Trabalho Prático 3: NuCache

Algoritmos e Estruturas de Dados III – 2017/2
Entrega: 03/12/2017

1 Introdução

Como todos sabem, o passa tempo favorito de Nubby é realizar operações em blocos de memória alocados sequencialmente. Recentemente Nubby soube que você está aprendendo sobre localidade de referência. Como Nubby acredita que este conteúdo pode ser útil para melhorar a eficiência dos algoritmos ele lhe propôs um novo método para multiplicação de matrizes.

Uma versão do algoritmo para multiplicação de matrizes quadradas (Algoritmo 1), recebe duas matrizes a serem multiplicadas $A \in \mathbb{Z}^{N \times N}$ e $B \in \mathbb{Z}^{N \times N}$ e uma terceira matriz $C \in \mathbb{Z}^{N \times N}$ que armazenará o resultado da multiplicação de A por B ou seja $C = AB$. O algoritmo consiste em realizar o produto interno das linhas de A pelas colunas de B de modo que $C_{ij} = \sum_{k=1}^{N}(A_{i,k} \cdot B_{k,j})$.

O novo algoritmo proposto por Nubby descrito em (Algoritmo 2), utiliza um conceito chamado de "blocking" o qual consiste em organizar as estruturas de dados abstratamente em grandes pedaços (blocos). Desse modo, quando necessário um bloco é carregado no cache L1, realiza-se todas as operações necessárias, descarta o bloco e carrega o próximo bloco e assim por diante. A ideia do algoritmo proposto é definir um tamanho de bloco (bsize) a ser carregado na cache, particionar as linhas das matrizes A e C em pedaços de tamanho $1 \times bsize$ e particionar B em blocos de tamanho $bsize \times bsize$. O loop interno (j, k) multiplica o pedaço de A por um bloco de B e acumula o resultado no respectivo pedaço de C. O loop (i) itera sobre as n linhas de A e C usando o mesmo bloco de B. Observe que esta estratégia explora o fato de que uma matriz pode ser particionada em submatrizes e estas submatrizes podem ser manipuladas como escalares.

Cabe a você a tarefa de analisar a eficiência do algoritmo proposto. Para cada uma das três matrizes A, B e C faça uma análise teórica da localidade espacial e temporal (Ex: se a localidade espacial da matriz A é boa ou ruim). Compare os dois algoritmos propostos utilizando as métricas de tempo de execução e taxa de cache miss variando o tamanho da matriz e tamanho do

bloco. Para simplificar a sua análise assuma que o tamanho do bloco seja divisor do número de linhas da matriz e que variáveis locais são armazenadas em registradores de modo que não seja necessário instrução de leitura ou escrita para acessar estas variáveis e assuma que a matriz sempre caiba na memória secundária (RAM) e que o bloco sempre caiba na primária (cache) (Em outras palavras que o tamanho da cache seja suficiente para armezenar os blocos que estão sendo multiplicados $|cache| = bsize^2 + 2 * bsize$).

Algorithm 1 Calcula C = AB sem blocos

```
for i = 0; i < N; i++ do
  for j = 0; j < N; j++ do
    C[i][j] ← 0
    for k = 0; k < N; k++ do
      C[i][j] ← C[i][j] + A[i][k] * B[k][j]
    end for
  end for
end for
```

Algorithm 2 Calcula C = AB com blocos

```
for i = 0; i < N; i++ do
  for j = 0; j < N; j++ do
    C[i][j] ← 0
  end for
end for
for kk = 0; kk < N; kk+ = bsize do
  for jj = 0; jj < N; jj+ = bsize do
    for i = 0; i < N; i++ do
      for j = jj; j < jj + bsize; j++ do
        sum ← C[i][j]
        for k = kk; k < kk + bsize; k++ do
          sum ← sum + A[i][k] * B[k][j]
        end for
        C[i][j] ← sum
      end for
    end for
  end for
end for
```

2 Entrada e saída

Entrada A primeira linha contém dois inteiros positivos: $N(1 \leq N \leq 2000)$ indicando o número de linhas e colunas das matrizes A, B (todas as matrizes são quadradas) a serem multiplicadas e $bsize$ ($1 \leq bsize \leq 50$, $N\%bsize = 0$) indicando tamanho do bloco a ser utilizado. A seguir, seguirão N linhas cada uma com N elementos inteiros contendo os elementos da matriz A. Similarmente, seguirão N linhas, cada uma com N elementos inteiros contendo os elementos da matriz B.

Exemplo de entrada

```
4 2
1 2 3 4
5 6 7 8
9 10 11 12
13 14 15 16
1 2 2 1
1 2 2 1
1 2 2 1
1 2 2 1
```

Saída A saída deve conter os valores da matriz $C = AB$ impressos em N linhas com N elementos em cada linha. **Exemplo de saída**

```
10 20 20 10
26 52 52 26
42 84 84 42
58 116 116 58
```

3 O que deve ser entregue

Deverá ser submetido um arquivo **.zip** contendo apenas uma pasta chamada tp3, esta pasta deverá conter: (i) Documentação **em formato PDF** e (ii) Implementação.

Documentação Deverá ter no máximo 10 páginas e seguir tanto os critérios de avaliação discutidos na Seção 4.1, bem como as diretrizes sobre a elaboração de documentações disponibilizadas no *moodle*. Além disso, a documentação deverá conter análise teórica das complexidades de tempo e espaço dos algoritmos, análise experimental validando a análise teórica e comparação

entre os resultados obtidos utilizando o algoritmo sem utilizar blocos e utilizando blocos.

Implementação O código fonte do seu TP deve estar separado em arquivos .c e .h.

Makefile Inclua um *makefile* na submissão que permita compilar o trabalho. **Deverão ser produzido um executável** chamado *block*.

É obrigatório o uso das *flags*: **-Wall -Wextra -Werror -std=c99 -pedantic -O2**. **Utilize estas flags para rodar os seus experimentos**

Execução A entrada e a saída devem ser **lida e impressa utilizando a entrada/saída padrão (stdin/stdout)**. Um exemplo de execução utilizando o executável *block* para um arquivo de entrada *in_1.in* e imprimindo o resultado em *out_1.out* será da seguinte forma:

$$./block < in\_1.in > out\_1.out$$

4 Avaliação

Uma lista **não exaustiva** dos critérios de avaliação que serão utilizados é apresentada a seguir.

4.1 Documentação

Introdução Inclua uma breve explicação do problema que está sendo resolvido no seu trabalho.

Solução do Problema Você deve descrever cada solução do problema de maneira clara e precisa, detalhando e justificando os algoritmos e estruturas de dados utilizados. Para tal, artifícios como pseudo-códigos, exemplos ou diagramas podem ser úteis. Note que documentar uma solução não é o mesmo que documentar seu código. **Não** é necessário incluir trechos de código em sua documentação nem mostrar detalhes de sua implementação, exceto quando estes influenciem o seu algoritmo principal, o que torna o detalhamento interessante.

Análise de Complexidade Inclua uma análise de complexidade de tempo e espaço dos principais algoritmos e estrutura de dados utilizados. Cada complexidade apresentada deverá ser devidamente **justificada** para que seja aceita. Faça uma análise para cada tipo de algoritmo implementado. Mostre vantagens e desvantagens de cada um deles.

Avaliação Experimental Sua documentação deve incluir os resultados de experimentos que avaliem o tempo de execução de seu código em função de características da entrada. Cabe a você gerar entradas para esses experimentos. Para tal, um gráfico mostrando o tempo de execução em função do tamanho da entrada pode ser interessante. Você também deve interpretar os resultados obtidos. Comente sobre cada gráfico ou tabela que você apresentar mostrando o que é possível concluir a partir dele.

Para cada experimento sobre um variável aleatória execute pelo menos 30 vezes e obtenha a média dos resultados obtidos como uma estimativa do valor real. Inclua todos os logs dos seus experimentos em um apêndice de forma que qualquer possoa seja capaz de reproduzir os seus resultados através dos logs. O apêndice pode possuir até 10 páginas além das 10 páginas da documentação.

4.2 Implementação

Linguagem & Ambiente O seu programa deverá ser implementado na linguagem **C** e poderá fazer uso de funções da biblioteca padrão da linguagem. Trabalhos que utilizem qualquer outra linguagem de programação ou utilizem outras bibliotecas que não seja padrão ou variáveis globais serão zerados. Além disso, é recomendado que se certifique que seu código compile e funcione corretamente nas máquinas **Linux** dos laboratórios do DCC.

Casos de teste A sua implementação passará por um processo de correção automatizado, portanto, o formato da saída do seu programa deve ser idêntico aquele descrito nessa especificação. Saídas com qualquer divergência serão consideradas erradas, mesmo que as divergências sejam *whitespaces*. e.g. espaços, *tabs*, quebras de linha, etc. Para auxilia-lo na depuração do seu código, será fornecido um pequeno, **não-exaustivo**, conjunto de entradas e suas respectivas saídas. É seu dever certificar-se que seu código funciona corretamente para qualquer entrada válida.

Alocação Dinâmica Algoritmos e estruturas de dados deverão fazer uso de memória alocada dinamicamente (`malloc()` ou `calloc()`). Certifique-

se que seu programa utiliza essas regiões de memória corretamente, pois os monitores penalizarão implementações que realizam *out-of-bounds access* e que tenham vazamento de memória (não desalocar memória dinâmica).
A alocação dinâmica deverá fazer uso das funções malloc() ou calloc() da biblioteca padrão C, bem como liberar tudo o que for alocado utilizando free(), para gerenciar o uso da memória. **DICA:** Utilize `valgrind` antes de submeter o seu TP.

Qualidade do código Seu código também será avaliado no quesito de legibilidade, dando atenção, porém não limitando-se, aos seguintes items: (i) **INDENTAÇÃO**; (ii) nomes de variável e função descritivos e claros; (iii) Modularização adequada; (iv) Comentários dentro de funções, explicando o que certos trechos mais complicados fazem; (v) Comentários fora de funções, explicando, em alto-nível, o que as funções mais importantes fazem; (vi) funções concisas que desempenham somente uma tarefa; (vii) **Proibido uso de variáveis globais**.

Atrasos **Não existe política de atraso para este trabalho. Não será aceito submissão após a data limite.**

5 Considerações Finais

Assim como em todos os trabalhos dessa disciplina **é estritamente proibida a cópia parcial ou integra de códigos, seja da internet ou de colegas**. Utilizaremos o algoritmo *MOSS* para detecção de plágio em trabalhos, seja honesto. Você não aprende nada copiando código de terceiros nem pedindo a outra pessoa que faça o trabalho por você. Se a cópia for detectada, sua nota será zerada e os professores serão informados para que as devidas providências sejam tomadas.