Fundamentos de Sistemas de Operação

Unix Windows NT Netware MacOS DOS/VS Vax/VMS
Linux Solaris HP/UX AIX Mach
Chorus

Processos Concorrentes:

Uma aplicação – multiplicação de matrizes

(desenvolvimento passo-a-passo: etapas fundamentais)

Concorrência vs. Paralelismo

- Execução paralela, quando:
  - existem duas ou mais “entidades activas” (das quais, por enquanto, apenas conhecemos os processos - da aplicação) em execução **simultânea** em processadores (ou cores) distintos
  - O foco da computação paralela é a redução do tempo de execução (da aplicação)
- Execução concorrente, quando:
  - O número de processadores (ou cores) disponíveis para executar as “entidades activas” (e.g., os processos da aplicação) é inferior às necessidades da computação - assim, o tempo de processador é distribuído pelos diferentes processos; nada pode, no entanto, ser inferido acerca da velocidade relativa da execução destes, e/ou de que forma são escalonados…

Álgebra...

$$c_{11} = (a_{11} \times b_{11}) + (a_{12} \times b_{21}) + (a_{13} \times b_{31}) + (a_{14} \times b_{41})$$

$$\begin{bmatrix} a_{11} & a_{12} & a_{13} & a_{14} \\ a_{21} & a_{22} & a_{23} & a_{24} \\ a_{31} & a_{32} & a_{33} & a_{34} \\ a_{41} & a_{42} & a_{43} & a_{44} \end{bmatrix} \times \begin{bmatrix} b_{11} & b_{12} & b_{13} & b_{14} \\ b_{21} & b_{22} & b_{23} & b_{24} \\ b_{31} & b_{32} & b_{33} & b_{34} \\ b_{41} & b_{42} & b_{43} & b_{44} \end{bmatrix} = \begin{bmatrix} c_{11} & c_{12} & c_{13} & c_{14} \\ c_{21} & c_{22} & c_{23} & c_{24} \\ c_{31} & c_{32} & c_{33} & c_{34} \\ c_{41} & c_{42} & c_{43} & c_{44} \end{bmatrix}$$

$$c_{ij} = \sum_{j=1}^{n} a_{ij} \times b_{ji}$$

NOTA: não precisam de ser matrizes quadradas!

Computação sequencial...

$$c_{11} = (a_{11} \times b_{11}) + (a_{12} \times b_{21}) + (a_{13} \times b_{31}) + (a_{14} \times b_{41})$$

$$\begin{bmatrix} a_{11} & a_{12} & a_{13} & a_{14} \\ a_{21} & a_{22} & a_{23} & a_{24} \\ a_{31} & a_{32} & a_{33} & a_{34} \\ a_{41} & a_{42} & a_{43} & a_{44} \end{bmatrix} \times \begin{bmatrix} b_{11} & b_{12} & b_{13} & b_{14} \\ b_{21} & b_{22} & b_{23} & b_{24} \\ b_{31} & b_{32} & b_{33} & b_{34} \\ b_{41} & b_{42} & b_{43} & b_{44} \end{bmatrix} = \begin{bmatrix} c_{11} & c_{12} & c_{13} & c_{14} \\ c_{21} & c_{22} & c_{23} & c_{24} \\ c_{31} & c_{32} & c_{33} & c_{34} \\ c_{41} & c_{42} & c_{43} & c_{44} \end{bmatrix}$$

Computação paralela...

4 CPUs $\rightarrow$ 4 processos: azul, amarelo, violeta, cinzento

Cada processo (cor) recebe:

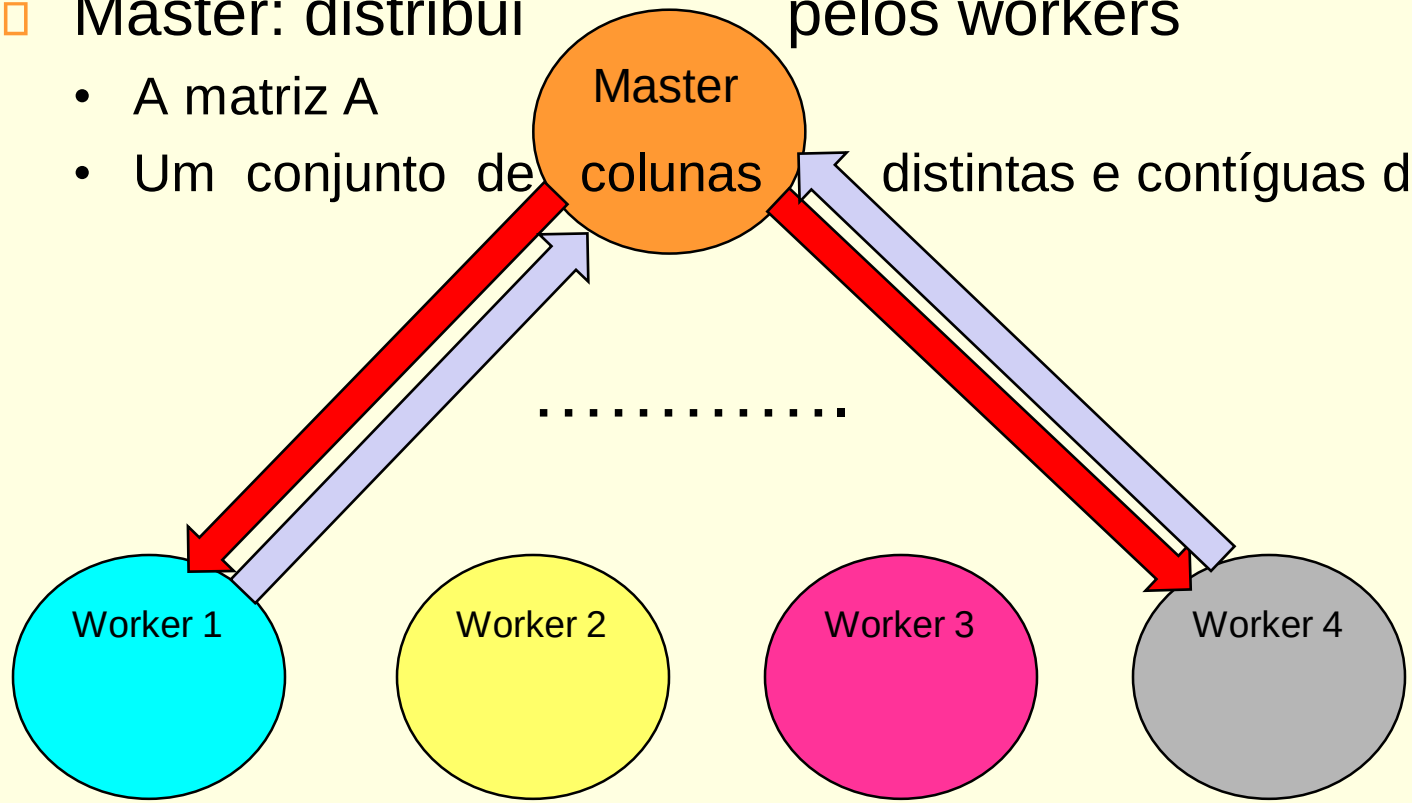
- a matriz A completa
- uma coluna da matriz B

$$\begin{pmatrix} a_{11} & a_{12} & a_{13} & a_{14} \\ a_{21} & a_{22} & a_{23} & a_{24} \\ a_{31} & a_{32} & a_{33} & a_{34} \\ a_{41} & a_{42} & a_{43} & a_{44} \end{pmatrix} \times \begin{pmatrix} b_{11} & b_{12} & b_{13} & b_{14} \\ b_{21} & b_{22} & b_{23} & b_{24} \\ b_{31} & b_{32} & b_{33} & b_{34} \\ b_{41} & b_{42} & b_{43} & b_{44} \end{pmatrix} = \begin{pmatrix} c_{11} & c_{12} & c_{13} & c_{14} \\ c_{21} & c_{22} & c_{23} & c_{24} \\ c_{31} & c_{32} & c_{33} & c_{34} \\ c_{41} & c_{42} & c_{43} & c_{44} \end{pmatrix}$$

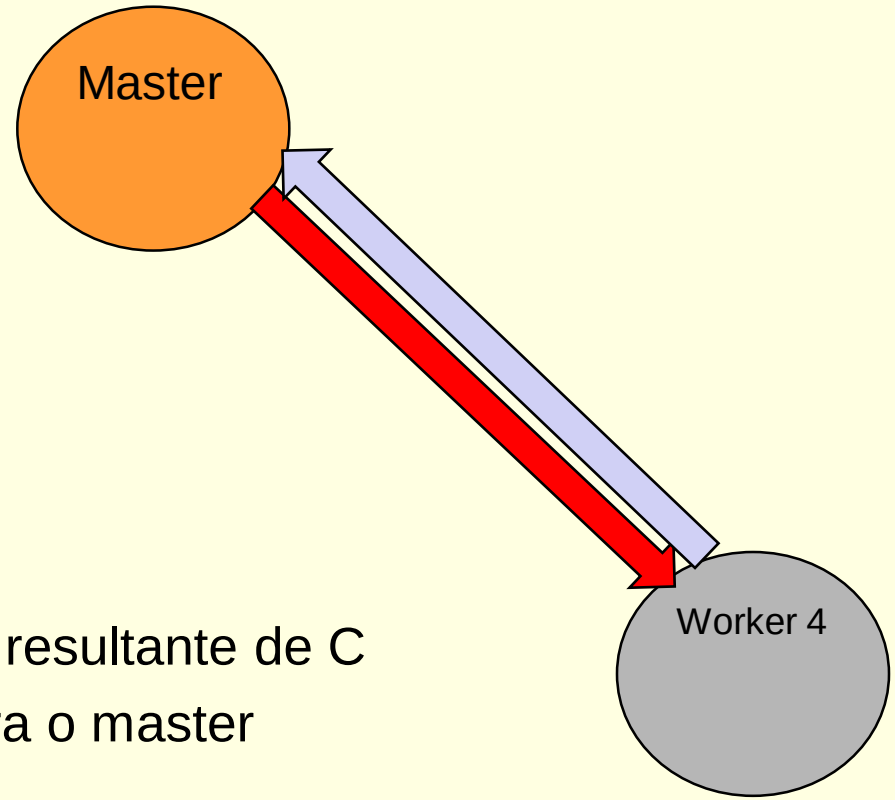
Assim, cada processo calcula uma coluna de C

Uma solução usando mensagens (1)

- Master: distribui pelos workers
  - A matriz A
  - Um conjunto de colunas distintas e contíguas de B

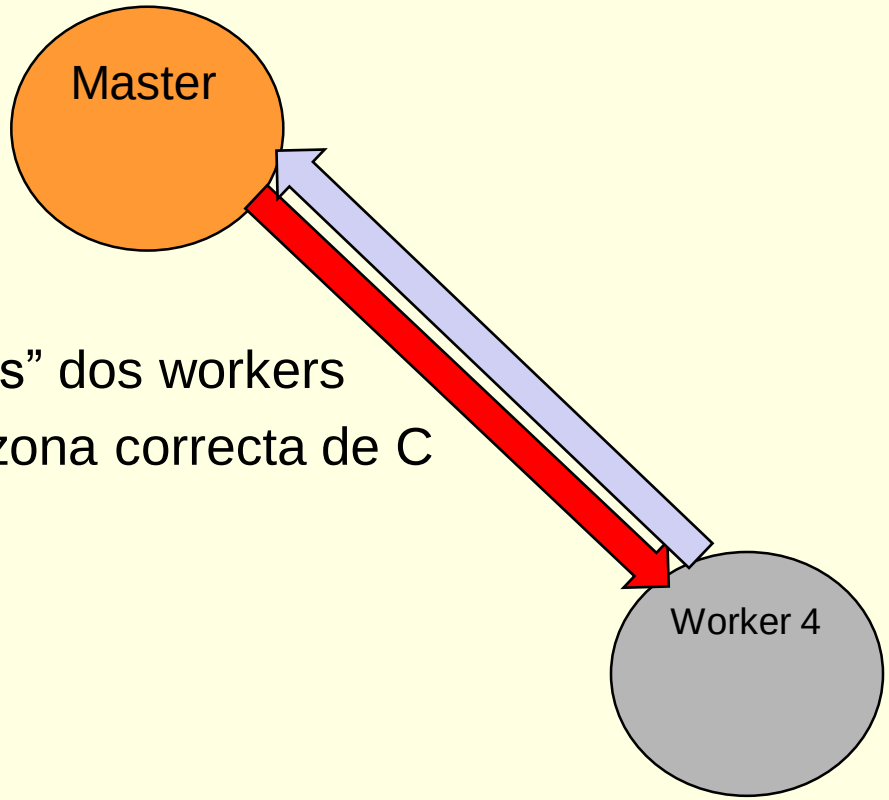


Uma solução usando mensagens (2)



- Cada worker
  - Calcula a “fatia” resultante de C
  - Envia a fatia para o master

Uma solução usando mensagens (3)



- O master
  - Recebe as “fatias” dos workers
  - “Arruma-as” na zona correcta de C

Uma solução com pipes: master (1)

- O master
  - Cria 4 pares de pipes (par: um canal para W e outro para R)
  - Lança 4 workers
  - Em cada par,
    - no canal W fecha o “extremo” R
    - No canal R fecha o “extremo” W
- Assumindo que os workers estão prontos **(é preciso?)**
  - Envia A para cada worker
  - Envia fatia de B para cada worker

Uma solução com pipes: master (2)

- Recepção dos resultados
  - "Percorrer" cada pipe de leitura (R), lendo
    - Hipótese 1: ler até receber tudo/não haver mais nada
    - Hipótese 2: ler, e se não tiver recebido tudo, avançar para outro pipe
  - Arrumar os dados recebidos na matriz C
- Teste
  - Comparar C com uma D = AxB, sendo D calculada pelo master usando um algoritmo sequencial

Uma solução com pipes: worker

- Fechar extremos supérfluos
  - no canal W fecha o “extremo” R
  - No canal R fecha o “extremo” W
- Recepção dos dados
  - Receber da pipe de leitura (R)
    - A matriz A **(como se sabe que já acabou?)**
    - A “fatia” de B (idem)
- Cálculo (algoritmo sequencial)
  - Como determinar as dimensões da matriz e sub-matriz?
- Envio da sub-matriz para o master; terminar worker

A reflectir…

- Se, no master
  - As matrizes A e B estão já carregadas com valores…
  - E o fork dos workers é executado depois,
    - É **mesmo necessário** enviar A e B para os workers via pipes?