Aberto: quarta-feira, 4 out. 2023, 15:00
Vencimento: sábado, 28 out. 2023, 23:59

EP-04 - Otimização de Código Serial

O objetivo deste exercício é melhorar a performance de funções de multiplicação de matrizes.

Você deverá usar melhorar a performance das operações abaixo, usando as técnicas vistas em aula, inclusive (mas não restritos a) *unroll & jam* + *blocking* e usando a ferramenta **LIKWID** para comparar as performances e tempos com e sem otimização:

1. `multMatVet()` --> multiplica uma matriz tipo **`MatRow`** por um vetor
2. `multMatMat()` --> multiplica duas matrizes de tipo **`MatRow`**

São definidos 2 tipos abstratos de dados (vide arquivo **matriz.h**):

- `MatRow`: tipo para representar uma matriz implementada como um único vetor cujo conteúdo são as linhas da matriz, em sequencia;
- `Vetor`: tipo para representar um vetor simples.

O fator de *unroll* **UF** e o fator de *blocking* **BK** devem ser definidos via macros em linguagem C no arquivo **matriz.h** .

Análise de Desempenho

Para analisar o desempenho das funções, você deve efetuar uma série de testes:

- **Cada teste** deve ser reportado sob a forma de um gráfico, onde a abcissa é o tamanho da matriz e ordenada é o valor dos indicadores abaixo.
- Cada teste deve ser executado para os seguintes tamanhos de matriz: **N**={64, 100, 128, 200, 256, 512, 600, 900, 1024, 2000, 2048, 3000, 4000};

Os seguintes testes devem ser executados para **cada função** de multiplicação (uma tabela para cada teste de cada função):

- **Teste de tempo:** mostra o **tempo médio** do cálculo da função, **em milisegundos** (utilize a função "timestamp()" para medir o tempo);
- **Banda de Memória:** utilizar o grupo **MEM** ou **L3** do LIKWID, e apresentar o resultado de "Memory bandwidth [MBytes/s]";
- **Cache miss L2:** utilizar o grupo **CACHE** ou **L2CACHE** do LIKWID, e apresentar o resultado de "**cache miss RATIO**";
- **Energia:** utilizar o grupo **ENERGY** do LIKWID, e apresentar o resultado de "**Energy [J]**";
- **Operações aritméticas:** utilizar o grupo **FLOPS_DP** do LIKWID e reportar **FLOPS_DP** e **`S_AVX** como duas colunas separadas na **mesma tabela**, em "MFLOP/s"

É imprescindível que sejam respeitadas as seguintes condições:

- Os códigos devem ser compilados com GCC e as opções: **-O3 -mavx2 -march=native**
- Os testes devem utilizar os mesmos parâmetros e em igualdade de condições;
- O código deve ser instrumentado com a biblioteca do **LIKWID** para que se possa separar os contadores de cada função.
- Você deve documentar a arquitetura do processador utilizado nos testes.
  Estas informações podem ser obtidas com o comando "**likwid-topology -g -c**".
- Os códigos devem ser compilados na mesma máquina utilizada para os testes;
  - Utilize seu computador pessoal com Linux *standalone,* ou, se isto não é possível, utilize remotamente os computadores do laboratório. NÃO É POSSÌVEL usar o LIKWID em máquinas virtuais.
  - Para acessar remotamente as máquinas dos laboratórios , deve-se executar
    ssh <login_DINF>@macalan.c3sl.ufpr.br e depois,
    a partir deste host, executar **ssh <maq>** , onde **<maq>** = {hxxn ixx, jxx, conforme o laboratório}.
  - Antes de rodar experimentos, é necessário FIXAR a frequencia do processador. Para isto, execute a seguinte linha de comando:

```
echo "performance" > /sys/devices/system/cpu/cpufreq/policy3/scaling_governor
```

- Utilizar o core de maior ordem de seu processador. Por exemplo, nos laboratórios do DINF, use sempre o **CORE 3** na hora de executar os experimentos:

  ```
  likwid-perfctr -C 3 -g ...
  ```

- Após os experimentos, retorne o computador à frequencia original

  ```
  echo "powersave" > /sys/devices/system/cpu/cpufreq/policy3/scaling_governor
  ```

O que deve ser entregue

Este exercício deve ser feito em **duplas** (preferencialmente, a dupla deve continuar a mesma para todos os demais EPs e Trabalhos práticos da disciplina). Apenas um dos alunos deve entregar os códigos-fonte do programa (em linguagem C), makefile e arquivos de teste usados, além de um arquivo **LEIAME** contendo os nomes dos alunos, limitações do programa (por exemplo, casos que o programa não funciona) e otimizações feitas.

O pacote deve ser arquivado e compactado com **zip** ou **tar**, em um arquivo chamado ***login1-login2.<ext>***, onde **login1** e **login2** são os logins (nos sistemas do DINF) dos alunos que compõem o grupo, e **<ext>** é **.tar**, ou **.zip**, ou **.tar.gz**, conforme o comando usado para arquivar o pacote.

O pacote deve ter a seguinte estrutura de diretório e arquivos:

- *./login1-login2/*: diretório principal;
- *./login1-login2/LEIAME*;
- *./login1-login2/Makefile;*
- *./login1-login2/*.c*
- *./login1-login2/*.h*

Note que a extração dos arquivos de ***login1-login2.<ext>*** deve criar o diretório ***login1-login2*** contendo todos os arquivo acima. Os arquivos fonte também devem estar contidos no diretório, ou em algum sub-diretório, desde que o *Makefile* funcione.

Acesso aos arquivos

Os arquivos para este exercício encontram-se no git abaixo. As implementações das funções indicadas no enunciado **não estão** otimizadas.

https://gitlab.c3sl.ufpr.br/nicolui/ci1164_2023-otimiz.git

ou via linha de comando:

git clone git@gitlab.c3sl.ufpr.br/nicolui/ci1164_2023-otimiz.git

Para implementação de funções auxiliares (como **timestamp()**), pode-se usar o git abaixo (pastas **utils** e **gnuplot**):

https://gitlab.c3sl.ufpr.br/nicolui/ci1164-utils

Adicionar envio

Status de envio

Número da tentativa	Esta é a tentativa 1 .
Status de envio	Nenhum envio foi feito ainda
Status da avaliação	Não há notas

Tempo restante	16 dias 11 horas restando
Última modificação	-
Comentários sobre o envio	▸ Comentários (0)