Programação Paralela

Prof. Marcelo Veiga Neves
marcelo.neves@pucrs.br

Introdução

Programação Paralela:

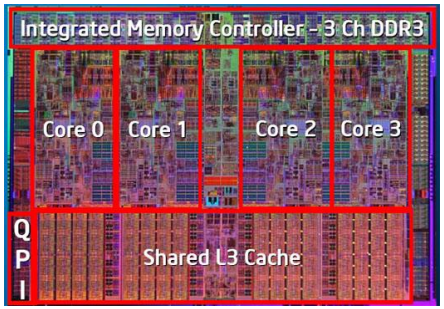
"A capacidade de dividirmos uma carga de trabalho entre várias pequenas tarefas para serem executadas em vários processadores de forma eficiente"

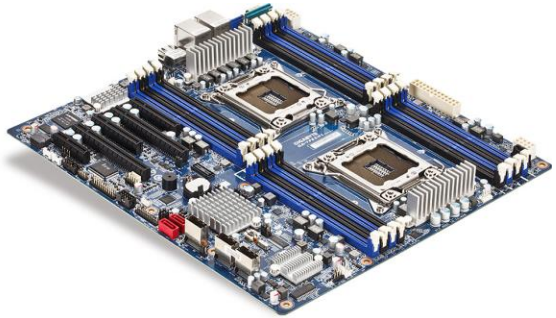
Computação Paralela:

"A execução concorrente destas tarefas com o objetivo de reduzir o tempo total de execução"

Motivação para paralelizar

- Aumentar desempenho de programas
  - Redução do tempo de execução (acelerar)
  - Melhorar a eficiência da aplicação
- Melhor aproveitamento dos recursos
  - Existência de arquitetura paralelas (ex: multicore, multiprocessadores)



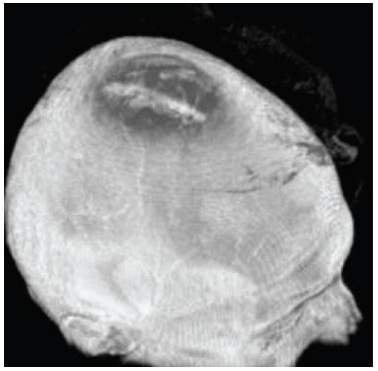


O que paralelizar?

Exploração de Petróleo

Ciência Molecular/Química

Atmosfera/Tempo/Clima



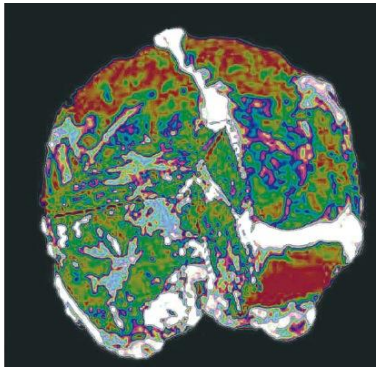
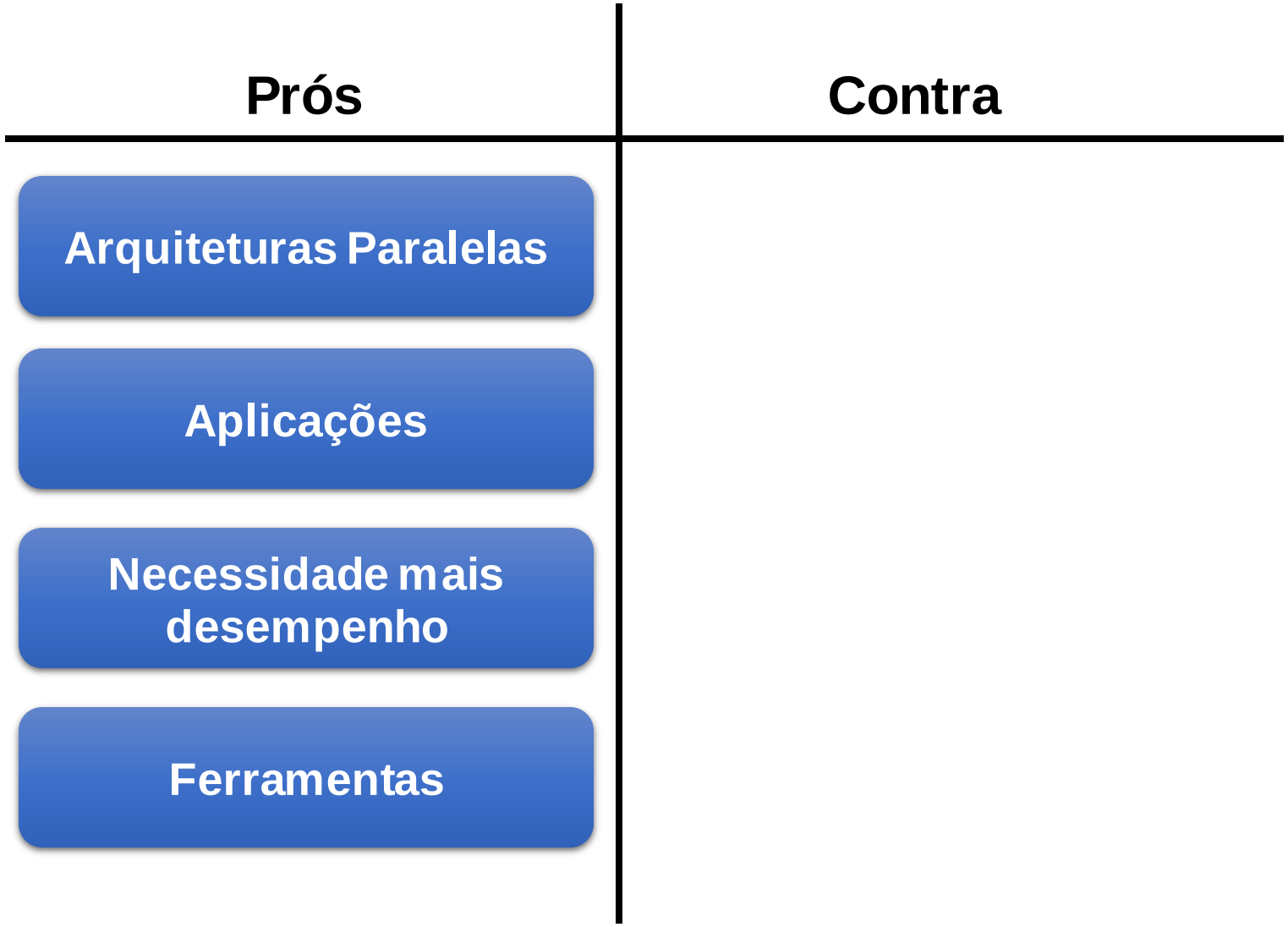


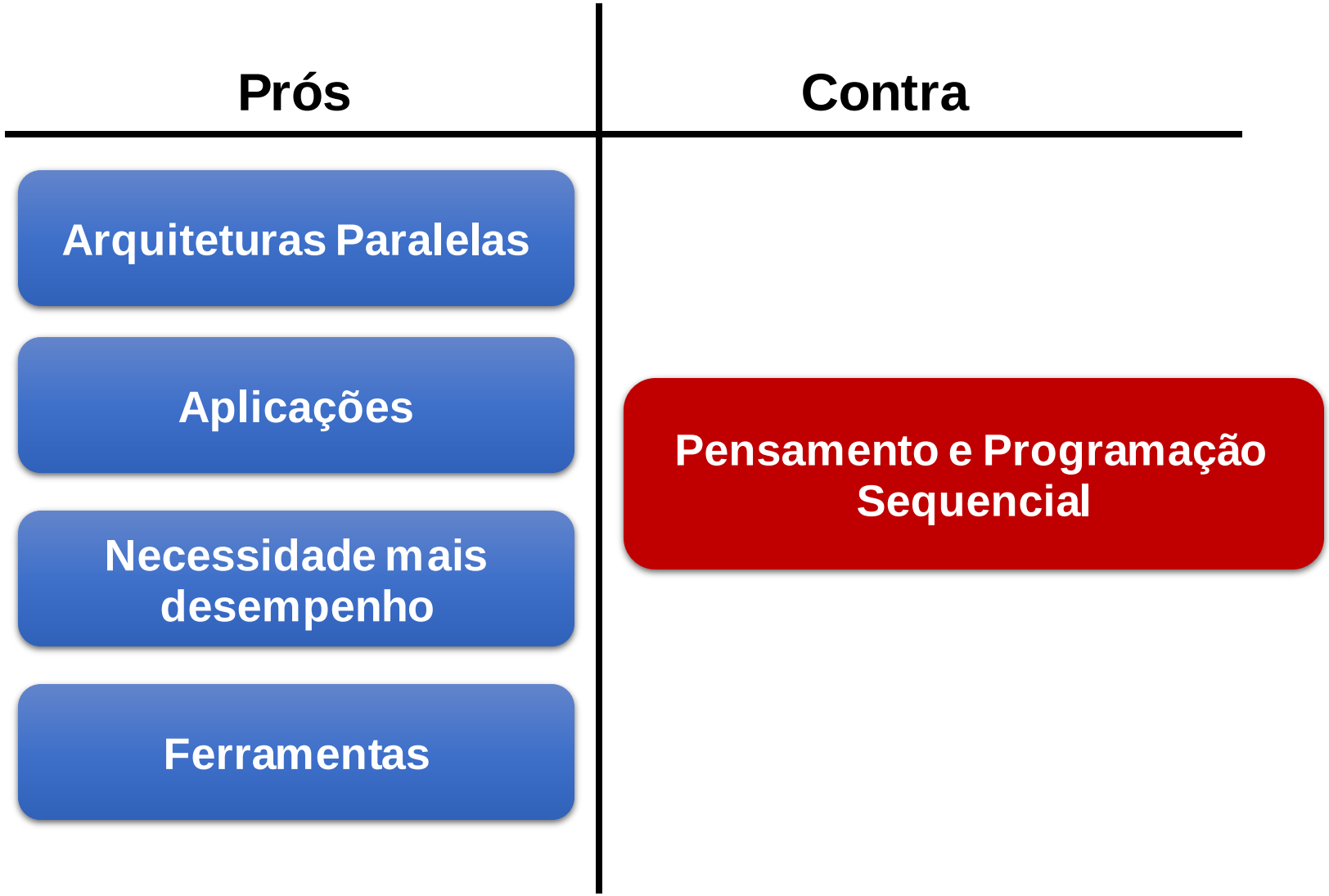
Imagem Médica e Diagnósticos

Modelagem Econômica

Cenário Atual



Cenário Atual



Programar paralelo
é díficil???

1. Preparar a tinta = 30s;
2. Pintar 300 estacas = 3000s;
3. Aguardar a tinta secar = 30s;

Quanto tempo levará um pintor??

1. Preparar a tinta = 30s;
2. Pintar 300 estacas = 3000s;
3. Aguardar a tinta secar = 30s;

Quanto tempo levará um pintor?? **3060s**

1. Preparar a tinta = 30s;
2. Pintar 300 estacas = 3000s;
3. Aguardar a tinta secar = 30s;

Quanto tempo levará um pintor?? **3060s**

Quanto tempo levarão dois pintores?

1530s

1560s

1. Preparar a tinta = 30s;
2. Pintar 300 estacas = 3000s;
3. Aguardar a tinta secar = 30s;

Quanto tempo levará um pintor?? **3060s**

Quanto tempo levarão dois pintores?

~~1[illegible]s~~

1560s

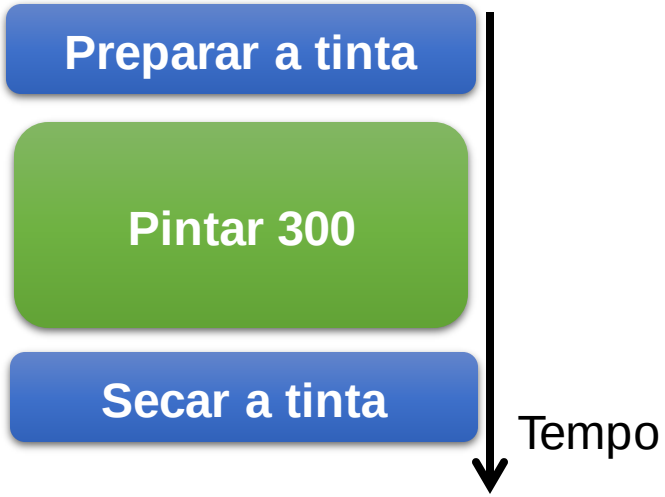
- Sempre existirão partes sequenciais em um programa!

1. Preparar a tinta
2. Pintar 300 estacas
3. Aguardar a tinta secar

- Sempre existirão partes sequenciais em um programa!

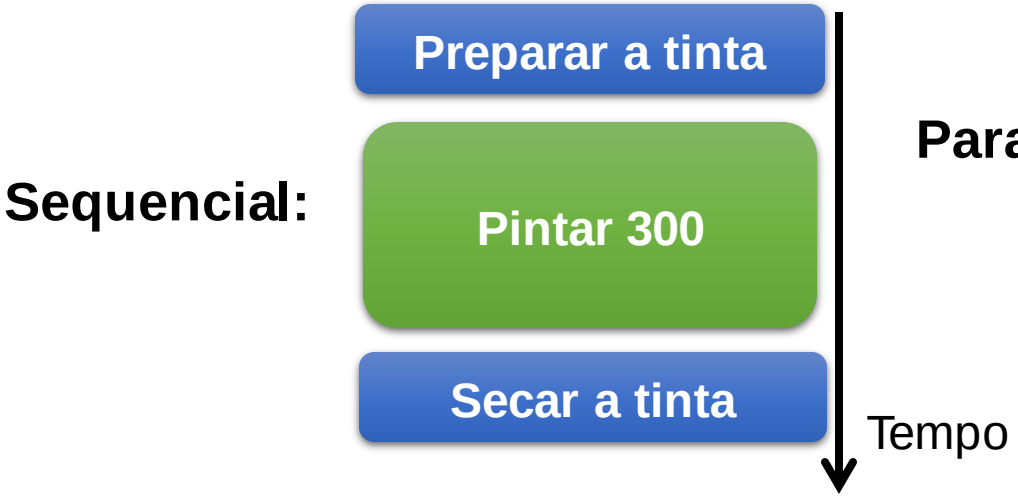
1. Preparar a tinta
2. Pintar 300 estacas
3. Aguardar a tinta secar

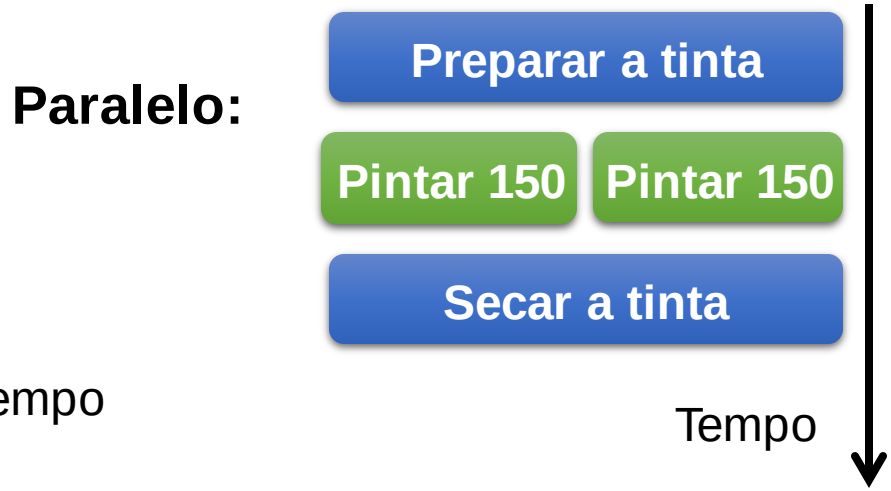
Sequencial:



- Sempre existirão partes sequenciais em um programa!

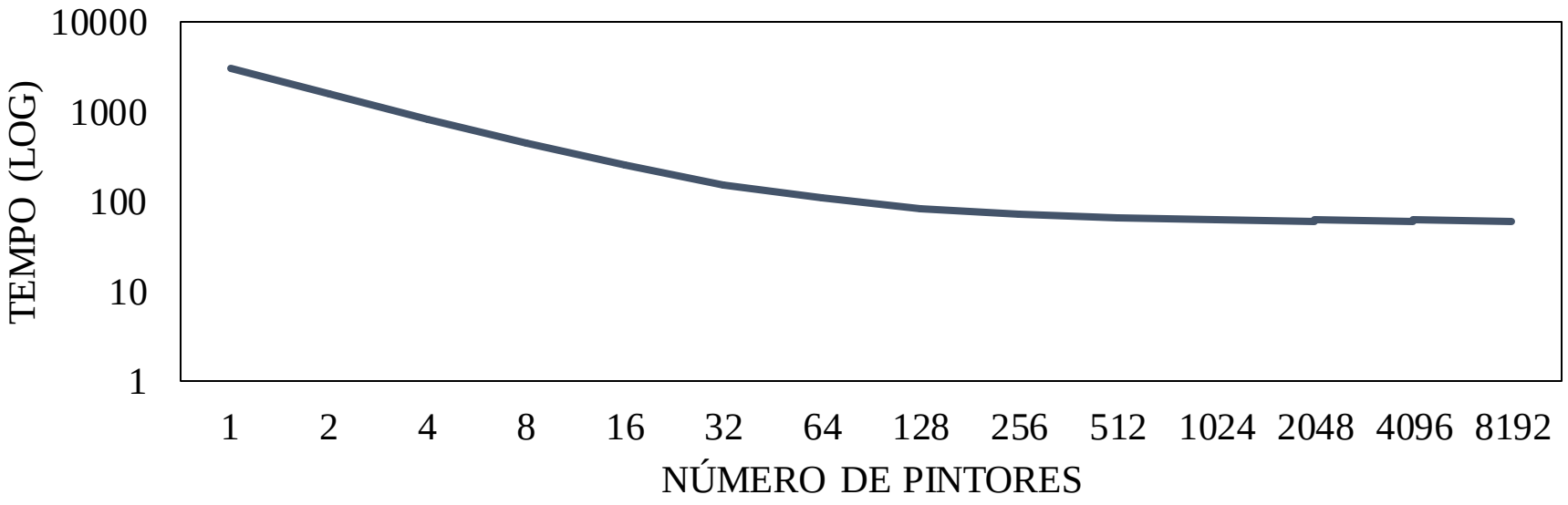
1. Preparar a tinta
2. Pintar 300 estacas
3. Aguardar a tinta secar





- E se ao invés de 2 pintores, forem:
  - 10 pintores
  - 100 pintores
  - 1000 pintores
- Ainda existiria melhora no desempenho?

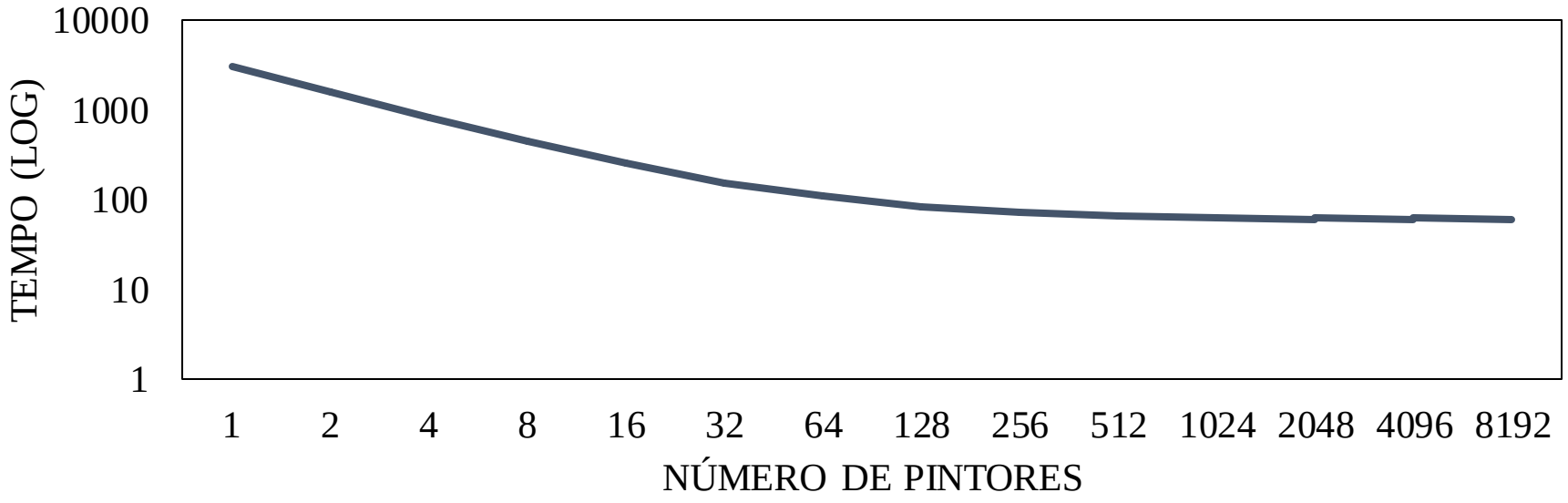
- E se ao invés de 2 pintores, forem:
  - 10 pintores
  - 100 pintores
  - 1000 pintores
- Ainda existiria melhora no desempenho?



- E se a
  - 10
  - 10
  - 10

Divisão das tarefas
Comunicação
Sincronização
Entre outros fatores

- Ainda existiria melhora no desempenho?



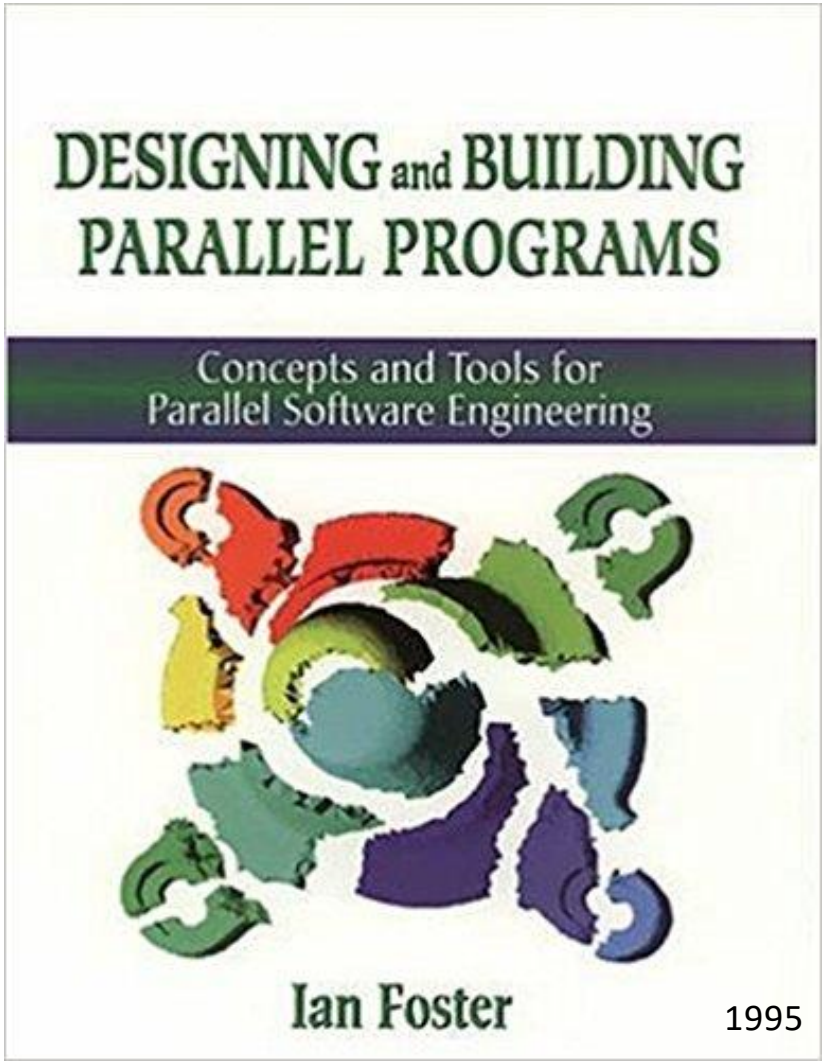
Conhecimento do Hardware

- Ideal seria não precisar conhecer detalhes da máquina
- Em PP o conhecimento da máquina influencia diretamente o desempenho
  - Paradigma de comunicação
    - Memória compartilhada x distribuída
    - Preciso comunicar?
  - Ferramenta de programação
    - Threads, openMP
    - MPI
  - Modelagem do problema

Identificando Oportunidades de Paralelismo

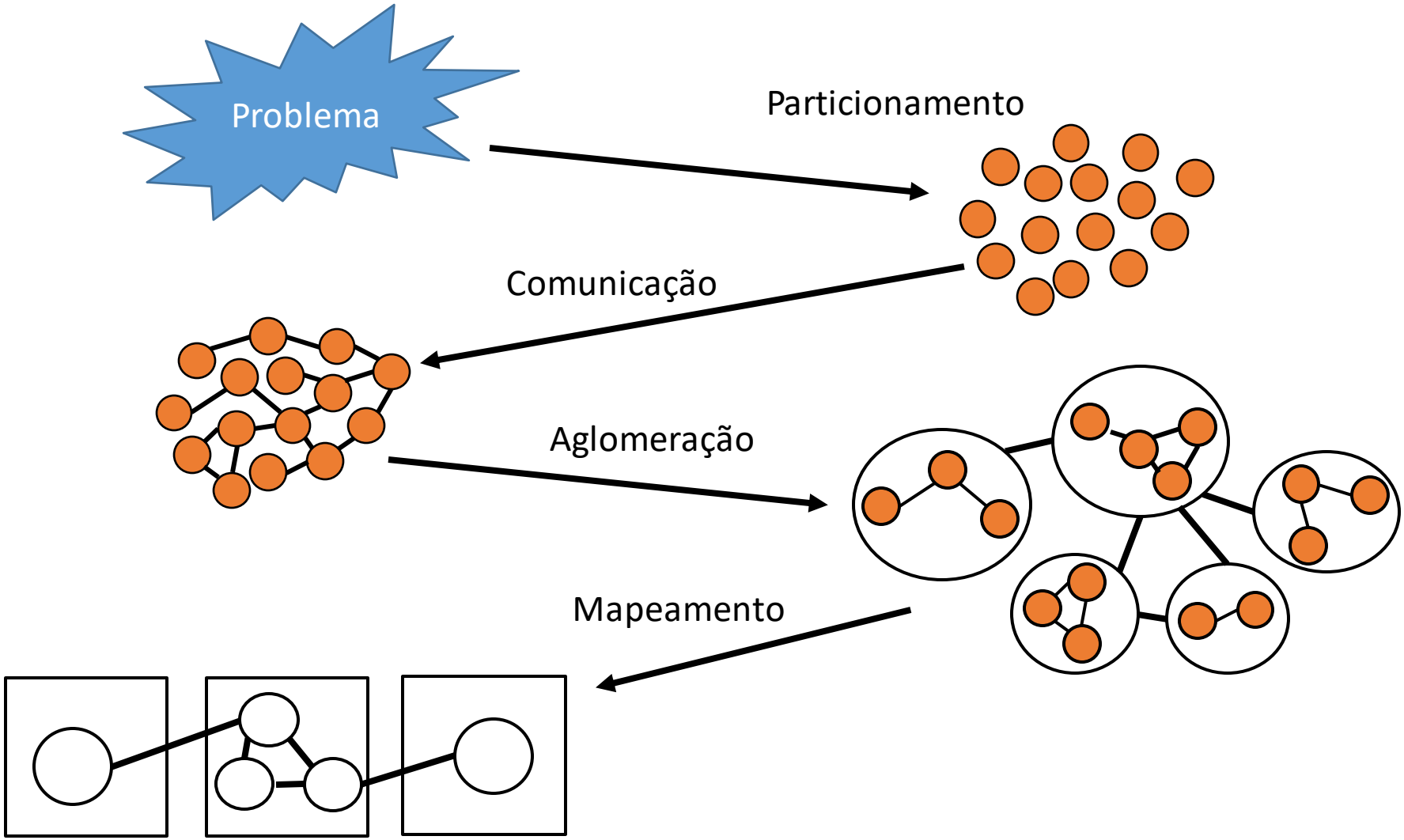
- Estudar a aplicação:
  - Tempo de execução
  - Ferramentas de geração de profile (gprof, por exemplo)
  - Avaliar potenciais funções e loops
- Identificar tarefas que possam ser executadas concorrentemente
- Definir a maneira mais eficiente de paralelizá-la
  - Obter um código paralelo ideal pode levar horas, dias, semanas e até meses.

Metodologia para desenvolvendo algoritmos paralelos



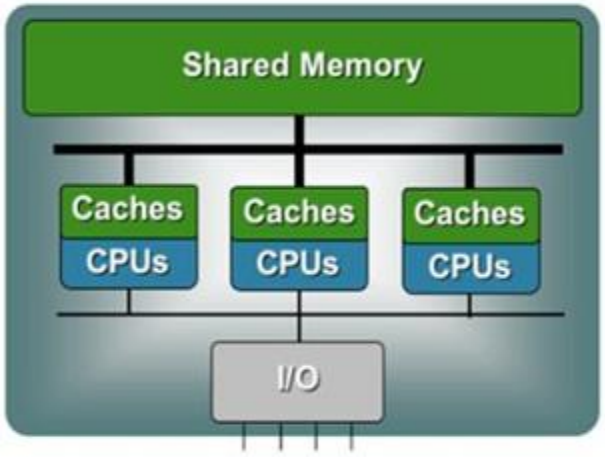
- Metodologia PCAM
  - Particionamento
  - Comunicação
  - Aglomeração
  - Mapeamento

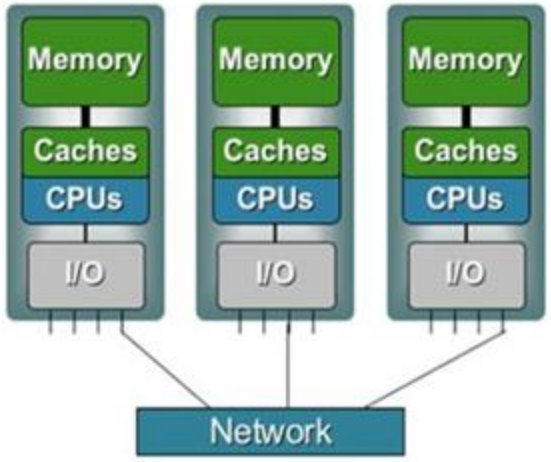
Desenvolvendo algoritmos paralelos (PCAM)



Foster, I. “Designing and Building Parallel Programs: Concepts and Tools for Parallel Software Engineering”, 1995

Modelos de Máquinas Paralelas





Memória Compartilhada (*Multicore, Maycore*)

Memória Distribuída (*Cluster*)

Interfaces de Programação Paralela

- Memória Compartilhada
  - Baseado em pragmas e diretivas:
    - OpenMP
    - Cilk++
  - Baseado em funções:
    - POSIX Threads
    - Intel TBB
- Memória Distribuída
  - MPI

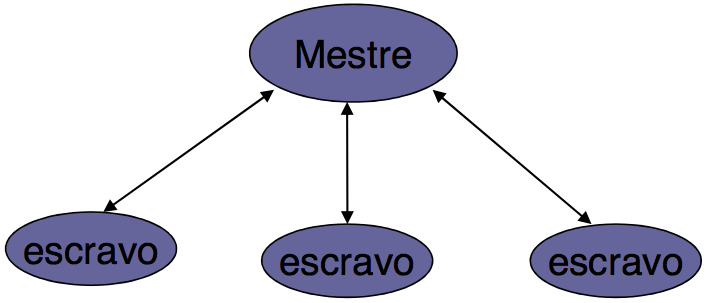
Modelos de Aplicação

- As aplicações são modeladas usando um grafo que relaciona as tarefas e trocas de dados
  - Vértices: tarefas
  - Arestas: trocas de dados (comunicação e/ou sincronização)

- Modelos básicos
  - Mestre/escravo
  - Pipeline
  - Divisão e conquista
  - Fases paralelas

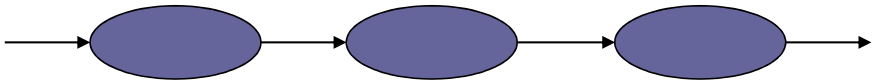
Modelo Mestre/escravo

- Mestre executa as tarefas essenciais do programa e divide o resto entre processos escravos
- Escalonamento centralizado (gargalo)
- Maior tolerância a falhas: um escravo para, não todo processamento



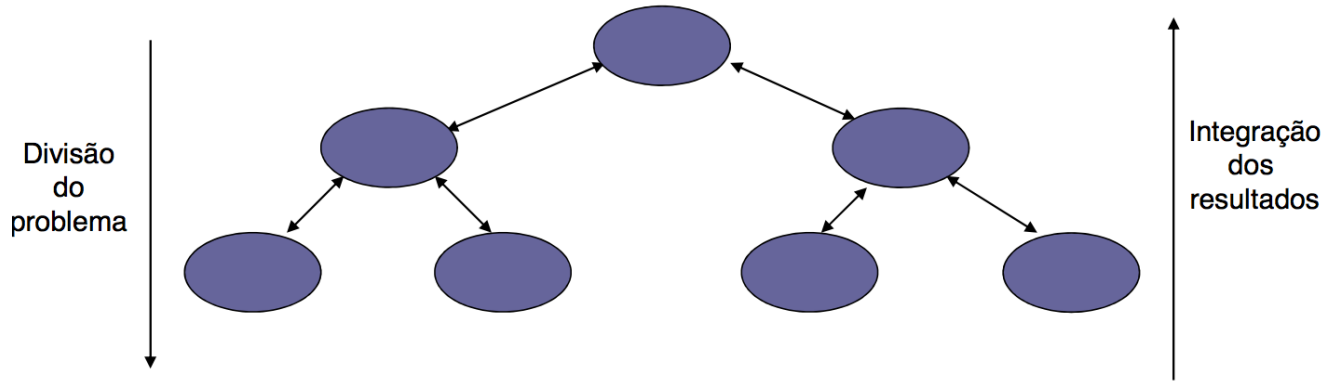
Pipeline

- Encadeamento de processos
- Pipeline virtual (rede não muda)
- Fluxo contínuo de dados
- Sobreposição de comunicação e computação
- Não tolerante a falhas – 1 para, todos param



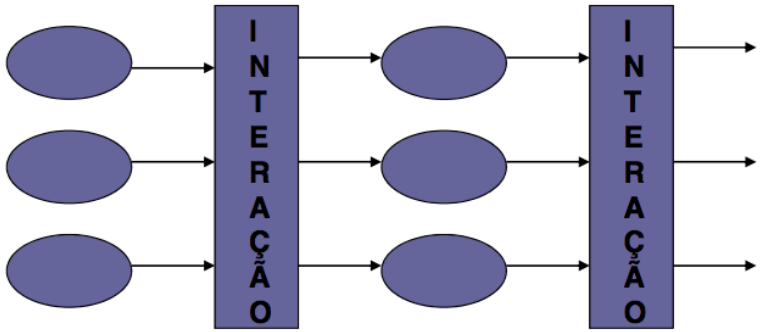
Divisão e conquista

- Processos organizados em uma hierarquia (pai e filhos)
- Processo pai divide trabalho e repassa uma fração deste ao seus filhos
- Integração dos resultados de forma recursiva
- Dificuldade de balanceamento de carga na divisão das tarefas



Fases Paralelas

- Etapas de computação e sincronização
- Problema de balanceamento devido ao sincronismo
  - Processos que acabam antes
- Overhead de comunicação
  - Toda comunicação é realizada ao mesmo tempo



Exemplo de problema

- Como paralelizar a ordenação de um grande vetor?
  - Mestre/escravo
  - Pipeline
  - Fases paralelas
  - Divisão e conquista

Mestre/Escravo

- O mestre envia partes do vetor para cada escravo, que retorna este ordenado e espera por um novo pedaço do vetor
  - Muita comunicação
  - Não é um bom modelo para o caso

Pipeline

- Cada processador faz uma troca simples, sendo necessário para um vetor de tamanho n, n-1 processadores
  - Só 1 vetor, só uma operação
  - Não é o modelo ideal

Fases Paralelas

- Cada processo inicia com uma parte do vetor, ordenam e interagem com os outros processos a fim de realizar trocas entre os limites de cada pedaço do vetor
  - Faz ordenação local, e mando o meu maior e recebo o menor do vizinho (% para vetores grandes)
  - Condição parada:
    - meu maior é menor do que o menor do meu vizinho para todos processadores?

Divisão e Conquista

- Subdivisão do vetor em partes menores a serem ordenadas por outros processos
  - Vai subindo na árvore intercalando o vetor preordenado dos filhos

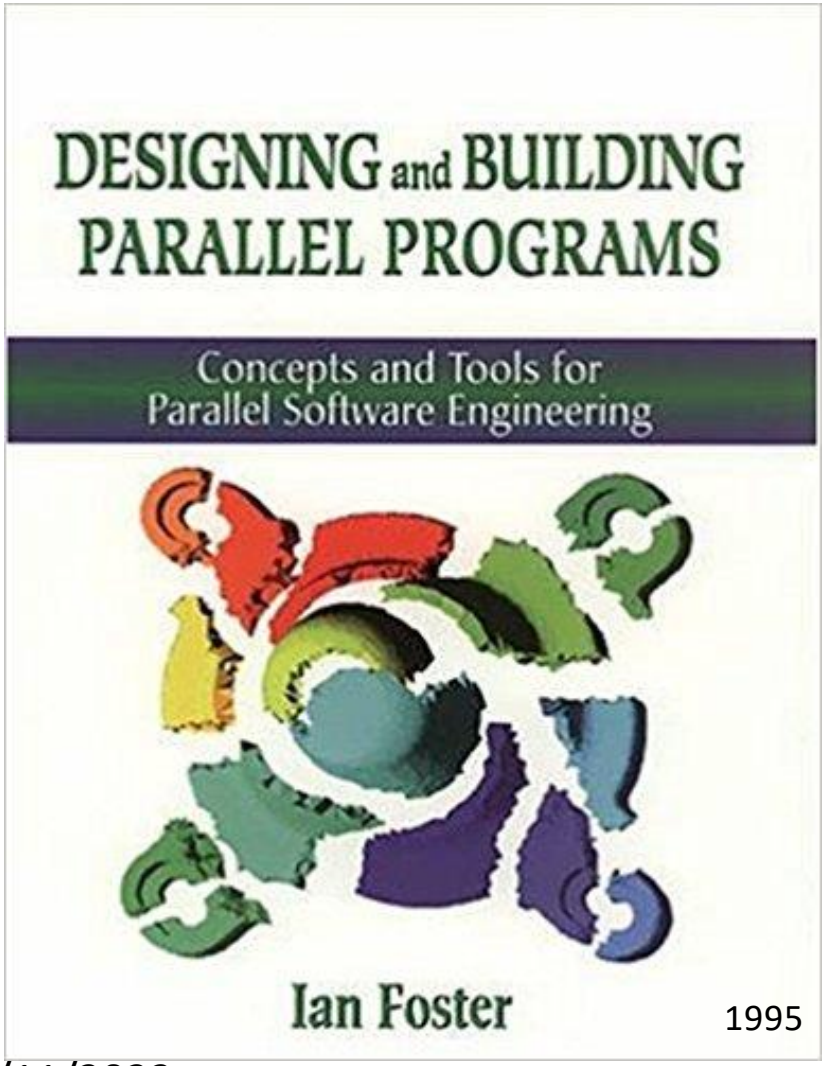
Referênicas

- Slides baseados no material dos Prof. Arthur F. Lorenzon e Antonio Carlos S. Beck da UFRGS

- Foster, I. “Designing and Building Parallel Programs: Concepts and Tools for Parallel Software Engineering”, 1995

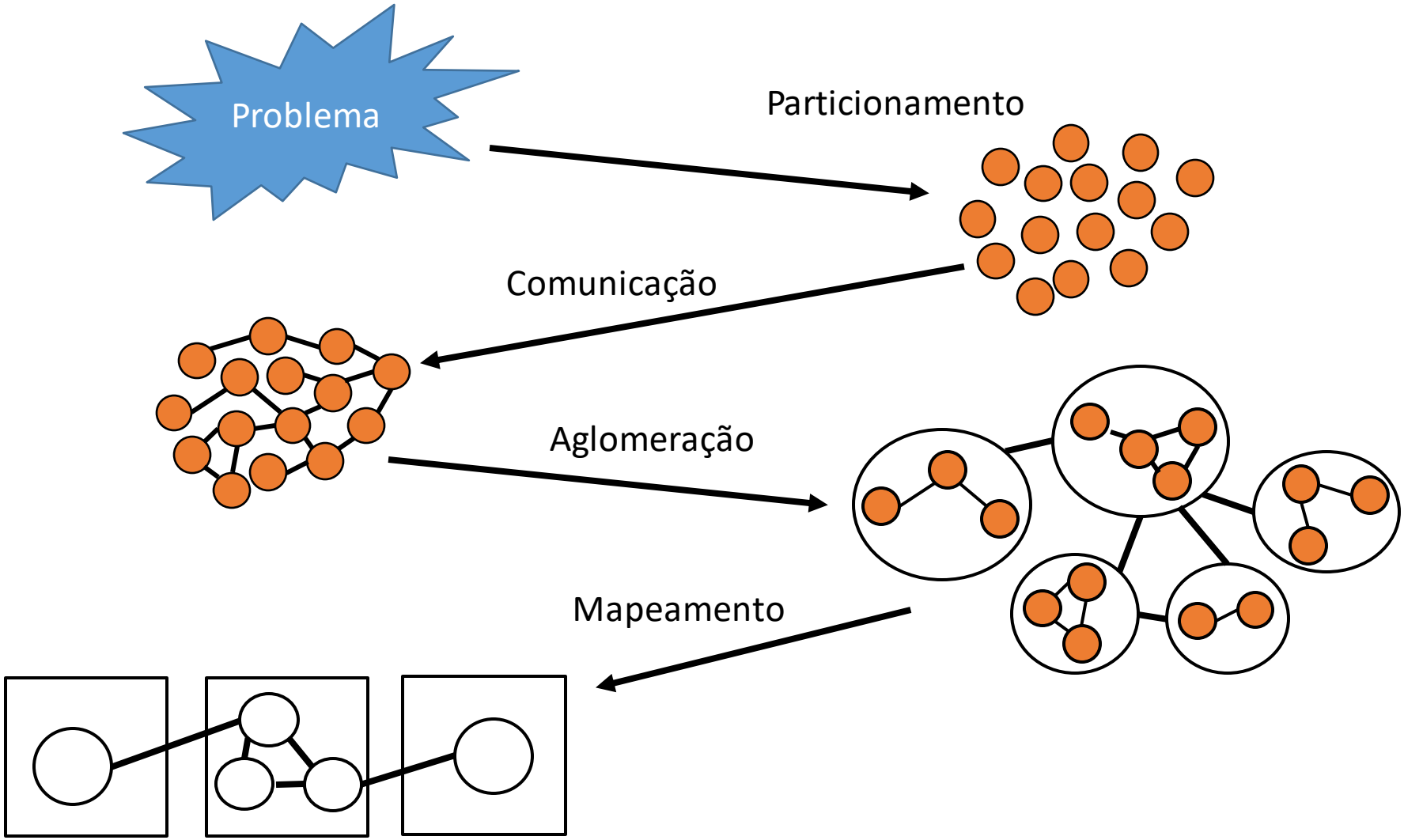
Material Adicional

Metodologia para desenvolvendo algoritmos paralelos



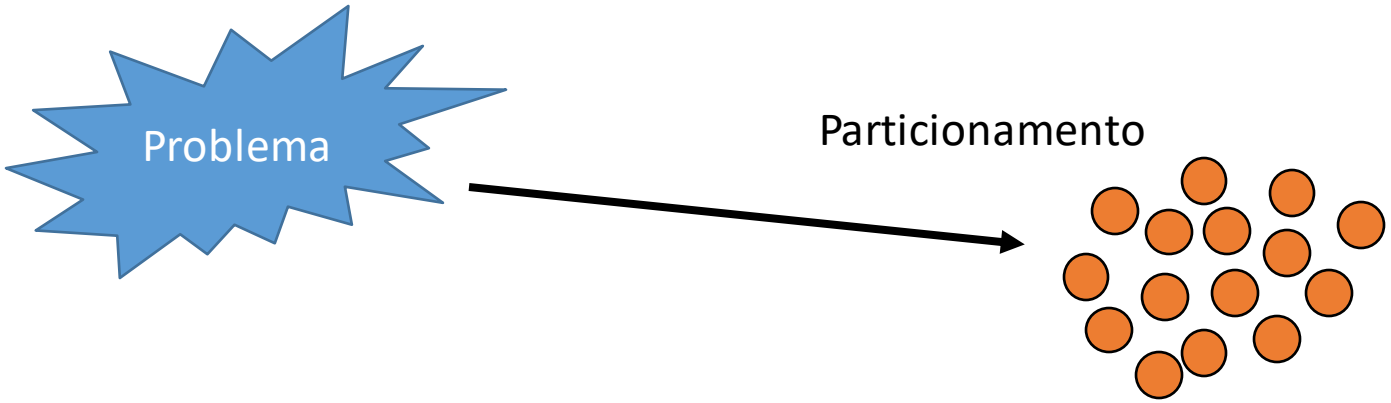
- PCAM
  - Particionamento
  - Comunicação
  - Aglomeração
  - Mapeamento

Desenvolvendo algoritmos paralelos (PCAM)



Foster, I. "Designing and Building Parallel Programs: Concepts and Tools for Parallel Software Engineering", 1995

Desenvolvendo algoritmos paralelos



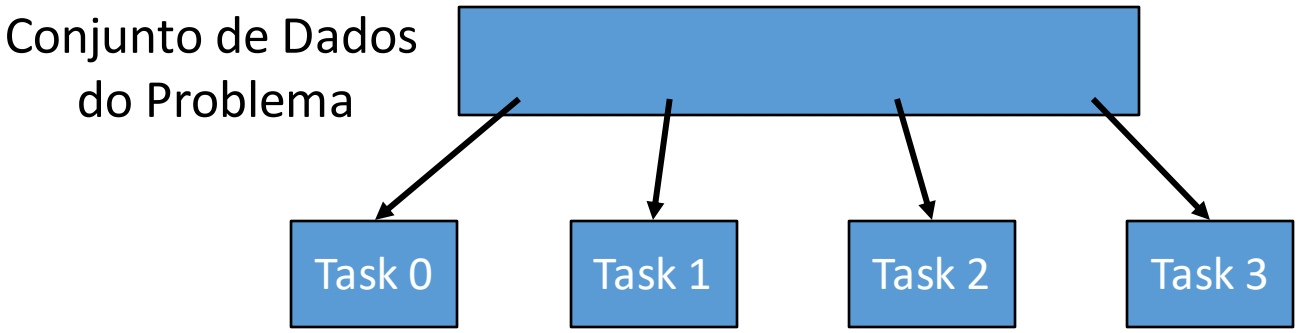
Particionamento

- Expor oportunidades de execução paralela
  - Como dividir os dados de um vetor, de uma matriz?
  - Como as funções serão computadas?

- Duas abordagens:
  - Decomposição de Domínio
  - Decomposição Funcional

Particionamento

- Decomposição de Domínio:



Particionamento

- Decomposição de Domínio:

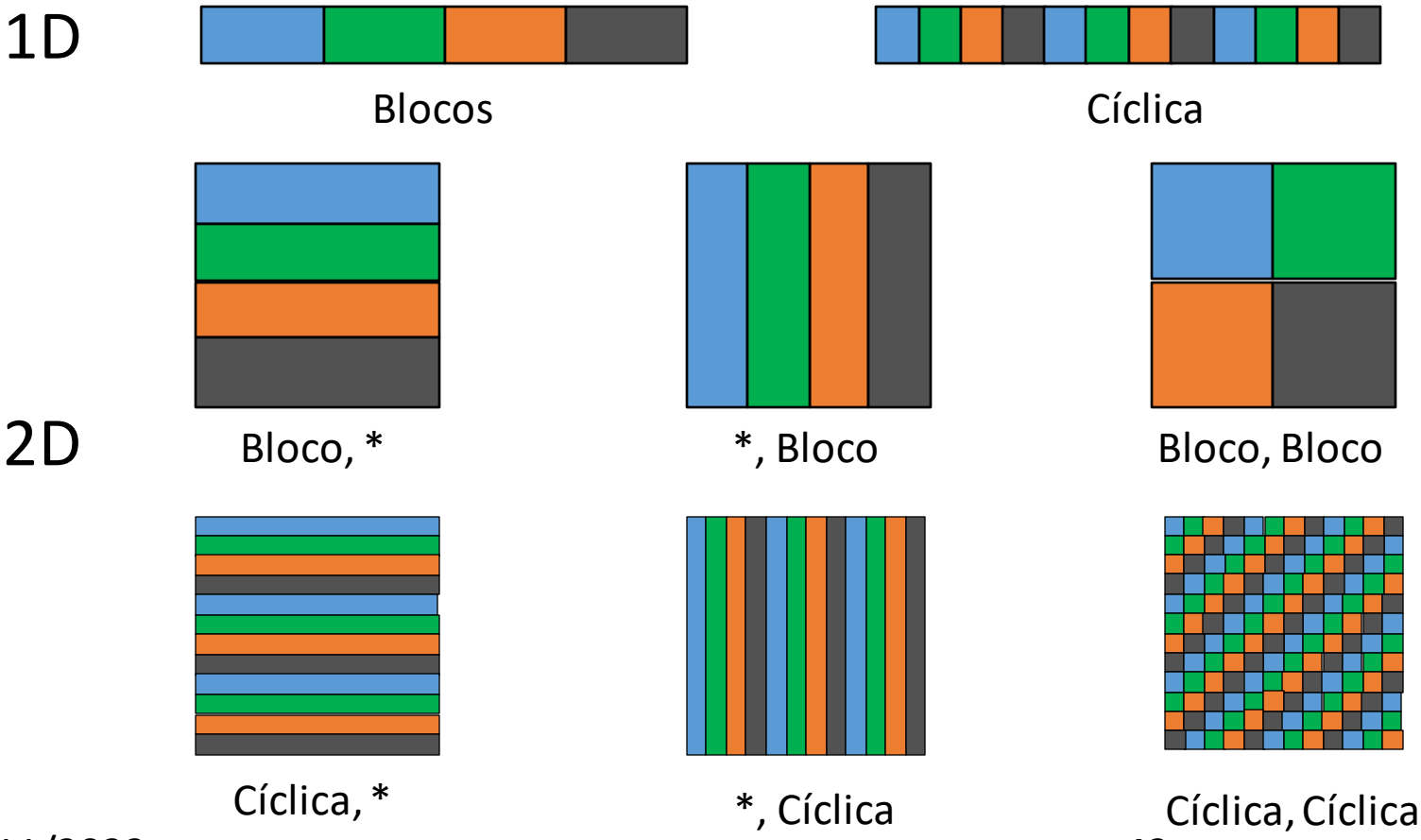
1D

Blocos

Cíclica

Particionamento

- Decomposição de Domínio:



Particionamento

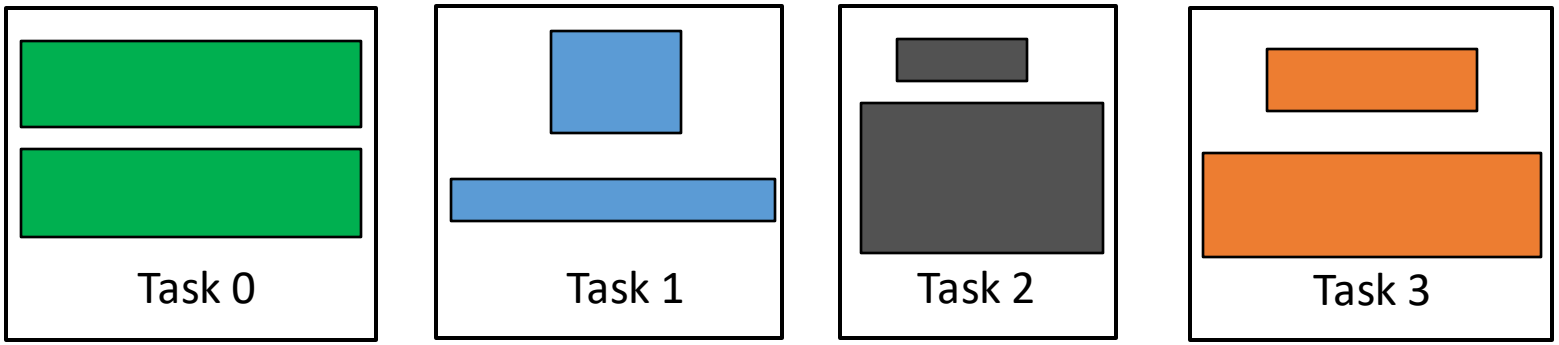
- Decomposição Funcional:

Conjunto de Instruções
do Problema

Particionamento

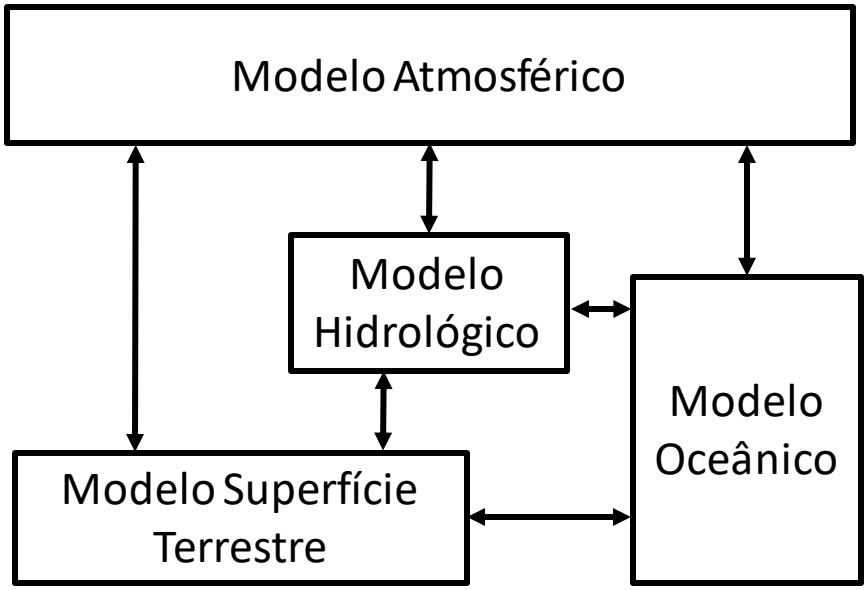
- Decomposição Funcional:

Conjunto de Instruções
do Problema



Particionamento

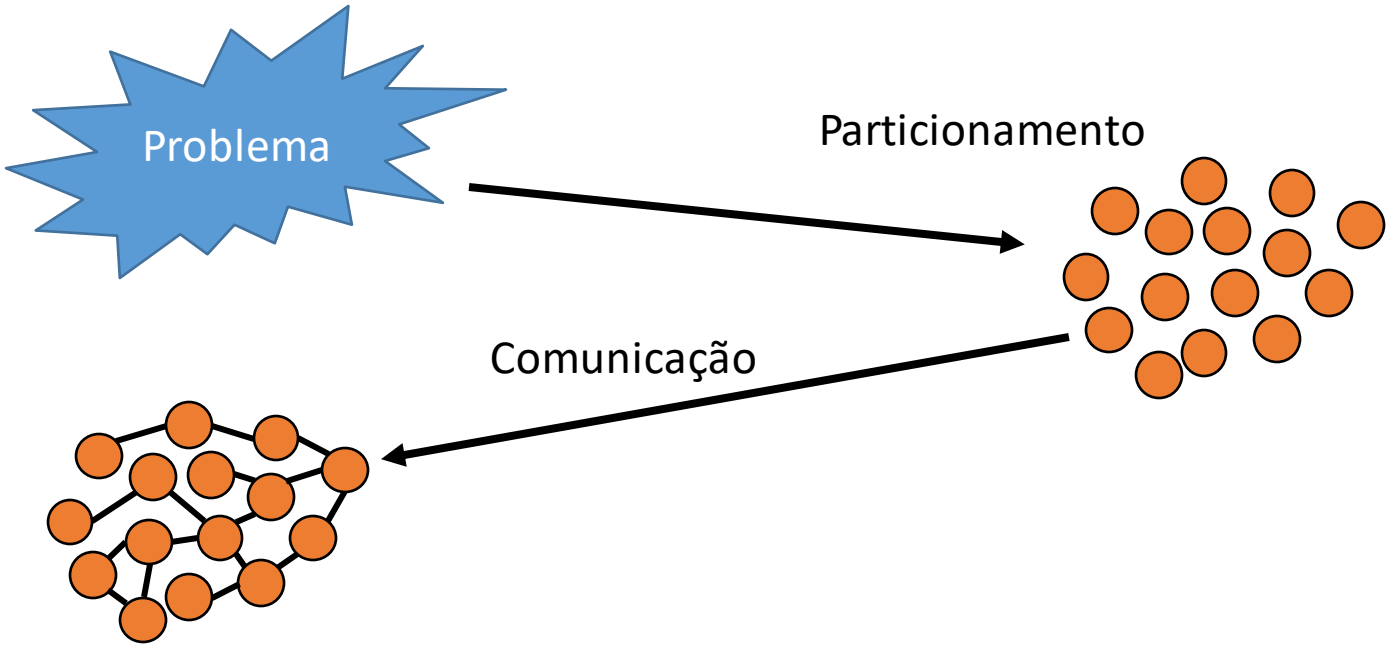
- Decomposição Funcional:



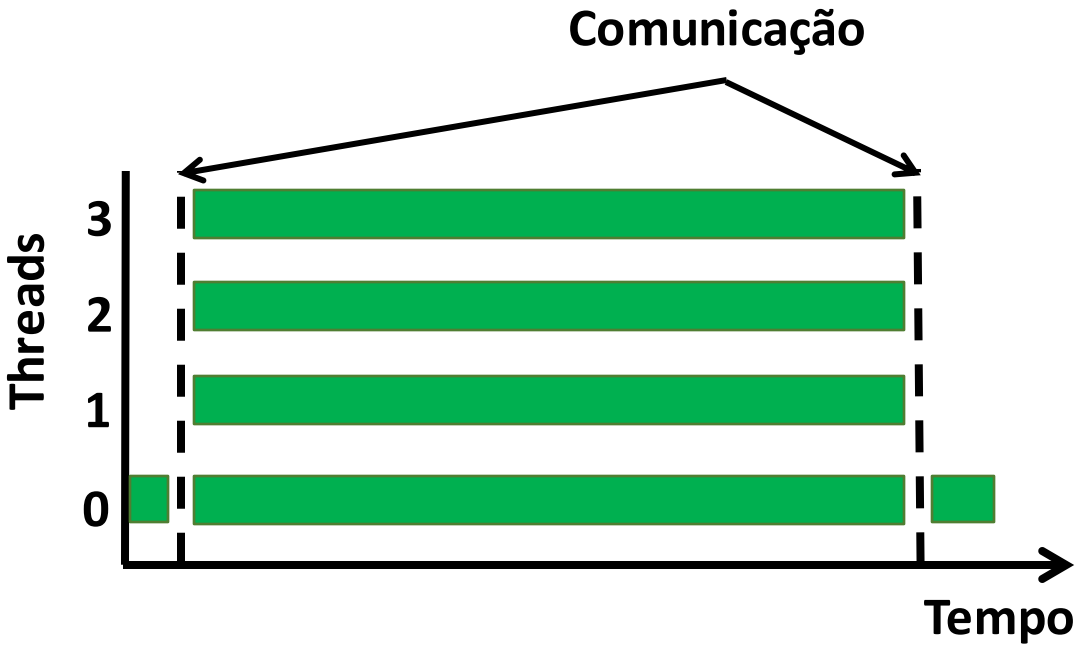
Particionamento

- Checklist:
  - O particionamento define mais tarefas do que o número de processadores do computador alvo?
  - O particionamento evita redundância de cálculo e operações de acessos a memória?
  - O número de tarefas escala com o tamanho do problema?
  - Foram idenfiticada partições alternativas?

Desenvolvendo algoritmos paralelos



Comunicação

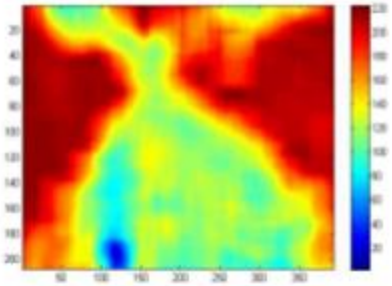


Comunicação

- A quantidade de comunicação depende do tipo de aplicação

Operações em imagens

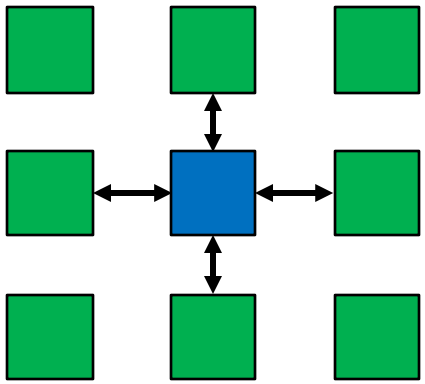




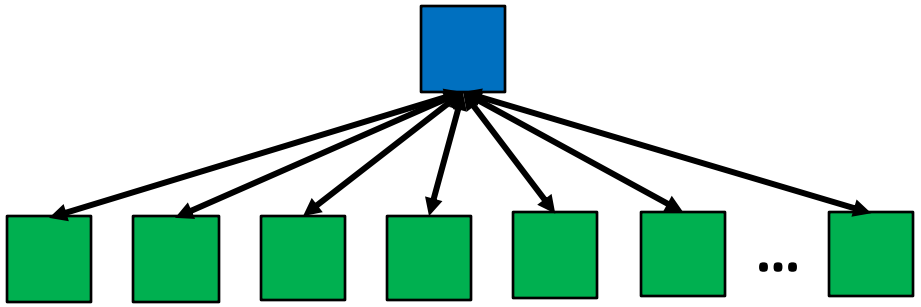
Aplicações relacionadas ao clima


Comunicação

Comunicação Local

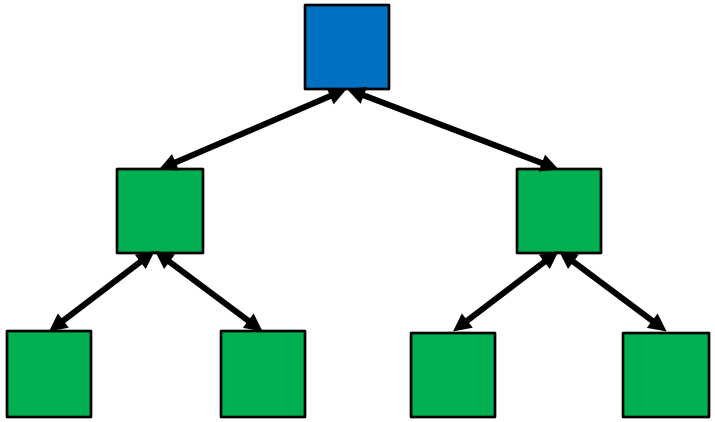


Comunicação Global

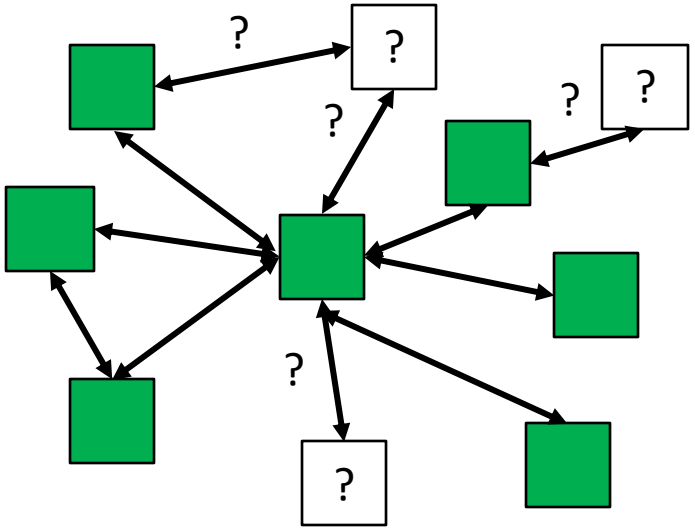


Comunicação

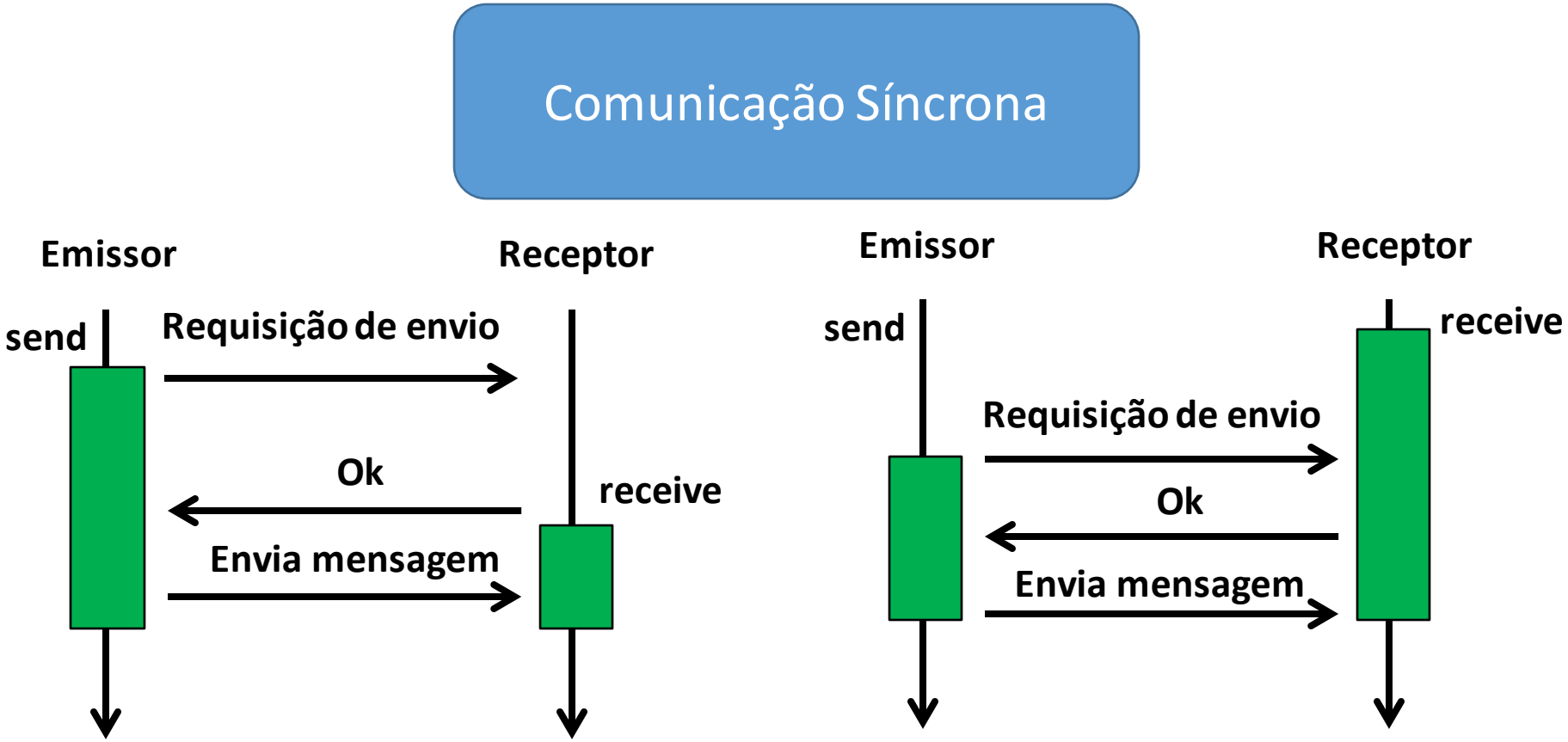
Comunicação Estruturada e Estática



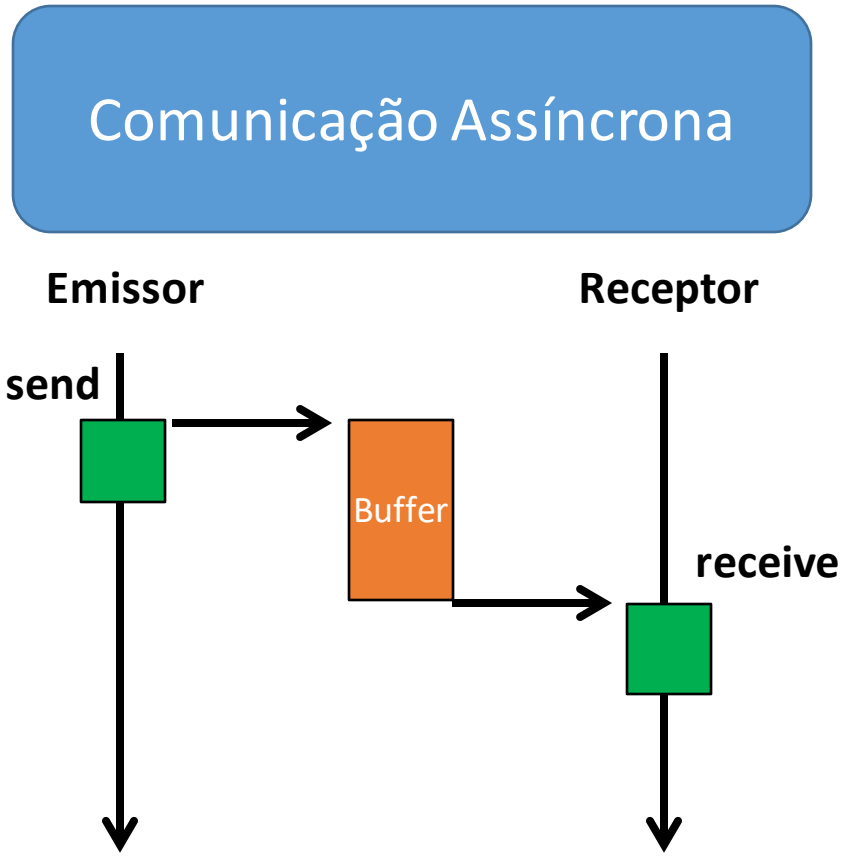
Comunicação Não Estruturada e Dinâmica



Comunicação



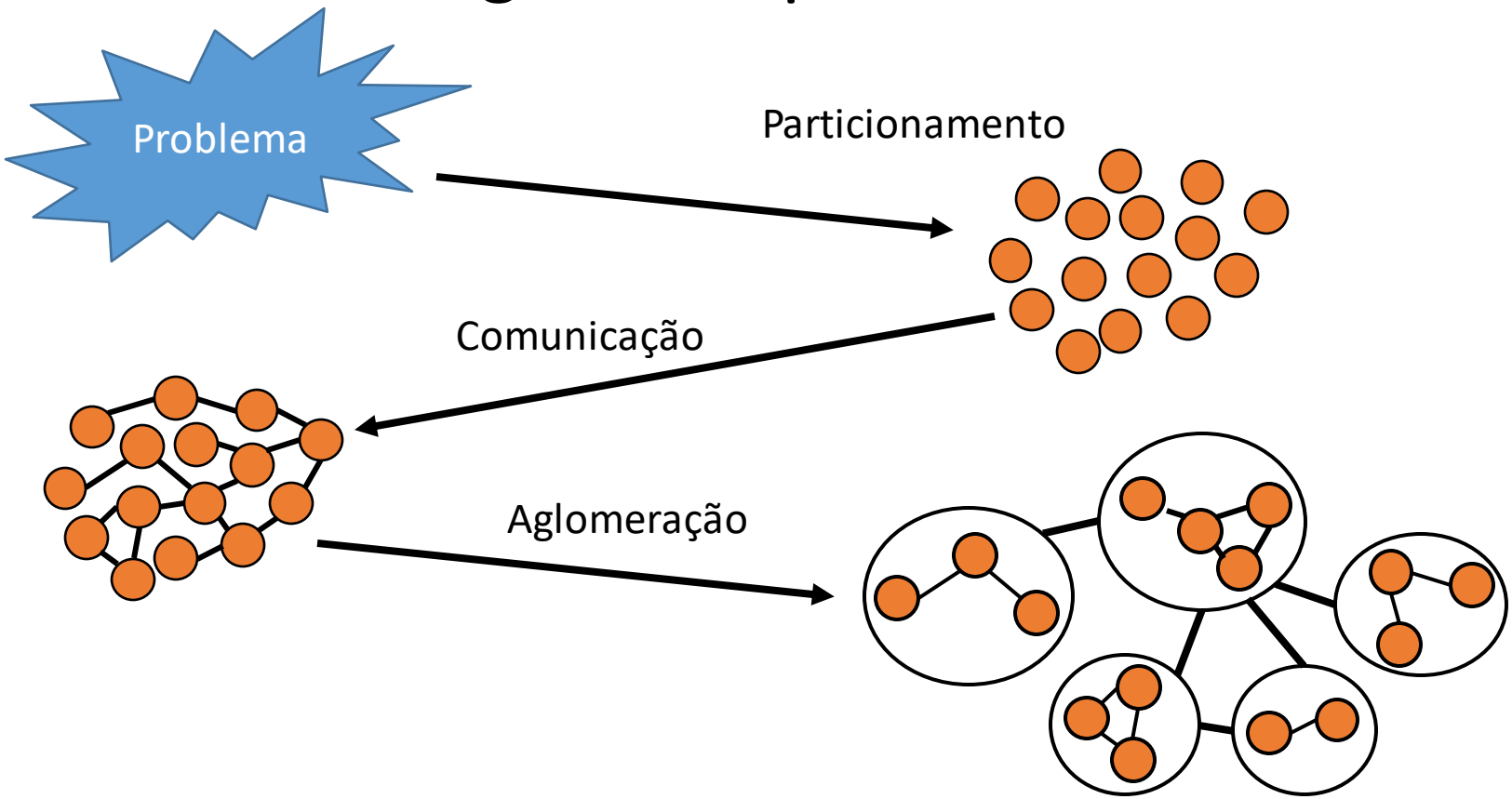
Comunicação



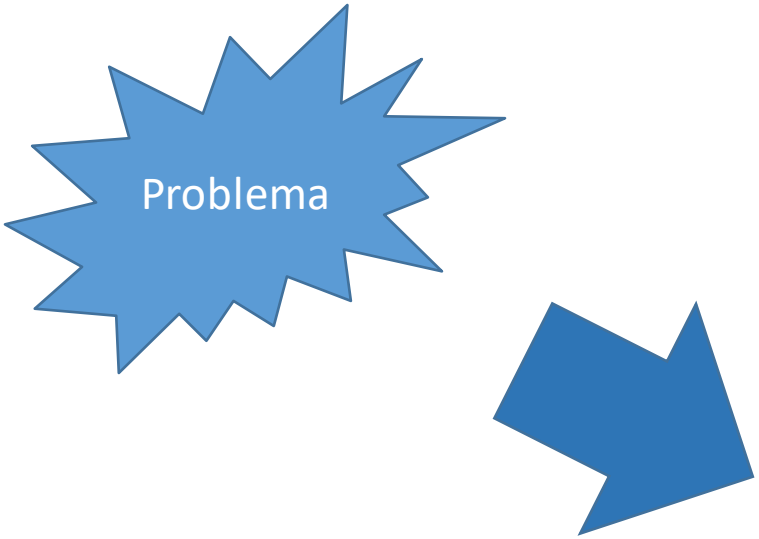
Comunicação

- Checklist:
  - Todas as tarefas realizam aproximadamente, o mesmo número de operações de comunicação?
  - Comunicações podem ser executadas concorrentemente com a computação das tarefas?

Desenvolvendo algoritmos paralelos

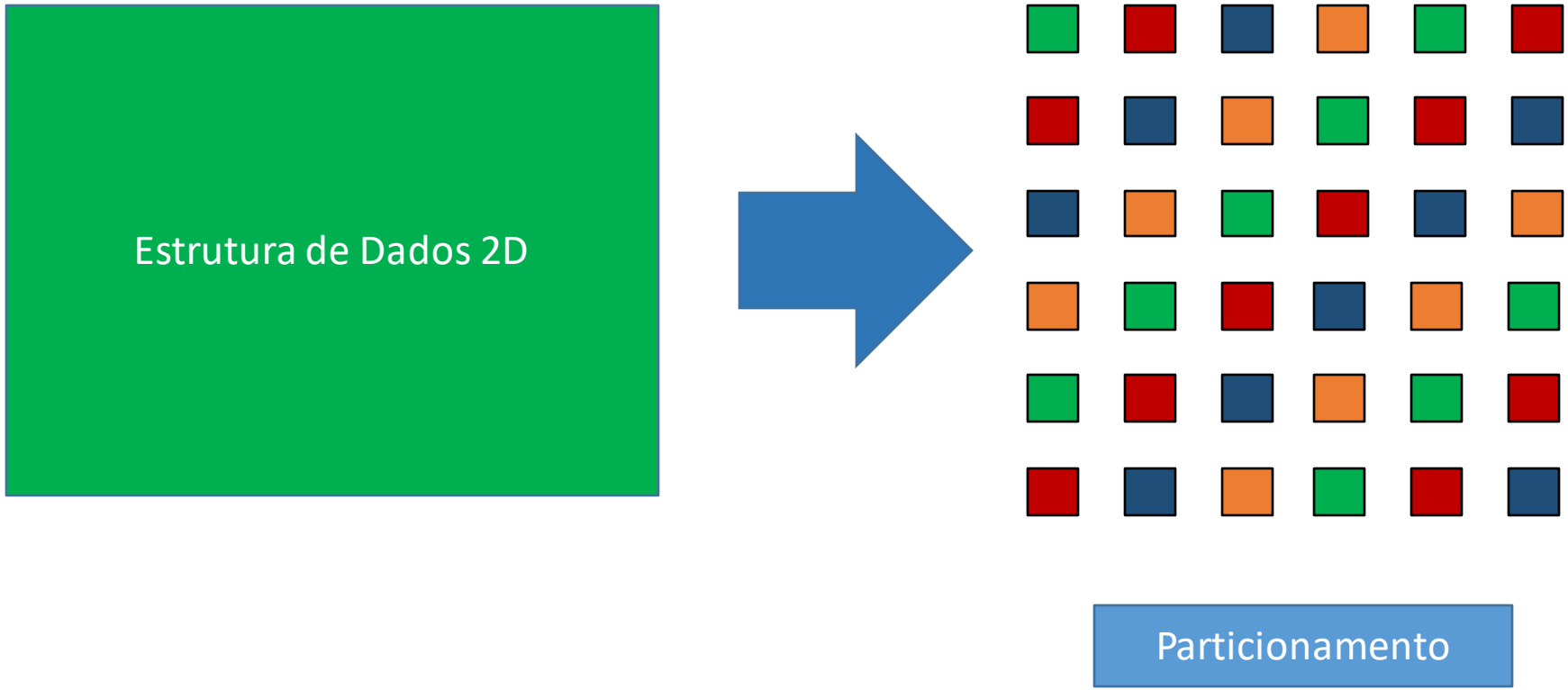


Aglomeração

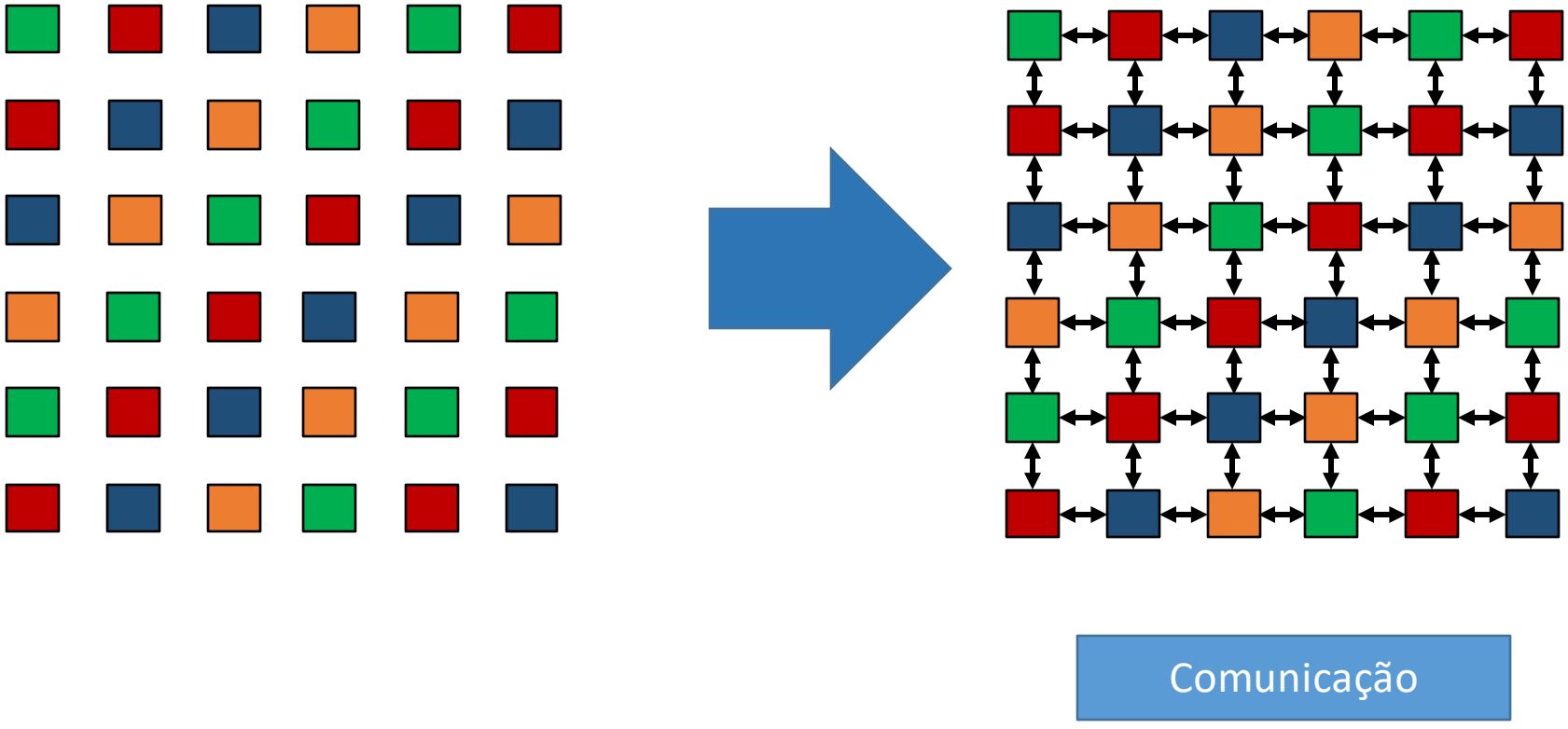


Estrutura de Dados 2D

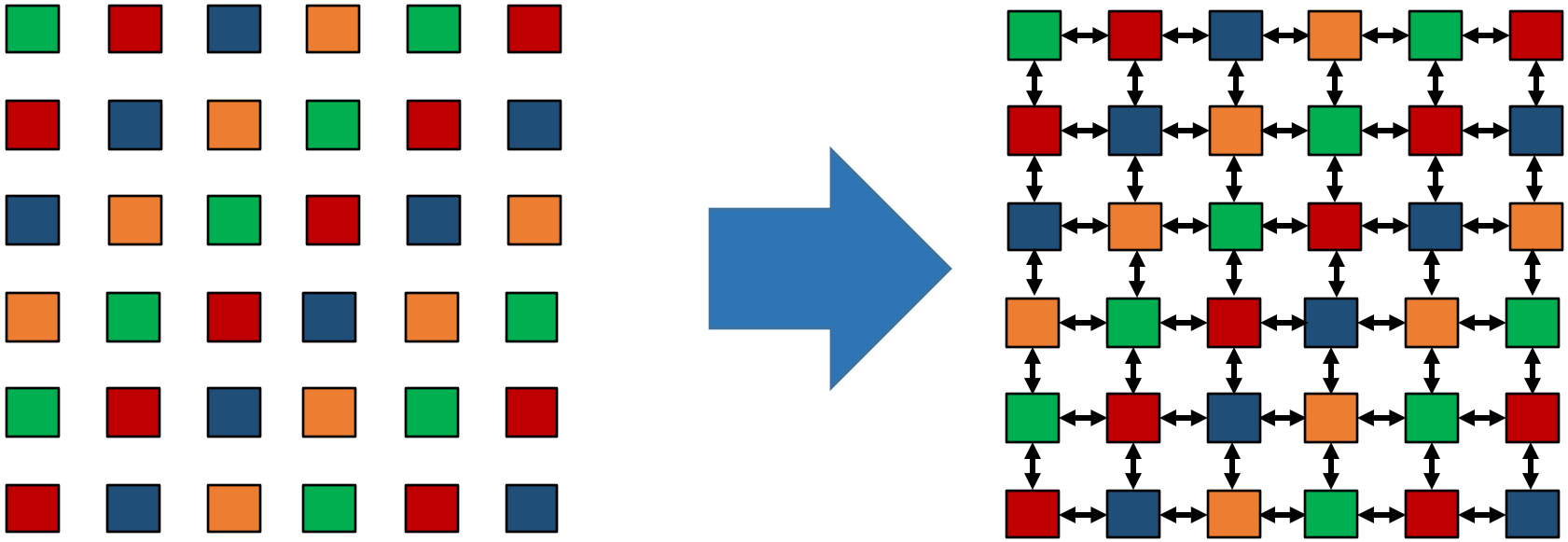
Aglomeração



Aglomeração

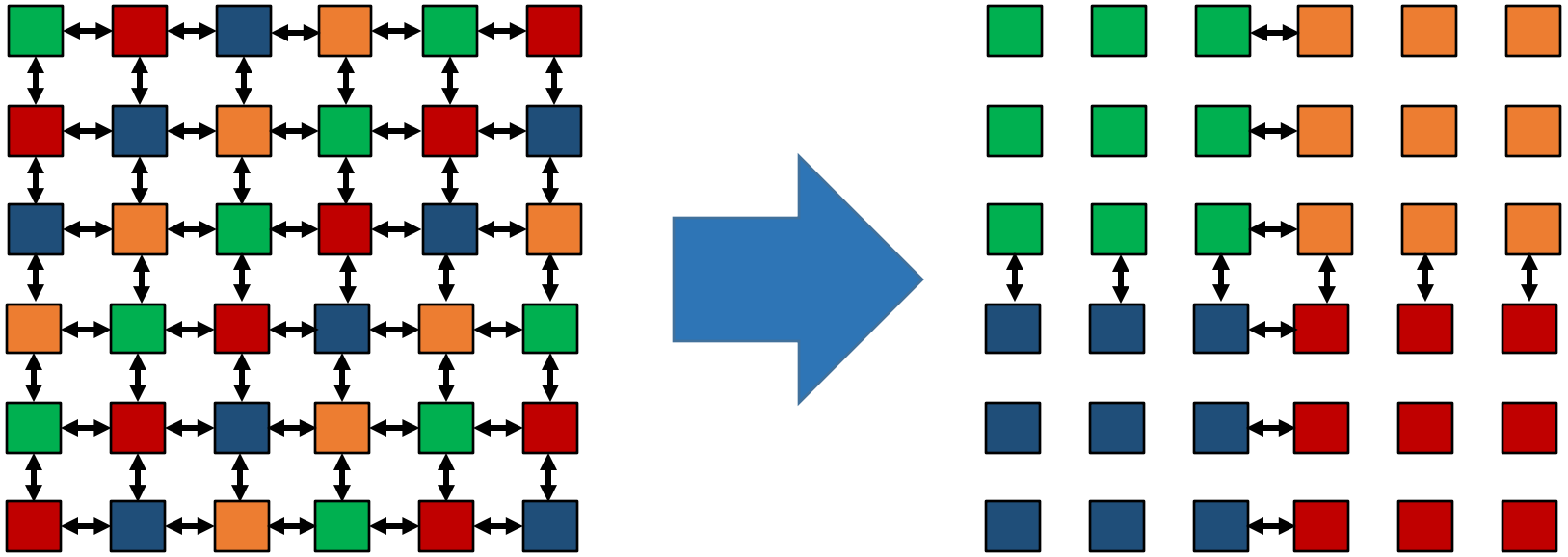


Aglomeração



A paralelização é eficiente?

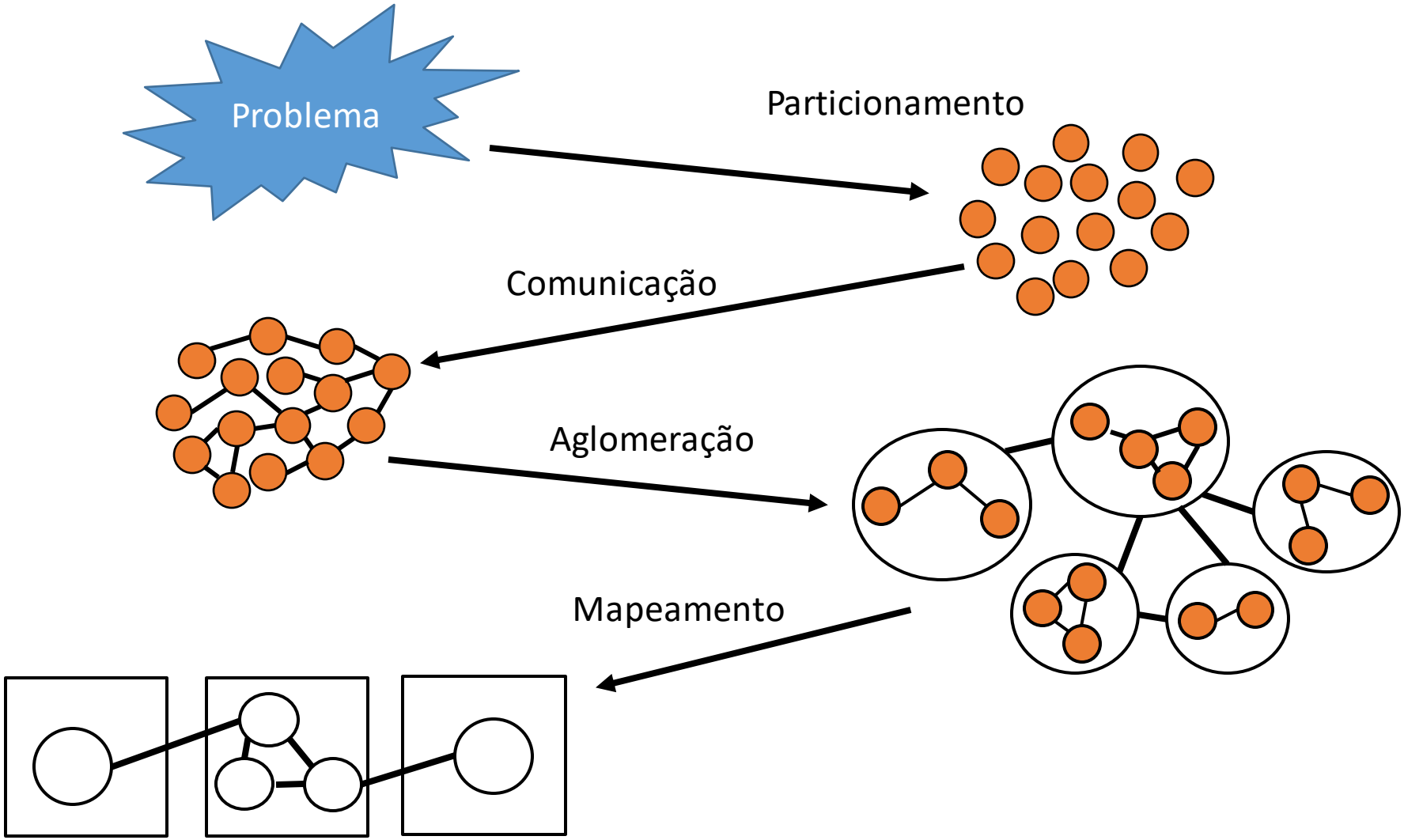
Aglomeração



Aglomeração

- Checklist:
  - Aglomeração reduziu o custo da comunicação e aumentou a localidade dos dados?
  - Se a aglomeração replicou computação, o benefício sobrepôs o custo da replicação?
  - Se a aglomeração replicou dados, o benefício sobrepôs o custo da comunicação adicional?
  - As tarefas ficaram com tempo similar de comunicação e computação?
  - O número de tarefas escala com o tamanho do problema?
  - ….

Desenvolvendo algoritmos paralelos



Mapeamento

- Definição de “onde” cada tarefa irá executar.
- Estratégias:
  - Alocar as tarefas em processadores diferentes, para aumentar a concorrência?
  - Alocar as tarefas no mesmo processador, para aumentar a localidade?
  - Qual a melhor?