Arquiteturas Paralelas e Distribuídas

Definição dos passos para a criação de um programa paralelo

Eduardo Furlan Miranda

Baseado em: GRAMA, Ananth; GUPTA, Anshul; KARYPIS, George; KUMAR, Vipin. Introduction to Parallel Computing. 2003. ISBN: 978-0-201-64865-2.

Resumo dos Passos

- Identificação do Problema e Análise de Paralelismo
- Decomposição do Problema (dados ou tarefas)
- Mapeamento das Tarefas para Processadores
- Comunicação e Sincronização
- Implementação do Programa Paralelo
- Avaliação de Desempenho e Otimização
- Validação e Correção de Erros

Identificação do Problema e Análise de Paralelismo

- Identificar o problema que será resolvido e determinar se ele pode ser paralelizado
- Nem todos os problemas são adequados para paralelização
  - Ex.: dependência de dados, algoritmos recursivos, etc.
- É essencial avaliar se há oportunidades para dividir o trabalho em tarefas independentes ou semi-independentes
- Identificar as partes do problema que podem ser executadas simultaneamente
- Determinar o grau de dependência entre as tarefas
  - Granularidade (tamanho das tarefas): fina ou grossa
- Avaliar o custo-benefício da paralelização
  - overhead de comunicação vs. ganho de desempenho

Decomposição do Problema

- O problema deve ser decomposto em subproblemas menores que podem ser executados em paralelo
- Existem duas abordagens principais para a decomposição:
  - Decomposição de Dados
    - Divide os dados em partes menores que podem ser processadas independentemente
      - Ex.: dividir uma matriz em blocos para processamento paralelo
  - Decomposição de Tarefas
    - Divide o problema em tarefas funcionais independentes
      - Ex.: em um sistema de simulação, diferentes componentes (como física, gráficos e IA) podem ser tratados por processos separados
- Escolher a estratégia de decomposição mais adequada ao problema
- Garantir que as dependências entre as tarefas sejam mínimas

Mapeamento das Tarefas para Processadores

- Após decompor o problema, é necessário mapear as tarefas para os processadores disponíveis
  - Decidir quais tarefas serão executadas em quais processadores ou núcleos
- Balanceamento de carga
  - Garantir que os processadores tenham uma carga de trabalho equilibrada
- Afinidade de Dados
  - Atribuir tarefas que acessam os mesmos dados a processadores próximos para minimizar a latência de comunicação
- Definir políticas de escalonamento estático (antes da execução) ou dinâmico (durante a execução)
- Considerar a topologia da rede (em sistemas distribuídos) para otimizar a comunicação

Comunicação e Sincronização

- Em programas paralelos, os processos frequentemente precisam compartilhar dados ou coordenar suas atividades
  - Isso requer mecanismos de comunicação e sincronização
- Comunicação por troca de mensagens ou memória compartilhada
- A sincronização garante que as tarefas ocorram na ordem correta, especialmente quando há dependências entre elas
  - Ex.: barreiras de sincronização podem ser usadas para garantir que todos os processos concluam uma fase antes de avançar para a próxima
- Implementar mecanismos de comunicação eficientes
  - Ex.: MPI para message passing
- Usar primitivas de sincronização, como semáforos, mutexes ou variáveis de condição

Implementação do Programa Paralelo

- Uso de linguagem ou framework que suporte paralelismo
- OpenMP : paralelismo em nível de thread em sistemas multicore
- MPI : comunicação entre processos em sistemas distribuídos
- CUDA/OpenCL : Para programação paralela em GPUs
- Existem várias outras opções e frameworks de alto nível
- Codificar as tarefas e a lógica de comunicação/sincronização
- Testar o programa em um ambiente controlado para identificar bugs ou gargalos

Avaliação de Desempenho e Otimização

- Após a implementação, o programa deve ser avaliado para medir seu desempenho e identificar áreas para otimização
- Métricas comuns incluem:
  - Speedup : quanto mais rápido o programa paralelo é em relação à versão sequencial
  - Eficiência : quão bem os recursos computacionais estão sendo utilizados
  - Escalabilidade : como o programa se comporta ao aumentar o número de processadores ou núcleos
- Usar ferramentas de profiling para identificar gargalos
- Otimizar a granularidade das tarefas e reduzir o overhead de comunicação
- Ajustar o balanceamento de carga e a sincronização

Validação e Correção de Erros

- Finalmente, o programa deve ser validado para garantir que produza resultados corretos e consistentes
- Em programas paralelos, erros comuns incluem race conditions, deadlocks e inconsistências de dados
- Realizar testes extensivos com diferentes entradas e configurações
- Usar ferramentas de depuração específicas para programas paralelos (ex.: Intel Inspector, Valgrind)
- Garantir que o programa seja robusto e tolerante a falhas, especialmente em sistemas distribuídos