Foster, Ian

Designing and Building Parallel Programs:
Concepts and Tools for Parallel Software Engineering,
Addison-Wesley Pub. Company, Inc., New York, 1994.

Capítulo 2
Projetando Algoritmos Paralelos:
Particionamento e Comunicação

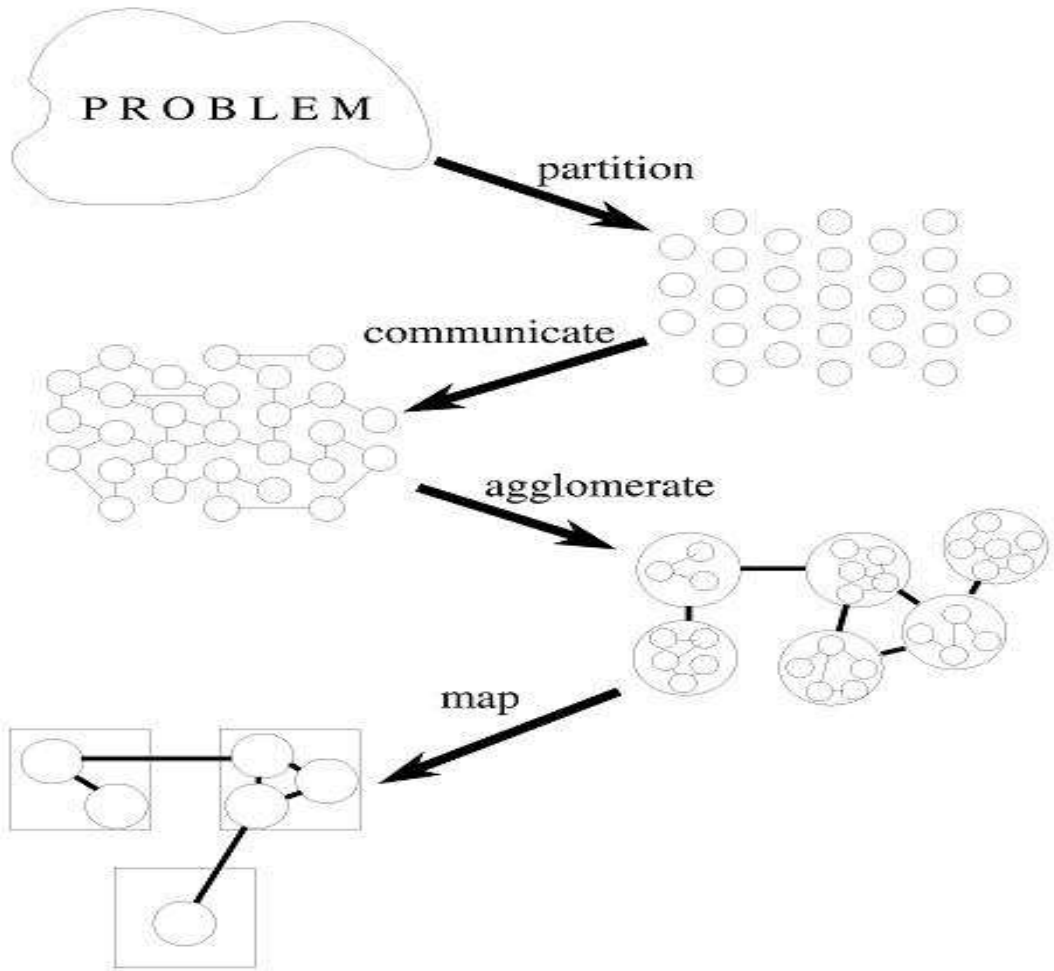
Slides por ***Paulo Sérgio Lopes de Souza***

Metodologia de Projeto

- **Há várias soluções** paralelas para um problema
  - a melhor solução paralela pode ser diferente da obtida a partir da versão sequencial!
- Proposta de Ian Foster
  - **inicialmente** preocupar-se com o **algoritmo**
    - focalizar atenção no algoritmo paralelo a ser desenvolvido
  - **posteriormente** adaptar o algoritmo à **máquina** que irá executá-lo
- Metodologia de projeto baseada nos estágios:
  - Particionamento **P**
  - Comunicação **C**
  - Aglomeração **A**
  - Mapeamento **M**

Metodologia de Projeto

- Metodologia de projeto **PCAM**



Metodologia de Projeto

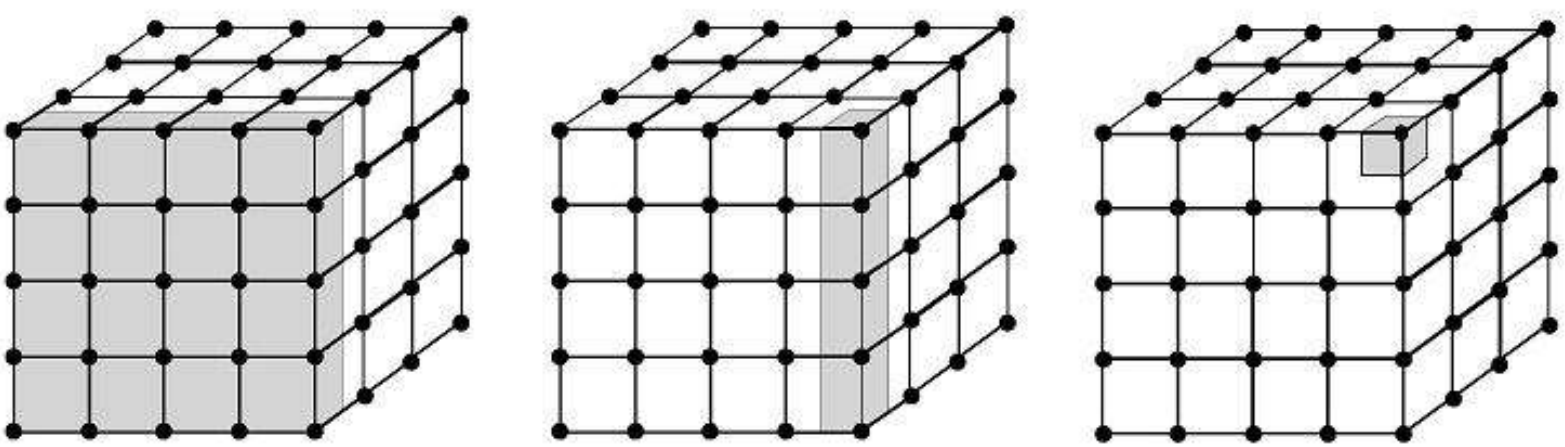
- Metodologia PCAM – **particionamento**
  - procurar por oportunidades de execuções paralelas
    - definir o maior número de pequenas tarefas
    - decomposição do problema com ***granulosidade fina***
      - oferece maior flexibilidade ao algoritmo paralelo
  - granulosidade pode tornar-se mais grossa
    - devido a fatores como:
      - comunicação, arquitetura, questões de ES
    - estágios posteriores determinarão se há essa necessidade
  - quando partições são criadas, devem ser considerados
    - **computação** associada ao problema
      - decomposição funcional
    - **dado** usado pela computação
      - decomposição por domínio
    - técnicas são complementares
  - importante neste estágio:
    - evitar replicação de computação e de dados
      - formar conjuntos disjuntos

Metodologia de Projeto

- Particionamento - **decomposição por domínio**
  - dividir dados em pequenos conjuntos balanceados
  - associar computação necessária a cada conjunto
  - caso sejam necessários dados de outras tarefas
    - comunicação deve ser inserida (analisada próximo estágio)
  - dados que serão decompostos podem ser:
    - de entrada, de saída ou valores intermediários

Metodologia de Projeto

- Particionamento - **decomposição por domínio**
  - —diferentes partições são possíveis
  - —devem ser consideradas:
    - maiores estruturas de dados (ED)
    - ED usadas mais frequentemente
  - —diferentes fases de computação podem necessitar de diferentes decomposições das ED

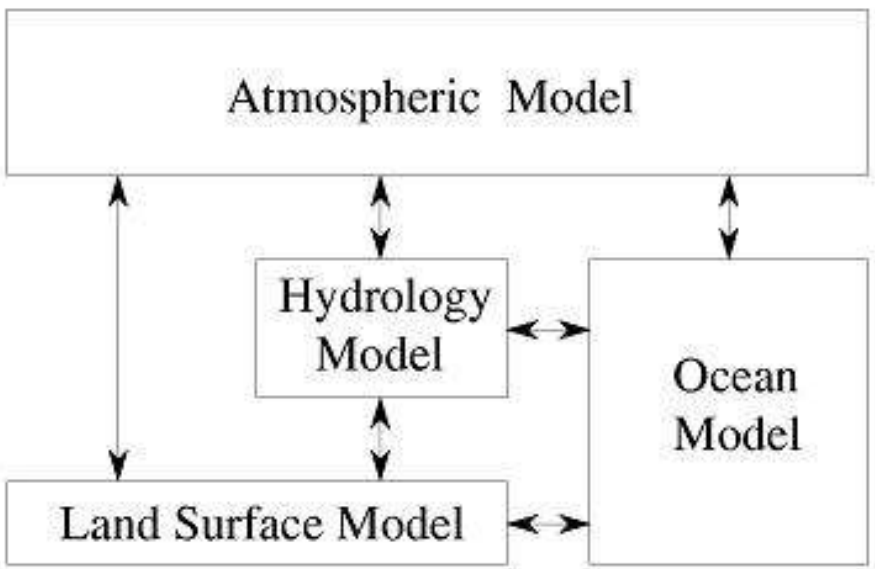


Metodologia de Projeto

- Particionamento - **decomposição funcional**
  - computação dividida em porções disjuntas
    - dados associados a cada partição são analisados depois
  - dados podem ser comuns às partições
    - devem ser replicados ou deve-se usar comunicação
    - se for o caso comum, usar decomposição por domínio
  - estudar se a computação a ser feita pode fornecer
    - alternativas à construção do algoritmo paralelo
    - estrutura paralela ao problema, otimizando a versão final

Metodologia de Projeto

- Particionamento - **decomposição funcional**
  - —pode reduzir a complexidade do problema
    - – divide-se o problema pelas suas funções maiores
    - – unem-se os módulos criados com uma interface apropriada
    - – Ex: Simulação climática da Terra



Metodologia de Projeto

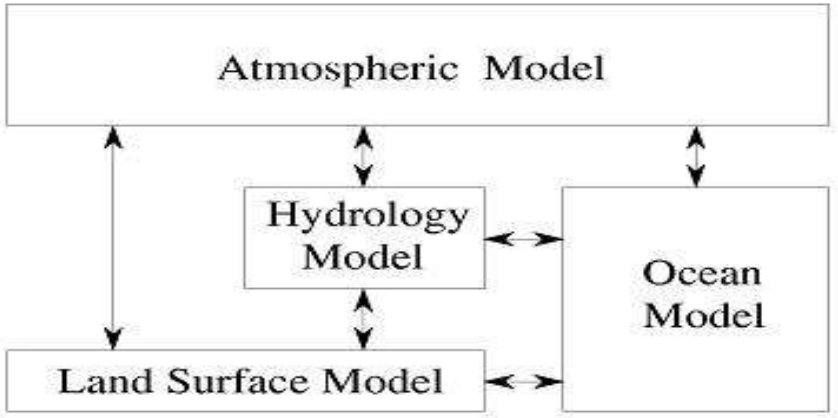
- Particionamento - ***checklist***
  - *Partição define no mínimo uma ordem de magnitude a mais de tarefas que de processadores?*
    - *senão... pouca flexibilidade nos estágios seguintes*
  - *Partição evita replicação de computação e de dados?*
    - *senão... algoritmo não terá escalabilidade*
  - *Tarefas têm tamanhos semelhantes?*
    - *senão... problemas durante o estágio de mapeamento*
  - *O número de tarefas aumenta de acordo com o crescimento do problema?*
    - *senão... aumentando apenas o trabalho em cada tarefa, dificulta escalabilidade*
  - *Foram identificados particionamentos alternativos?*
    - *técnicas de decomposição são complementares*
    - *maximizar flexibilidade nos estágios superiores*

Metodologia de Projeto

- Comunicação (Estágio de)
  - —Tarefas precisam trocar dados e sincronizar
  - —Canais lógicos formalizam a comunicação
    - permitem a troca de mensagens entre tarefas
  - — Duas fases:
    - 1ª definição da estrutura do canal ligando produtores e consumidores
    - 2ª especificação das mensagens enviadas pelos canais
  - —Estrutura fornece uma visão:
    - quantitativa sobre localidade & custos da comunicação

Metodologia de Projeto

- Comunicação
  - Custos: definir canais e enviar a mensagem
  - Comunicação desnecessária deve ser evitada!
    - Mas organizada de modo a otimizar a concorrência
  - Decomposição funcional facilita a construção da comunicação



  - Decomposição por domínio dificulta essa visão
    - detectar necessidade de troca de dados não é natural

Metodologia de Projeto

- Comunicação
  - Pode apresentar as seguintes categorias:
    - **Local / Global**
      - + apenas vizinhos / com muitas tarefas
    - **Estruturada / Não Estruturada**
      - + tarefas e vizinhos formam estrutura regular (grid, árvore) /
      - + ou então a rede de comunicação forma grafo irregular
    - **Estática / Dinâmica**
      - + padrões de comunicação não muda durante o tempo
      - + padrões de comunicação variam em tempo de execução
    - **Síncrona / Assíncrona (cuidado com esses termos!)**
      - + produtores e consumidores executam coordenadamente
      - + consumidores obtêm dados sem a colaboração dos produtores

Metodologia de Projeto

- Comunicação - **local**
  - —Ex: Método de Diferença Finita de Jacobi
    - usando decomposição por domínio,
      - + cada tarefa contém um ponto do *grid* bi-dimensional
      - + cada tarefa realiza todas as iterações referentes a um ponto

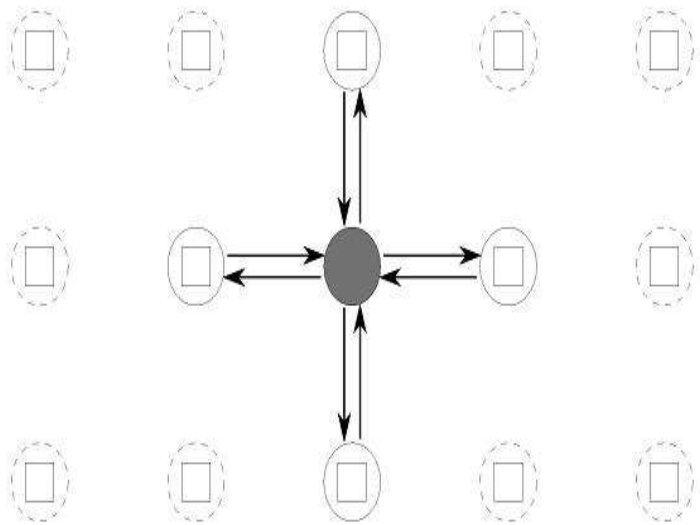
for t = 0 to T-1

send $X_{i,j}^{(t)}$ ***o each neighbor***

recv $X_{i-1,j}^{(t)}, X_{i+1,j}^{(t)}, X_{i,j-1}^{(t)}, X_{i,j+1}^{(t)}$ ***from neighbors***

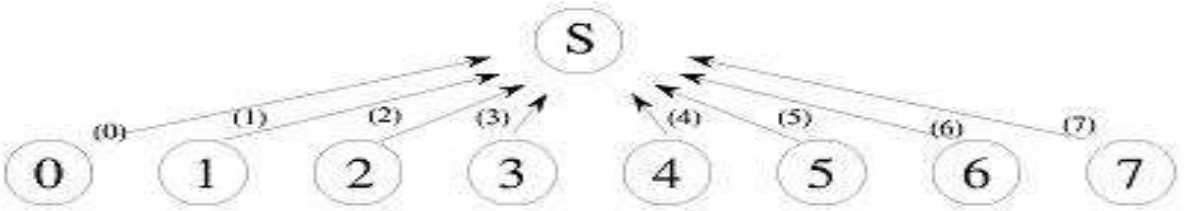
compute $X_{i,j}^{(t+1)}$ ***using Equation 2.1***

endfor



Metodologia de Projeto

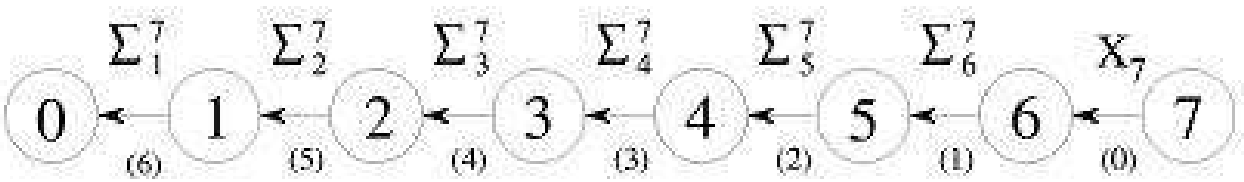
- Comunicação - **global**
  - —envolve muitas tarefas na comunicação
  - —identificar pares de prod/cons não é o suficiente
    - abordagem gera muita comunicação e limita concorrência
    - Ex: Operação de redução $S = \sum_{i=0}^{N-1} X_i$, em paralelo e com um mestre



    - **Problemas** desse algoritmo:
      - + centralizado: não distribui computação e comunicação. Mestre deve participar em toda a operação (gargalo)
      - + sequencial: não permite computação e comunicação concorrentes. Mestre recebe 1 valor por vez e o soma.

Metodologia de Projeto

- Comunicação - **global**
  - —Distribuição da Comunicação e Computação
    - cada tarefa I, 0<I<N-1, computa: $S_i = X_i + S_{i-1}$
    - recebe valor do vizinho da direita, soma e repassa resultado para o vizinho da esquerda



    - evita o gargalo do processo mestre
    - distribui N-1 computações e comunicações
      - + porém, ainda leva N-1 passos para solução
    - execução concorrente apenas se há múltiplas reduções a fazer
      - + ***pipeline***

Metodologia de Projeto

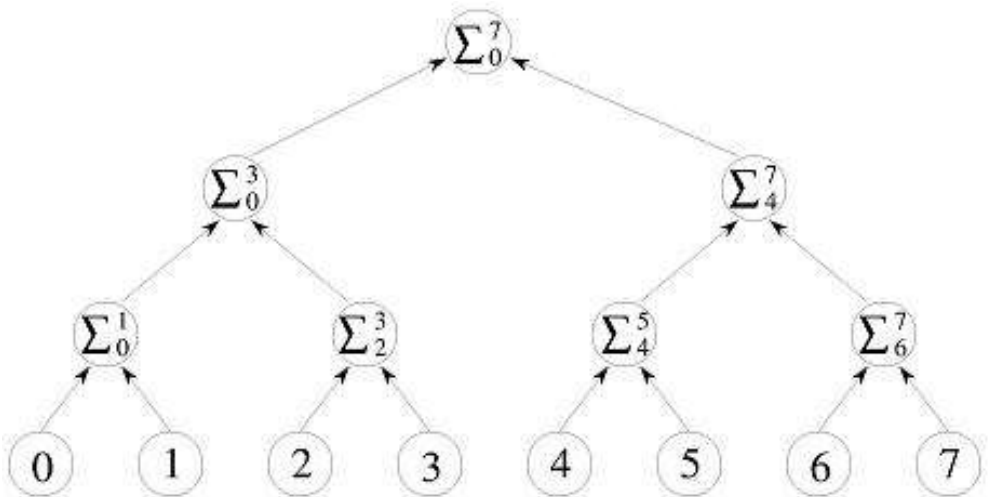
- Comunicação - **global**
  - Estratégia **divisão e conquista**
    - **oferece mais oportunidades para explorar concorrência entre computação e comunicação**

```
procedure divide_and_conquer
begin
    if base case then
        solve problem
    else
        partition problem into subproblems L and R
        solve problem L using divide_and_conquer
        solve_problem R using divide_and_conquer
        combine solutions to problems L and R
    endif
end
```

Metodologia de Projeto

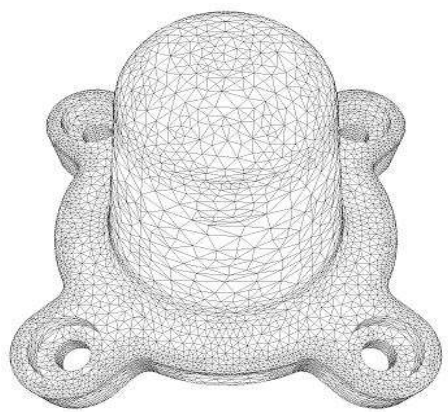
- Comunicação - **global**
  - —Estratégia **divisão e conquista**
    - Exemplo da operação de redução
      - considerando $N = 2^n$, n é um inteiro > 0 (aqui $N = 8$ e, obviamente, $n = 3$)

$$\sum_{i=0}^{2^n-1} = \sum_{i=0}^{2^{n-1}-1} \; 2^{n-1} \sum_{i=0}^{2^n-1}$$



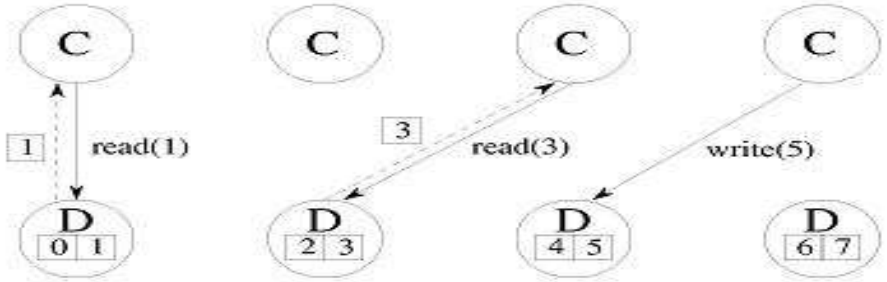
Metodologia de Projeto

- Comunicação – **dinâmica e não estruturada**
  - —Nem sempre ocorre de maneira regular e estática durante a execução
  - —Ex: Métodos de elementos finitos
    - + *grid* computacional formando um objeto irregular
    - + cada ponto no *grid* é irregular, dependente de dados e pode mudar durante a execução, conforme o *grid* é refinado
  - —Não dificulta primeiros estágios
    - – complica aglomeração e mapeamento



Metodologia de Projeto

- Comunicação – **assíncrona**
  - produtores não sabem quando enviar mensagem
    - consumidores devem requisitar dados aos produtores
  - Ex: Aplicação paralela com uma ED com muitos dados
    - + acessos frequentes por todas as tarefas
    - soluções:
      - + distribuir dados entre as tarefas que realizam a computação
        - o falta de modularidade, grande comunicação e interrupção da computação
      - + criar um segundo conjunto de tarefas (servidores de dados)
        - o modular, perda de localidade dos dados, custo com chaveamento de contexto entre tarefas de dados e de computação



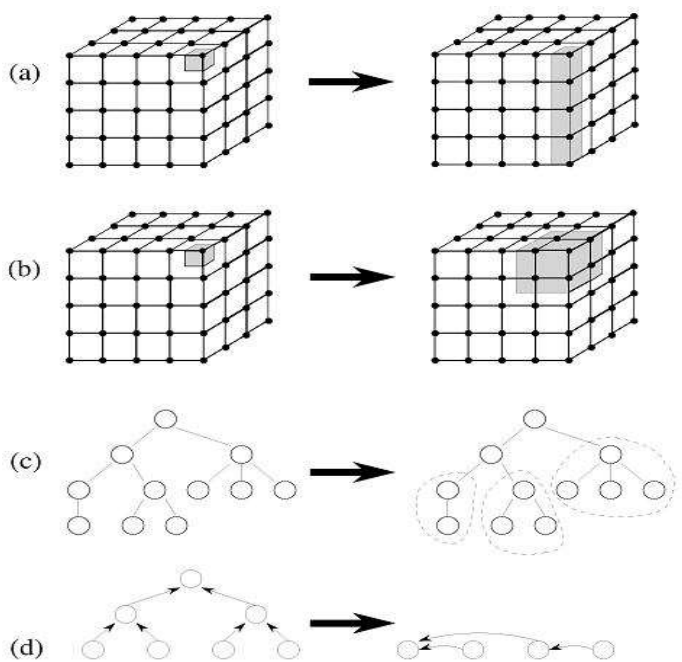
      - + usar um computador e modelo de programação *shared-memory*
        - o usuário deve garantir exclusão mútua

Metodologia de Projeto

- Comunicação – ***checklist***
  - O número de comunicações nas tarefas está balanceado?
    - problema com escalabilidade e gargalos
  - Cada tarefa comunica-se apenas com seus vizinhos?
    - problema com escalabilidade
  - Operações de comunicação ocorrem concorrentemente?
    - problema com ineficiência e escalabilidade
  - Operações associadas com diferentes tarefas podem ser realizadas concorrentemente?
    - problema com ineficiência e escalabilidade

Metodologia de Projeto

- Aglomeração – (Estágio de)
  - Particionamento e comunicação produzem algoritmo
    - abstrato
    - não especializado para execução eficiente em uma máquina
    - **problemas** comuns com a **granulosidade fina**
  - Nesta fase:
    - 1ª iniciativa para mover-se do abstrato ao concreto
    - considera a necessidade de aglomerar tarefas
      - + diminuir número de tarefas, deixando-as com mais "peso"
      - + restando ***P*** tarefas em ***P*** processadores, tem-se SPMD e não há estágio mapeamento
    - considera a necessidade de replicar dados e/ou computação



Metodologia de Projeto

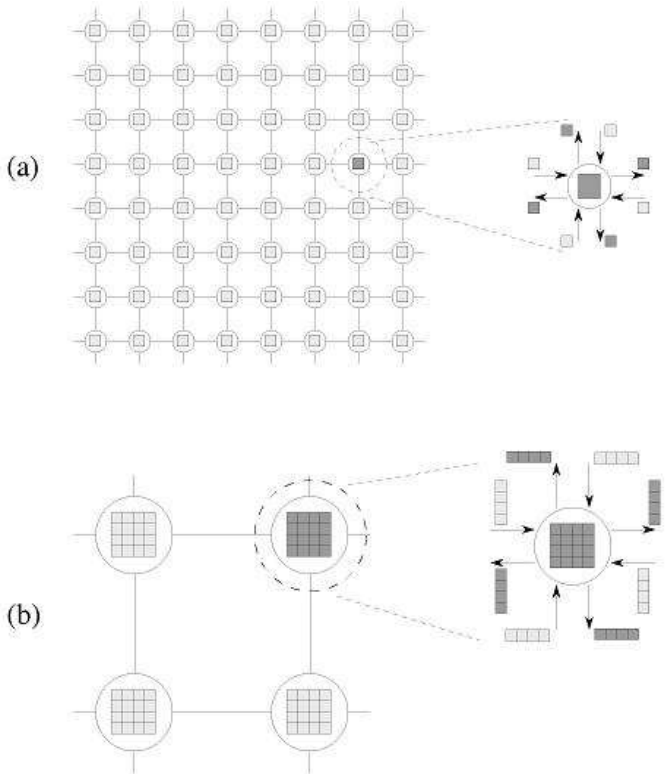
- Aglomeração
  - —**Três metas** principais (podem ser conflitantes)
    - reduzir custos com a comunicação
      - + aumentando granulosidade
    - manter flexibilidade em relação à escalabilidade e decisões de mapeamento
    - reduzir os custos da engenharia de software

Metodologia de Projeto

- Aglomeração – **aumentar granulosidade**
  - —grande número de pequenas tarefas não produz, necessariamente, um algoritmo eficiente
  - —custos da comunicação afetam desempenho
    - – muitos computadores **param computação para comunicar**
    - – enviar menos dados melhora desempenho
  - —**mensagens maiores são mais eficientes que menores**
    - – para a mesma quantidade de dados a serem transmitidos
      - + custo de *startup* a mensagem + custo para o envio
  - —há também o **custo para criação de várias tarefas**

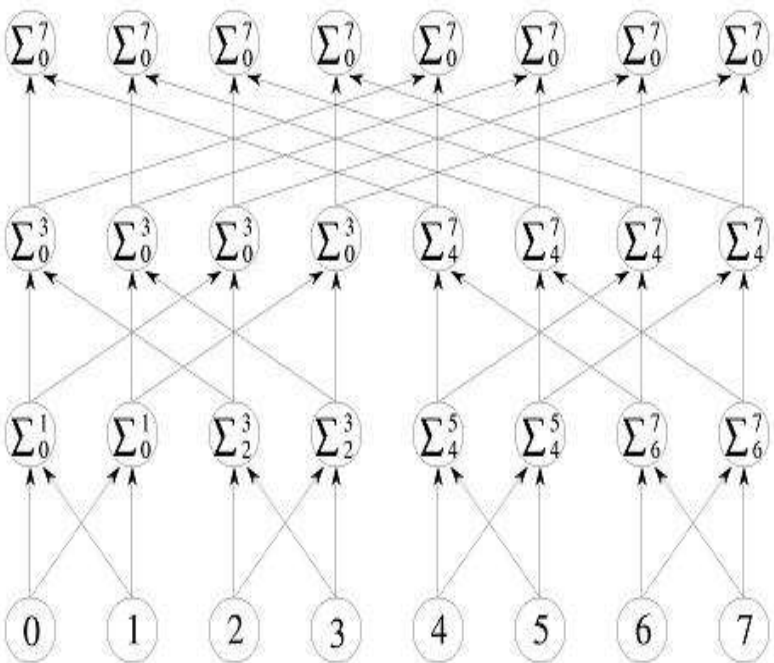
Metodologia de Projeto

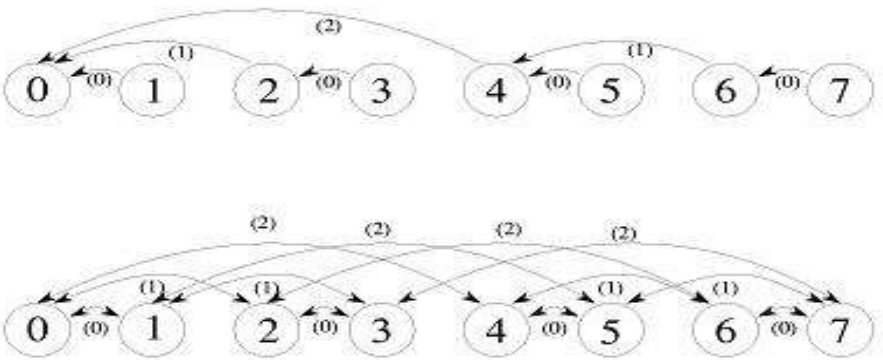
- Aglomeração – **aumentar granulosidade**
  - —Efeito superfície-por-volume
    - responsável pela redução de custos quando aumenta-se a granulosidade
    - superfície indica a comunicação
      - + aumentando a granulosidade, diminui-se a comunicação
    - volume indica a computação
      - + demanda por computação é proporcional ao subdomínio volume
    - Ex.: Problema diferença finita bi-dimensional

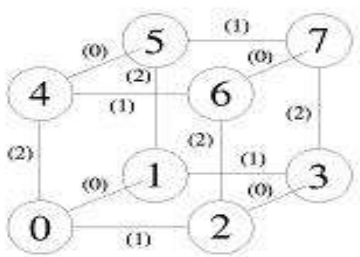


Metodologia de Projeto

- Aglomeração – **aumentar granulosidade**
  - —Evitar comunicação
    - + dependência de dados pode impedir emprego da concorrência
    - + aglomerar tarefas neste caso evita comunicação desnecessária
  - – Ex:op. de redução em paralelo, todas as tarefas têm resultado
    - + tarefas podem explorar concorrência apenas no passo (0, 1 ou 2)







Metodologia de Projeto

- Aglomeração – **preservar flexibilidade**
  - é fácil a aglomeração prejudicar a escalabilidade
    - normalmente torna decisões de particionamento estáticas
      - impedem adaptação do algoritmo para uma máquina melhor
  - ter mais tarefas que processadores contribui para:
    - adaptar a aplicação para uma nova plataforma
    - sobrepor comunicação com computação
      - tarefa bloqueada não bloqueia necessariamente o processador
        - este pode atender outra tarefa
    - facilitar o balanceamento da carga no estágio de mapeamento
  - estudos empíricos e modelagem analítica
    - **estudos complementares** que podem determinar nº ótimo de tarefas
  - nº maior de tarefas que processadores
    - **não é obrigatório** para flexibilidade
    - **importante**: não incorporar limites desnecessários à criação de

Metodologia de Projeto

- Aglomeração – **redução de custos da ES**

  - diferentes estratégias de particionamento encarecem projeto
    - **mudanças** no código **devem ser evitadas**

  - partições podem reaproveitar código já existente
    - elas facilitam o **reuso** de componentes

Metodologia de Projeto

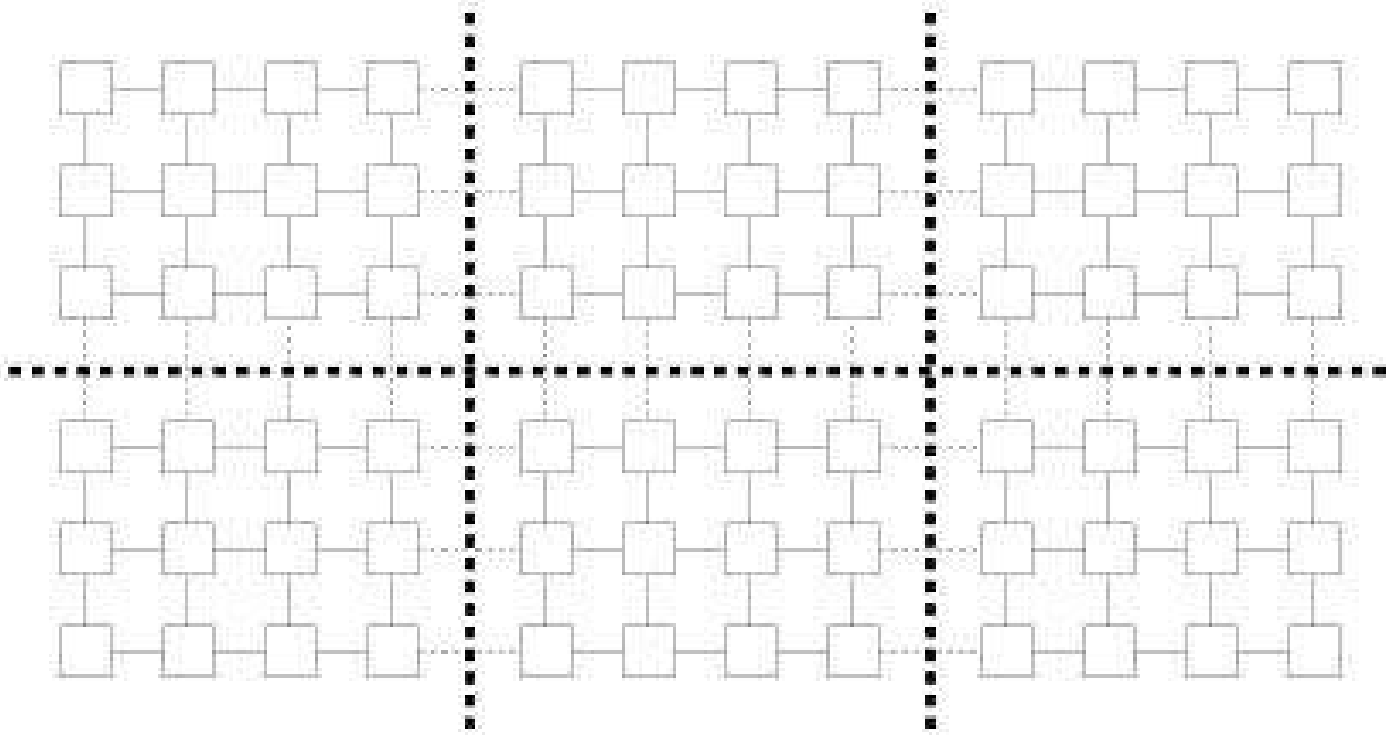
- Aglomeração – ***checklist***
  - Aglomeração reduziu os custos de comunicação com o aumento da localidade?
  - Se a aglomeração replicou a computação, verificaram-se os benefícios frente aos custos
    - considerados tamanhos variados de problemas e/ou quantidades de processos?
  - Se a aglomeração replicou dados, verificou-se a escalabilidade?
    - considerados tamanhos variados de problemas e/ou quantidades de processos?
  - Tarefas têm tamanhos aproximados (computação e comunicação)?
  - O número de tarefas é condizente com o tamanho do problema e com o computador usado?
  - O número de tarefas pode ser reduzido ainda mais, sem prejudicar escalabilidade, flexibilidade ou custos de ES?

Metodologia de Projeto

- Mapeamento – (Estágio de)
  - —especifica onde cada tarefa vai executar
  - —não se aplica em monoprocessadores
    - ou máquinas onde o escalonamento de procesos é automático
  - —Problema com objetivo de:
    - escalonar tarefas que executam concorrentemente em diferentes processadores (**aumentar concorrência**)
    - escalonar tarefas que comunicam frequentemente nos mesmos processadores (**aumentar localidade**)
    - Essas estratégias são conflitantes!
  - —Problema difícil (NP-completo)
    - estratégias e heurísticas específicas a determinados problemas

Metodologia de Projeto

- Mapeamento
  - —mais fácil de ser empregado se
    - – decomposição por dados
    - – número fixo de tarefas balanceadas
    - – comunicação local/global estruturada



Metodologia de Projeto

- Mapeamento
  - algoritmos mais complexos empregam:
    - algoritmos de **balanceamento de carga**
      - empregam políticas de escalonamento com heurísticas
    - balanceamento de carga **probabilísticos**
      - menos *overhead* que os baseados na estrutura da aplicação
    - **balanceamento de carga dinâmico**
      - consideram variações no número de tarefas, quantidade de comunicação e/ou computação por tarefa
      - geram *overhead* em tempo de execução
        - comunicação local é preferível à global
    - algoritmos de **escalonamento de tarefas**
      - alocam tarefas para processadores ociosos (ou que se tornarão ociosos)
      - usados quando aplicação:
        - usa decomposição funcional
        - computação é formada por várias tarefas de curta duração
          coordenam com outras tarefas só no início e fim da execução

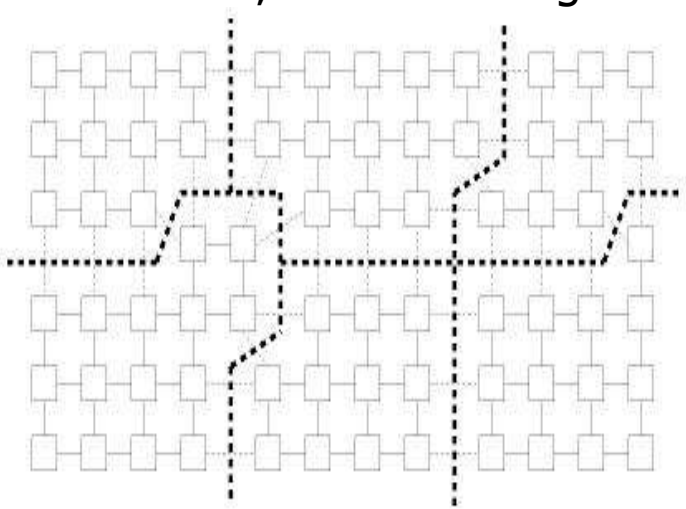
Metodologia de Projeto

- Mapeamento - **balanceamento de carga**
  - —há diversos algoritmos para o balanceamento
  - —aqui foram agrupados em:
    - – métodos de biseção recursiva,
    - – algoritmos locais
    - – métodos probabilísticos e
    - – mapeamentos cíclicos

  - —**biseção recursiva**
    - – divide recursivamente domínio em subdomínios (tamanhos =)
      - + objetivo é minimizar custos de comunicação
      - + reduzindo número de canais que cruzam limites dos subdomínios
      - + estratégia pode ser feita em paralelo
    - – há variações deste método
      - + **particionam computação x particionam comunicação**

Metodologia de Projeto

- Mapeamento - **balanceamento de carga**
  - **—algoritmos locais**
    - – técnicas baseadas na biseção recursiva são caras
      - + necessitam conhecimento global da computação a ser feita
    - – algoritmos locais tomam decisões com conhecimentos locais
      - + no máximo considerando carga dos vizinhos
      - + processadores comparam periodicamente suas cargas locais com as cargas dos vizinhos
        - o caso a diferença for maior que um limite, transfere carga
      - + aplicados quando há variação constante das cargas
        - o são mais suscetíveis a picos de variação de carga
        - o podem se tornar lentos



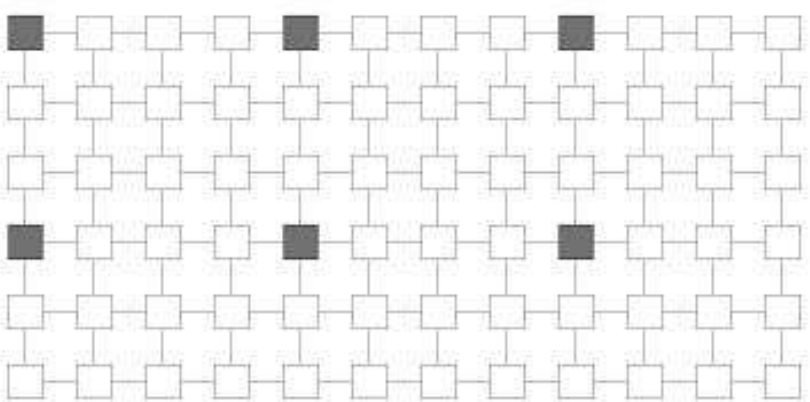
Metodologia de Projeto

- Mapeamento - **balanceamento de carga**

  - **métodos probabilísticos**
    - distribuem tarefas aleatoriamente aos processadores
    - processadores devem ser pulverizados por igual
      - + balanceamento da carga é atingido com menos custo
      - + favorece escalabilidade (adapta-se a um grande nº de tarefas)
      - + desvantagem: não considera comunicação entre tarefas
        - o custos de comunicação provavelmente aumentam

Metodologia de Projeto

- Mapeamento - **balanceamento de carga**
  - **—mapeamentos cíclicos (ou espalhados)**
    - usados quando sabe-se que há
      - variação da carga computacional por ponto no grid
      - significante localidade espacial nos níveis de carga
    - Ex: tarefas são enumeradas (o "como" não importa)
      - p^{th} tarefa vai para o processador p
    - considerados uma forma de método probabilístico
      - meta: na média haverá uma distribuição homogênea na carga
      - problema: aumenta custos com a comunicação



Metodologia de Projeto

- Mapeamento - **escalonamento de tarefas**
  - aplicado quando decomposição funcional cria várias tarefas com pouca localidade
  - gerente ou "pool de gerentes" decidem a distribuição
  - algoritmo é dividido em tarefas escravas (*worker tasks*)
  - ponto complicado:
    - escolher a heurística para realizar a distribuição das tarefas
    - escolha assume compromisso com objetivos conflitantes:
      + independência de operações
        - para reduzir custos de comunicação
      + conhecimento global sobre o estado da computação
        - para melhorar o balanceamento de carga
  - abordagens:
    - mestre / escravo
    - mestre / escravo hierárquico
    - esquemas descentralizados
    - detecção de finalização
      + impede que escravos solicitem trabalho indefinidamente

Metodologia de Projeto

- Mapeamento - ***checklist***
  - Se optou-se por SPMD para o um problema complexo, criação dinâmica de tarefas não seria melhor?
    - algoritmo pode ser mais simples
  - Se a opção foi pela criação dinâmica de tarefas, um projeto SPMD também foi analisado?
    - algoritmo SPMD facilita controle sobre escalonamento, computação e comunicação (mas pode ser mais complexo)
  - Se balanceamento é centralizado, ele não será um gargalo na execução?
  - Se balanceamento é dinâmico, foram analisados os custos durante a execução?
  - Se método é probabilístico, são criadas mais tarefas suficientes para garantir balanceamento?

Metodologia de Projeto

- ***Feedback***
  - —Apresentada uma metodologia de quatro estágios
    - particionamento
      - + decomposição por dados/domínio e tarefas/funcional
    - communicação
      - + local/global, estática/dinâmica, estruturada/desestruturada, síncrona/assíncrona
    - aglomeração
      - + reduzir comunicação, manter flexibilidade em processos, reduzir custos ES
    - mapeamento
      - + reduzir custos de comunicação, manter flexibilidade para arquiteturas, balanceamento da carga, escalonamento de tarefas