INTRODUCÃO A COMPUTAÇÃO DE ALTO DESEMPENHO
TRABALHO PRÁTICO 1
OTIMIZAÇÃO

DATA DE ENTREGA: 06/06/2016.

INSTRUÇÕES:

1. Deve ser elaborado um relatório contendo os resultados e discussões dos exercícios abaixo. Quando for o caso, o código-fonte também ser ser enviado. O relatório deve ser salvo no formato PDF. O relatório tem peso predominante na nota final de trabalho (60%). Assim, produza um relatório de qualidade!!
2. A entrega será pelo email: camata@nacad.ufrj.br.
3. Serão aceitos trabalhos enviados até às 23:59 do dia da entrega.
4. **ATENCÃO:** No assunto da mensagem, use a seguinte regra:
    a. CAD-2016 - TRAB1 - NOME COMPLETO
5. Trabalhos “similares” podem ser penalizados.

Exercício 1: Medindo a taxa de transferência da mémoria in MB/s.

Em computação, benchmarking é o ato de executar um programa de computador, um conjunto de programas ou outras operações, a fim de avaliar o desempenho relativo de um objeto, normalmente executando uma série de testes padrões e ensaios nele. Nesse exercício, gostariamos de obter a taxa de transferência de memória de uma determinada máquina usando o programa de benchmark STREAM.

O benchmark STREAM é um programa de benchmark sintético, escrito em Fortran 77 (com uma versão correspondente em C) que mede o desempenho de quatro operações com vetores descritas abaixo:

		(por iteracão)	
Nome	Operacão	Bytes	FLOPS
COPY	*a(i) = b(i)*	16	0
SCALE	*a(i) = q*b(i)*	16	1
SUM	*a(i) = b(i) + c(i)*	24	1
TRIAD	*a(i) = b(i) + q*c(i)*	24	2

Cada uma dessas quatro operações fornece informações independentes para os resultados:

- COPY: mede a taxa de transferência na ausência de operacões aritméticas.
- SCALE: adiciona uma operacão aritmetica simples
- SUM: adiciona um terceiro operando
- TRIAD: permite operacão do tipo FMA.

O código do programa está em anexo e sua compilação é extremamente simples:

```
gcc -O -DSTREAM_ARRAY_SIZE=<N> stream.c -o stream
```

onde <N> deve substituido pelo tamanho do array. Por exemplo, para executar as operacões com vetores de tamanho 1000, deve-se usar a seguinte linha para a compilacão:

```
gcc -O -DSTREAM_ARRAY_SIZE=1000 stream.c -o stream
```

Deseja-se:

1. Medir a taxa de transferência de uma determinada máquina. Para tanto, execute o programa STREAM com diferentes tamanhos <N>.
2. Gere um gráfico <N> x MB/s para cada uma das operações do benchmark.
3. Obtenha a configuração de hardware da máquina onde os testes foram realizados.
4. Coloque no relatório os resultados obtidos.

Mais informacões, consulte:
STREAM webpage: http://www.cs.virginia.edu/stream/

Observacão: Quando uma operacão não involve instrucões aritméticas, a taxa de transferência (MB/s) pode ser usada como métrica de desempenho ao invés da taxa de operacões com ponto-flutuante por segundo (FLOPS).

Exercício 2: Trabalhando com perfiladores

O objetivo desse exercício é demonstrar o funcionamento de dois principais perfiladores de códigos: PAPI e HPCToolkit. A seguir é detalhado o processo de instalação e utilização dessas duas ferramentas.

- Instalando o PAPI:
  - Faça o download do código fonte: http://icl.cs.utk.edu/papi/software/index.html
  - Descompacte-o e siga o processo padrão de compilação em ambientes unix/linux.
    - `./configure --prefix=$HOME/local/papi`
    - `make`
    - `make install`
  - Provavelmente será necessário modificar as variáveis de ambiente PATH e LD_LIBRARY_PATH:
    - `export PATH=$PATH:$HOME/local/papi/bin`
    - `export LD_LIBRARY_PATH=$LD_LIBRARY_PATH:$HOME/local/papi/lib`
  - Abre um novo terminal e teste a instalacão com os comandos:
    - papi_avail
    - Papi_mem_info
- Instalando o HPCToolKit:
  - A instalação é realizada em três etapas. Veja o site com os pacotes necessários:
    - http://hpctoolkit.org/software.html
  - Etapa 1: Instalando softwares externos necessários ao hpctoolkit:
    - Hpctoolkit-externals: Segue a instruções e instale em `$HOME/local/hpctools-external`
  - Etapa 2: Instalando o hpctoolkit
    - Segue as instruções do site. Observe que na configuracão (./configure) deve-se indicar o diretório onde os pacotes externos foram instalados na etapa anterior. Instale o hpctoolkit em `$HOME/local/hpctoolkit`.
  - Etapa 3: Instalando o HPCViewer:
    - Basta fazer o donwload, descompactar e clicar no executável *hpcviewer*.

Se tudo sair bem você já estará apto em perfilar um código. Para o primeiro teste, vamos perfilar o programa STREAM. Siga os passos descritos abaixo:

1. Compilando o STREAM com a opcão -g
   a. `gcc -g -O -DSTREAM_ARRAY_SIZE=<N> stream.c -o stream`

2. Rodando com o HPCToolkit
    a. `hpcrun ./stream`
    b. `hpcstruct ./stream`

3. Gerando relatório de perfilagem
    a. `hpcprof -S stream.hpcstruct -I* hpctoolkit-stream-measurements`

4. Abrindo o hpcviewer selecionado o diretório hpctoolkit-stream-database-* o qual foi criado na etapa anterior.

Identifique qual(is) linha(s) do código do programa STREAM gasta(m) o(s) maior(es) tempo de CPU? Qual a taxa de erro de cache (L1 e/ou L2) para cada uma das operacões do benchmark para diferentes <N>? Coloque os resultados obtidos no relatório. Compacte o diretório database criado pelo hpctoolkit para também ser enviado ao professor.

Exercício 3: Operacões com matrizes

Programe uma subrotina que multiplique duas matrizes A e B quadradas de ordem n. Sua primeira implementação deve considerar a ordem dos laços como apresentado abaixo. Considere diferentes tamanhos de n (n=1000, 2000, 4000, etc);

```
...
for (j=0; j < n; j++)       // Loop J
   for(i=0; i < n;i++)      // Loop I
      for(k=0; k < n; k++) // Loop K
          C[i][j] = A[i][k]*B[k][j];
...
```

- Compile com diferentes níveis de otimização. Meça os tempos para cada um deles. Reporte os resultados.
- Implemente uma versão do código com os laços I e J invertidos. Ou seja, primeiro o laço i e depois o j. Novamente, meça o tempo de processamento usando a rotina time. Utilize o HPCToolkit e os eventos do PAPI para detectar os gargalos e as razões para isso.
- Escolha a melhor implementacão acima, altere-a utilizando blocagem dos dados. Encontre um tamanho de bloco que melhor aproveite a hierarquia de memória.

Exercício 4: Otimizando um código científico.

A propagação da onda acustica em meio isotrópico é dada pela seguinte equação diferencial parcial:

$$\frac{\partial^2 p}{\partial t^2} = c^2 \nabla^2 p$$

Onde $\nabla^2 p$ é o operador laplaciano, *p* é o campo de pressão que se deseja obter e c o campo de velocidade de propagacão da onda. Desenvolvendo a equacão acima, chega-se:

$$\frac{\partial^2 p}{\partial t^2} = c^2 \left(\frac{\partial^2 p}{\partial x^2} + \frac{\partial^2 p}{\partial y^2} + \frac{\partial^2 p}{\partial z^2} \right).$$

Cuja formula discreta por diferencas finitas é dada por:

$$\frac{p_{t+1}[x,y,z] - 2p_t[x,y,z] + p_{t-1}[x,y,z]}{\Delta t^2} = c^2 (FD_x + FD_y + FD_z)$$

onde

$$FD_x = \frac{1}{\Delta x^2} \left(c_0 p_t[x,y,z] + \sum_{i=1}^{n} c_i (p_t[x+i,y,z] + p_t[x-i,y,z]) \right)$$

$$FD_y = \frac{1}{\Delta y^2} \left(c_0 p_t[x,y,z] + \sum_{i=1}^{n} c_i (p_t[x,y+i,z] + p_t[x,y-i,z]) \right)$$

$$FD_z = \frac{1}{\Delta z^2} \left(c_0 p_t[x,y,z] + \sum_{i=1}^{n} c_i (p_t[x,y,z+i] + p_t[x,y,z-i]) \right)$$

Aqui $c_0, c_1, \ldots, c_n$ são constantes referente as método das diferencas finitas.

Então, o campo de pressão em um próximo passo de tempo pode ser obtido a partir do campo presente e do passo anterior como se segue:

$$p_{t+1}[x,y,z] = 2p_t[x,y,z] - p_{t-1}[x,y,z] + \Delta t^2 . c^2 (FD_x + FD_y + FD_z)$$

Em wave.tar.gz está o código-fonte (escrito em C) que implementa a equacões acima. O código está dividido em três partes:

1. Alocacão e inicializacão:
    a. Aloca os vetores e fornece um impulso inicial.
2. Cálculo da equacão da propagacão da onda
    a. Partindo do impulso inicial em t0, calcula a propagacao da onda nos instantes t1, t2,..., tmax.
    b. Escreve resultados a cada 10 instantes de t.
3. Desalocacão.

Para compilá-lo basta chamar o comando make (via linha de comando) dentro do diretorio contendo o código-fonte.

Os resultados podem ser visualizados no *gnuplot*. Serão gerados sempre arquivos *wave_****.dat* e *wave_*****.plot*. No primeiro, encontra-se a solucão do problema em determinados pontos. O segundo é um script gnuplot que gera uma imagem png dos dados contidos no arquivo .dat. Para gerar esse gráfico, basta fazer, por exemplo,

```
gnuplot wave_0000.plot
```

A partir desse código, vocês deverão perfilá-lo e encontrar os gargalos computacionais. Além disso, vocês devem propor otimizacões usando as métricas obtidas pelo perfilador para demonstrar os ganhos de desempenho. Todo o processo de otimizacão deve estar presente no relatório.

Observação: A solução numérica é obtida em uma malha com 256 x 256 x 256 pontos. Melhor precisão dos resultados podem ser obtidos com malhas maiores, tipo: 512 x 512 x 512.