

Roofline



UNIVERSIDADE FEDERAL
DO RIO DE JANEIRO

- Pedro Tubenchlak Boechat - DRE 119065050
– `pedroboechat@poli.ufrj.br`

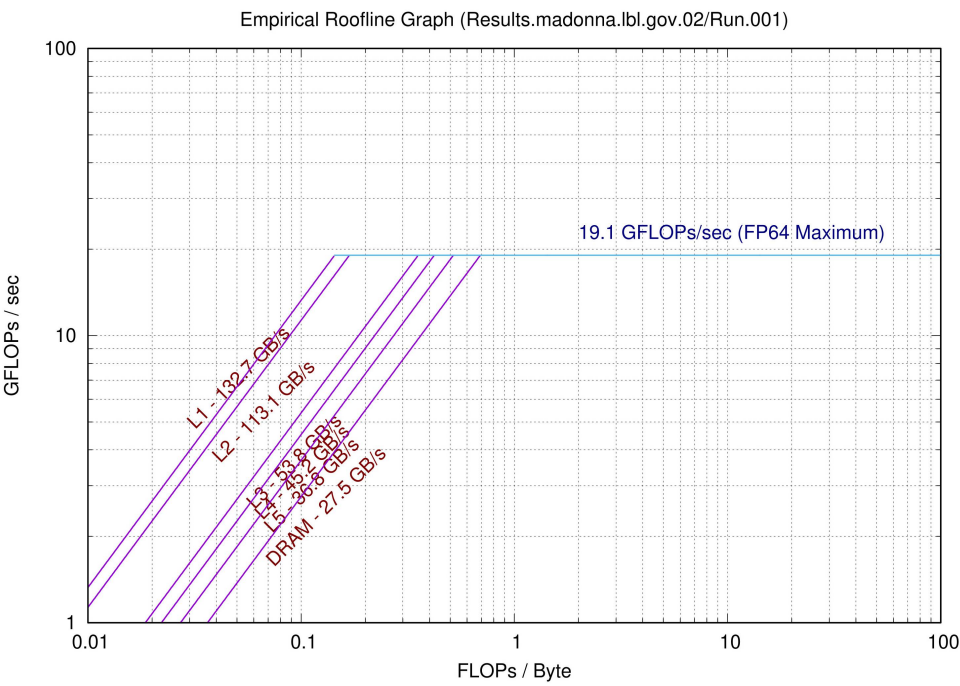
O código-fonte desse trabalho está disponível no meu GitHub
(<https://github.com/pedroboechat/UFRJ/tree/main/COC472/Roofline>).

Introdução

Para esse trabalho foi utilizado o código *laplace.cxx* com os valores fixos dentro do código, ao invés de receber pelo *stdin*.

Roofline do sistema

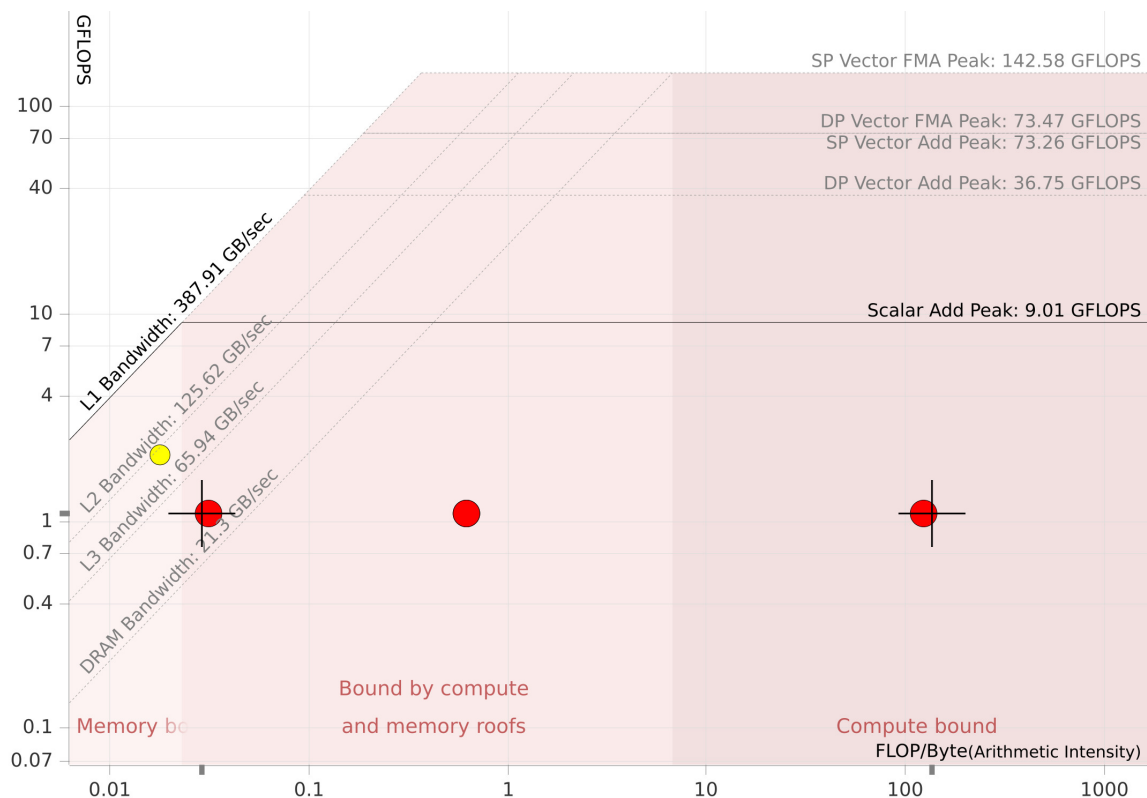
Utilizando o Empirical Roofline Tool (ERT), foi calculado o roofline do sistema composto por uma CPU Intel i7-10700 e memória RAM DDR4 de frequência 2666 MHz. Os resultados mostram que o roofline é de 19.1 GFLOPs/s.



Desempenho do código

Utilizando a ferramenta Intel Advisor do pacote OneAPI, foi calculado o desempenho do código *laplace.cxx*, cujo resultado foi:

▶ GINTOPS: ⓘ	3,58	32% of 10,92 Integer GINTOPS
▼ GFLOPS: ⓘ	1,09	12% of 9,01 GFLOPS
GFLOP: ⓘ	0,27	
▼ CARM (L1 + NTS) Bandwidth: ⓘ	37,39	9% of 387,91 GB/sec
Traffic: ⓘ	9,33	
FP Arithmetic Intensity: ⓘ	0,03	
INT Arithmetic Intensity: ⓘ	0,10	
▶ L2 Bandwidth: ⓘ	1,61	1% of 125,62 GB/sec
▶ L3 Bandwidth: ⓘ	1,60	2% of 65,94 GB/sec
▼ DRAM Bandwidth: ⓘ	0,01	0% of 21,30 GB/sec
Traffic: ⓘ	0,00	
FP Arithmetic Intensity: ⓘ	136,02	
INT Arithmetic Intensity: ⓘ	445,19	



As cruzes no gráfico representam, da esquerda para a direita, respectivamente, a performance da L1 e da DRAM. Dessa forma é possível identificar a intensidade aritmética para ambas:

CARM (L1 + NTS) Program Total for All Functions and Loops
 Performance: **1.09 GFLOPS**
 Arithmetic Intensity: **0.029 FLOP/Byte**

DRAM Program Total for All Functions and Loops
 Performance: **1.09 GFLOPS**
 Arithmetic Intensity: **136.02 FLOP/Byte**

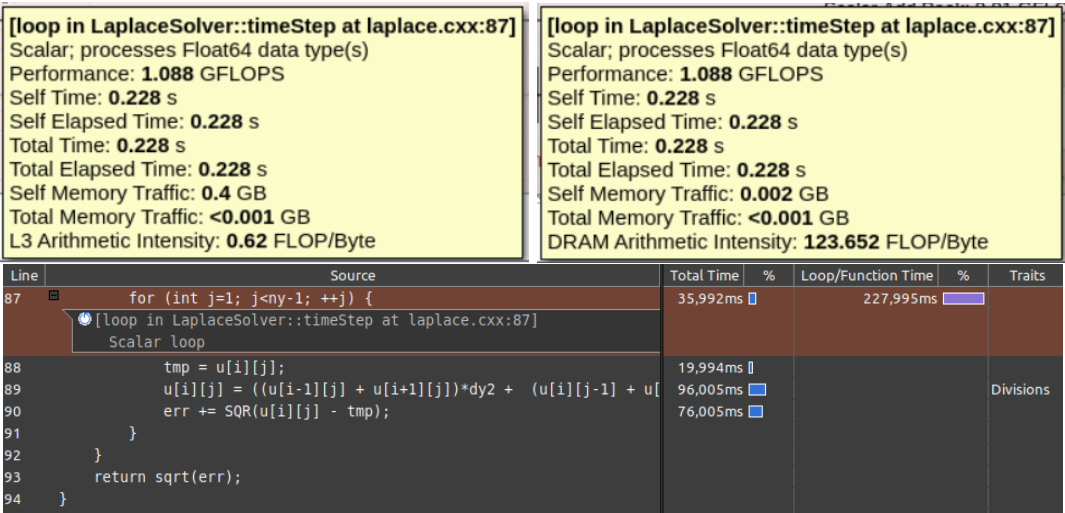
Também é possível identificar dois hotspots nas funções *LaplaceSolver::timeStep* (vermelhos) e *SQR* (amarelo), as mesmas identificadas anteriormente pelo *Gproof* (Trabalho 2). Analisando essas funções individualmente, temos que:

- *LaplaceSolver::timeStep*

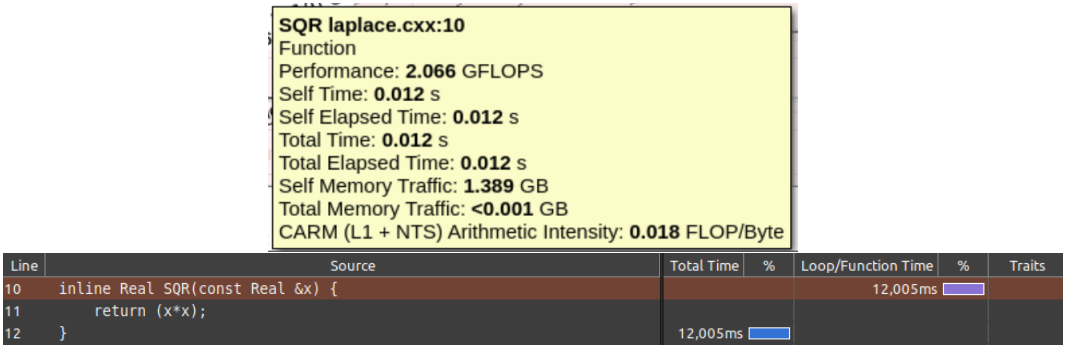
Esta função é um hotspot em L1, L2, L3 e DRAM (L2 e L3 estão sobrepostos). Os maiores tempos de execução são encontrados na linha 89 e 90. Na linha 89, como também identificado no Trabalho 2, há uma multiplicação por constante que inclui a operação de divisão, como corretamente sinalizado na coluna "Traits". Já a linha 90 é onde há a chamada para a função *SQR*. Pela análise do gráfico, essa função é *compute and memory bound* (causa possível: muitas operações aritméticas).

[loop in LaplaceSolver::timeStep at laplace.cxx:87]
 Scalar; processes Float64 data type(s)
 Performance: **1.088 GFLOPS**
 Self Time: **0.228 s**
 Self Elapsed Time: **0.228 s**
 Total Time: **0.228 s**
 Total Elapsed Time: **0.228 s**
 Self Memory Traffic: **7.937 GB**
 Total Memory Traffic: **<0.001 GB**
 CARM (L1 + NTS) Arithmetic Intensity: **0.031 FLOP/Byte**
 Bounded By: **Scalar L1 Bandwidth** (Utilization: 38%)

[loop in LaplaceSolver::timeStep at laplace.cxx:87]
 Scalar; processes Float64 data type(s)
 Performance: **1.088 GFLOPS**
 Self Time: **0.228 s**
 Self Elapsed Time: **0.228 s**
 Total Time: **0.228 s**
 Total Elapsed Time: **0.228 s**
 Self Memory Traffic: **0.402 GB**
 Total Memory Traffic: **<0.001 GB**
 L2 Arithmetic Intensity: **0.616 FLOP/Byte**



- *SQR*
Esta função é um hotspot em L1. Pela análise do gráfico, essa função é *memory bound* (causa possível: acessa diversos endereços de memória).



Conclusão

Após a análise dos resultados do ERT e do Intel Advisor podemos reforçar as conclusões do Trabalho 2 sobre os hotspots do código *laplace.cxx*, além de perceber com mais clareza como o código mal-otimizado influencia na utilização do potencial completo de computação em um sistema.