Insper

SuperComputação

Aula 11 – Introdução a paralelismo

2020 – Engenharia

Luciano Soares <lpsoares@insper.edu.br>
Igor Montagner <igorsm1@insper.edu.br>

Revisão

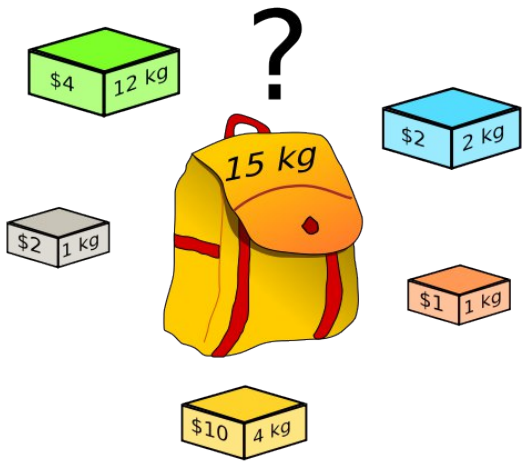
A mochila binária

Quais escolhas podem ser feitas?

- Quais produtos pegar?

Qual é a função objetivo?

- Maximizar valor dos objetos guardados



Quais são as restrições?

- Peso dos objetos não pode exceder capacidade da mochila

Resolução do problema

- Heurísticas
- Busca local
- Busca exaustiva
  - Branch and Bound (propriedades do problema)

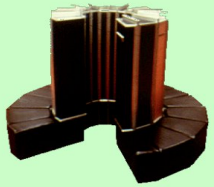
Discussão: o quanto algoritmos bons ajudam?

Atividade prática

Medindo desempenho I (25 minutos)

1. Criar rotinas de medição de desempenho automáticas
2. Visualizar resultados usando gráficos

Mas e o paralelismo?



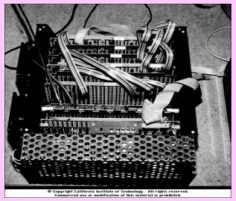
Cray 1 (1976)

Cray 2 (1985)

Cray C-90 (1991)

Vector Computers

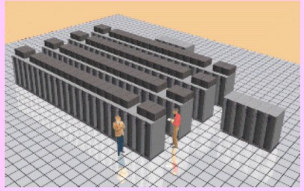
SMP computers



Cosmic cube (1983)

Massively Parallel Processors (MPP)

Paragon (1993)



ASCI Red (1997)

Clusters (late 80's)

Cluster Computers

Linux PC Clusters
(~1995)

Insper

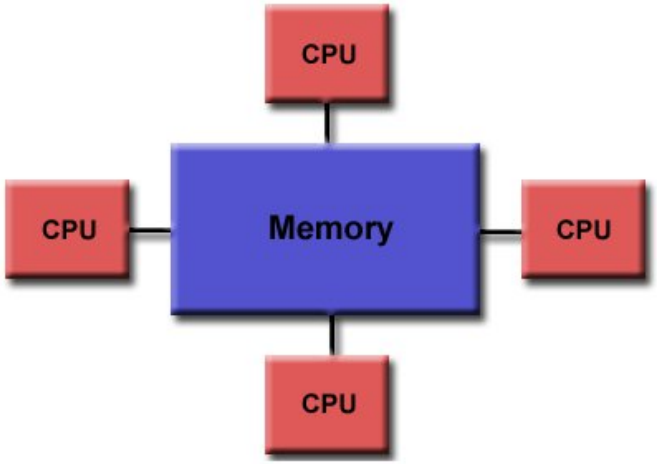
Solução de alto desempenho

1. Algoritmos eficientes
2. Implementação eficiente
    - Cache, paralelismo de instrução
    - Linguagem de programação adequada
3. Paralelismo

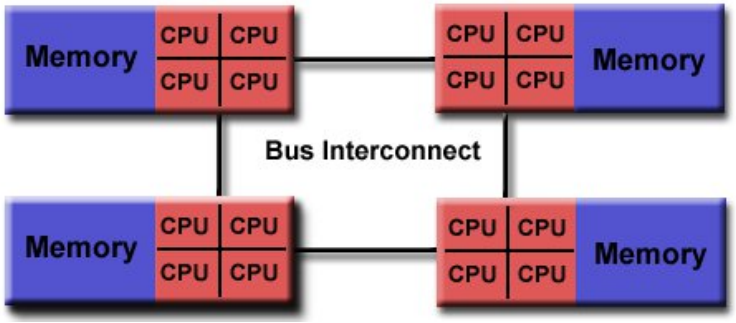
Solução de alto desempenho

1. **Algoritmos eficientes**
2. Implementação eficiente
    - Cache, paralelismo de instrução
    - Linguagem de programação adequada
3. **Paralelismo**

Sistemas Multi-core



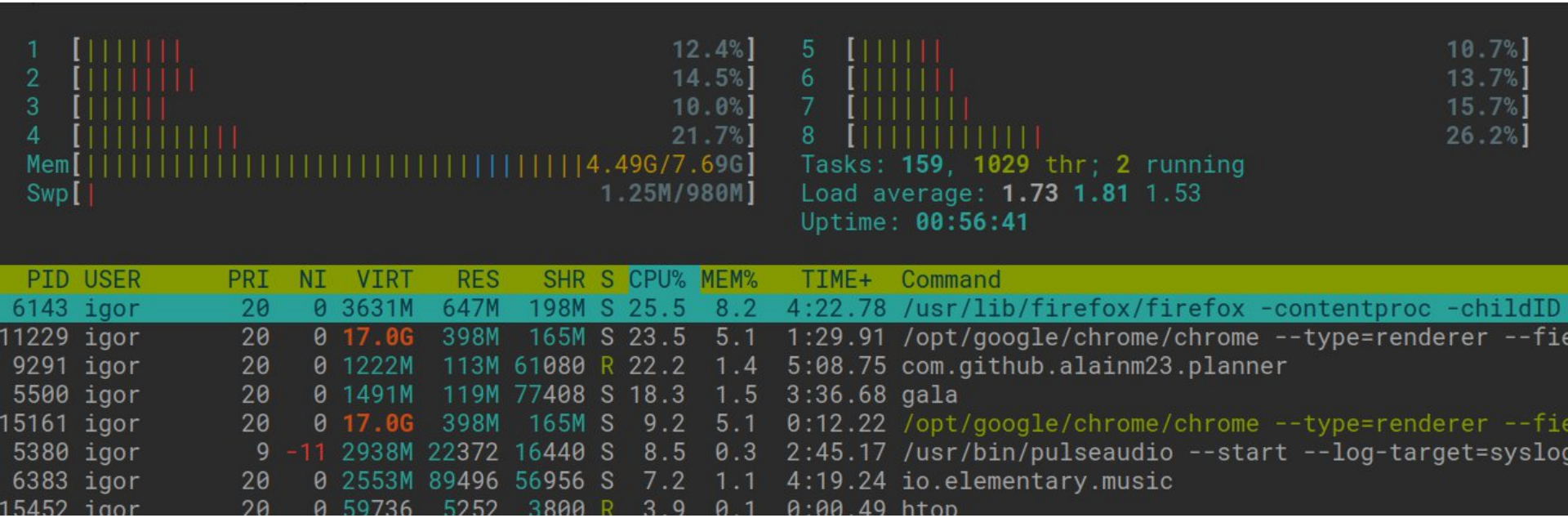
Uniform Memory Access



Non-Uniform Memory Access

Insper

Sistemas multi-core



Insper

Discussão I: qual expectativa de melhoria de velocidade?

Exemplo 1

```
vector<double> dados;
vector<double> resultados;
for (int i = 0; i < dados.size(); i++) {
    resultados[i] = funcao_complexa(dados[i]);
}
```

Exemplo 1

```
vector<double> dados;
vector<double> resultados;
for (int i = 0; i < dados.size(); i++) {
    resultados[i] = funcao_complexa(dados[i]);
}
```

Tempo total divido por 8!

Exemplo 2

```
vector<double> dados;
vector<double> resultados;
resultados[0] = 0;
for (int i = 1; i < dados.size(); i++) {
    resultados[i] = funcao_complexa(dados[i], resultados[i-1]);
}
```

Exemplo 2

```
vector<double> dados;
vector<double> resultados;
resultados[0] = 0;
for (int i = 1; i < dados.size(); i++) {
    resultados[i] = funcao_complexa(dados[i], resultados[i-1]);
}
```

Nenhum ganho! Depende da iteração anterior :(

Conceito 1: Dependência

Um loop tem uma **dependência** de dados sua execução correta depende da ordem de sua execução.

Isto ocorre quando **uma iteração depende de resultados calculados em iterações** anteriores.

Quando não existe nenhuma dependência em um loop ele é dito **ingenuamente paralelizável**.

Exemplo 3

```
vector<double> dados;
vector<double> resultados1;
vector<double> resultados2;
resultados1[0] = resultados2[0] 0;
for (int i = 1; i < dados.size(); i++) {
    resultados1[i] = funcao_complexa(dados[i], resultados1[i-1]);
    resultados2[i] = funcao_complexa2(dados[i], resultados2[i-1]);
}
```

Exemplo 3

```
vector<double> dados;
vector<double> resultados1;
vector<double> resultados2;
resultados1[0] = resultados2[0] 0;
for (int i = 1; i < dados.size(); i++) {
    resultados1[i] = funcao_complexa(dados[i], resultados1[i-1]);
    resultados2[i] = funcao_complexa2(dados[i], resultados2[i-1]);
}
```

Podemos fazer resultados1 e resultados2 em paralelo!

Conceito 2: Paralelismo

Paralelismo de dados: faço em paralelo a mesma operação (lenta) para todos os elementos em um conjunto de dados (grande).

Paralelismo de tarefas: faço em paralelo duas (ou mais) tarefas independentes. Se houver dependências quebro em partes independentes e rodo em ordem.

Discussão II: Busca local

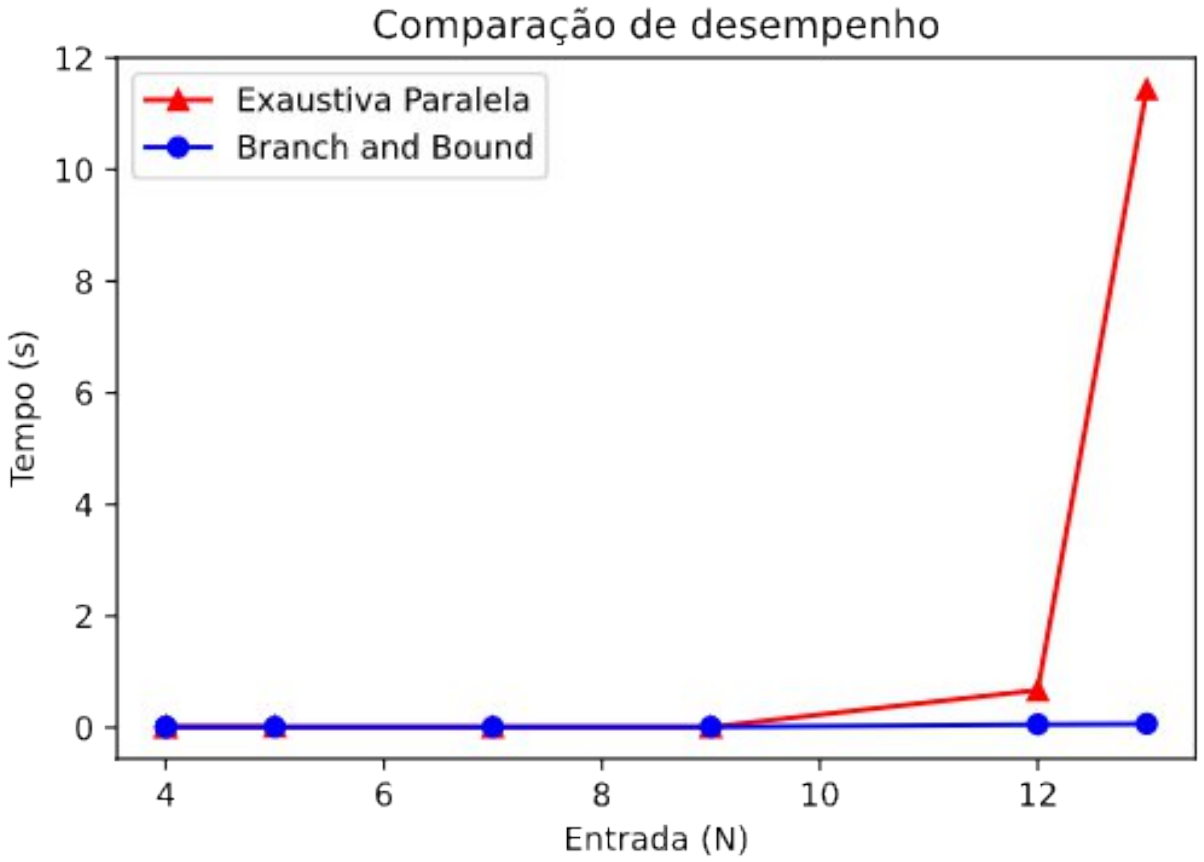
Discussão III: Busca exaustiva

Atividade prática

Expectativas de melhorias com paralelização (30 minutos)

1. Medir desempenho de algoritmos sequenciais e paralelos
2. Comparar desempenho esperado com desempenho medido.

Algoritmos fazem a diferença! (II)



Insper

Resumo

1. Automatizar trabalho é importante
2. Recursos visuais são úteis para comparar duas implementações do mesmo algoritmo
3. Paralelismo pode dar ganhos expressivos
    - mas não faz milagre.
    - algoritmo esperto ganha de qualquer paralelização

Insper

www.insper.edu.br