SSC-0143
PROGRAMAÇÃO CONCORRENTE

Aula 06 – Modelos de Programação

Prof. Jó Ueyama e Julio Cezar Estrella

Créditos

Os slides integrantes deste material foram construídos a partir dos conteúdos relacionados às referências bibliográficas descritas neste documento

Visão Geral da Aula de Hoje

1. Overview
2. Modelo de Memória Compartilhada
3. Modelo de Threads
4. Modelo de Passagem de Mensagens
5. Modelo Paralelo de Dados
6. Modelo Híbrido
7. Exercício e Leitura Recomendada

Overview

Modelos de Programação

- Há vários modelos de programação paralela em uso atualmente:
  - Memória compartilhada
  - Threads
  - Passagem de Mensagens
  - Dados paralelos
  - Híbrido

  Modelos de programação paralela existem como uma abstração acima da arquitetura de hardware e de memória

Modelos de Programação

- Apesar de não ser evidente, os modelos não são específicos para um determinado tipo de arquitetura
- Qualquer modelo pode (teoricamente) ser implementado em qualquer hardware
- Exemplos

Modelos de Programação

- Modelo de memória compartilhada em uma máquina de memória distribuída: Kendall Square Research (KSR)
  - A memória é fisicamente distribuída, mas aparece para o usuário como uma memória única (espaço de endereçamento global).
    - ***Memória virtual compartilhada***

Modelos de Programação

- Modelo de Passagem de Mensagens em uma máquina de memória compartilhada
- Cada tarefa tem acesso direto à memória global
- Exemplo:
  - MPI em SGI Origin

Modelos de Programação

- A SGI Origin utiliza o tipo de arquitetura de memória compartilhada CC-NUMA
  - Cada tarefa tem acesso direto à memória global. No entanto a capacidade para enviar e receber mensagens com MPI, como é realizado em uma máquina com memória distribuída, não somente é possível, como é comumente utilizada

Modelos de Programação

- **Reflexões**
  - Qual modelo utilizar é em geral decidido:
    - Com base no que há disponível
    - Escolha pessoal
- Não existe o melhor modelo. No entanto há melhores implementações de alguns modelos sobre outros
  - Vamos descrever os modelos e discutir algumas de suas implementações reais

MODELO DE MEMÓRIA COMPARTILHADA

Modelo de Memória Compartilhada

- Neste modelo as tarefas compartilham um espaço de endereço comum, onde elas podem ler e escrever simultaneamente
- Mecanismos como semáforos podem ser utilizados para controlar o acesso à memória compartilhada
- Vantagem → Para o programador
  - Não existe a necessidade de comunicação explicita de dados entre as tarefas

Modelo de Memória Compartilhada

- Desvantagem
  - Dificuldade de entender e gerenciar a localidade dos dados
- Manter os dados locais ao processador que os opera minimiza os acessos à memória que ocorrem quando múltiplos processadores utilizam o mesmo dado.

Modelo de Memória Compartilhada

- **Implementações**
  - Neste tipo de organização os compiladores nativos traduzem as variáveis do programa do usuário em endereços de memória, que são globais
- Os SMPs (e.g. multicore) representam uma implementação comum de uma arquitetura de memória compartilhada
  - A KSR fornece uma visão de memória compartilhada mesmo com a memória física sendo distribuída

Modelo de Threads

Modelo de Threads

- No modelo de threads, cada processo pode ter múltiplos caminhos de execução concorrente

Modelo de Threads

- O programa principal é escalonado para ser executado pelo sistema operacional nativo e adquire todos os recursos necessários para ser executado.
- O programa principal executa sequencialmente, mas cria tarefas (threads) que podem ser escalonadas e executadas simultaneamente pelo SO
- Cada thread tem dados locais e compartilha recursos com o programa principal. Isso diminui o overhead associado com a replicação dos recursos entre vários processos. Cada thread também se beneficia de uma visão global de memória, porque ela compartilha o espaço de memória com o programa principal.

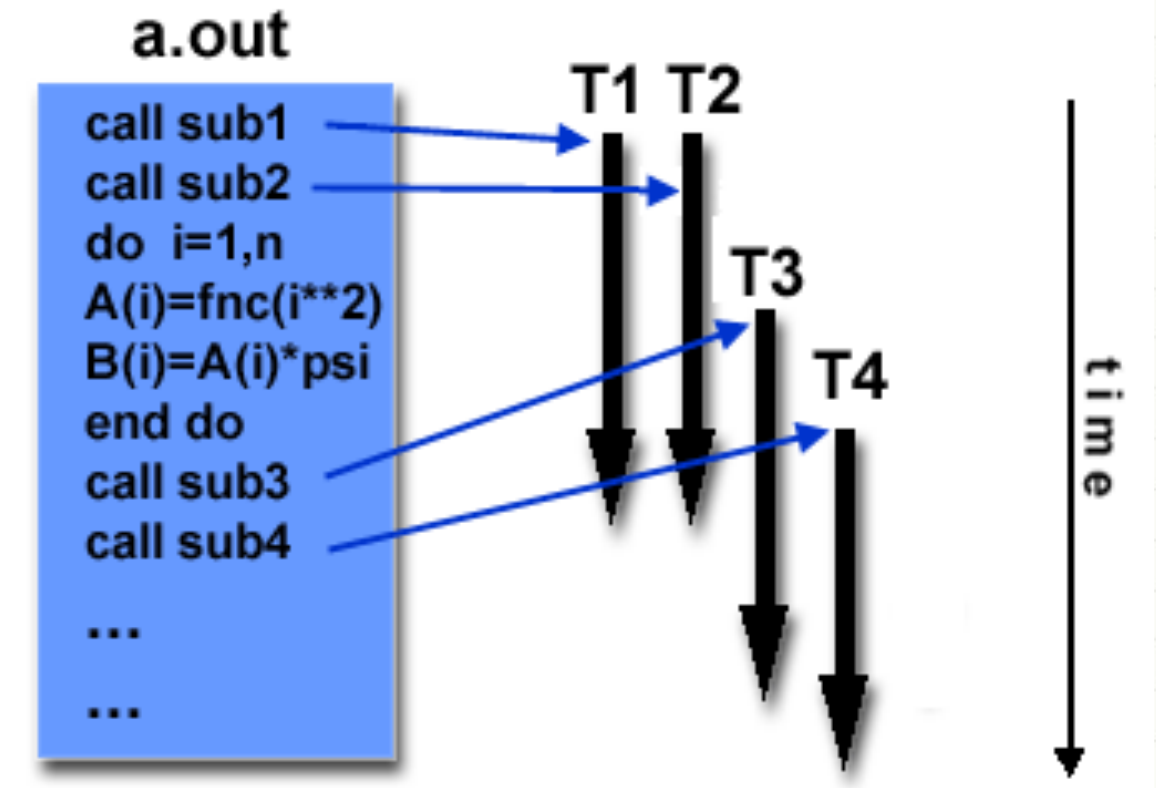
Modelo de Threads

- Threads se comunicam umas com as outras por meio da memória global (atualizando endereços locais). Isso requer sincronização para garantir a consistência dos dados durante a execução das threads.
- Threads podem ser criadas ou destruídas, mas o programa principal continua ativo para fornecer os recursos compartilhados necessários até que a tarefa seja concluída.

Modelo de Threads

- Threads são comumente associadas com arquiteturas de memória compartilhada e sistemas operacionais

Modelo de Threads



Modelo de Threads

- **Implementação inclui**:
  - Uma biblioteca de sub-rotinas que são chamadas a partir do código fonte paralelo
  - Um conjunto de diretivas de compilador no código fonte paralelo ou serial. Em ambos os casos, o programador é responsável por determinar todo o paralelismo

Modelo de Threads

- **OpenMP**
  - Padrão definido e apoiado pelos maiores fabricantes de hardware e software do mundo
  - API OpenMP Fortran → 28/10/1957
  - API OpenMP C/C++ → Final de 1998
  - Portável e multiplataforma (Unix e Windows)
  - É fácil e simples de usar → Prevê paralelismo incremental
  - A Microsoft tem sua própria implementação de threads
    - Não relacionada com o padrão Unix POSIX ou OpenMP

Modelo de Threads

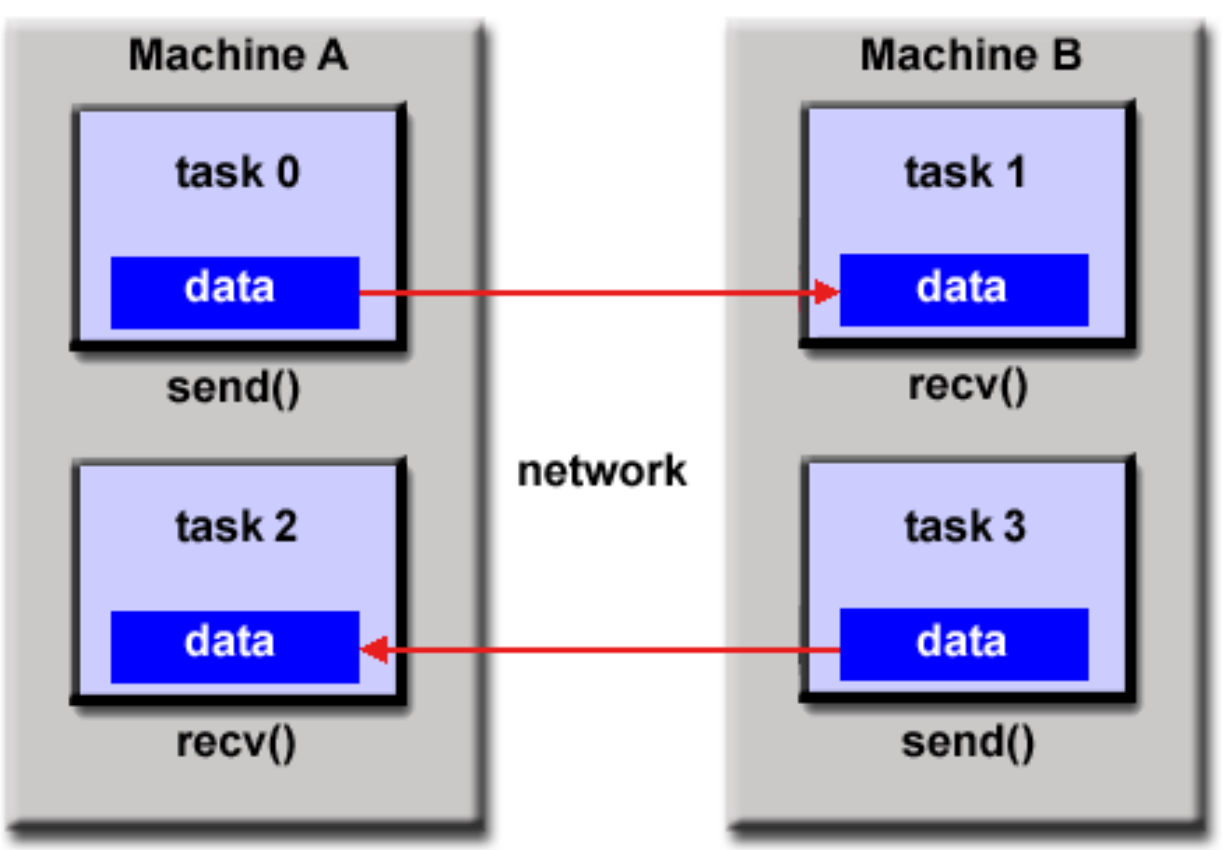
- **Posix Threads**
  - Especificado pelo padrão IEEE POSIX 1003.1c (1995).
  - Somente linguagem C
  - Conhecido como Pthreads
  - A maioria dos fabricantes de hardware já fornecem pthreads
  - Paralelismo explícito; requer muita atenção do programador para detalhes de codificação

Modelo de Passagem de Mensagens

Modelo de Passagem de Mensagens

- **Características**
  - Conjunto de tarefas que utilizam a memória local durante a computação. Várias tarefas podem residir na mesma máquina.
  - Tarefas trocam dados através de comunicação, enviando e recebendo mensagens.
  - A transferência de dados em geral requer operações de cooperação a serem executadas por cada processo. Exemplo:
    - Operação de envio deve ter uma operação de recepção correspondente

Modelo de Passagem de Mensagens



Modelo de Passagem de Mensagens

- O programador é responsável por determinar todo o Paralelismo
- Várias bibliotecas disponíveis desde a década de 1980
  - Muitas diferenças → Dificuldade de desenvolver aplicações portáteis
- 1992 → Forum MPI estabelece um padrão de interface para implementação de passagem de mensagens
- 1994 → Parte 1
- 1996 → Parte 2 (MPI-2),

- ***MPI***: Padrão da indústria para ambientes de passagem de mensagens

Modelo de Passagem de Mensagens

- Para arquiteturas de memória compartilhada, as implementações de MPI geralmente não utilizam redes para a comunicação entre as tarefas (overhead de comunicação)
- Ao invés disso é utilizada memória compartilhada (cópias de memória) por razões de desempenho

Modelo Paralelo de Dados

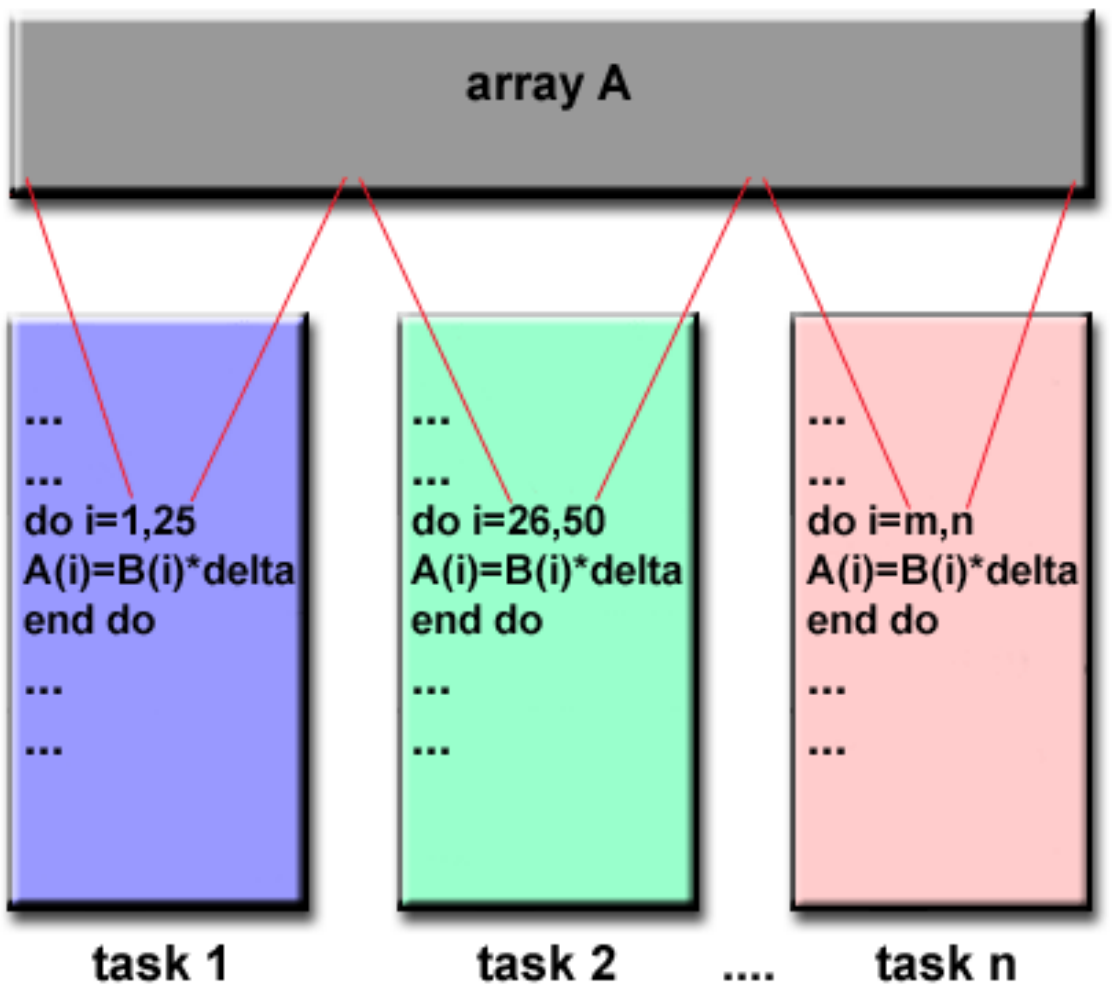
Modelo Paralelo de Dados

- **Características**
  - A maioria da tarefas paralelas envolve a execução de operações em um conjunto de dados. Os dados são em geral organizados em uma estrutura, como uma matriz ou um cubo
  - Um conjunto de tarefas trabalham coletivamente na mesma estrutura de dados, no entanto, cada tarefa opera em uma partição diferente da mesma estrutura de dados

Modelo Paralelo de Dados

- Tarefas executam a mesma operação em sua partição de trabalho
- Exemplo:
  - Adicionar 4 a cada elemento de um array
- Em arquiteturas de memória compartilhada, todas as tarefas podem ter acesso à estrutura de dados armazenada na memória global.
- Em arquiteturas de memória distribuída, a estrutura de dados é dividida e distribuída na memória local de cada processador

Modelo Paralelo de Dados



Modelo Paralelo de Dados

- **Implementação**
  - Escreve-se um programa definindo a organização paralela dos dados
  - A organização pode ser definida por chamadas a uma biblioteca de dados paralelos, ou por diretivas do compilador

Modelo Paralelo de Dados

- **Implementação**
  - Fortran 90 e 95: extensões ISSO/ANSI padrão para Fortran 77
  - Contém tudo o que está no Fortran 77
  - Novo formato de código-fonte; adições ao conjunto de caracteres
  - Adições à estrutura do programa e aos comandos
  - Adições de variáveis – métodos e argumentos
  - Ponteiros e alocação de memória dinâmica adicionados
  - Matriz de transformação (matrizes tratadas como objetos)
  - Recursividade e novas funções intrínsecas

Modelo Paralelo de Dados

- **Implementações**
  - High Performance Fortran (HPF): Extensões do Fortran 90 para suportar distribuição de dados
  - Tudo o que tem o Fortran 90
  - Diretivas para dizer ao compilador como distribuir os dados
  - Assertivas que podem melhorar a otimização do código gerado
  - Construção paralela de dados (parte do Fortran 95)

Outros Modelos

Outros Modelos

- Além dos modelos já mencionados há outros modelos de programação paralela. Três dos mais comuns são descritos:
  - Híbrido
  - SPMD
  - MPMD

Outros Modelos

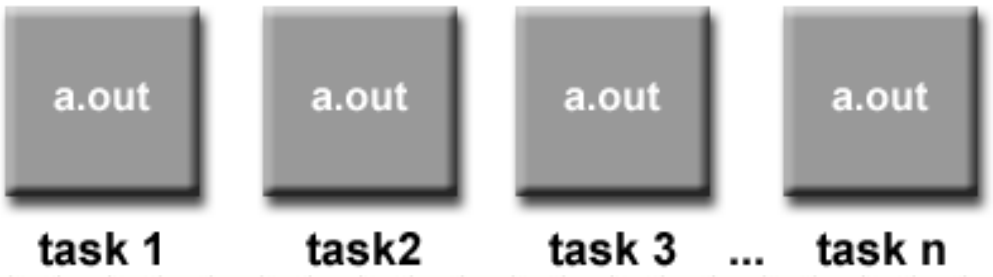
- **Híbrido**
  - Combinação de dois ou mais modelos de programação paralela
    - Exemplo:
      - Combinação do modelo de passagem de mensagens (MPI) com o modelo de threads (POSIX Threads) ou o modelo de memória compartilhada (OpenMP)
      - Combinação de dados paralelos com passagem de mensagens em arquiteturas de memória distribuída. Na verdade utiliza-se passagem de mensagens para transmitir dados entre as tarefas, de forma transparente para o programador

Outros Modelos

- **SPMD – Single Program Multiple Data**
  - Modelo de programação de alto nível que pode ser construído a partir da combinação dos modelos de programação paralela previamente mencionados
- Aqui o processamento é colaborativo entre todas as máquinas
- O código executado em todas é o mesmo (single program)
- Os dados (multiple data) são trocados de modo coordenado entre os processos

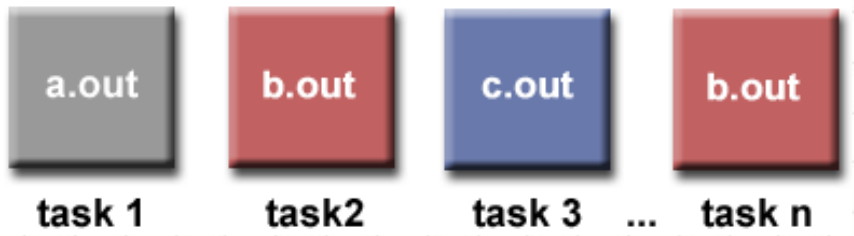
Outros Modelos

- Programas SPMD geralmente possuem a lógica necessária em seu programa para permitir que diferentes tarefas se ramifiquem ou condicionalmente executem apenas as partes do programa que se destinam a executar.
- Todas as tarefas podem utilizar diferentes dados



Outros Modelos

- **MPMD – Multiple Program Multiple Data**
  - Modelo de programação de alto nível que pode ser construído a partir da combinação dos modelos de programação paralela previamente mencionados
- Um único processo chamado mestre controla todas as tarefas que serão dadas para os outros processos (escravos/trabalhadores)
- Todos podem executar o mesmo programa ou diferentes programas



Outros Modelos

- Todas as tarefas podem utilizar dados diferentes
- O mestre executa a coordenação e pode ou não contribuir para a computação
- Os trabalhadores executam as tarefas independentemente. E só se comunicam com o mestre para passar o resultado da tarefa por ele executada e obter a sua próxima tarefa
- Esse modelo tem gargalo na comunicação causado por todos os trabalhadores, pois eles precisam se comunicar com o processo mestre

EXERCÍCIO E LEITURA RECOMENDADA

Exercício

- Acessar o Moodle

Leitura Recomendada

- Introduction to Parallel Computing, Ananth Grama, Anshul Gupta, George Karypis, Vipin Kumar - 2ª ed., Addison Wesley
- Introduction to Parallel Computing
  - https://computing.llnl.gov/tutorials/parallel_comp/

Dúvidas

Próxima Aula...

- Técnicas de Desenvolvimento de Programas Paralelos – Parte 1