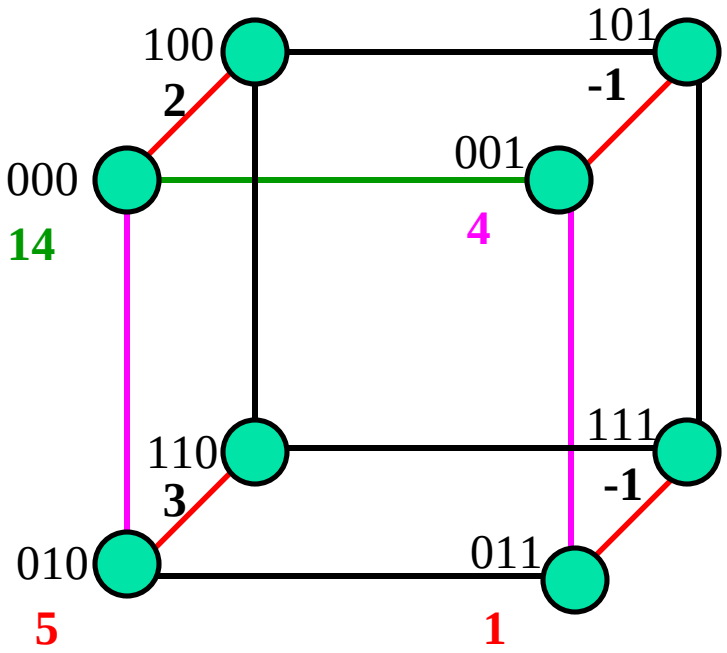
Hipercubo



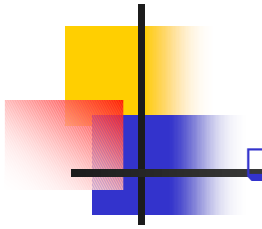
Hipercubo



Hipercubo

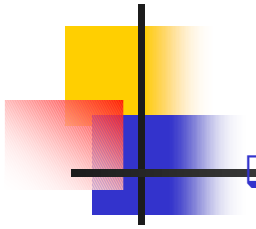


Algoritmos CGM / MPI

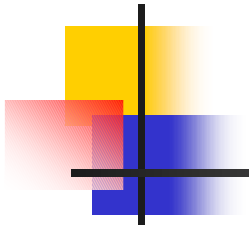


- Introdução
  - Sistemas de Computação Paralela e Distribuída
  - Algoritmos Paralelos e Complexidade
  - Modelos de Computação Paralela e Distribuída
    - Modelo de Memória Compartilhada
    - Modelo de Rede
    - Modelos Realísticos
    - Comparação
  - Implementação de Algoritmos Paralelos
    - MPI
    - PVM

Algoritmos CGM / MPI

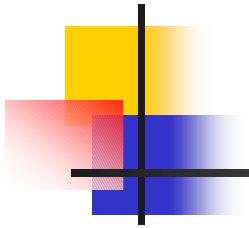


- Técnicas Básicas
  - Soma de Prefixos
  - Ordenação
  - List Ranking
- Alguns Algoritmos para Problemas em Grafos
  - Euler Tour em Árvores
  - Ancestral Comum mais Baixo
  - Mínimo Intervalar



Modelos Realísticos

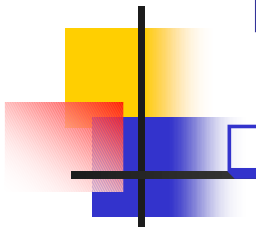
- No final dos anos 80, a área de algoritmos paralelos atravessava uma série crise.
- Vários resultados teóricos e para máquinas específicas (meshes e hipercubos).
- Resultados teóricos obtinham speedups desapontadores.
- Programas sem portabilidade.



Modelos Realísticos

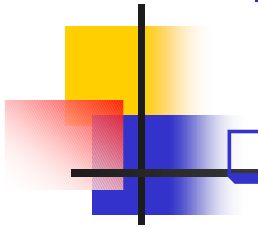
- Anos 90: mudança significativa com a introdução de modelos de computação de granulosidade grossa.
- BSP – Bulk Synchronous Parallel Model.
- CGM – Coarse Grained Multicomputers.

Modelo BSP



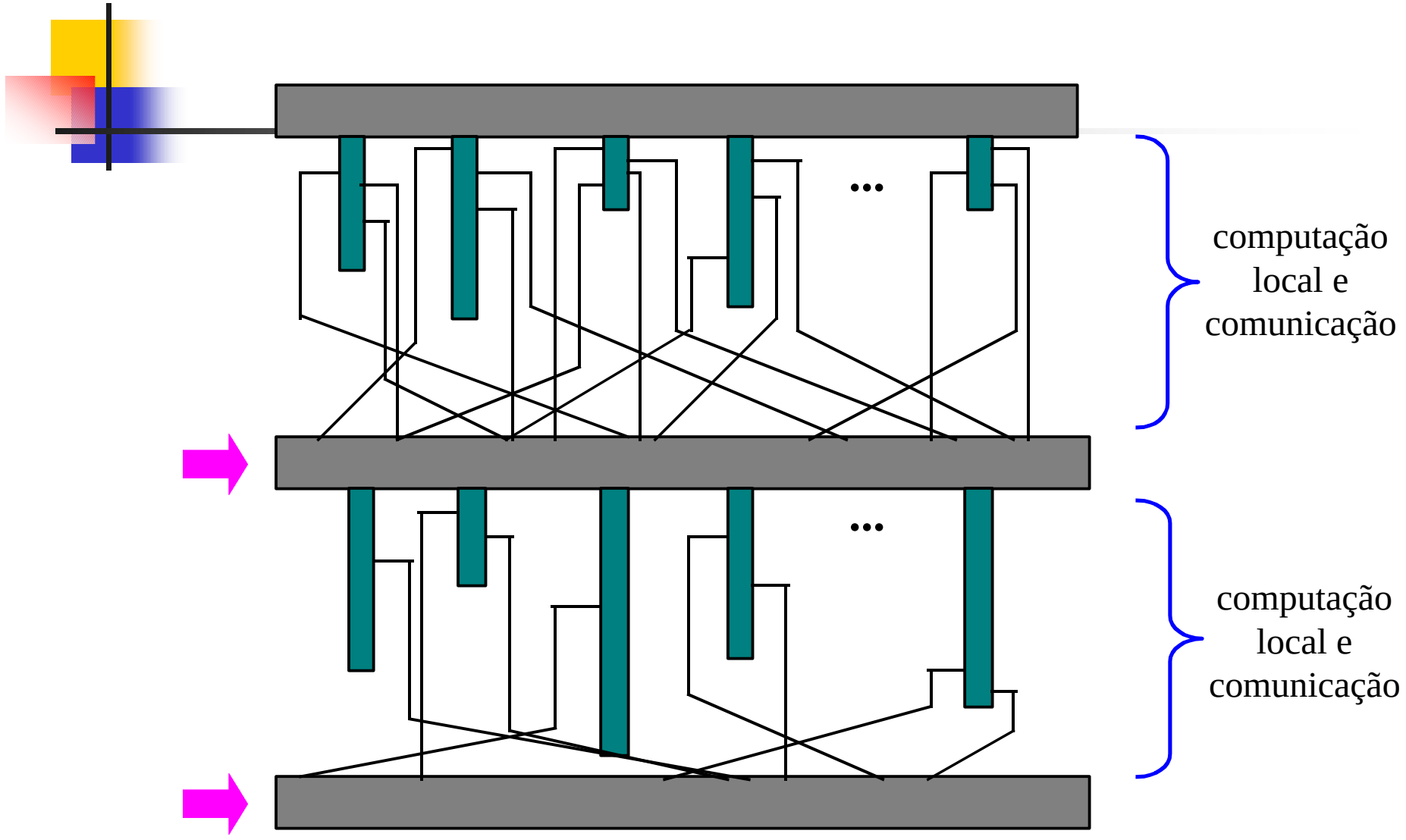
- O modelo BSP (*Bulk Synchronous Parallel*) foi introduzido por Valiant.
- Uma máquina BSP consiste de *p* processadores com memória local.
- Os processadores se comunicam através de um meio de interconexão, gerenciados por um roteador.
- Também oferece a capacidade de sincronizar os processadores em intervalos regulares de *L* unidades de tempo.

Modelo BSP



- **Um algoritmo BSP consiste de uma seqüência de *superpassos*.**
- **Em cada superpasso cada processador executa uma combinação de computações locais, transmissão e recebimento de mensagens de outros processadores.**
- **Após cada período de *L* unidades de tempo, uma barreira de sincronização é realizada.**

Modelo BSP


computação
local e
comunicação
computação
local e
comunicação

Modelo BSP

- **Parâmetros:**
  - ***n*** **: tamanho do problema.**
  - ***p*****: número de processadores com memória local.**
  - ***L*****: tempo máximo de um superpasso (periodicidade) – latência.**
  - ***g*****: taxa de eficiência da computação/comunicação.**
- **Custo (superpasso *i*):** $w_i + gh_i + L$, $w_i = \max\{L, t_1, .., t_p\}$ e $h_i = \max\{L, c_1, \ldots, c_p\}$
- **Custo Total:** $W + gH + LT$, $W = \Sigma\, w_i$, $H = \Sigma\, h_i$,

Soma – BSP

Entrada: número do processador i; O número p de processadores; $B = a((i-1)r+1:ir)$, $r = n/p$.

Saída: p_i calcula $z = B(1)+...+B(n)$ e envia o resultado para p_1. P_1 calcula $S = z_1+...+z_p$.

1) Z := b[1] + ... + b[r];
2) Se i = 1 então S := z;
   Caso contrário send(z,p_1);
3) Se i = 1 então.
   Para i = 2 até p faça.
   receive(z,p_i);
   S := S + z;

Soma - BSP - Detalhes

- **Troca de mensagens:**
  - **Mestre/escravo.**
    - Mestre inicializa os escravos (processos) na máquina virtual e distribui os dados.
    - Cada escravo (processo) realiza a sua soma e envia o resultado para o mestre.
  - **SPMD.**
    - Processo 0 faz a distribuição dos dados.
    - Processo 0 faz a soma.
  - **MPMD.**
    - Processo i faz a distribuição dos dados.
    - Processo i faz a soma.

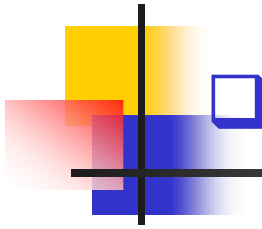
Soma – BSP – Complexidade

- **Passo 1: cada p_i efetua r operações.**
- **Passo 2: P_1 efetua uma operação e os demais p_i's enviam uma MSG.**
- **Passo 3: P_1 recebe p-1 mensagens e efetua p-1 operações.**
- **Uma sincronização.**
- **Um superpasso.**
- **$O(n/p)$.**

Modelo CGM

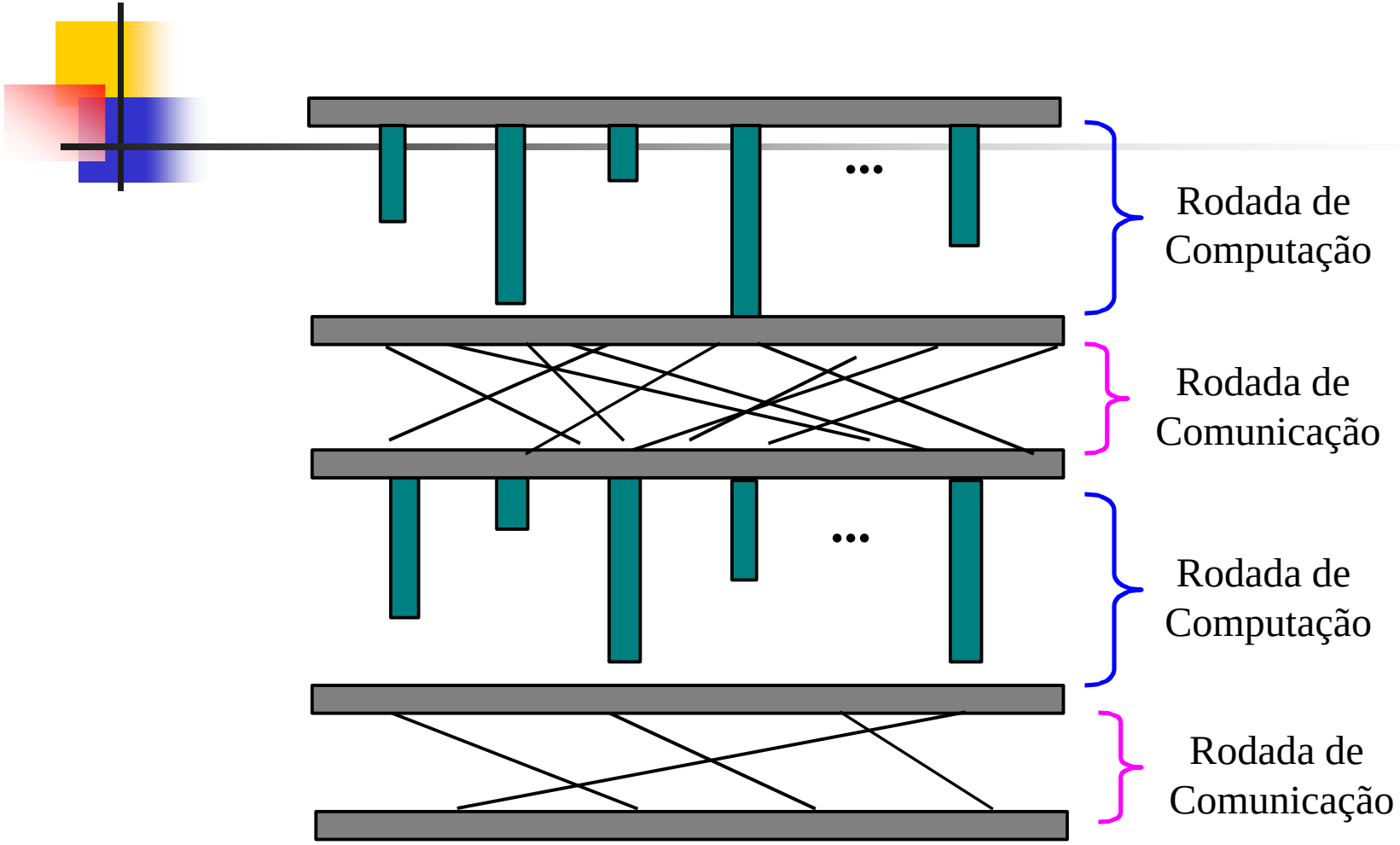
- O modelo CGM (*Coarse Grained Multicomputer*) foi proposto por Dehne et al.
- Utiliza apenas dois parâmetros:
  - *N*: tamanho da entrada.
  - *P*: número de processadores.
- Uma máquina CGM consiste de um conjunto de *p* processadores, cada um com memória local de tamanho $o(n/p)$. Os processadores se comunicam através de um meio de interconexão.

Modelo CGM



- **Um algoritmo CGM consiste de uma seqüência alternada de rodadas de computação e rodadas de comunicação separadas por uma barreira de sincronização.**
- **Na fase de comunicação cada processador troca no máximo um total de $o(n/p)$ dados com outros processadores.**
- **Custo (λ rodadas de comunicação):**
  - **Computação: análogo ao BSP.**
  - **Comunicação: $\lambda H_{n,p}$ onde $H_{n,p}$ é o custo único para cada rodada de comunicação.**

Modelo CGM


Rodada de Computação
Rodada de Comunicação
Rodada de Computação
Rodada de Comunicação

Soma – CGM

Entrada: número do processador i; O número p de processadores; $B = a((i-1)r+1:ir)$, $r = n/p$.

Saída: p_i calcula $z = B(1)+...+B(n)$ e envia o resultado para p_1. P_1 calcula $S = z_1+...+z_p$.

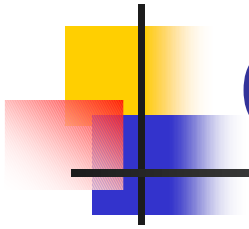
1) Z := b[1] + ... + b[r];
2) Se i = 1 então S := z;
   Caso contrário send(z,p_1);
3) Se i = 1 então.
   Para i = 2 até p faça.
   receive(s[i],p_i);
4) S := S + s[2] + ... s[p];

Soma - CGM - Detalhes

- **Troca de Mensagens:**
  - **Mestre/Escravo**
    - Mestre inicializa os escravos (processos) na máquina virtual e distribui os dados.
    - Cada escravo (processo) realiza a sua soma e envia o resultado para o mestre.
  - **SPMD**
    - Processo 0 faz a distribuição dos dados.
    - Processo 0 faz a soma.
  - **MPMD**
    - Processo i faz a distribuição dos dados.
    - Processo i faz a soma.

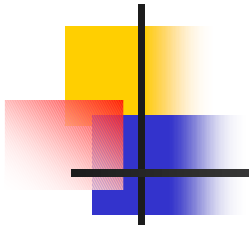
Soma – CGM – Complexidade

- **Passo 1: cada p_i efetua r operações.**
- **Passo 2: P_1 efetua uma operação e os demais p_i's enviam uma MSG.**
- **Passo 3: P_1 recebe p-1 mensagens.**
- **Passo 4: P_1 efetua p-1 operações.**
- **Uma rodada de comunicação.**
- **$O(n/p)$.**



Comparação

- Cada modelo apresenta vantagens, dependendo da situação.
- Memória compartilhada: projeto e análise.
- Rede: pouca portabilidade.
- Modelos Realísticos tem se mostrado mais adequados.



Implementação

- Arquitetura
- Formato dos dados
- Velocidade computacional
- Carga da máquina
- Carga da rede
- PVM – Parallel virtual Machine
- MPI – Message Passing interface