MAC 438 - Programação Concorrente
EP 1

Daniel de Angelis Cordeiro - 3286158
Igor Ribeiro Sucupira - 3286499

7 de abril de 2003

Programas Implementados

- Gerador.java - este programa é um gerador de matrizes. As matrizes são preenchidas com valores pseudo-aleatórios, do tipo double. Para gerar matrizes n por m, execute o programa da seguinte maneira:

    java Gerador n m nomeDoArquivoDeSaida -i, onde nomeDoArquivoDeSaida é o nome do arquivo onde será guardada a matriz gerada. A opção -i é opcional e faz com que a matriz gerada seja impressa na saída padrão, em formato humanamente legível. Isso pode ser interessante, já que as matrizes são gravadas utilizando um formato mais econômico (através de um objeto DataOutputStream).

- Impressor.java - este programa pode ser utilizado para imprimir, na saída padrão, em formato humanamente legível, as matrizes geradas pelo gerador ou pelo multiplicador (leia abaixo sobre o multiplicador). Para executá-lo, utilize:

    java Impressor nomeDoArquivoDeEntrada, onde nomeDoArquivoDeEntrada é o nome do arquivo que contém a matriz que se deseja visualizar.

- Multiplicador.java - este é o principal programa de nossa implementação. Para executá-lo, utilize:

    java Multiplicador arqA arqB arqC numThreads -i, onde:
    arqA e arqB são os arquivos que contêm as matrizes de entrada (A e B).
    arqC irá conter a matriz de saída C (resultado de A x B).
    numThreads especifica o número de threads que deverá ser utilizado. Caso seja utilizado um número maior que n (a quantidade de linhas da matriz A), o programa utilizará n threads.

-i (opcional) faz com que a matriz resultante (C) seja impressa na saída padrão, em formato humanamente legível.

Classes auxiliares

- TarefaDeMultiplicacao - subclasse de Thread. Os objetos desta classe serão responsáveis pela multiplicação de algumas linhas da matriz A pela matriz B. A multiplicação propriamente dita será feita por um objeto da classe Multiplicador, que é passado para o construtor dos objetos TarefaDeMultiplicacao. Esse objeto Multiplicador possui todas as informações necessárias para a multiplicação, recebendo das tarefas apenas as linhas que devem ser multiplicadas.

Para mais informações, favor conferir os abundantes comentários no código. :-)

Gráficos de variação de desempenho

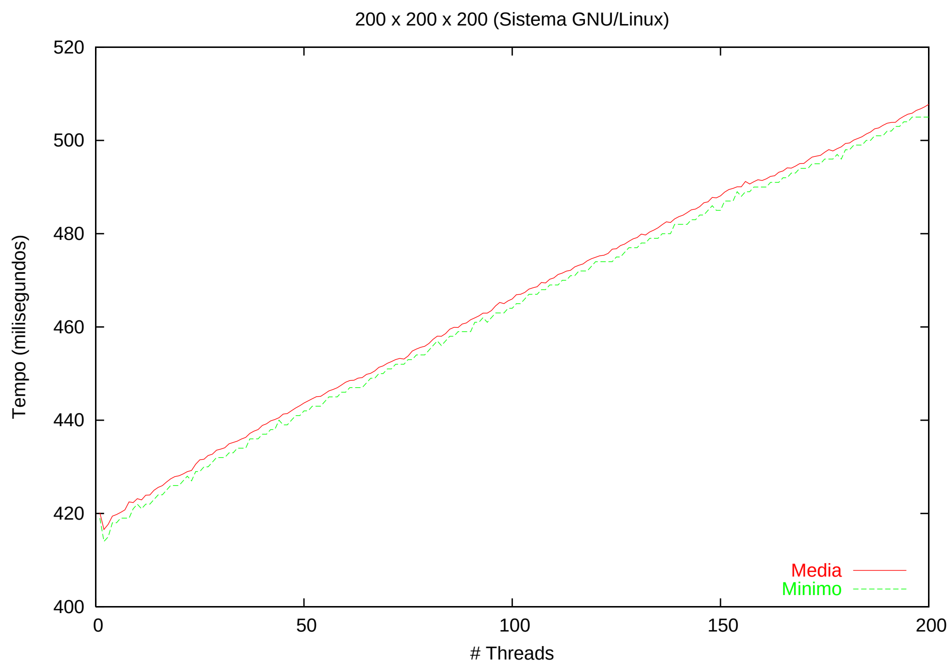
Os testes foram realizados nos sistemas operacionais GNU/Linux e Microsoft Windows XP.

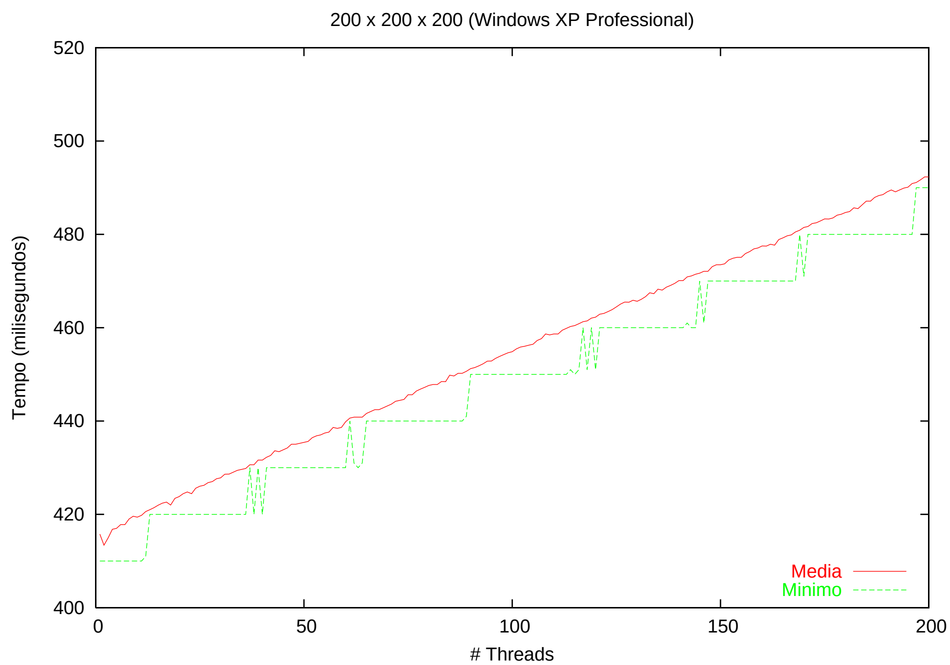
Micro com Linux: Celeron 1.0 Ghz, 128 MB RAM

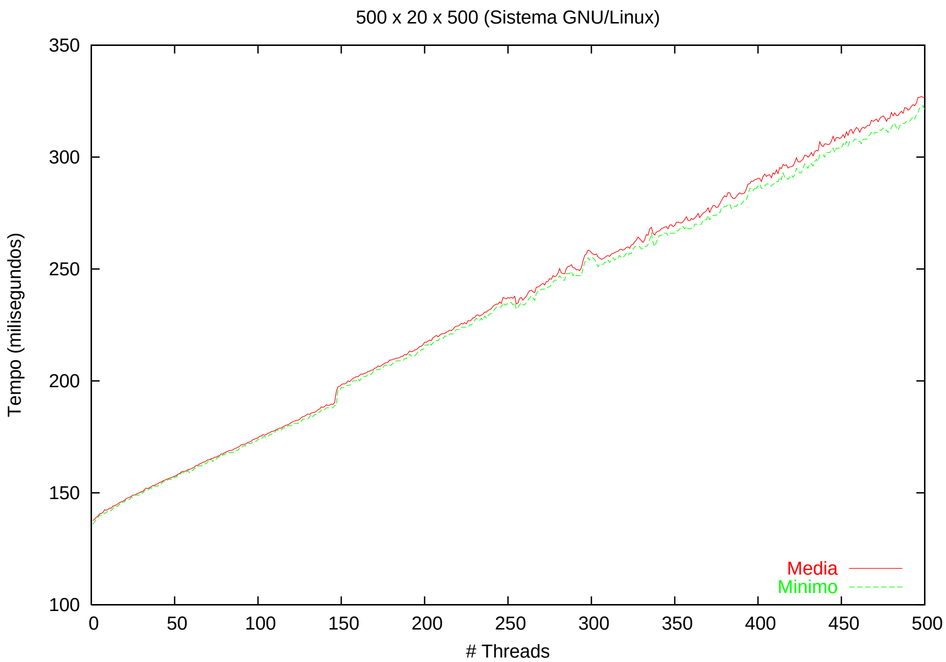
Micro com Windows: Celeron 1.0 Ghz, 128 MB RAM

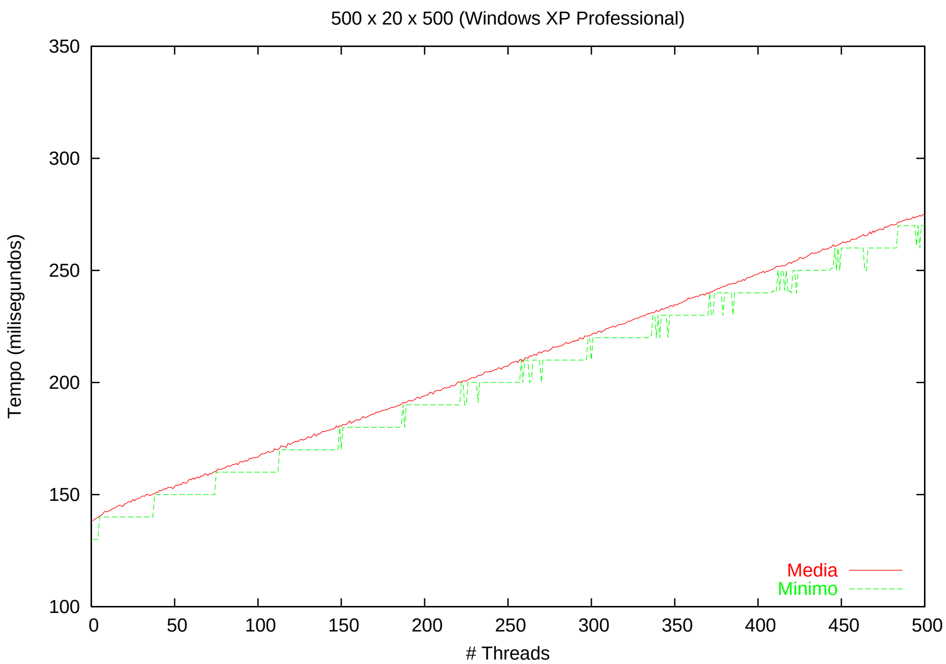
A idéia inicial, para a geração dos gráficos, foi rodar 30 vezes cada configuração. Escolhemos esse número devido à Lei dos Grandes Números, segundo a qual, numa amostra com n >= 30 dados, a distribuição de probabilidades para a média é bem próxima de uma Gaussiana. Devido às características da distribuição Normal, a probabilidade de que a média "real" esteja bem próxima da média calculada entre os 30 elementos da amostra é bem próxima de 1.

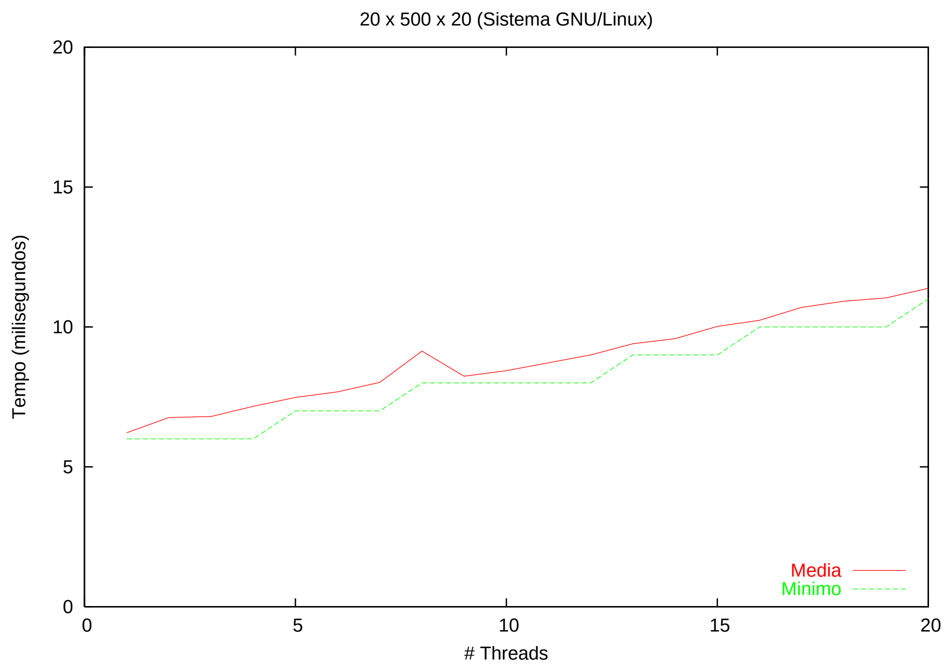
Porém, após a execução de alguns testes, verificamos que executar 30 vezes cada configuração resultava em um gráfico com muitas variações. Assim, empiricamente, acabamos escolhendo o valor 50 para o número de execuções de cada configuração.

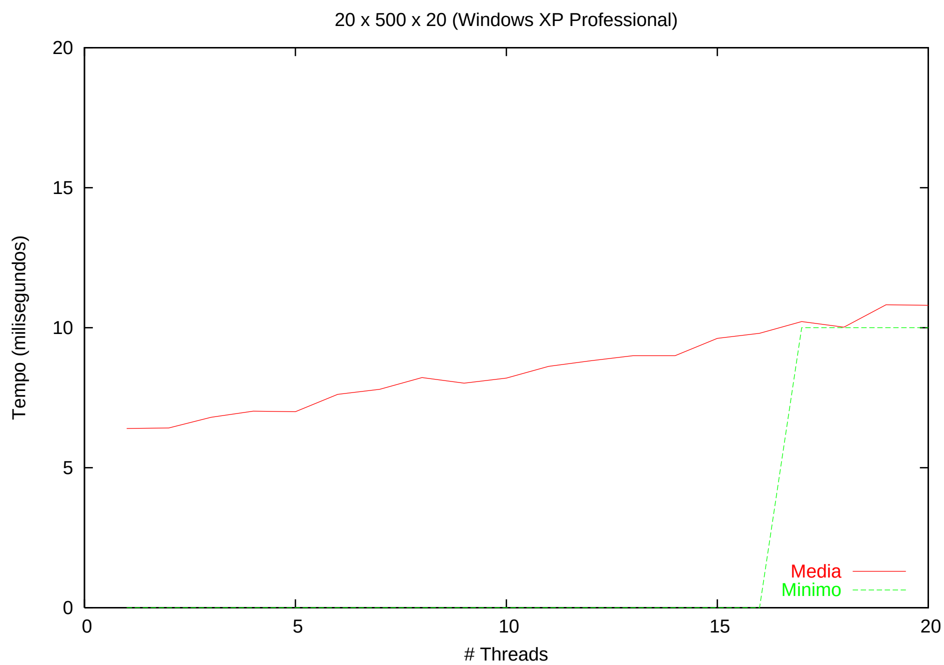

200 x 200 x 200 (Sistema GNU/Linux)
Tempo (milisegundos)
Threads
Media
Minimo
520
500
480
460
440
420
400
0
50
100
150
200


200 x 200 x 200 (Windows XP Professional)
Tempo (milisegundos)
Threads
Media
Minimo
520
500
480
460
440
420
400
0
50
100
150
200


500 x 20 x 500 (Sistema GNU/Linux)
Tempo (milisegundos)
Threads
Media
Minimo
350
300
250
200
150
100
0
50
100
150
200
250
300
350
400
450
500


500 x 20 x 500 (Windows XP Professional)
Tempo (milisegundos)
Threads
Media
Minimo
350
300
250
200
150
100
0
50
100
150
200
250
300
350
400
450
500


20 x 500 x 20 (Sistema GNU/Linux)
Tempo (milisegundos)
Threads
Media
Minimo
0
5
10
15
20


20 x 500 x 20 (Windows XP Professional)
Tempo (milisegundos)
Threads
Media
Minimo
0
5
10
15
20

Análise dos resultados

Para fazer os gráficos, utilizamos o número de threads no eixo horizontal e o tempo de execução (em milissegundos) no eixo vertical. Cada gráfico representa os resultados, em um sistema operacional, de uma multiplicação A x B, sendo que a legenda n x m x l em cada gráfico indica o número de linhas da matriz A (n), o número de colunas de A (m = número de linhas da matriz B) e o número de colunas de B (l). Ainda, em cada gráfico, são exibidas duas funções: uma delas baseia-se no cálculo da média entre as 50 execuções para cada número de threads; a outra função calcula o mínimo de cada um desses conjuntos de 50 execuções.

Foram testados 3 tipos de multiplicação, tanto no GNU/Linux quanto no Microsoft Windows XP:

A1(200x200) x B1(200x200)

A2(500x20) x B2(20x500)

A3(20x500) x B3(500x20)

Em ambos os sistemas, os gráficos das médias assemelham-se aos gráficos de funções lineares crescentes. Isso também vale para os gráficos dos mínimos, no caso do GNU/Linux. Esses fatos indicam que a sobrecarga de tempo para alocação e comutação de tarefas faz com que a resolução de um mesmo problema utilizando várias threads no mesmo processador demore um tempo linearmente crescente com o número de threads.

Ainda, em alguns gráficos, há uma pequena queda no tempo de execução, quando se passa de uma thread para algumas (poucas) threads. Esse fato, provavelmente, ocorre devido à dedicação de mais tarefas para a resolução do mesmo problema, quando a sobrecarga provocada por essas tarefas ainda é muito pequena. Por exemplo: se há apenas mais uma tarefa sendo executada pelo processador, nosso problema com uma thread obterá metade do tempo da CPU. Já o mesmo problema com duas threads obterá 2/3 do tempo da CPU.

Observe que o fato acima nem sempre ocorre, já que, se houver várias outras tarefas executando no sistema operacional, a diferença de tempo de CPU entre resolver o problema com uma tarefa e resolvê-lo com duas é muito pequena e é compensada pela sobrecarga de alocação e comutação de tarefas.

Outro fator importante é a diferença entre os sistemas operacionais GNU/Linux e Windows XP, no que diz respeito aos coeficientes angulares das retas obtidas nos gráficos. Nota-se que, nos gráficos do Windows, as retas são menos inclinadas, o que pode indicar uma maior eficiência do Windows na comutação de tarefas e/ou no tempo de alocação das mesmas.

Por fim, uma característica desafiadora foi observada nas funções de mínimos geradas no Windows: embora as funções de médias apresentem-se aproximadamente lineares, as funções de mínimos têm aparência semelhante a "escadas".

Uma explicação para isso pode estar no fato de que as atualizações no relógio disponível para o interpretador Java são menos freqüentes no Windows, o que faz com que o tempo de execução obtido nos testes tenha mais variações que o tempo de execução real, já que o tempo, em nossos testes, é obtido dentro do programa. Assim, a média se preserva, aproximadamente, mas os mínimos são afetados.

Explicando de maneira mais informal: se o relógio é atualizado com menor freqüência, os tempos de execução calculados serão múltiplos do período entre uma atualização e outra. Assim, alguns valores acabam sendo "arredondados" para cima e outros para baixo.

Os aumentos bruscos no valor da função (que dão a aparência de escada) ocorrem, provavelmente, porque, nos trechos em que o tempo mínimo calculado aparece constante (digamos *min*) no gráfico, o tempo mínimo real está aumentando, de forma que, em algum momento, nenhum tempo obtido será mais "arredondado" para *min*.