ORGANIZAÇÃO E ARQUITETURA DE COMPUTADORES

Arquiteturas Paralelas

Medidas de desempenho

Alexandre Amory

Edson Moreno

Índice

Introdução

- **Aumento de desempenho dos PCs não atende diversas aplicações**
  - Previsão do tempo
  - Prospecção de petróleo (análise de local para perfuração de poços de petróleo)
  - Simulações físicas (aerodinâmica; energia nuclear)
  - Matemática computacional (análise de algoritmos para criptografia)
  - Bioinformática (simulação computacional da dinâmica molecular de proteínas)

- **Problemas**
  - Falta de processamento
  - Falta de memória

- **Alternativas dos usuários destas aplicações**
  - Modelos mais abstratos
  - Heurísticas
  - Aceleradores (HW especial)
  - Executar aplicação em máquinas mais poderosas (arquiteturas especiais ou arquiteturas paralelas)

Arquiteturas Especiais

- **Melhoram desempenho, replicando número de unidades ativas**
  - Normalmente processadores
- **Principal objetivo**
  - Reduzir o tempo total de execução
- **Outros objetivos**
  - Tolerância a falhas
    - Reduz a probabilidade de falhas em cálculos → cada unidade ativa calcula o mesmo problema → no final ocorre uma votação
  - Modelagem
    - Reduz a complexidade da modelagem e implementação da aplicação, utilizando uma linguagem que expresse paralelismo
  - Aproveitamento de recursos
    - Aproveita melhor os recursos disponíveis, executando uma aplicação com múltiplos processos

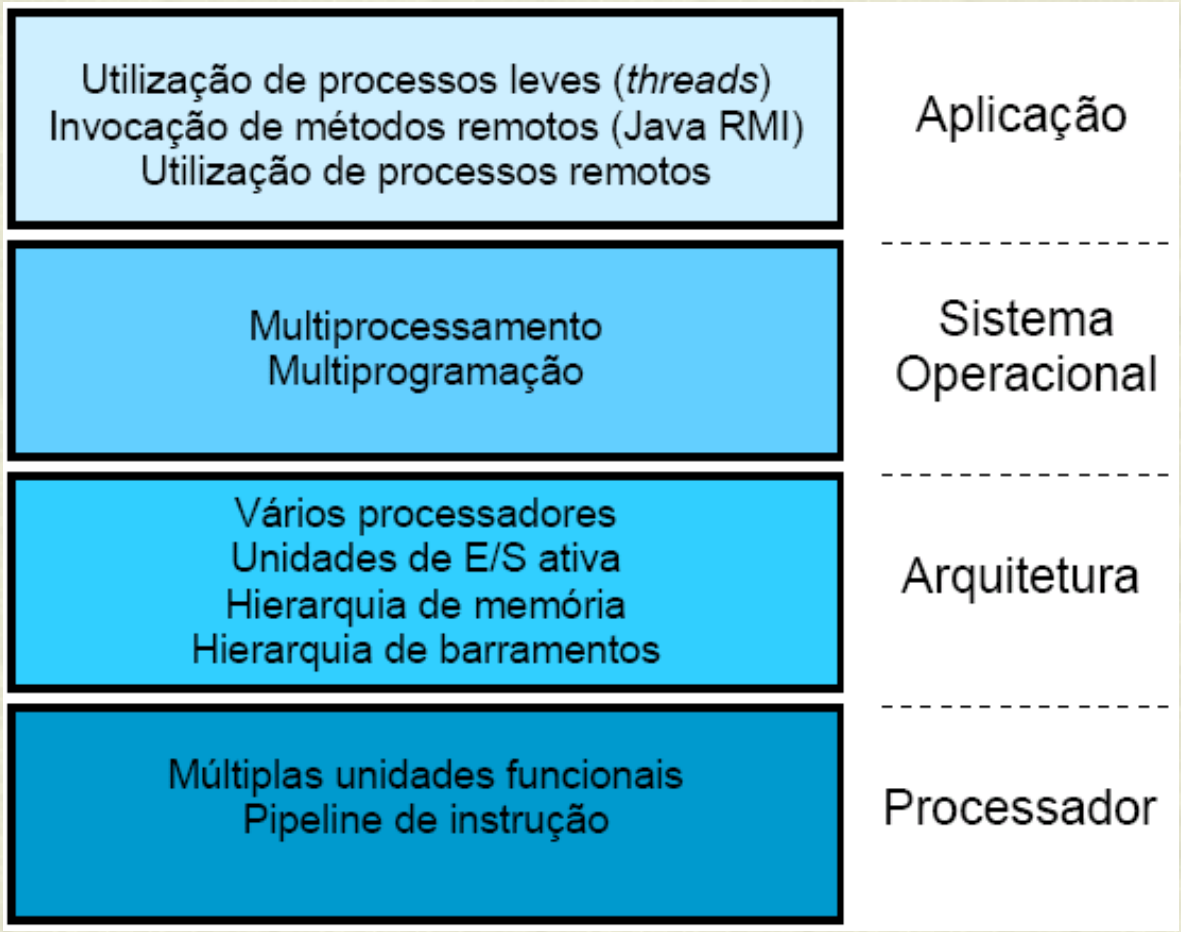
Processamento Paralelo

- **Dificuldades inerentes ao paralelismo**
  - Dificuldade de alimentar vários processadores com dados
    - Gera custo de comunicação entre processadores
  - Programas mais complexos devido ao particionamento em unidades ativas
    - O particionamento do programa não é realizado pelo SO
  - Dificuldade em extrair o paralelismo da aplicação
    - Programa que não foi preparado com paralelismo ***não*** executa mais rápido, apenas por estar em máquina paralela

- **Exemplos de programas paralelos**
  - Aplicação com várias *threads*
  - Aplicação usando RMI (*Remote Method Invocation*)
  - Aplicação com vários processos que se comunicam por *sockets*

Níveis de exploração de paralelismo

- **Exploração de paralelismo está presente nos diversos níveis de um sistema**



Granularidade de paralelismo

- **Definição**
  - Grau de agrupamento de funcionalidades em unidades ativas
- **Grão grosso**
  - Particionamento em unidades de trabalho grandes
  - Alto custo de processamento. Processador é a parte mais complexa do sistema
  - Baixo custo de comunicação. Infra-estrutura de comunicação de menor complexidade
- **Grão fino**
  - Particionamento em unidades de trabalho pequenas
  - Baixo custo de processamento. Processador é a parte menos complexa do sistema
  - Alto custo de comunicação. Infra-estrutura de comunicação de alta complexidade
- **Grão médio**
  - Caso intermediário entre os dois anteriores

Process. Paralelo x Distribuído

- **Ambas áreas têm os mesmos complicadores**
  - Custo de comunicação
  - Distribuição dos dados
  - Dependências
- **Motivações diferentes**
  - Processamento paralelo
    - Ganho de desempenho
      - Unidades ativas estão normalmente na mesma máquina
        - » Custos de comunicação menores (baixa latência)
  - Processamento distribuído
    - Modelagem e o aproveitamento de recursos
      - Unidades ativas estão normalmente afastadas
        - » Custos de comunicação maiores (alta latência)

Process. Paralelo x Distribuído

- **Aumento de unidades ativas não aumenta *necessariamente* o desempenho**
  - Fatores que limitam o número de unidades ativas usadas eficientemente
    - Quantidade de trabalho
    - Arquitetura alvo
  - Muitas unidades ativas para uma quantidade limitada de trabalho prejudica o desempenho do sistema
    - O tempo pode aumentar!!
  - Complicadores
    - Dependências de dados
    - Distribuição dos dados
    - Sincronização

Exemplo (cenário)

- **Construção de um muro**
  - 1 pedreiro faz este muro em 3 horas
  - 2 pedreiros fazem em 2 horas
  - 3 pedreiros fazem em 1 hora e meia
  - 4 pedreiros fazem em 2 horas (aumentou o tempo !!!)
  - 8 pedreiros fazem em 3 horas e meia (tempo pior que a versão sem paralelismo !!!)

- **Avaliação dos dados obtidos e porque ocorreu o aumento de tempo?**
  - Incidência de muitos complicadores
    - Ganho de desempenho *não é proporcional* ao acréscimo de unidades ativas
    - Duplicação do número de pedreiros (1→2) não reduz o tempo de execução pela metade

Exemplo (Complicadores)

- **Complicadores encontrados e a analogia com a computação paralela**
  - O muro só pode ser feito de baixo para cima
    - Dependência de dados
  - Os tijolos têm que ser distribuídos entre os pedreiros
    - Distribuição dos dados
  - Um pedreiro não pode levantar o muro do seu lado muito na frente dos outros pedreiros. Ritmo de subida do muro é dado pelo pedreiro mais lento
    - Sincronização
  - Se só existir um carrinho com cimento este será disputado por todos os pedreiros
    - Áreas críticas

Índice

Medidas básicas de desempenho

- **Definição**
  - Permite indicar o desempenho de diferentes aspectos de um sistema paralelo
- **Medidas**
  - Desempenho da aplicação
    - Aqui considera-se o efeito conjunto do processamento, armazenamento e comunicação
  - Desempenho da rede de interconexão
    - Aqui considera-se o efeito apenas da comunicação

Desempenho de aplicação (Speed-Up)

- **Medida usada para avaliar o fator de aceleração**
  - Indica quantas vezes o programa paralelo é mais rápido que a versão seqüencial para executar uma dada tarefa
  - É calculado pela razão entre o melhor tempo seqüencial e o melhor tempo da versão paralela

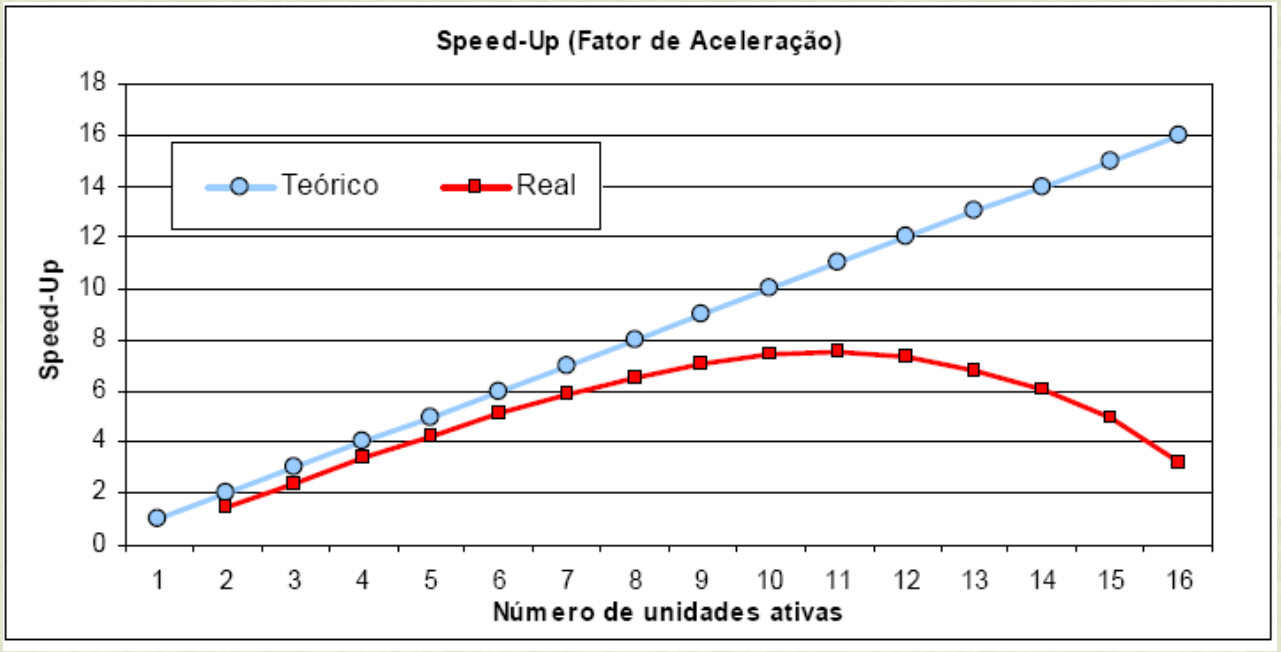
SpeedUp(p) ou SU(p) = TS / TP(p)

  - Onde:
    - *TS é o tempo de execução da aplicação na versão seqüencial*
    - *TP é o tempo de execução na versão paralela*
    - *p* é o número de unidades ativas (processadores) utilizadas

- **Se *SU* > 1 a versão paralela reduziu o tempo de execução (ficou mais rápido que a seqüencial)**
- **Se *SU* < 1 a versão paralela aumentou o tempo de execução (ficou mais lenta que a seqüencial)**

Speed-Up (Teórico x Real)

- **Cada aplicação tem sua curva que depende**
  - Do processamento
  - Da incidência de complicadores
- **Toda aplicação tem um número ideal de unidades ativas para a obtenção do melhor desempenho em uma dada arquitetura alvo**
  - Não sendo verdade que quanto mais unidades ativas melhor



Dempenho da aplicação

- **Eficiência**
  - Indica a taxa de utilização média das unidades ativas
  - Mostra se os recursos foram bem aproveitados
  - É calculado pela razão entre o *Speed-Up* e o número de *unidades ativas* utilizadas

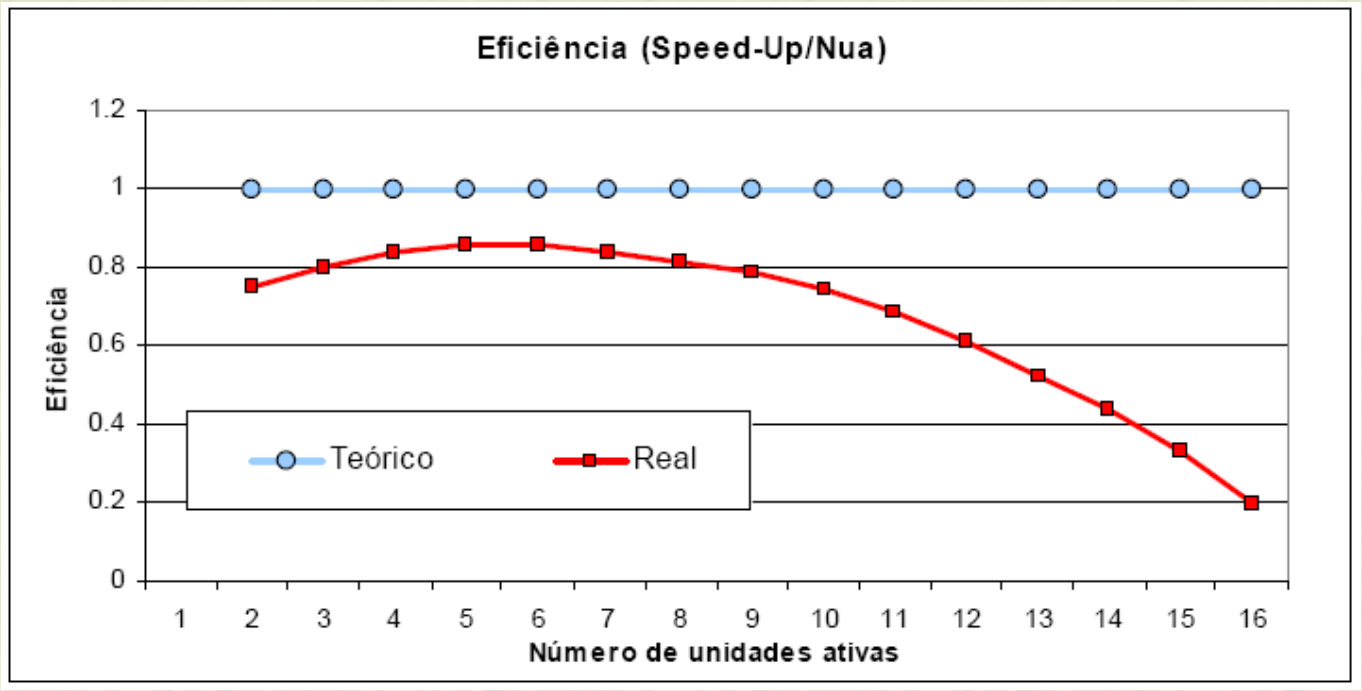
  - Eficiência(p) ou E(p) = SU(p) / p

- **Normalmente, as unidades ativas ficam parte de seu tempo esperando por resultados de vizinhos**
  - Reduz sua taxa de utilização e conseqüentemente a eficiência

Eficiência

- **Eficiência ideal**
  - Cada unidade ativa com 100% do tempo ativa (linha azul)

- **A melhor taxa de utilização média não significa o menor tempo de execução**
  - Exemplo: o menor tempo de execução ocorreu com 11 unidades ativas e a melhor taxa de utilização média com 5 unidades ativas



Lei de Amdahl

- **Em programação, sempre existe algum trecho sequencial**
  - Não pode ser paralelizada
  - Se s for a parte de trabalho seqüencial o que resta (1-s) será a parte susceptível de ser paralelizada

- **Mesmo que a parte paralela seja perfeitamente escalável o desempenho (Speedup) está limitado pela parte seqüencial**

Lei de Amdahl

Tempo de execução após melhoria =
Tempo de execução não afetado +
(Tempo de execução afetado / Quantidade de melhoria)

- **Exemplo:**

  “Suponha que um programa seja executado em 100 segundos em uma máquina, com multiplicação responsável por 80 segundos desse tempo. O quanto precisamos melhorar a velocidade da multiplicação se queremos que o programa seja executado 4 vezes mais rápido?”

Lei de Amdahl

Tempo de execução após melhoria =
Tempo de execução não afetado +
(Tempo de execução afetado / Quantidade de melhoria)

- **Exemplo:**

  “Suponha que um programa seja executado em 100 segundos em uma máquina, com multiplicação responsável por 80 segundos desse tempo. O quanto precisamos melhorar a velocidade da multiplicação se queremos que o programa seja executado 4 vezes mais rápido?”

  Que tal torná-lo 5 vezes mais rápido?