



Programação Paralela

Centro Universitário de Araraquara
Prof. MSc. Rodrigo D. Malara



Agenda


- Computação Paralela
- Arquiteturas Paralelas
- Algoritmos Paralelos
- Modelos comuns de programação
 - Passagem de Mensagens
 - Espaço de Endereçamento Compartilhado



Introdução a Computação Paralela


■ Lei de Moore

- O no. de transistores que podem ser colocados em um circuito integrado vão dobrar a cada 18 meses.
- Profecia
 - Objetivo do arquiteto de computadores
 - Premissa para o desenvolvedor de software



Introdução à Computação Paralela


- Impedimentos a Lei de Moore
 - Limite Teórico
 - O que fazer com todo espaço no CI?
 - Complexidade de Projeto
 - Como atingir o aumento de performance?



Introdução à Computação Paralela


■ Paralelismo

- Continuar a aumentar a performance através de paralelismo.




Introdução à Computação Paralela

- De um ponto de vista de software, necessário para resolver alguns problemas
 - Simulações na área da Engenharia
 - Aplicações científicas
 - Aplicações de uso comerciais
- Precisam da performance que a computação paralela oferece



Introdução à Computação Paralela


- Exemplos de Aplicações na Engenharia
 - Aerodinâmica
 - Eficiência de Motores



Introdução à Computação Paralela

■ Aplicações Científicas


- ☐ Bioinformática
- ☐ Processos termonucleares
- ☐ Modelagem climática



Introdução à Computação Paralela


■ Aplicações Comerciais

- ☐ Processamento de transações financeiras
 - Criptomoedas
- ☐ Data mining
- ☐ Indexação da Internet



Introdução à Computação Paralela

- Infelizmente aumenta enormemente a complexidade do código
 - Coordenar tarefas concorrentes
 - Paralelizar algoritmos
 - Falta de ambientes padronizados e suporte



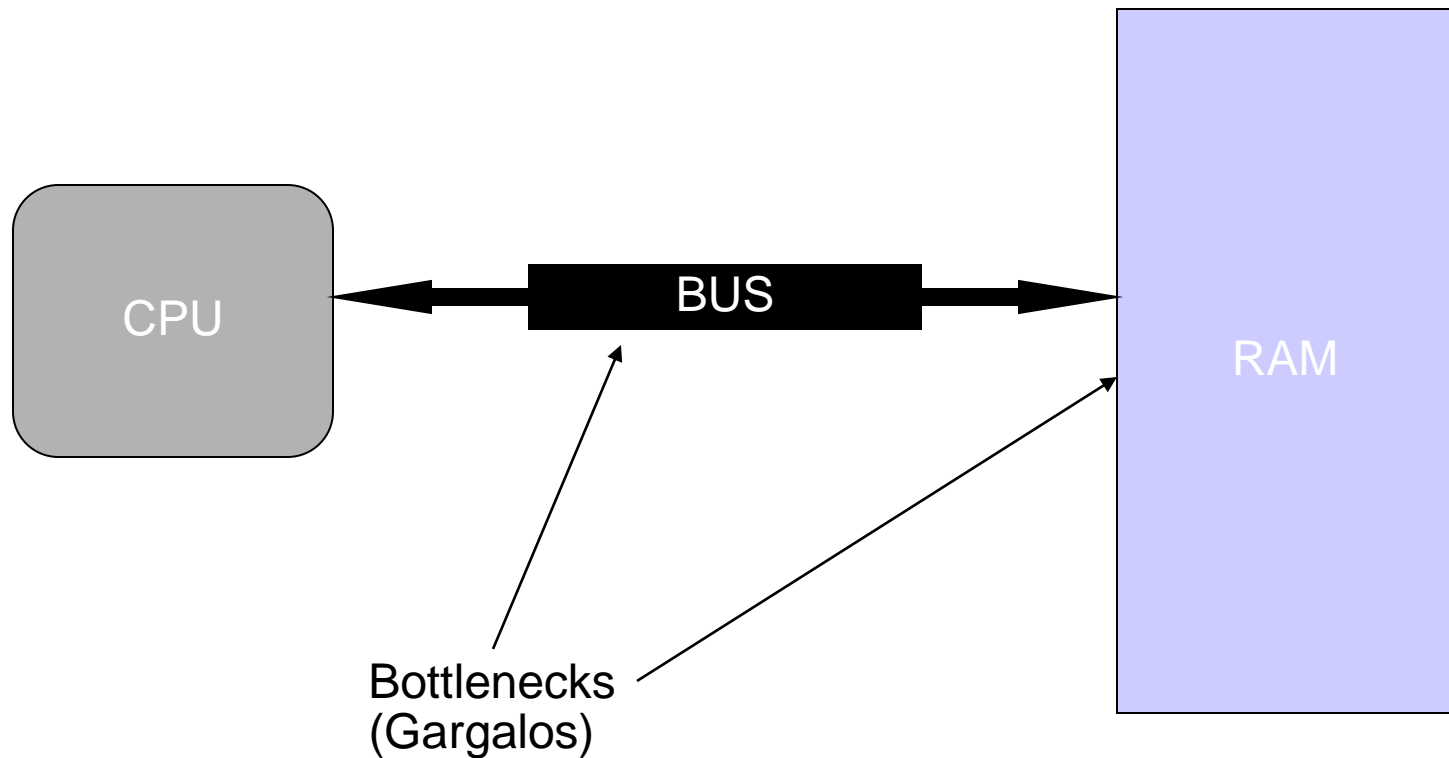
Introdução to Paralela Computação

■ O desafio

- Disponibilizar as abstrações, paradigmas de programação e algoritmos necessários para projetar, implementar e manter efetivamente as aplicações que explorem o paralelismo disponibilizado pelo hardware de forma a resolver problemas modernos.

Arquiteturas Paralelas

- Arquitetura padrão sequencial





Arquiteturas Paralelas

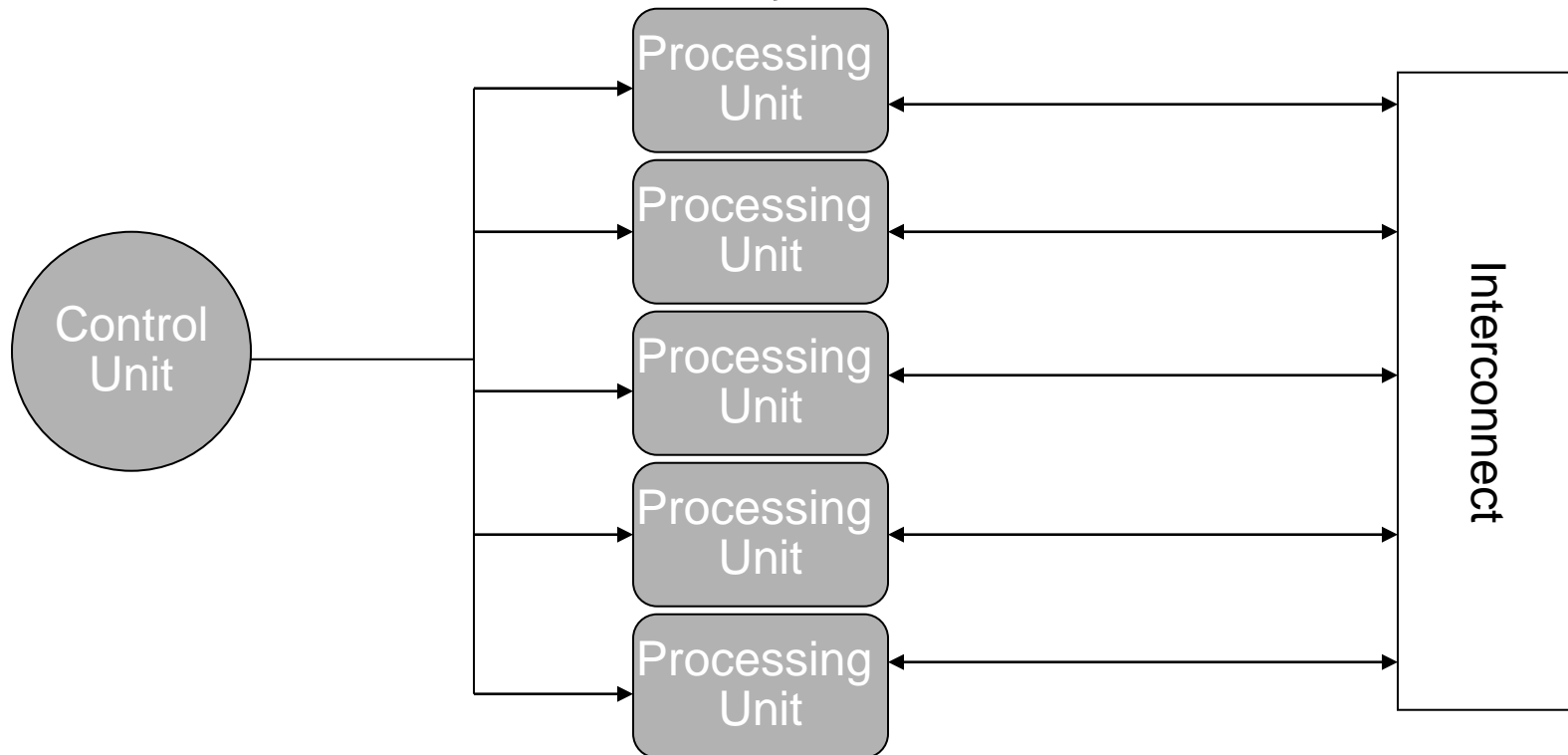
- Usar múltiplos
 - Caminhos para dados
 - Unidades de memória
 - Unidades de processamento

Arquiteturas Paralelas

■ SIMD

□ Single instruction stream, multiple data stream

- Fluxo simples de instruções, fluxos múltiplos de dados





Arquiteturas Paralelas

■ SIMD

□ Vantagens

- Realiza bem operações com vetores/matrizes
 - EX: instruções Intel MMX
 - CUDA (nVidia)

□ Desvantagens

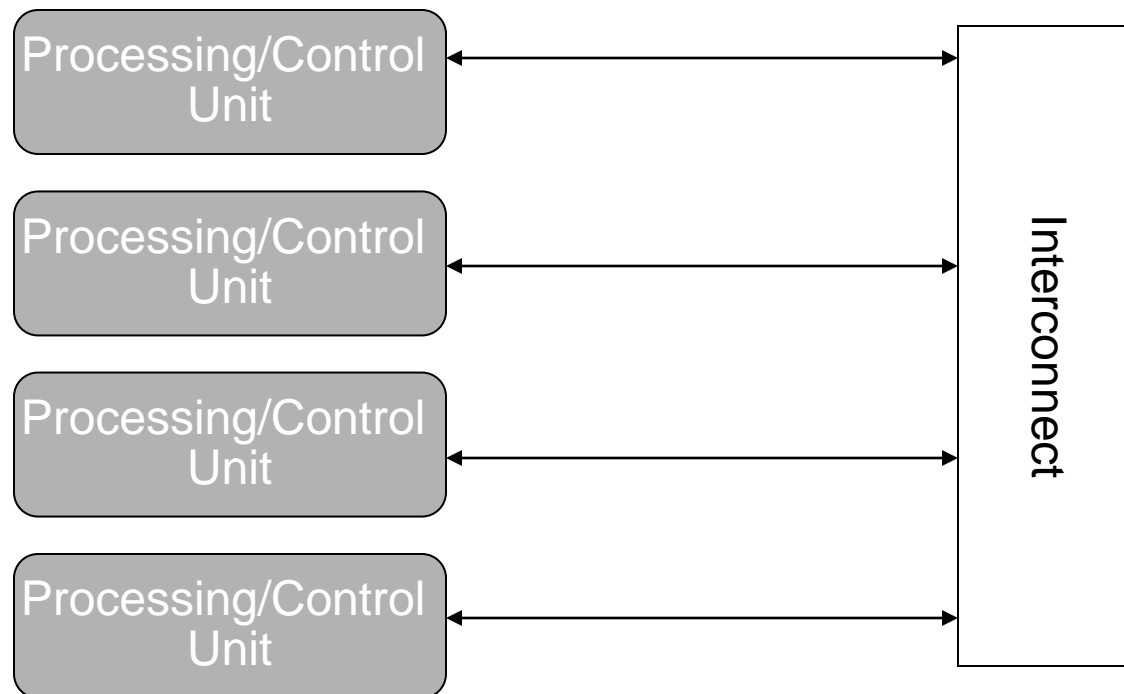
- Muito dependente do tipo de computação
 - EX: Computação Gráfica
- Performance e utilização de recursos não permitem ganho de desempenho se a computação não for “paralela”.

Arquiteturas Paralelas

■ MIMD

□ Multiple instruction stream, multiple data stream

■ Múltiplo fluxo de instruções e múltiplo fluxo de dados





Arquiteturas Paralelas

■ MIMD

□ Vantagens

- Pode ser construída de componentes comerciais
- Mais apropriada para acesso a dados irregulares

□ Desvantagens

- Requer mais hardware (!compartilha unidade de controle)
- Cada processador executa um programa diferente

■ Ex: Computadores pessoais



Arquiteturas Paralelas

■ Comunicação entre Tarefas

□ Memória compartilhada

- Usar memória comum para trocar dados
- Comunicação e replicação estão implícitas

□ Troca de mensagens

- Usar primitivas `send()/receive()` para trocar dados
- Comunicação e replicação são explícitas
 - Trabalho extra para o desenvolvedor



Arquiteturas Paralelas

■ Troca de mensagens

- Cada unidade de processamento tem sua memória privada
- Passagem de mensagens para trocar dados
- APIs
 - Message Passing Interface (MPI)
 - Parallel Virtual Machine (PVM)



Algoritmos Paralelos

■ Algoritmo

- Sequencia de instruções finitas, geralmente usadas para processamento de dados ou realização de cálculos.

■ Algoritmo Paralelo

- Um algoritmo que pode ser executado, uma parte por vez em vários e diferentes processadores e então reunidos novamente no final, de forma a gerar um resultado correto



Algoritmos Paralelos

■ Desafios

- ☐ Identificar o trabalho que possa ser executado paralelamente.
- ☐ Mapear o trabalho aos processadores.
- ☐ Distribuir o trabalho
- ☐ Gerenciar o acesso a dados compartilhados
- ☐ Sincronizar os vários estágios da computação.



Algoritmos Paralelos

■ Performance

- Geralmente compara-se com a versão sequencial e paralela

■ Speedup (ou Ganho)

- S_p : Speedup em p processadores
- T_1 : Tempo sequencial
- T_p : Tempo em p processadores

$$S_p = \frac{T_1}{T_p}$$

Algoritmos Paralelos

■ Lei de Ahmdal

- Define o quanto uma melhoria em determinada parte de um sistema pode gerar um ganho de performance
- Determinar o speedup máximo teórico

■ Exemplo

- Tempo sequencial de um programa: 480 minutos
 - Tempo para inicialização e envio das tarefas: 15 m
 - Tempo para processamento **paralelizável**: 430 m
 - Tempo para receber resultados, consolidar e terminar: 15 m
- Tempo mínimo teórico do trecho paralelizável em 12p = $430/12 = 35,83$ m
- Tempo mínimo teórico geral = $15m + 35,83m + 15m = 65,83m$
- $S_{12-t} = 480m / 65,83m = 7,3$ x mais rápido em 12 processadores



Algoritmos Paralelos

■ Tipos de Problemas

□ Granularidade fina

■ Relação Computação/Comunicação baixa

- Difícil ganho de performance em troca de mensagens porém pode ser viável para memória compartilhada

□ Granularidade Grossa

■ Relação Computação/Comunicação alta

■ Cenário ideal

□ Embarçosamente Paralelos

■ Assemelha-se à Granularidade Grossa

■ Pouca comunicação entre os nós

■ Pouco esforço para se codificar



Algoritmos Paralelos

■ Modelos

- Uma maneira de estruturar um algoritmo paralelo, selecionando técnicas de decomposição e mapeamento para minimizar interações.



Algoritmos Paralelos

■ Modelos

- ☐ Paralelismo por dados
- ☐ Grafo de tarefas
- ☐ Pool de Processos
- ☐ Mestre-escravo
- ☐ Pipeline
- ☐ Híbrido



Algoritmos Paralelos

- Paralelismo por Dados (Data-parallel)
 - Mapeamento de Trabalho
 - Estático -> Durante codificação
 - Tarefas-> Representadas por Processos ou Threads
 - Mapeamento dos Dados
 - Itens de dados independentes atribuídos aos processos

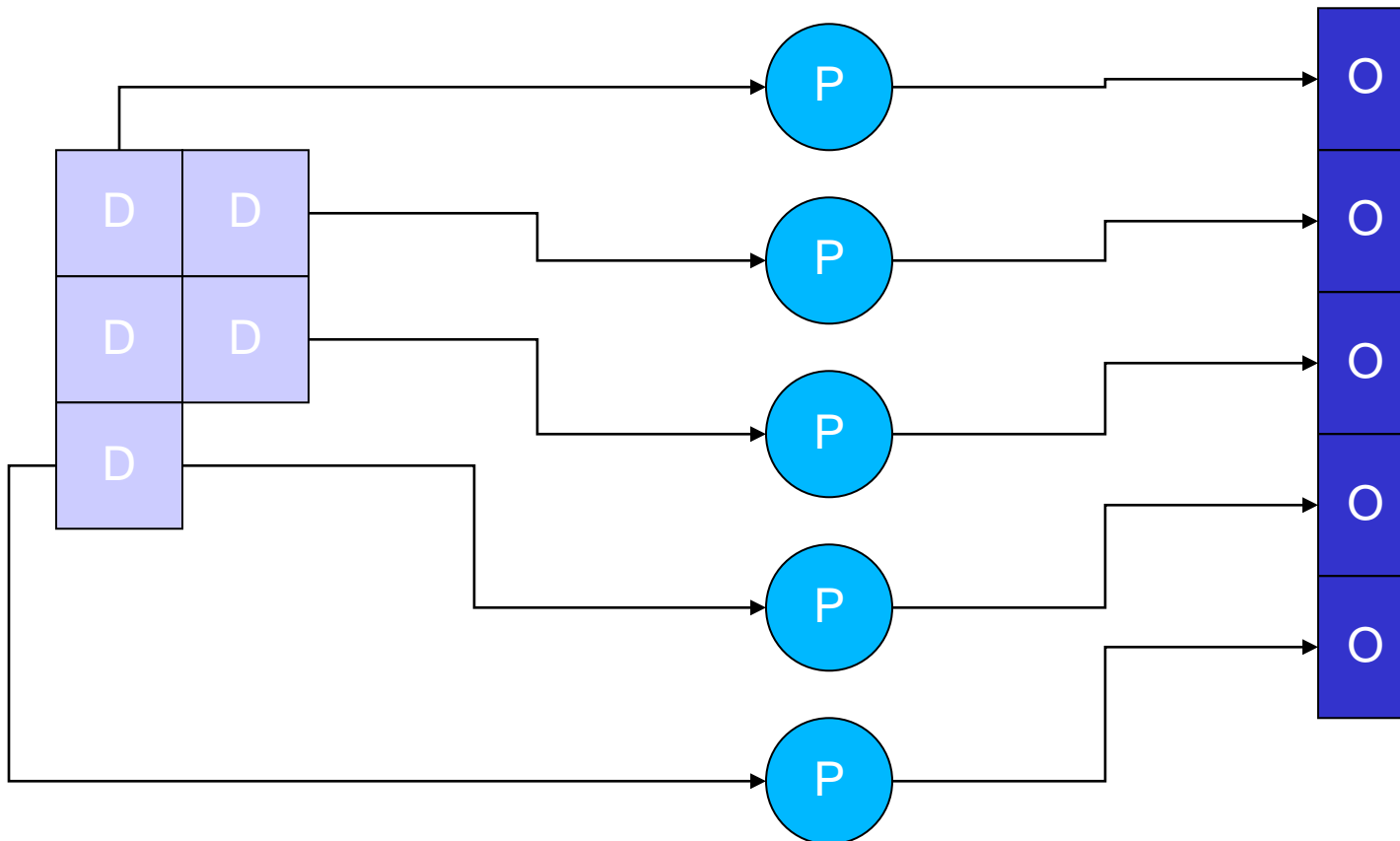


Algoritmos Paralelos

- Paralelismo por Dados (Data-parallel)
 - Computação
 - Tarefas processam os dados, sincronizam para pegar novos dados ou para devolver resultados e continuam até que esteja tudo processado
 - Balanceamento de Carga
 - Depende da uniformidade na distribuição dos dados
 - Sincronização
 - Mínima e pode necessitar de uma barreira no final
 - Ex: Processamento de imagens, simulações em semi-condutores, games e multimídia

Algoritmos Paralelos

■ Paralelismo por Datos (Data-parallel)





Algoritmos Paralelos

■ Pool de Processos

□ Mapeamento de Trabalho/Dados

- Mapeamento à priori é desnecessário
- Qualquer tarefa é realizada por qualquer processo

□ Computação

- Processos trabalham assim que os dados são disponibilizados (ou chegue uma requisição)



Algoritmos Paralelos

■ Pool de Processos

□ Balanceamento de Carga

- Mapeamento de tarefas a processos é dinâmico

□ Sincronização

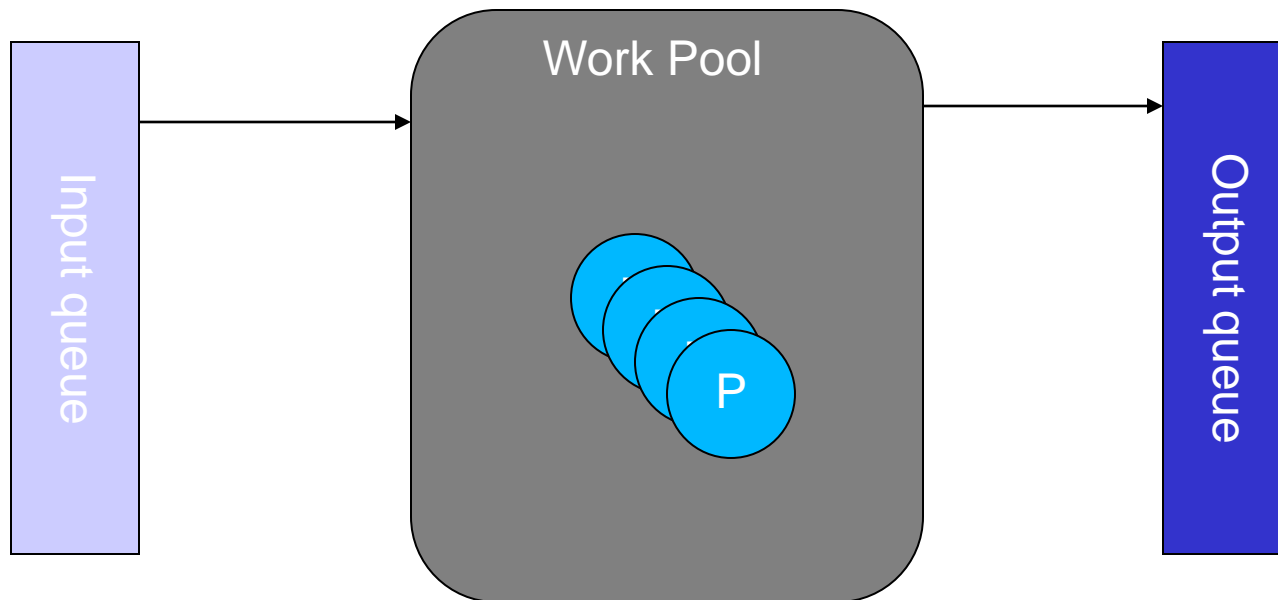
- Adicionar ou remover trabalho de uma fila

□ Ex: servidores WEB e de Bancos de Dados

- Pode ser em cluster ou não

Algoritmos Paralelos

■ Pool de Processos





Algoritmos Paralelos

■ Mestre-escravo

□ Modificação no Pool de Processos

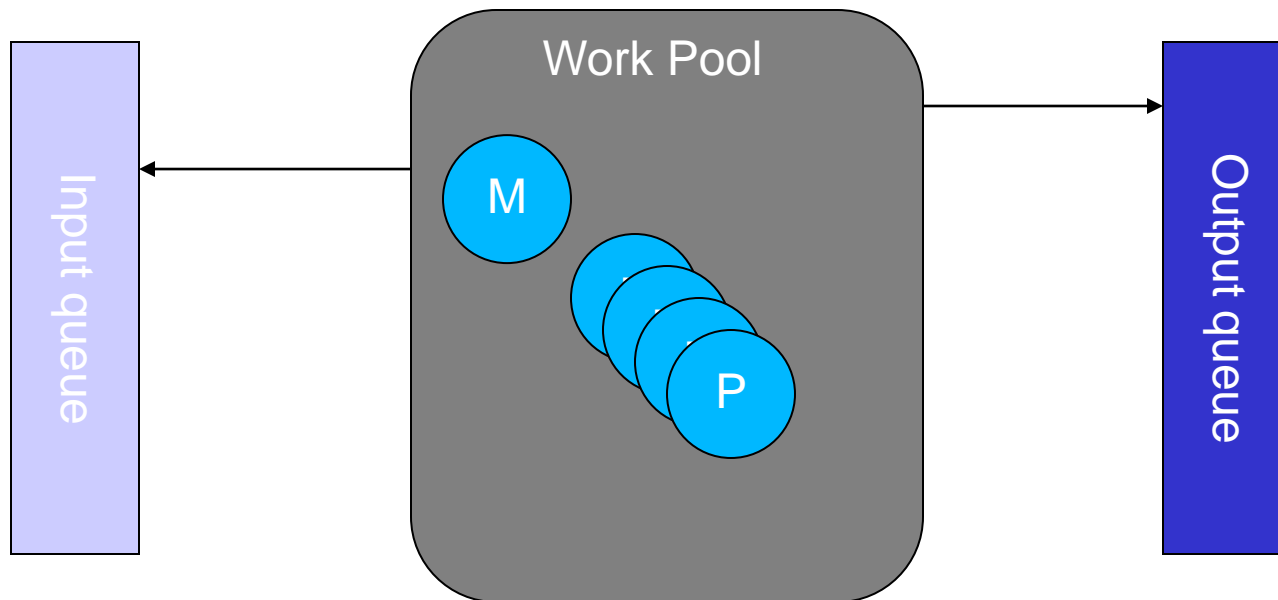
- Um ou mais processos mestre gerando e distribuindo trabalho aos processos escravos


□ Balanceamento de Carga

- Um processo mestre pode melhorar a distribuição de trabalho aos escravos, implementar alguma política baseada em alguma métrica

Algoritmos Paralelos


■ Mestre-Escravo





Modelos comuns de Programação Paralela


- Troca de mensagens
- Memória compartilhada



Modelos comuns de Programação Paralela

■ Troca de mensagens

- Mais usado para programar computadores paralelos (clusters)
- Atributos chave:
 - Memória particionada
 - Paralelização explícita
- Interação entre processos
 - Envio e recebimento de dados



Modelos comuns de Programação Paralela

■ Comunicação

□ Primitivas

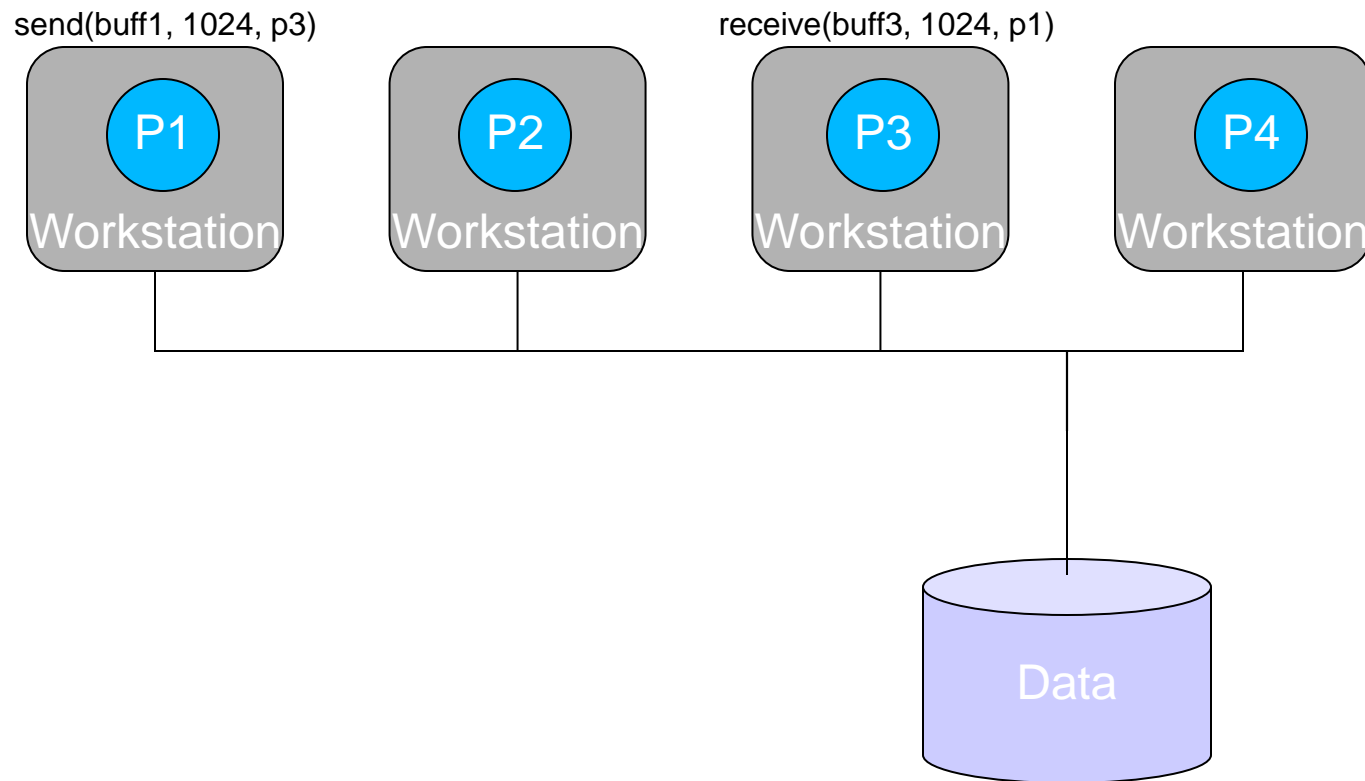
- `send(buff, size, destination)`
- `receive(buff, size, source)`
- Bloqueante vs não-bloqueante
- Buferizada vs não-buferizada


■ Message Passing Interface (MPI)

- API popular para passage de mensagens
 - Disponível para Linguagem C e Fortran
- ~125 funções

Modelos comuns de Programação Paralela

■ Troca de mensagens






Modelos comuns de Programação Paralela

■ Memória Compartilhada

- Programação mais utilizada em máquinas SMP (multicore chips)
- Atributos chave
 - Memória Compartilhada
 - Threads
 - Shmget/shmat UNIX operations
 - Paralelização implícita
- Comunicação entre Processos/Threads
 - Leitura/Escrita em memória



Modelos comuns de Programação Paralela

■ Memória Compartilhada

□ Comunicação

■ Leitura/Escrita em memória

□ EX: c++;

□ API Posix Threads

■ API popular

■ Operações

□ Criação/remoção de threads

□ Sincronização (mutexes, semaphores)

□ Gerenciamento de Threads

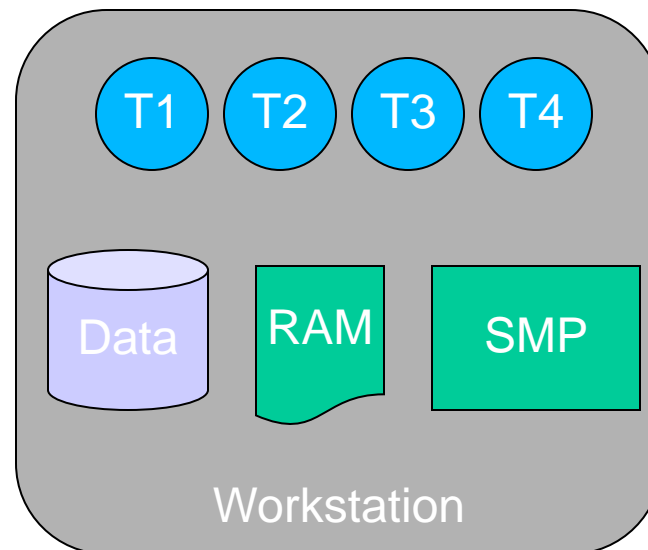
□ OpenMP

■ Disponível para C e Fortran

■ Paralelismo por dados orientado por diretivas

Modelos comuns de Programação Paralela

■ Memória Compartilhada





Desafios da Programação Paralela

- Sincronização
 - Deadlock ou impasses
 - Justiça
- Eficiência
 - Maximizar paralelismo
- Confiabilidade
 - Corretitude
 - Depuração



Referências

- Introduction to Parallel Computing, Grama et al., Pearson Education, 2003
- Distributed Systems
 - <http://code.google.com/edu/parallel/index.html>
- Distributed Computação Principles and Applications, M.L. Liu, Pearson Education 2004
- Designing and Building Parallel Programs, Foster, I. <http://www.mcs.anl.gov/~itf/dbpp>