Universidade Tecnológica Federal do Paraná (UTFPR)
Campus Curitiba
Programa de Pós-graduação em Engenharia e Informática (CPGEI)
Laboratório de Bioinformática

Fundamentos de Computação Paralela

Heitor Silvério Lopes

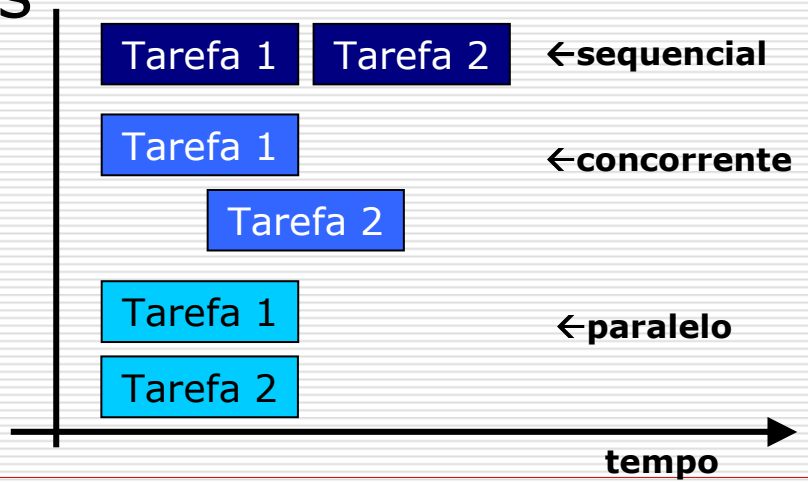
hslopes@utfpr.edu.br

Computação Paralela

- Objetivo principal:
  - Reduzir o tempo total de processamento
- Natureza dos problemas que demandam computação paralela:
  - Grande quantidade de dados a serem processados
  - Grande quantidade de interações para processar dados
  - Ambos
- Metodologia:
  - Identificação e exploração de eventos concorrentes da computação
- Principais fatores limitantes:
  - Quantidade e natureza do hardware disponível
  - Dificuldade de projeto, implementação e teste de hardware/software paralelo

Concorrência x Paralelismo

- Concorrência: oposto da sequencialidade, tarefas que podem ser entrelaçadas no tempo. Ex. SO Multitarefa, threads
- Paralelismo: tarefas podem ser executadas simultaneamente em processadores reais independentes



Histórico das arquiteturas paralelas

- No passado (70-80) os multiprocessadores eram muito caros e de baixa escalabilidade
- Recentemente (90-) passou-se a utilizar componentes standard de baixo custo individual e grande escalabilidade:
  - PCs, processadores multicore, clusters, grids, etc.

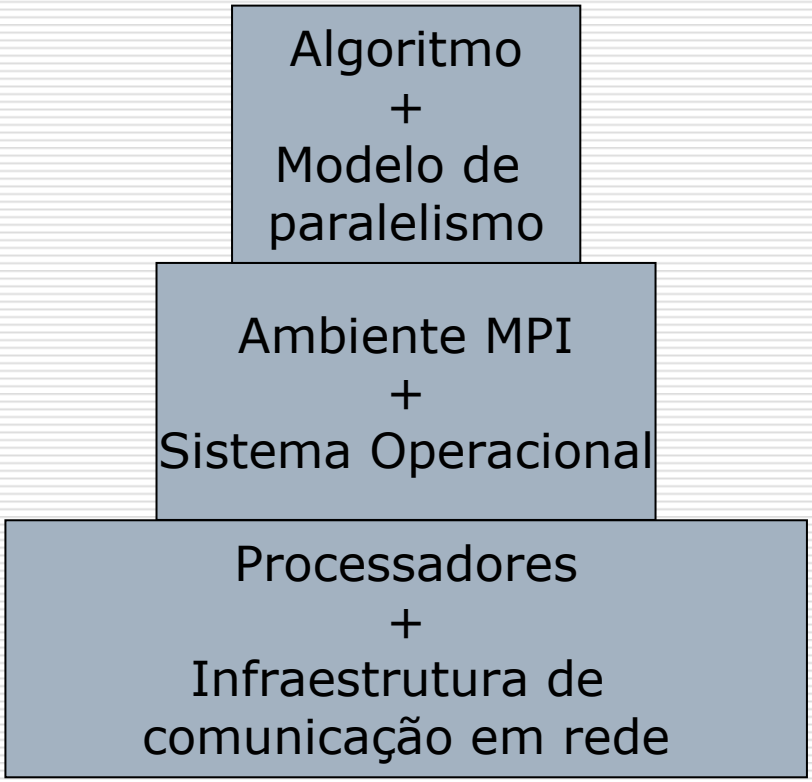
Arquiteturas paralelas (cap.1 Roosta)

- Flynn propôs 4 grupos computadores, de acordo com os mecanismos de controle (ordem de execução) e de fluxo de dados (acesso aos operandos):
  - SISD (Single Instruction stream Single Data stream)
  - SIMD (Single Instruction stream Multiple Data stream)
  - MISD (Multiple Instruction stream Single Data stream)
  - MIMD (Multiple Instruction stream Multiple Data Stream)

Arquiteturas paralelas

- SISD: inclui computadores pessoais monoprocessados, baseados no modelo de Von Neumann (década de 40)
- SIMD: uma única instrução é executada por varios EPs sobre diferentes dados
- MISD: de fato não existem, as mais próximas são as arquiteturas pipeline
- MIMD: inclui os multiprocessadores, processadores multicore, clusters de computadores e arquiteturas dataflow

Arquitetura de uma aplicação paralela



Metodologia para projeto de algoritmos paralelos (cap.5 Roosta)

- Considerar primeiro os aspectos independentes do hardware utilizado, tais como concorrência de operações e decomposição de tarefas
- Posteriormente considerar aspectos dependentes do hardware: custo de comunicação e número de processadores
- A metodologia de projeto tem 4 fases:
  1. Particionamento
  2. Comunicação
  3. Agrupamento
  4. Mapeamento e agendamento

1- Particionamento

- Identificar as oportunidades de paralelização dividindo o processamento em várias partes pequenas. Métodos:
  - Particionamento do domínio: foco nos dados associados com o problema e determina a computação apropriada para estes dados
  - Particionamento funcional: foco na decomposição do processamento em tarefas disjuntas
- Os métodos podem ser aplicados a diferentes partes do problema
- É fundamental a análise de dependência/ordenamento de tarefas, dependência/compartilhamento de dados
- Deve-se evitar replicação de tarefas ou de dados
- Deve-se determinar o paradigma de paralelização: pipeline, master-slave, cliente-servidor, heartbeat, broadcasting

2- Comunicação

- Objetiva estabelecer uma estrutura de comunicação entre tarefas e um meio de coordenar a execução de tarefas
- Envolve a transferência de dados entre tarefas e o controle de execução
- Há duas estruturas envolvidas:
  - Estrutura de canal de comunicação: diz respeito ‘a ligação direta ou indireta de tarefas que requerem dados de tarefas que produzem dados
  - Estrutura de passagem de mensagens: especifica as mensagens válidas que são enviadas pelo canal
- É uma fase fundamental devido ao custo envolvido na comunicação
- Utilizar os mecanismos do ambiente (p.ex.MPI) para criar, terminar, sincronizar e gerenciar processos, bem como distribuir/receber dados

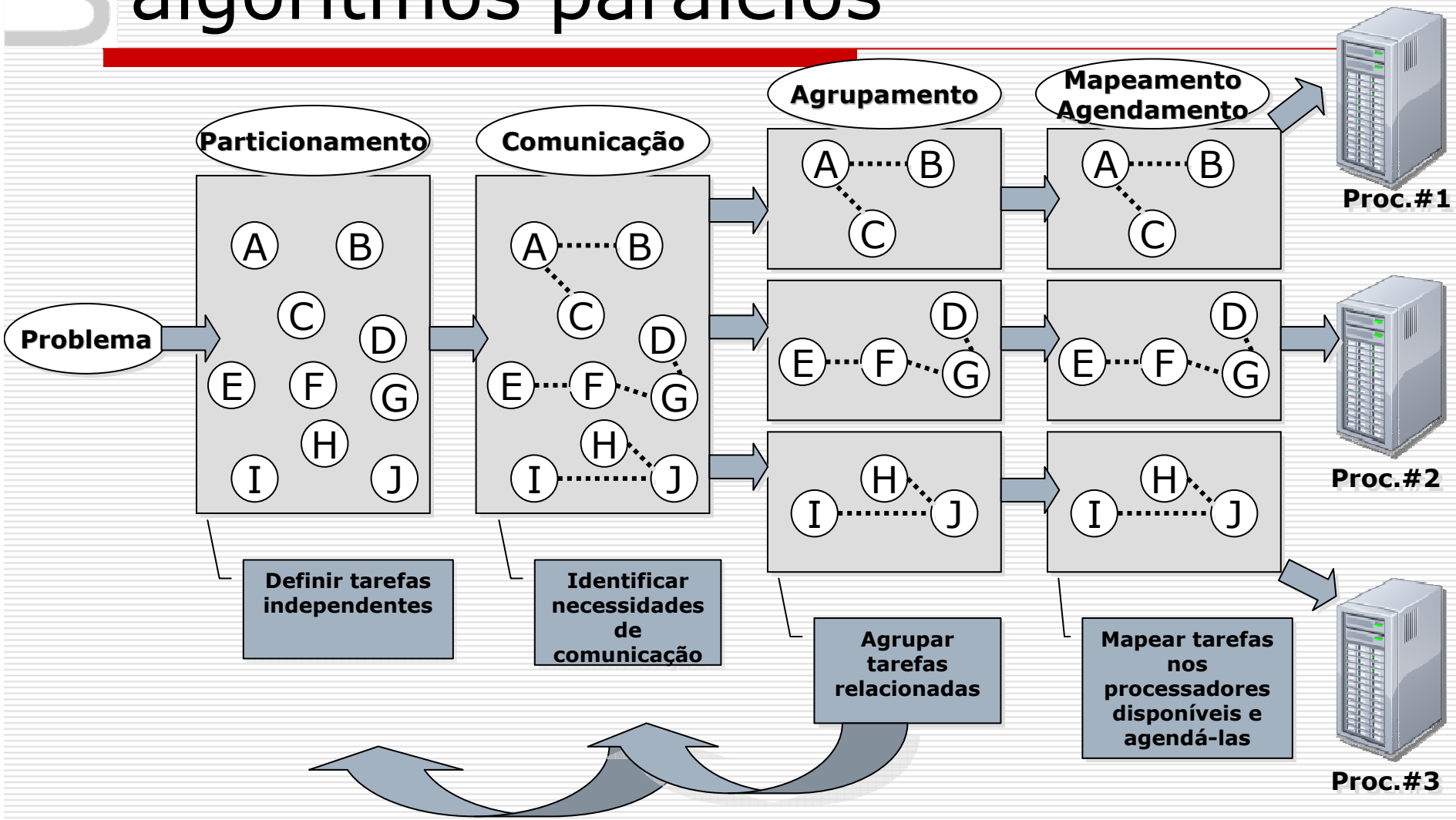
3- Agrupamento

- Avalia a granularidade das tarefas e o custo de comunicação entre elas
- Deve levar em conta o hardware disponível e, se necessário, voltar `as duas etapas anteriores
- As tarefas que tiverem maior dependência de dados ou maior correlação entre si devem ser agrupadas

4- Mapeamento e Agendamento

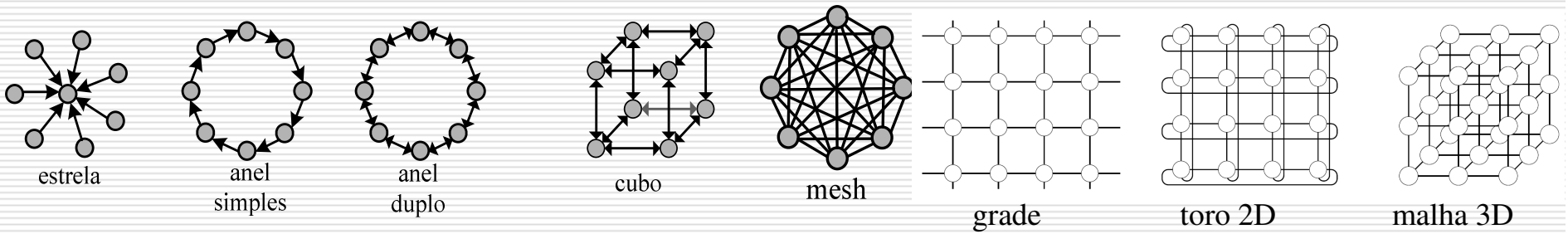
- Especifica onde e quando cada (grupo de) tarefas deverá ser fisicamente executado
- Deve ser maximizada a utilização dos processadores e minimizada a comunicação entre eles
- Usualmente é feita estaticamente, embora seja desejável ser feita dinamicamente

Metodologia para projeto de algoritmos paralelos



Topologias e sincronismo

Topologias de interligação de processos



Sincronismo:

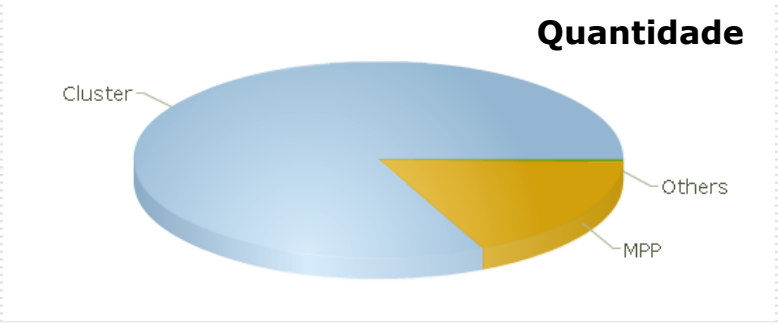
- Síncrono: alterna passos de processamento e comunicação em ciclos. Desvantagem: overflow na comunicação. Vantagem: previsibilidade
- Assíncrono: os passos de processamento e comunicação são variáveis para os processadores. Vantagem/desvantagem: pode aproveitar ou desperdiçar tempo

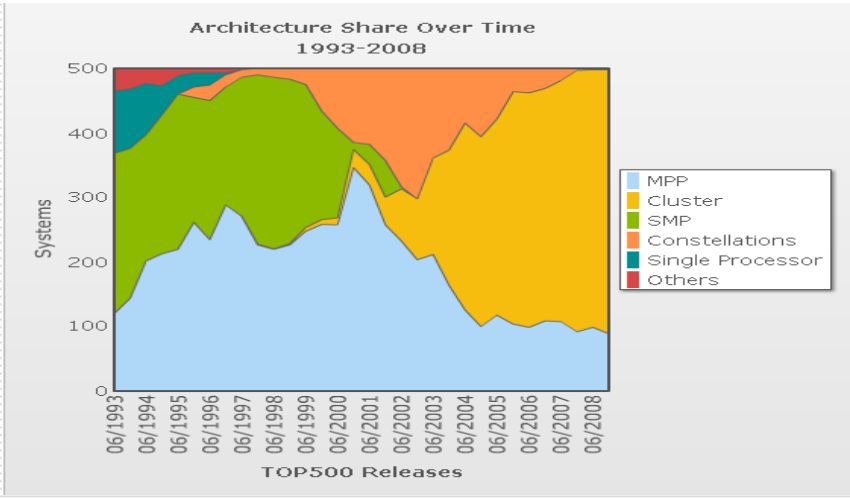
Exemplos negativos de paralelização de um algoritmo para somar $1+2+...10^6$ em 4 proc.

1) Balanço de carga impróprio: P1 faz tudo, P1..P3 parados. Nenhum ganho.
2) Comunicação excessiva: P0 tem todos os dados, divide em 4 partes e distribui, cada um processa ¼, P0 recebe resultados parciais e conclui. Grande custo para transferir massa de dados para P1..P3 e piora com mais processadores
3) Gargalo sequencial: Dados estão distribuídos em P0..P3. Balanço de carga ideal, mínima comunicação. No entanto, suponha que os dados devam ser somados na ordem original (como na série de Fibonacci). O algoritmo não é paralelizável. É importante conhecer o comprimento do caminho mais longo do gráfico dataflow. Se for excessivo, mudar o algoritmo.

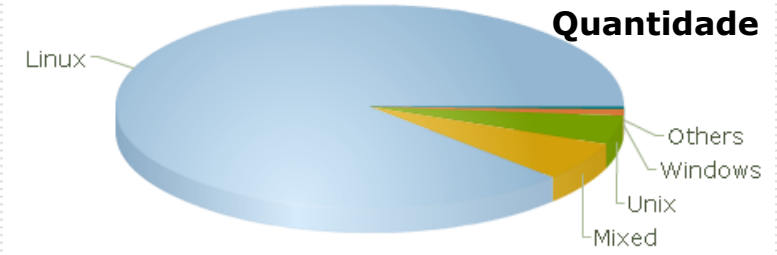
Tendências dos supercomputadores (www.top500.org)

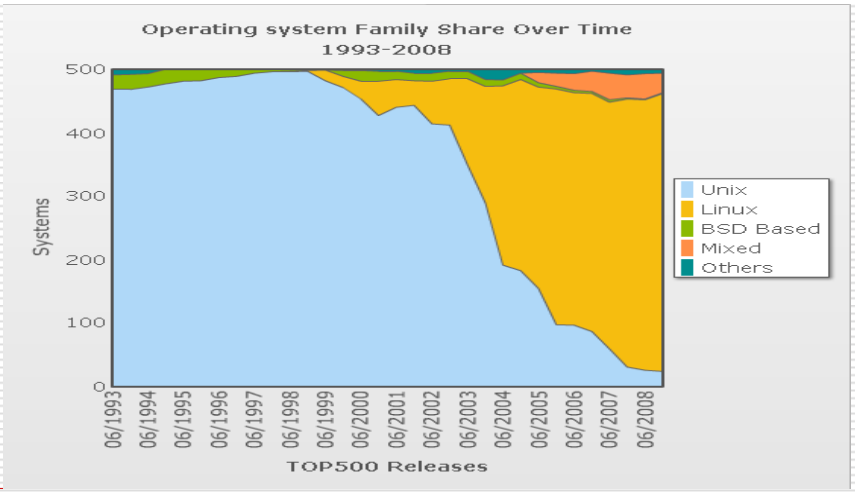
- Tipo de arquitetura





- Sistema operacional





Alguns clusters de última geração

- APPRO GreenBlade: 10 x 2 Xeon (6 core), 64 GbRAM

- Cluster: até 80 GreenBlades = 960 cores

- NVIDIA: 2 Tesla GPU (240 proc.)+2 Xeon 5500 (4 core), 4 GbRAM, U$ 8300

- Até 304 CPU/18240 GPU