Universidade Federal do Paraná – UFPR

Setor de Ciências Exatas Departamento de Informática - DInf

Disciplina: ci164 - Introdução à Computação Científica

Rubens Zandomenighi Laszlo- GRR20206147

**Relatório do trabalho prático de Otimização do Desempenho para Sistemas Lineares Esparsos com Pré-Condicionador de Jacobi utilizando Matrizes Esparsas**

**Introdução**

Otimização dos seguintes métodos efetuado no trabalho 1.

- A operação op1 de iteração do método de Gradiente Conjugado com Pré-condicionador de Jacobi;

- A operação op2 de cálculo do resíduo a partir da matriz de coeficientes e vetor de termos independentes originais;

Os resultado foram gravados em pastas conforme a versão do trabalho, sendo

./v1/outputs : Pasta com outputs de execução do programa versão 1.

./v1/logs: Pasta com outputs de execução do likwid da versão 1.

./v2/outputs : Pasta com outputs de execução do programa versão 2.

./v2/logs: Pasta com outputs de execução do likwid da versão 2.

**Topologia**

Para o desenvolvimento do trabalho, foi utilizada a máquina DInf. As especificações da mesma obtidas por $ likwid-topology -c -g são:

--------------------------------------------------------------------------------

CPU name: Intel(R) Core(TM) i5-7500 CPU @ 3.40GHz

CPU type: Intel Coffeelake processor

CPU stepping: 9

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Hardware Thread Topology

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Sockets: 1

Cores per socket: 4

Threads per core: 1

--------------------------------------------------------------------------------

HWThread Thread Core Socket Available

0 0 0 0 \*

1 0 1 0 \*

2 0 2 0 \*

3 0 3 0 \*

--------------------------------------------------------------------------------

Socket 0: ( 0 1 2 3 )

--------------------------------------------------------------------------------

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Cache Topology

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Level: 1

Size: 32 kB

Type: Data cache

Associativity: 8

Number of sets: 64

Cache line size: 64

Cache type: Non Inclusive

Shared by threads: 1

Cache groups: ( 0 ) ( 1 ) ( 2 ) ( 3 )

--------------------------------------------------------------------------------

Level: 2

Size: 256 kB

Type: Unified cache

Associativity: 4

Number of sets: 1024

Cache line size: 64

Cache type: Non Inclusive

Shared by threads: 1

Cache groups: ( 0 ) ( 1 ) ( 2 ) ( 3 )

--------------------------------------------------------------------------------

Level: 3

Size: 6 MB

Type: Unified cache

Associativity: 12

Number of sets: 8192

Cache line size: 64

Cache type: Inclusive

Shared by threads: 4

Cache groups: ( 0 1 2 3 )

--------------------------------------------------------------------------------

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

NUMA Topology

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

NUMA domains: 1

--------------------------------------------------------------------------------

Domain: 0

Processors: ( 0 1 2 3 )

Distances: 10

Free memory: 3610.48 MB

Total memory: 7834.9 MB

--------------------------------------------------------------------------------

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Graphical Topology

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Socket 0:

+---------------------------------------------+

| +--------+ +--------+ +--------+ +--------+ |

| | 0 | | 1 | | 2 | | 3 | |

| +--------+ +--------+ +--------+ +--------+ |

| +--------+ +--------+ +--------+ +--------+ |

| | 32 kB | | 32 kB | | 32 kB | | 32 kB | |

| +--------+ +--------+ +--------+ +--------+ |

| +--------+ +--------+ +--------+ +--------+ |

| | 256 kB | | 256 kB | | 256 kB | | 256 kB | |

| +--------+ +--------+ +--------+ +--------+ |

| +-----------------------------------------+ |

| | 6 MB | |

| +-----------------------------------------+ |

+---------------------------------------------+

**Principais alterações**

Estrutura:  
Alteração da estrutura do Sistema Linear para o armazenamento apenas das diagonais não nulas, sendo que foi feito o armazenamento linha por linha, para aproveitamento da localidade espacial no acesso à memória.

Código:  
Alteração do código da OP2 para minimização do custo de functions calls e facilitação de Loop fusion entre as rotinas chamadas na iteração, além do uso de variáveis auxiliares para rotinas semelhantes.   
Utilização de Unroll and Jam e Loop Fusion nos laços mais críticos dentro das operações em que se buscava ganho de desempenho. Sendo utilizado fator de UNROLL como 7, já que assim eram utilizadas as linhas da cache sem exceder o seu limite, otimizando o uso de linhas de cache.   
Memória:   
Utilização da função aligned\_alloc() para alocação da memória com alinhamento, para aproveitamento, sendo utilizado alinhamento de 16 para utilizar endereços de memória divisíveis por 16, assim beneficiando o uso de instruções SIMD.  
Para a facilitação da utilização de instruções SIMD, foi utilizado a palavra chave restrict, assim informando o compilador que não existe dependência de dados, em funções em que parâmetros utilizam ponteiros para o mesmo tipo de dados.  
Utilização da função memset e memcpy para inicialização das matrizes quando necessário para otimizar o custo de acesso a memória.

**Resultados Obtidos**

- Tempo de Execução:

Foi observada uma pequena melhora no tempo, principalmente no cálculo do resíduo. Contudo, o ganho não foi significativo. As estratégias de Unroll and Jam e de Loop Fusion não surtiram tanto efeito quanto era esperado. Podemos observar isso no gráfico abaixo

-Banda de Memória:

A banda de memória medida em L3 foi consistentemente maior em v2 em relação a v1, conforme era esperado pela otimização do uso dos dados. Desde o início, foi buscado um uso eficiente dos dados, evitando assim acessos desnecessários à memória. Segue o gráfico abaixo:

- Desempenho em MFLOP/s

Esse foi o setor de maior melhora na otimização. Os resultados em v2 foram extremamente melhores do que os vistos em v1. Isso indica que as estratégias de Unroll and Jam e de Loop Fusion surtiram um efeito enorme em um ganho de desempenho em operações de pontos flutuantes double. Outro aspecto importante foi o alinhamento da memória com aligned\_alloc( ) , o que possibilita vetorização e uso de instruções SIMD por parte do compilador. O ganho de desempenho pode ser visto abaixo no gráfico