Projeto de Algoritmos Concorrentes: metodologia PCAM

Paulo Sérgio Lopes de Souza
pssouza@icmc.usp.br

Universidade de São Paulo / ICMC / SSC – São Carlos
Laboratório de Sistemas Distribuídos e Programação Concorrente

CNPq Conselho Nacional de Desenvolvimento Científico e Tecnológico
CAPES
FAPESP

Metodologia PCAM

- Há várias soluções paralelas para um problema
  - Soluções paralelas eficientes podem requerer códigos diferentes da versão sequencial!
- Proposta de Ian Foster
  - *Quatro etapas determinam como o programa paralelo deve ser construído*
  - **inicialmente** focar no **problema/aplicação**
    - Foco em obter paralelismo e entender comunicação/sincronização
  - **posteriormente** focar na **máquina** que irá executar o algoritmo projetado
    - Foco em minimizar custos da versão paralela na plataforma alvo

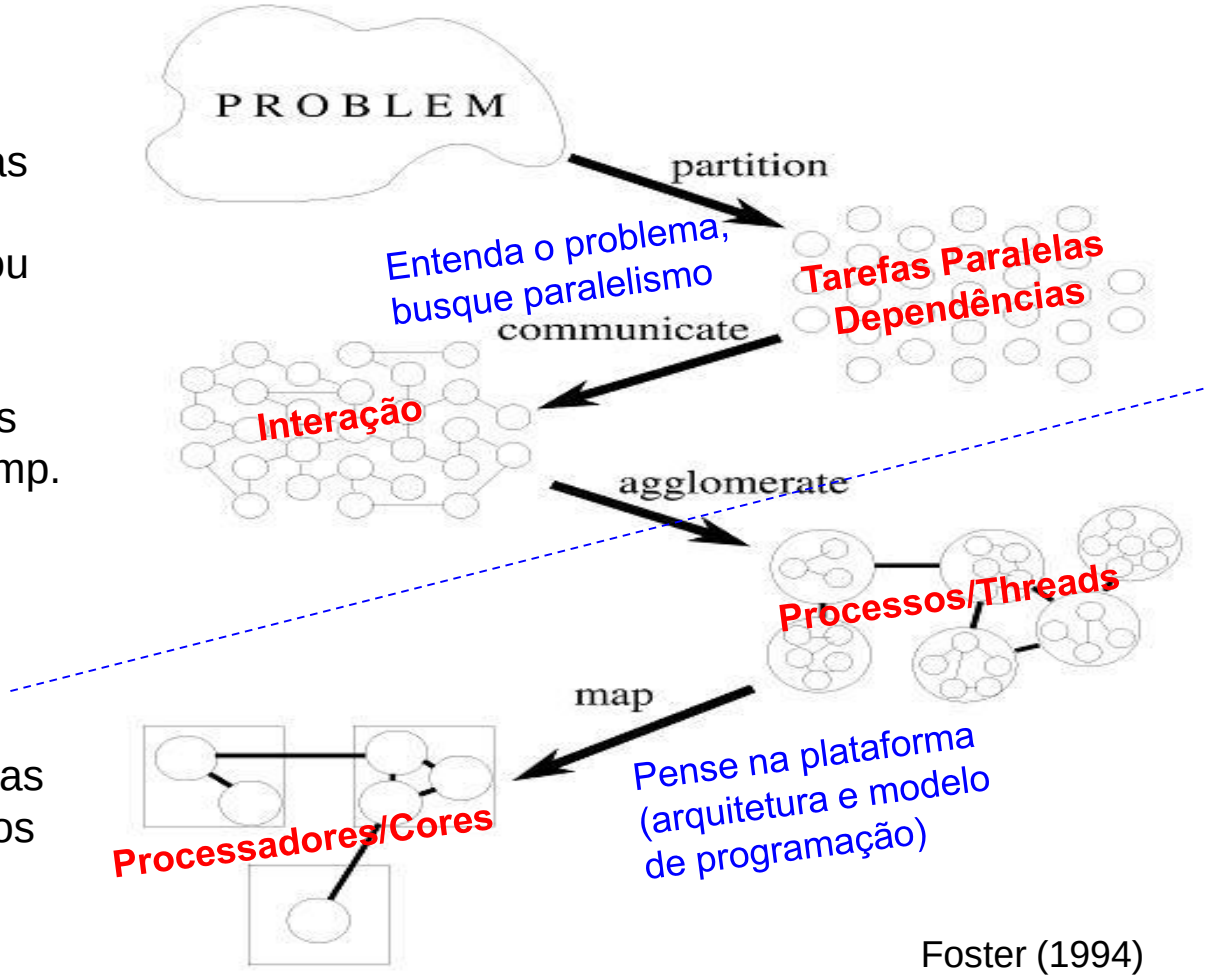
- Metodologia de projeto baseada nos estágios:

• Particionamento	**P**	Foco no problema com granulação mais fina. Identifique paralelismo, dependências e padrões de comunicação.
• Comunicação	**C**	
• Aglomeração	**A**	Ajuste o projeto à plataforma alvo. Ajuste a granulação para a plataforma alvo. Atribua tarefas a processos e os processos a processadores. Minimize sobrecargas da programação concorrente.
• Mapeamento	**M**	

Foster (1994)

Metodologia PCAM

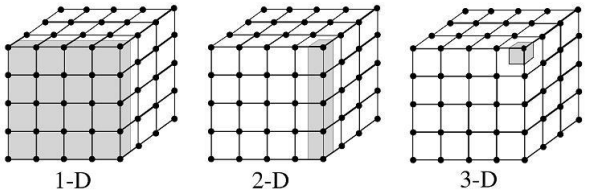
- Há dois fluxos complementares, basicamente:
  - Execução (ou tarefas) ou dados (ou domínio)
    - **Paralelismo funcional (ou de tarefas)**
    - **Paralelismo de dados**
- Evite replicações ao particionar um problema
- Foque nas maiores demandas
- Etapa de **Particionamento** ou Decomposição **por dados**
  - Quando usar?
  - Divida primeiro os dados
    - Depois associe comp.
  - Pode considerar dados
    - Entrada
    - Intermediários
    - Saída
  - Comunicação quando dados de outras tarefas são necessários



Foster (1994)

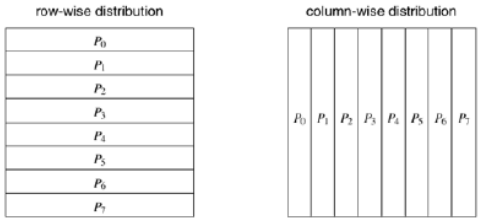
Metodologia PCAM

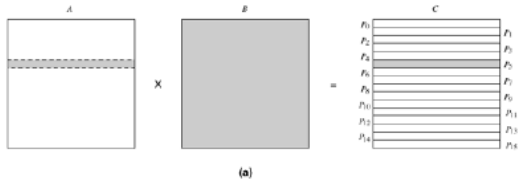
- Ainda sobre o **Particionamento por dados**
  - Dados podem ser particionados de maneiras diferentes
    - Uma ou mais dimensões
    - Afeta escalabilidade e complexidade do algoritmo

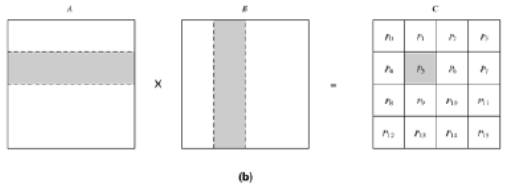


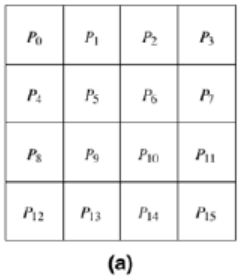
  - Dados podem ser particionados por tamanho ou frequência de acesso
    - Outros fatores podem ser considerados também (ex: custo de acesso)

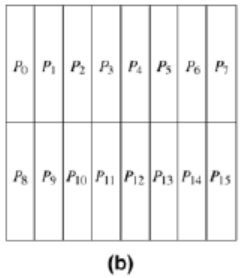
- Algumas variações:
  - **Blocos de dados**











Foster (1994), Grama et al. (2003)

Metodologia PCAM

- Ainda sobre o **Particionamento por dados**
- Algumas variações:
  - Blocos de dados
  - **Distribuição cíclica de blocos**
- Exemplo: Solução de Sistema Linear por Fatoração LU

Figure 3.28. A typical computation in Gaussian elimination and the active part of the coefficient matrix during the *k*th iteration of the outer loop.

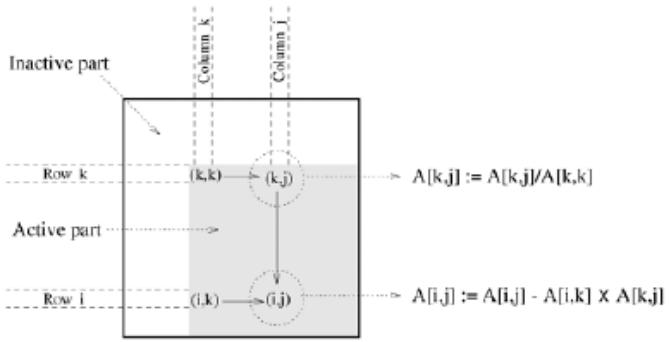
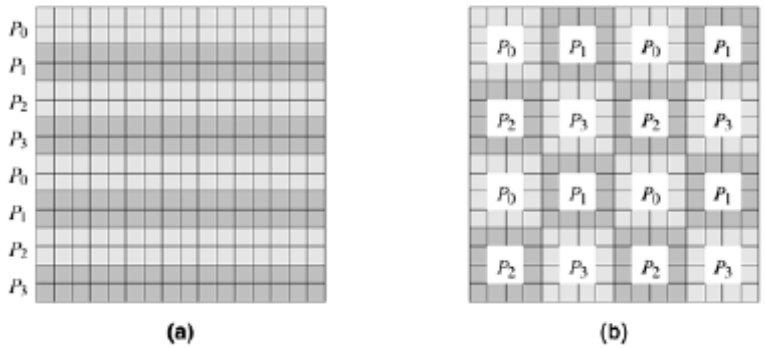


Figure 3.29. A naive mapping of LU factorization tasks onto processes based on a two-dimensional block distribution.

P_0 T_1	P_3 T_4	P_6 T_5
P_1 T_2	P_4 T_6 T_{10}	P_7 T_8 T_{12}
P_2 T_3	P_5 T_7 T_{11}	P_8 T_9 T_{13} T_{14}

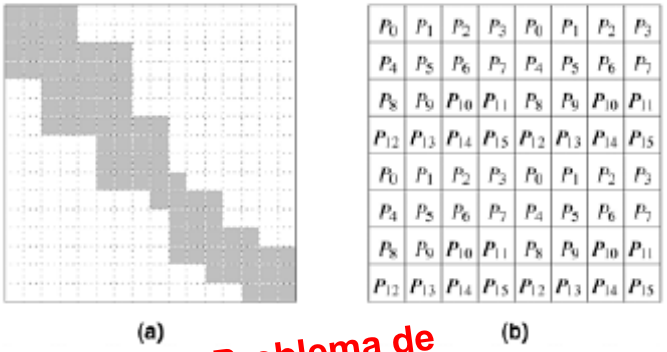
Problema de balanceamento



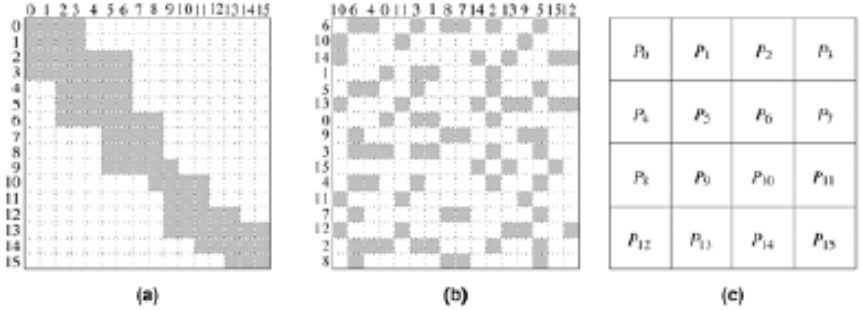
Foster (1994), Grama et al. (2003)

Metodologia PCAM

- Ainda sobre o **Particionamento por dados**
- Algumas variações:
  - Blocos de dados
  - Distribuição cíclica de blocos
  - **Distribuição aleatória de blocos**



Problema de balanceamento



Foster (1994), Grama et al. (2003)

Metodologia PCAM

- Ainda sobre o **Particionamento por dados**
- Algumas variações:
  - Blocos de dados
  - Distribuição cíclica de blocos
  - Distribuição aleatória de blocos
  - **Particionamento por grafos**

Figure 3.34. A mesh used to model Lake Superior.

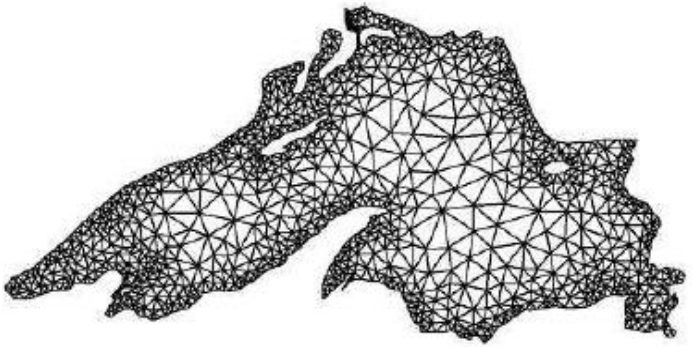


Figure 3.35. A random distribution of the mesh elements to eight processes.

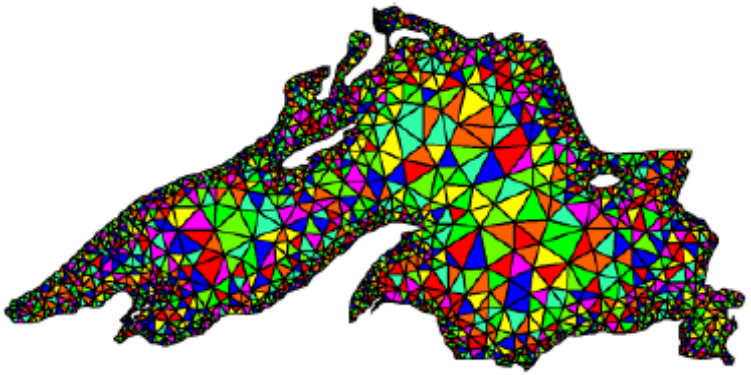
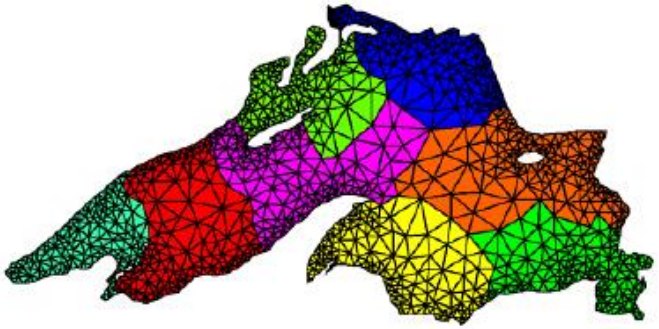


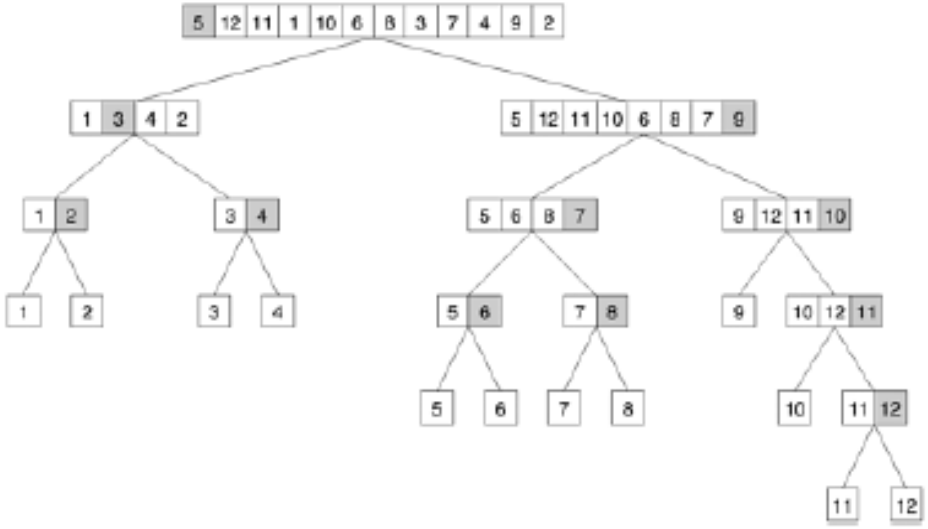
Figure 3.36. A distribution of the mesh elements to eight processes, by using a graph-partitioning algorithm.



Foster (1994), Grama et al. (2003)

Metodologia PCAM

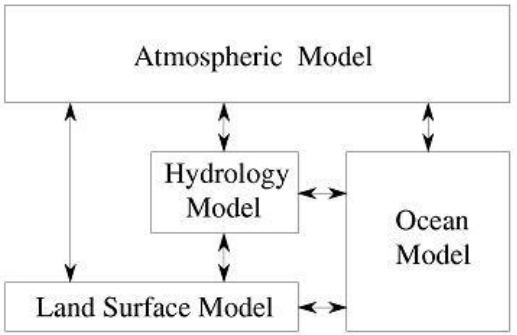
- Ainda sobre o **Particionamento por dados**
- Algumas variações:
  - Blocos de dados
  - Distribuição cíclica de blocos
  - Distribuição aleatória de blocos
  - Particionamento por grafos
  - **Recursiva** (pode ser vista como funcional/tarefas também)



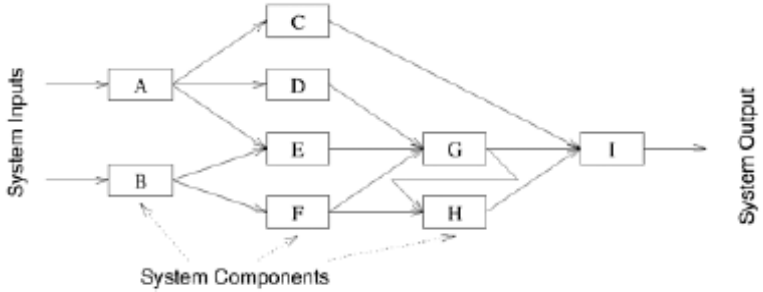
Foster (1994), Grama et al. (2003)

Metodologia PCAM

- **Particionamento por tarefas/funcional**
  - Computação dividida em tarefas disjuntas
    - Dados são associados às tarefas depois
  - Comunicação necessária quando a tarefa precisa de dados remotos
  - Útil quando o problema tem porções disjuntas bem claras



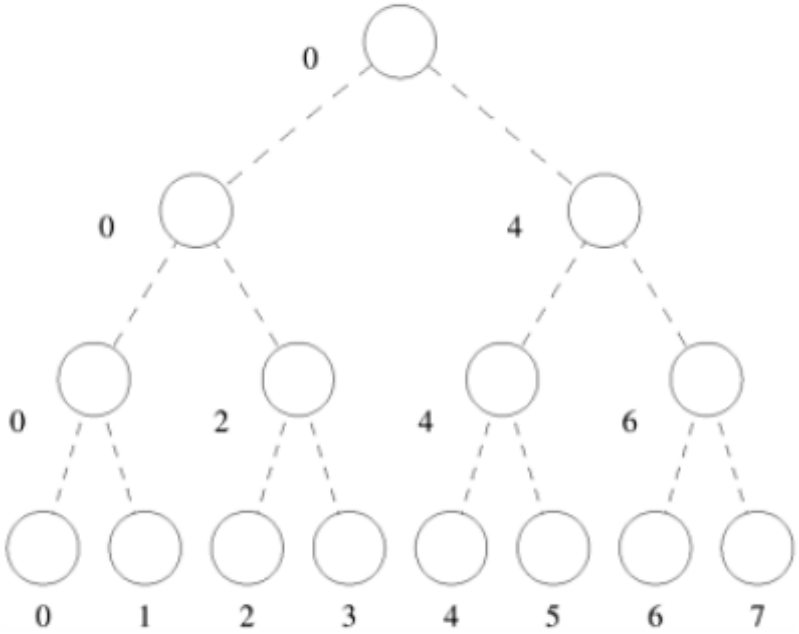
- Algumas variações
  - **Decomposição especulativa**



Foster (1994), Grama et al. (2003)

Metodologia PCAM

- **Particionamento por tarefas/funcional**
  - Computação dividida em tarefas disjuntas
    - Dados são associados às tarefas depois
  - Comunicação necessária quando a tarefa precisa de dados remotos
  - Útil quando o problema tem porções disjuntas bem claras
- Algumas variações
  - Decomposição especulativa
  - **Baseado na dependência das tarefas**

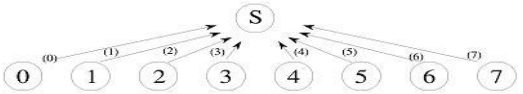


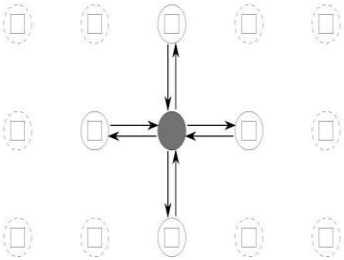
Foster (1994), Grama et al. (2003)

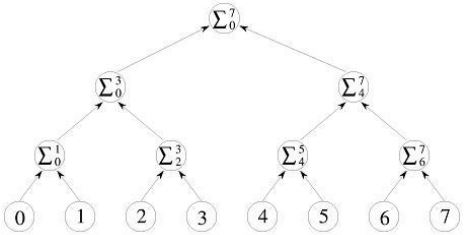
Metodologia PCAM

- Etapa de Análise da **Comunicação & Sincronização**
  - Indica quando tarefas necessitam trocar dados ou garantir ordem de execução

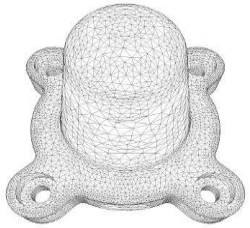
  - Fornece uma visão quantitativa de:
    - Localidade dos dados
    - Custos da comunicação







  - Pode ser:
    - **Local vs global**
      - Viz ou mais tarefas
    - **Estruturada vs não estruturada**
      - Estrs. Regulares
        - grid/árvore
      - Estrs. Irregulares
        - grafo irregular
    - **Estática vs dinâmica**
      - Varia na execução?
    - **Síncrona / Assíncrona**
      - Snd/rcv coordenados
      - Rcvs obtêm dados sem colaboração dos snds



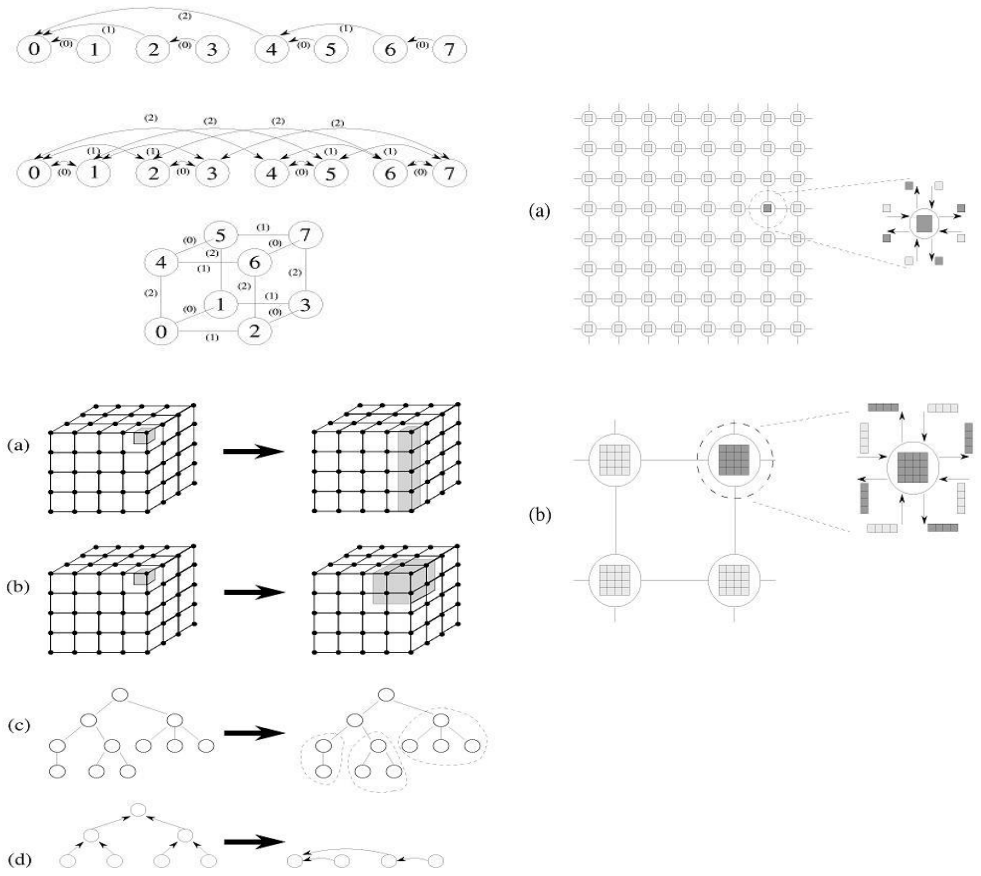
Foster (1994)

Metodologia PCAM

- Etapa de **Aglomeração** de tarefas em processos
  - Atribuir tarefas aos processos ou *threads*

- Particionamento & Comunicação fornecem uma visão abstrata com foco no problema
  - **Aglomeração começa a instanciar o projeto para uma máquina real**

- Objetivos para **reduzir custos**:
  - **Geração de processos**
    - Tarefas dependentes
  - **Reduzir comunicação**
    - Aumentar granulação
    - Adequar à plataforma
  - **Equilibrar cargas de trabalho**
  - **Desenvolvimento & Manutenção**
  - **Preservar Flexibilidade**



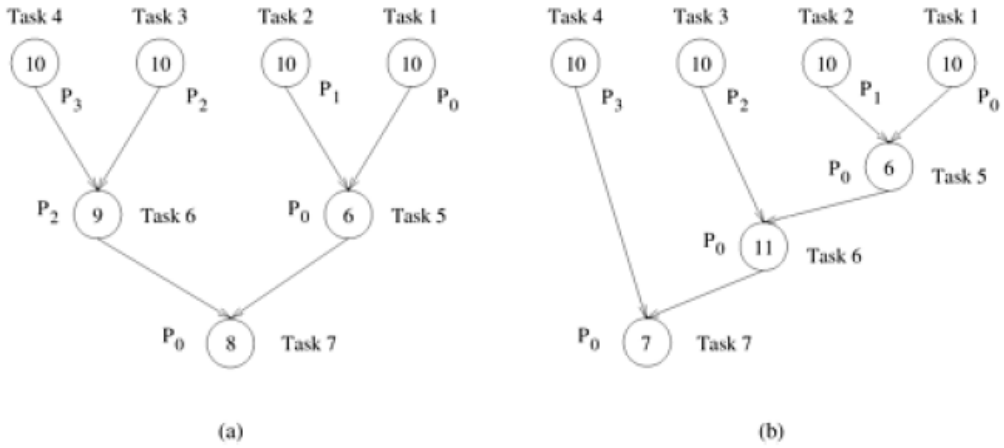
Foster (1994)

Metodologia PCAM

- Etapa de **Aglomeração** de tarefas em processos
  - Atribuir tarefas aos processos ou *threads*

- Particionamento & Comunicação fornecem uma visão abstrata com foco no problema
  - **Aglomeração começa a instanciar o projeto para uma máquina real**

- Objetivos para **reduzir custos**:
  - **Geração de processos**
    - Tarefas dependentes
  - **Reduzir comunicação**
    - Aumentar granulação
    - Adequar à plataforma
  - **Equilibrar cargas de trabalho**
  - **Desenvolvimento & Manutenção**
  - **Preservar Flexibilidade**



Grama et al. (2003)

Metodologia PCAM

- Etapa de **Aglomeração**
  - **O número de processos deve observar o tamanho e características da plataforma!**
    - Em plataformas MIMD com MPI ou OpenMP:
      - Gere processos em função do número de processadores e/ou cores
      - Não gere processos em função do tamanho da carga de trabalho

      - Exemplo: em uma soma de dois vetores de tamanho N, em P máquinas MIMD, não gere N processos, um para cada par de elementos; mas sim considere gerar um número de processos em função de P
        - Gere processos compatíveis com o número de recursos de processamento (i.e. P)
        - Máquinas MIMD não suportam uma granulação muito fina, nem a geração de milhares/milhões/... de processos ou *threads*

    - Em máquinas no estilo SIMD, como GPUs com CUDA:
      - Gere *threads* em função do tamanho da carga de trabalho, respeitando a granulação necessária para se obter desempenho
        - São geradas muito mais *threads* em CUDA que em OpenMP e MPI

Foster (1994)

Metodologia PCAM

- Etapa de **Aglomeração**
  - A aglomeração deve gerar diferentes modelos de programação:
    - Para máquinas SIMD ou MIMD
      - Processos devem ser gerados em função do modelo aquitetural
    - **SPMD** (*Single Program Multiple Data*)
      - Um único binário computando dados distintos
    - MPMD (*Multiple Program Multiple Data*)
      - Múltiplos binários computando dados distintos
    - **Mestre Escravo**
      - Um processo mestre atribui carga aos demais processos escravos
    - Grafo de Tarefas
      - Processos gerados em função do grafo de dependências
    - ***Workpool***
      - Processos escravos buscam tarefas em um pool de trabalho
    - Produtor – Consumidor
      - Processo que produzem a computação e processos que consomem a computação
    - Cliente – Servidor
      - Processos clientes enviam requisições a servidores processem tarefas
    - Pipeline
      - Processos atuam em diferentes estágios de uma computação
    - Híbridos
      - Permitem modelos distintos em diferentes computações da aplicação

Foster (1994)

Metodologia PCAM

- Etapa de **Mapeamento** de processos em processadores
  - Atribuir processos/*threads* aos processadores/núcleos
    - *Escalonar tarefas independentes em processadores diferentes*
      - *Explora melhor a concorrência para extrair mais desempenho*
    - *Escalonar tarefas que se comunicam frequentemente em um processador*
      - *Aumentar a localidade espacial e temporal*
    - *Estratégias podem ser conflitantes*
  - *Escalonamento é um problema difícil (NP-completo)*

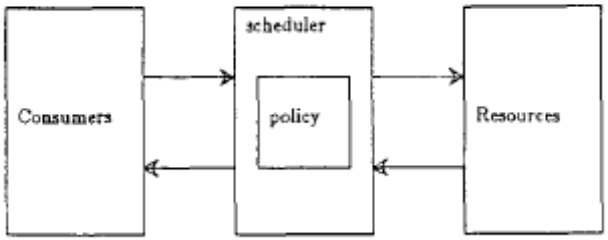


Fig. 1. Scheduling system.

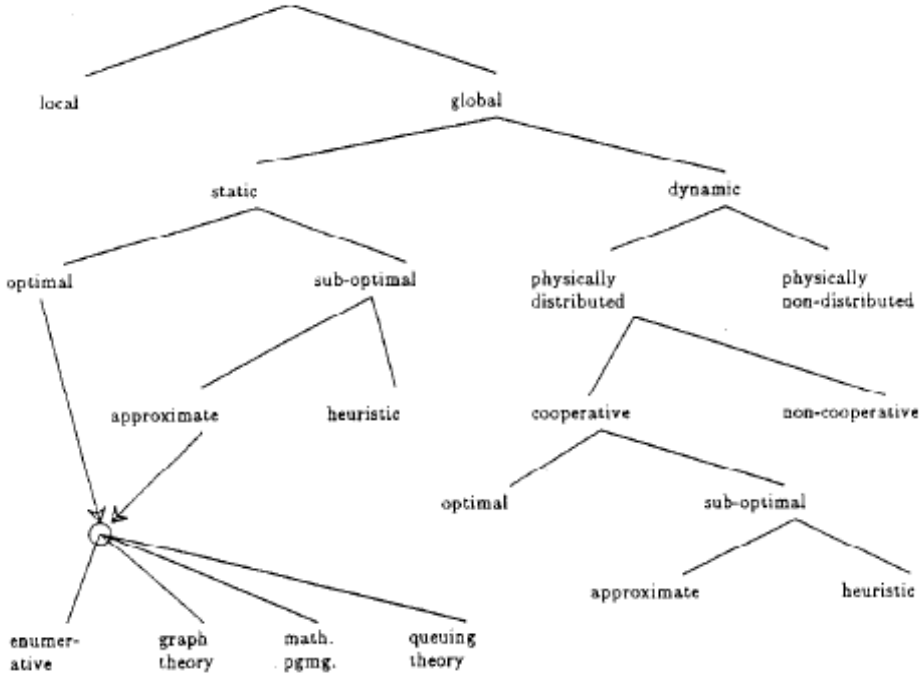


Fig. 2. Task scheduling characteristics.

Casavant & Kuhl (1988)

Metodologia PCAM

- Exemplo de aplicação do PCAM:
  - Peneira de Eratosthenes (entendendo o problema)
    - Algoritmo para encontrar os números primos ≤ N (um número inteiro positivo).
    - Algoritmo segue estes passos básicos:

1. Crie um vetor de números inteiros vet = {2, 3, 4, ..., N}, sendo que nenhum item de vet é marcado como primo (i.e., são iniciados com zero);
2. Determine K = 2, sendo K o primeiro número não marcado em vet;
3. Repita
(a) marque todos os múltiplos de K entre K^2 e N;
(b) encontre em vet o menor número maior que K que não está marcado e atribua a K este novo valor;
até $K^2 > N$
4. Os números não marcados em vet são todos números primos.

Casavant & Kuhl (1988)

Metodologia PCAM

- Exemplo de aplicação do PCAM: Peneira de Eratosthenes

Exemplo do algoritmo Peneira de Eratosthenes para N = 60.

a) Criar a lista/o vetor vet de números naturais. Nenhum está marcado.

XX	02	03	04	05	06	07	08	09	10	11	12	13	14	15	16	17	18	19	20
21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40
41	42	43	44	45	46	47	48	49	50	51	52	53	54	55	56	57	58	59	60

(b) K = 2 (marcar nrs: 4, 6, 8, 10, 12, 14, 16, 18, 20, 22, 24, 26, 28, 30, … , 60)

XX	02	03	04	05	06	07	08	09	10	11	12	13	14	15	16	17	18	19	20
21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40
41	42	43	44	45	46	47	48	49	50	51	52	53	54	55	56	57	58	59	60

(c) K = 3 (marcar nrs: 9, 12, 15, 18, 21, 24, 27, 30, 33, 36, 39, 42, 45, 48, 51, 54, 57 e 60)

XX	02	03	04	05	06	07	08	09	10	11	12	13	14	15	16	17	18	19	20
21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40
41	42	43	44	45	46	47	48	49	50	51	52	53	54	55	56	57	58	59	60

(d) K = 5 (marcar nrs: 25, 30, 35, 40, 45, 50, 55 e 60)

XX	02	03	04	05	06	07	08	09	10	11	12	13	14	15	16	17	18	19	20
21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40
41	42	43	44	45	46	47	48	49	50	51	52	53	54	55	56	57	58	59	60

(e) K = 7 (marcar nrs: 49 e 56)

XX	02	03	04	05	06	07	08	09	10	11	12	13	14	15	16	17	18	19	20
21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40
41	42	43	44	45	46	47	48	49	50	51	52	53	54	55	56	57	58	59	60

(f) como K = 11, onde $\mathbf{K^2 > N}$, então sai do **repita...até**.

Quinn (2003)

Metodologia PCAM

Faça um projeto de algoritmo paralelo seguindo a metodologia PCAM para o problema da peneira de Eratosthenes. Descreva, explicitamente, de forma textual e, se necessário, gráfica, o particionamento, a comunicação, a aglomeração e o mapeamento.

Considere que este algoritmo paralelo deve executar com o menor tempo de resposta, que ele será carregado para execução em um cluster de computadores e que o seu projeto deve usar o particionamento por dados.

Por último, considere que forneceremos este projeto que vocês estão fazendo agora, para outro grupo implementá-lo remotamente. Espera-se que o seu projeto seja suficientemente detalhado para que a outra equipe possa fazer a implementação adequadamente frente ao projeto proposto aqui.

Metodologia PCAM

- Exemplo de aplicação do PCAM:
  - Peneira de Eratosthenes: **particionamento** por dados

O foco da paralelização estará no passo 3a, i.e., na marcação de todos os múltiplos de K entre K^2 e N. Para a versão paralela com particionamento por dados, vet é particionado em N tarefas, sendo que cada tarefa marca vet, caso a divisão de N por K resulte em um resto 0, a partir de N >= K^2. Isso precisa ser feito a cada iteração, sendo que na primeira delas o valor de K == 2.

A partir da segunda iteração um novo valor de K precisa ser descoberto (passo 3b), por uma operação de redução, de ordem logarítmica. Como detalhado na seção de comunicação, o novo valor de K precisará ser repassado a todas as tarefas novamente, para que o passo 3a se repita.

Metodologia PCAM

- Exemplo de aplicação do PCAM:
  - Peneira de Eratosthenes: descrevendo a **comunicação** das tarefas

Há duas comunicações globais no particionamento por dados acima, a cada iteração. Há uma operação de redução para determinar o novo valor de K e um *broadcast* para fornecer este novo valor para todas as tarefas ainda em atividade na aplicação, que possuam um valor para o valor de vet maior que K^2.

Estas comunicações são caras, se comparadas ao custo da possível marcação que cada tarefa deve fazer iterativamente.

Metodologia PCAM

- Exemplo de aplicação do PCAM:
  - Peneira de Eratosthenes: **aglomeração** das tarefas em processos

Dada a plataforma alvo deste algoritmo (um cluster de computadores), deve-se agrupar as N tarefas em P processos, onde P equivale ao número de elementos de processamento (ou núcleos) disponíveis. Esta aglomeração permitirá aumentar a granularidade de cada processo e diminuir a comunicação necessária para atingir o objetivo final.

Cada processo receberá um bloco de dados de vet, contendo (N / P) elementos. Caso não seja possível obter uma divisão exata de (N / P), as posições excedentes de vet devem ser atribuídas ao último processo disponível. Uma alternativa ao resto de (N / P) é o espalhamento desses elementos excedentes aos primeiros processos disponíveis para processamento.

Com esta aglomeração, a operação de redução e o broadcast não ocorrerão mais em função de N valores de vet, mas sim em função de P processos.

Metodologia PCAM

- Exemplo de aplicação do PCAM:
  - Peneira de Eratosthenes: **mapeamento** dos processos em processadores

Considerando que os nós do cluster possuem um desempenho esperado homogêneo, o mapeamento de P processos em PROC Elementos de Processamento ocorrerá por meio de uma fila circular (Round-Robin). Neste caso, se P == PROC, então cada Elemento de Processamento receberá exatamente um processo.

Uma avaliação empírica pode ser aplicada para determinar se P > PROC poderia resultar em melhores desempenhos na plataforma em particular. Neste caso, a fila circular ainda pode ser utilizada.

Caso o desempenho dos nós do cluster seja diferente, então o mapeamento dos nós do cluster pode ser dinâmico, determinado em tempo de execução, considerando a heurística de atribuição ao nó com a menor carga de trabalho em andamento (determinada por alguma métrica de desempenho com o número de processos na fila de pronto para execução, uso de CPU, quantidade de bytes trocados em swap de disco, entre outros).

Referências

FOSTER, I. Designing and Building Parallel Programs, Addison-Wesley Publishing Company, 1994.

GRAMA,A.; KUMAR, U.; GUPTA,A.; KARYPIS, G. Introduction to Parallel Computing, 2nd Edition, 2003.

CASAVANT T. L.; KUHL, J. G.; A taxonomy of Scheduling in General-Purpose Distributed Computing Systems, IEEE Transactions on Software Engineering, v.14, nr.2, 1988.

QUINN, M. J. Parallel Programming in C with MPI and OpenMP, McGraw-Hill,Published 2003.

Grama et al. (2003)

Projeto de Algoritmos Concorrentes: metodologia PCAM

Paulo Sérgio Lopes de Souza
pssouza@icmc.usp.br

Universidade de São Paulo / ICMC / SSC – São Carlos
Laboratório de Sistemas Distribuídos e Programação Concorrente