



Programação Paralela

Aula 1



Objetivos

- Compreender o paradigma paralelo
- Apresentar diferentes técnicas de programação paralela
- Desenvolver soluções paralelas utilizando modelos diferentes
- Calcular o Speed Up e a eficiência da solução proposta



Conteúdo Programático

- Apresentar o paradigma de desenvolvimento paralelo
- Apresentar a Arquitetura Paralela
- Modelos de Programação Paralela
- Métricas de avaliação de solução paralela
- Metodologia para o desenvolvimento de soluções paralelas
- Threads em JAVA e Python
- OPENMP
- MPI
- Programação em CUDA



Bibliografia

- TANENBAUM, Andrew S. Distributed operating systems. New Jersey: PrenticeHall, c1995. 614 p
- COULOURIS, G., DOLLIMORE, J. & KINDBERG, T. Distributed Systems: Concepts and Design. 4ª edição. Addison Wesley. 2005.
- Na Introduction to Parallel Programming – Peter Pacheco - 2010



Avaliação

- 2 Provas
 - 6 pontos
- 3 ou 4 trabalhos
 - 4 pontos



Histórico

- No primórdio da computação não tinha-se hardware e nem arquitetura para dar suporte à programação Paralela e Distribuída
- O campo da programação concorrente iniciou-se uma explosiva expansão a partir de 1968
- Ambos os campos surgiram a partir do campo da programação concorrente



Histórico

- Já a Programação Paralela
 - Conhecida como uma programação concorrente
 - Porém com mais linhas de execução
 - Onde os processos são divididos em sub-processos, conhecidos como Threads
 - Na qual são executados paralelamente com o processo pai
 - Já na distribuída, o sistema é executado em vários ambientes interligados por uma rede de comunicação

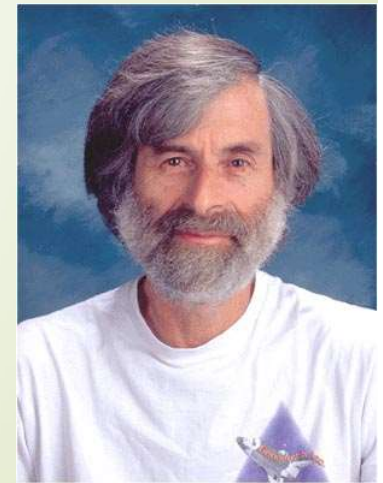
Histórico

- **1968** *E. W. Dijkstra*: Cooperando Processos Seqüenciais. Semáforos
- **1971** *E. W. Dijkstra*: Ordem hierárquica de processos sequenciais. Prioridade
- **1973** *C. L. Liu e J. W. Layland* : Algoritmos de escalonamento para multiprogramação em ambiente de tempo real.



Histórico

- 1974 C. A. R. Hoare: Monitores - conceito para estruturar sistemas operacionais.
- 1974 Lamport: Uma nova solução para o problema da programação concorrente de Dijkstra. **Relógios de Lamport.**



Leslie Lamport

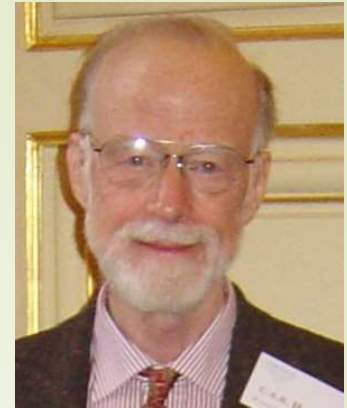
Histórico

- **1976** *J. H. Howard*: Provando monitores.
- **1976** *S. Owicki e D. Gries*: Verificando propriedades de programas paralelos: uma abordagem axiomática.
- **1977** *P. Brinch Hansen*: A arquitetura de programas concorrentes.



P. Brinch Hansen

Histórico



C. A. R. Hoare

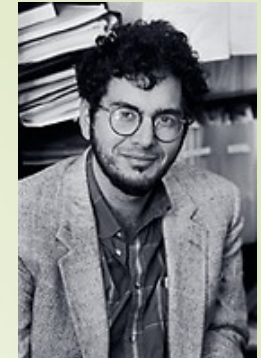
- **1978** *C. A. R. Hoare*: Comunicação de Processos Sequenciais. Um modelo usado para especificar interações entre processos concorrentes.
- **1978** *E. W. Dijkstra, L. Lamport, A. J. Martin, C. S. Sholten e E. F. M. Steffens*: Um exercício em cooperação para “garbage collection”.
- **1980** *E. W. Dijkstra e C. S. Sholten*: Detecção de terminação.



Histórico

- **1981** *G. Ricart e A. Agrawala*: Um algoritmo ótimo pra exclusão mútua distribuída.
- **1981** *G. L. Peterson*: O problema da exclusão mútua. Um algoritmo de programação concorrente para exclusão mútua.
- **1982** *J. Misra e K. M. Chandy*: Detecção de terminação em *Communicating Sequential Processes*.

Histórico



David Gelernter

- **1983** G. L. Peterson: Uma nova solução para o problema de programação concorrente de Lamport usando variáveis compartilhadas para n processos
- **1985** D. Gelernter: A Linguagem Linda
 - Considerada uma linguagem para programação paralela. Primeiro produto comercial a implementar o conceito de memória virtual compartilhada




Histórico

- De 1986 – 2002, os microprocessadores aumentaram o seu desempenho cerca de 50% por ano
- A partir de 2002 o aumento foi de 20%




Uma solução inteligente

- Como melhorar o desempenho
- Ao invés de projetar e construir microprocessadores mais rápidos (frequência de clocks mais elevados)
- Era mais viável colocar vários processadores em um único circuito integrado
 - Custo
 - Limitações físicas
 - Aquecimento



Agora, cabe aos programadores

- Adicionando mais processadores não ajuda muito se os programadores não estão cientes deles ou não sabe como utilizá-los
- Programas seriais não são beneficiados por esta abordagem




Por que precisamos de desempenho cada vez maior

- Demanda computacional cresce ilimitadamente
- Ela sempre está acima da disponibilidade de recursos computacionais existentes
- O desenvolvimento científico exige muito computação de alto desempenho
 - simulações numéricas, armazenamento de dados e visualização de informações
 - Decodificação genoma humano
 - Modelagem climática
 - Dobramento de proteínas
 - Novas drogas
 - Pesquisas em energia
 - Análise de dados



Por que Construimos Sistemas Paralelos

- Até agora , os aumentos de desempenho foram atribuídos ao aumento da densidade de transistores
- Mas há problemas inerentes



Uma pequena lição de física

- Transistores menores = processadores mais rápidos
- Processadores mais rápidos = aumento no consumo de energia
- Aumento no consumo de energia = aumento de calor
- Aumento de calor = processadores não confiáveis



Solução

- Em vez de criar um processador muito rápido
- Crie vários núcleos de processadores mais lentos
- Desta forma, precisamos da programação paralela



Por que precisamos escrever programas paralelos

- ▶ Executar várias instâncias de um programa serial, muitas vezes não é útil
- ▶ Pense executando várias instâncias do seu jogo favorito.
- ▶ O que você realmente quer é para que ele seja executado mais rapidamente.



Abordagens para o problema serial

- Reescrever programas seriais de modo que eles se tornem paralelos
- Escrever programas que convertam automaticamente programas seriais para paralelos



Como escrever programas em paralelo

- ▶ Paralelismo de tarefas
 - ▶ Particionar as várias tarefas a serem executadas
 - ▶ Resolvendo o problemas entre os núcleos
- ▶ Paralelismo de dados
 - ▶ Particionar os dados utilizados na resolução do problema entre os núcleos
 - ▶ Cada núcleo realiza operações semelhantes sobre a sua parte dos dados



Coordenação

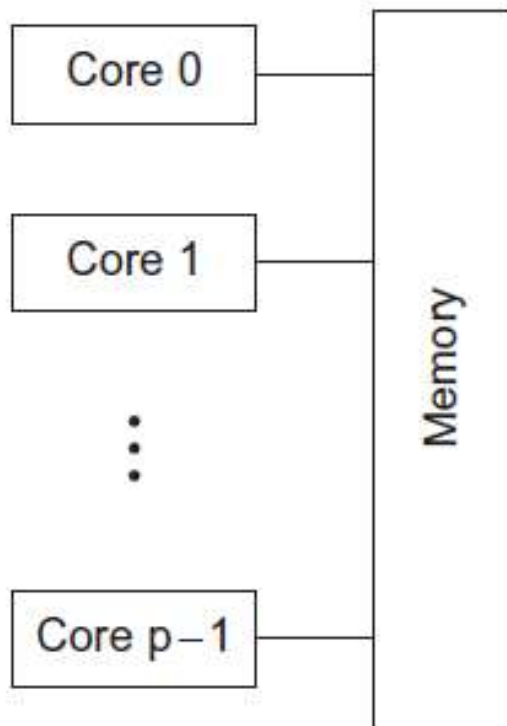
- Os núcleos precisam coordenar o seu trabalho
- Comunicação
 - Um ou mais núcleos enviam seus resultados parciais para outros
- Balanceamento de carga
 - compartilhar o trabalho uniformemente entre os núcleos de modo que um não fique muito carregado
- Sincronização
 - cada núcleo trabalha em seu próprio ritmo , certifique-se que eles não fiquem muito à frente do resto.



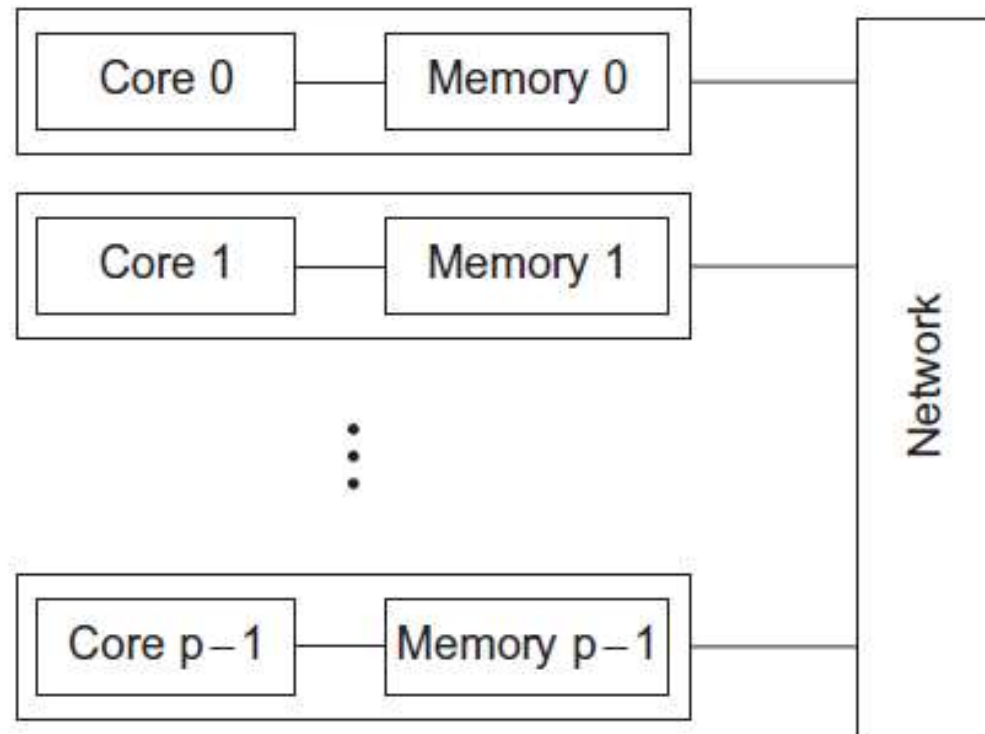
O que vamos fazer?

- Aprender a escrever programas que são explicitamente paralelos
- Usando a linguagem de programação JAVA Python e C
 - Threads
 - OpenMP
 - MPI


Tipos de sistemas



(a)

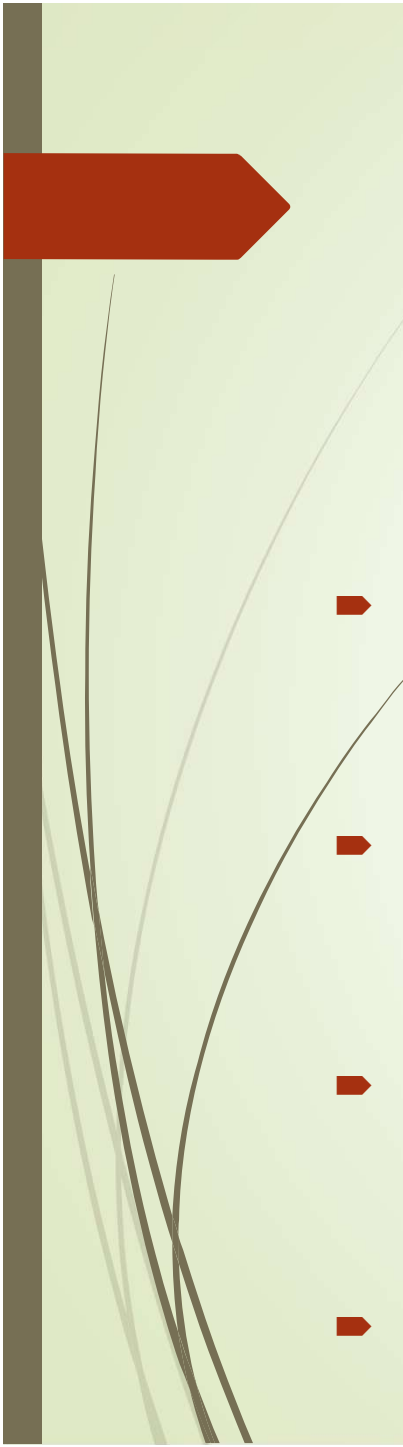


(b)



Técnicas para a Programação em Modelos de Memória Compartilhada

- Requerem uso de técnicas que possibilitem a criação de threads e que implementem mecanismos de sincronização entre elas
- As ferramentas mais comuns
 - Threading Explícito
 - Diretivas (normas, instruções) de Compilação
 - Troca de mensagem
 - Linguagens Paralelas



Técnicas para a Programação em Modelos de Memória Compartilhada

- Threading Explícito
 - Pthreads (POSIX Threads, win32 Threads)
- Diretivas de Compilação
 - OpenMP
- Troca de Mensagem
 - MPI (Message Process Interface)
- Linguagens Paralelas
 - HPF – High Performance Fortran



Enfim

- Computação Concorrente
 - É um programa na qual múltiplas tarefas podem estar rodando no mesmo tempo
- Computação Paralela
 - É um programa na qual múltiplas tarefas cooperam rigorosamente para resolver um problema
- Computação Distribuída
 - Um programa pode ter que cooperar com outro para resolver um problema



Observações

- Os limites físicos nos fizeram a ir em direção da abordagem da tecnologia multicore
- Programas seriais não são beneficiados pela tecnologia multicore
- Geração de programas paralelos a partir de programas seriais feito de forma automática, não é uma abordagem eficiente para conseguir alto desempenho de computadores multicores



Observações

- Aprender a escrever programas paralelos envolve aprender como coordenar os cores
- Programas paralelos geralmente são complexos, portanto, necessitam de técnicas para o seu desenvolvimento
- Programas paralelos utilizam a arquitetura de memória compartilhada