**HPC Profiling Report**

**CS22B2014 Navaneethakrishnan R**

**Profiling Using Gprof**

**Flat profile:**

% cumulative self self total

time seconds seconds calls s/call s/call name

61.00 39.17 39.17 30 1.31 1.31 backward\_pass -- **HOTSPOT**

37.17 63.03 23.86 4268971 0.00 0.00 attention\_score -- **HOTSPOT**

1.73 64.14 1.11 30 0.04 0.83 gnn\_layer

0.10 64.21 0.07 12000 0.00 0.00 softmax

0.00 64.21 0.00 12001 0.00 0.00 sigmoid

0.00 64.21 0.00 12000 0.00 0.00 binary\_cross\_entropy

0.00 64.21 0.00 12000 0.00 0.00 layer\_norm

0.00 64.21 0.00 12000 0.00 0.00 relu

0.00 64.21 0.00 30 0.00 0.00 clip\_gradients

0.00 64.21 0.00 30 0.00 0.00 update\_parameters

0.00 64.21 0.00 2 0.00 0.00 initialize\_parameters

0.00 64.21 0.00 1 0.00 0.00 predict\_link

**Inferences**

The **backward\_pass** **(61%)** and **attention\_score** **(37.17%)** dominate execution time, making them key optimization targets. **The attention\_score function is called ~4.27M times and that function is mostly consists matrix multiplication**, heavily impacting performance. Parallelizing or optimizing these functions could significantly improve efficiency.

**Call graph**

index % time self children called name

<spontaneous>

[1] 100.0 0.00 64.21 main [1]

39.17 0.00 30/30 backward\_pass [2]

1.11 23.93 30/30 gnn\_layer [3]

0.00 0.00 1/1 predict\_link [6]

0.00 0.00 12000/12001 sigmoid [7]

0.00 0.00 12000/12000 binary\_cross\_entropy [8]

0.00 0.00 30/30 clip\_gradients [11]

0.00 0.00 30/30 update\_parameters [12]

0.00 0.00 2/2 initialize\_parameters [13]

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

39.17 0.00 30/30 main [1]

[2] 61.0 39.17 0.00 30 backward\_pass [2]

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

1.11 23.93 30/30 main [1]

[3] 39.0 1.11 23.93 30 gnn\_layer [3]

23.86 0.00 4268970/4268971 attention\_score [4]

0.07 0.00 12000/12000 softmax [5]

0.00 0.00 12000/12000 layer\_norm [9]

0.00 0.00 12000/12000 relu [10]

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

0.00 0.00 1/4268971 predict\_link [6]

23.86 0.00 4268970/4268971 gnn\_layer [3]

[4] 37.2 23.86 0.00 4268971 attention\_score [4]

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

0.07 0.00 12000/12000 gnn\_layer [3]

[5] 0.1 0.07 0.00 12000 softmax [5]

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

0.00 0.00 1/1 main [1]

[6] 0.0 0.00 0.00 1 predict\_link [6]

0.00 0.00 1/4268971 attention\_score [4]

0.00 0.00 1/12001 sigmoid [7]

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

0.00 0.00 1/12001 predict\_link [6]

0.00 0.00 12000/12001 main [1]

[7] 0.0 0.00 0.00 12001 sigmoid [7]

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

0.00 0.00 12000/12000 main [1]

[8] 0.0 0.00 0.00 12000 binary\_cross\_entropy [8]

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

0.00 0.00 12000/12000 gnn\_layer [3]

[9] 0.0 0.00 0.00 12000 layer\_norm [9]

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

0.00 0.00 12000/12000 gnn\_layer [3]

[10] 0.0 0.00 0.00 12000 relu [10]

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

0.00 0.00 30/30 main [1]

[11] 0.0 0.00 0.00 30 clip\_gradients [11]

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

0.00 0.00 30/30 main [1]

[12] 0.0 0.00 0.00 30 update\_parameters [12]

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

0.00 0.00 2/2 main [1]

[13] 0.0 0.00 0.00 2 initialize\_parameters [13]

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

**Profiling Using Gcov**

CODE : gcov -b -c main.c

File 'main.c'

Lines executed:100.00% of 170

Branches executed:100.00% of 90

Taken at least once:95.56% of 90

Calls executed:100.00% of 27

Creating 'main.c.gcov'

Lines executed:100.00% of 170

Code: gcov -a main.c

File 'main.c'

Lines executed:100.00% of 170

Creating 'main.c.gcov'

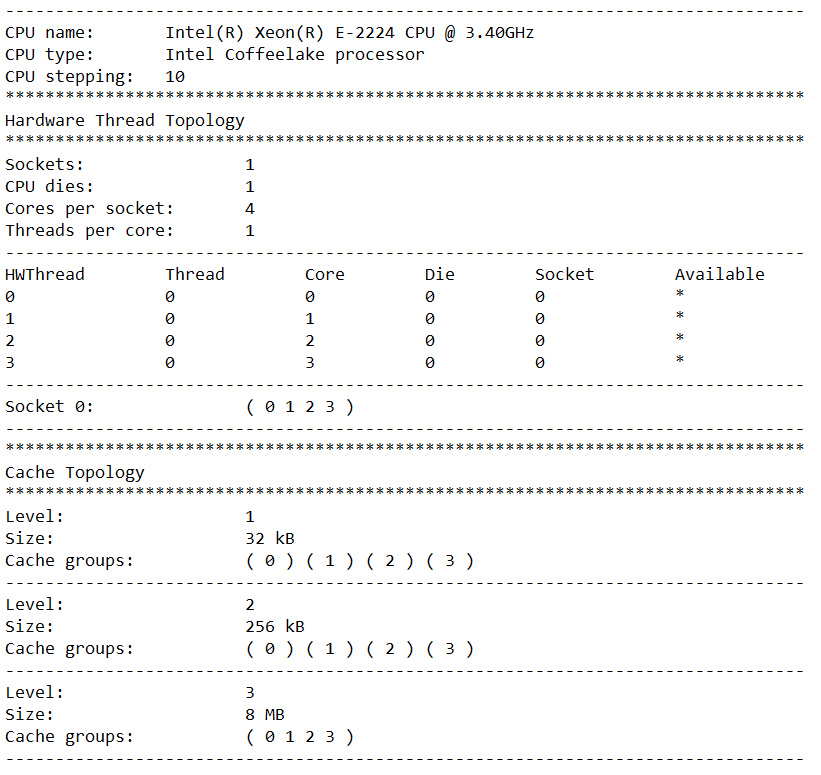
Lines executed:100.00% of 170

**Inference:**

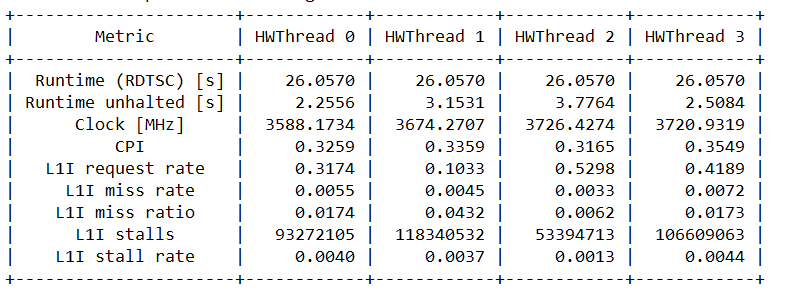
The code in main.c achieved 100% line execution coverage (170/170) and 100% function call execution (27/27). Branch execution is also 100%, with 95.56% of branches taken at least once, indicating thorough testing with minor uncovered branch paths.

**Profiling Using LIKWID**

**Architecture Topology**



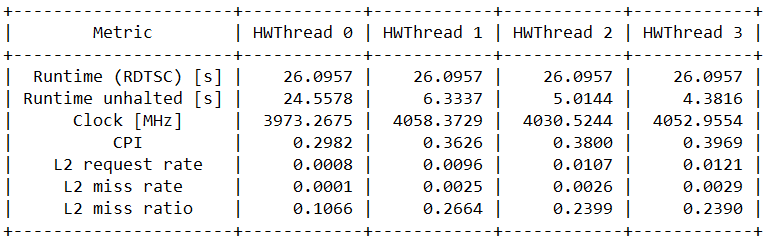
**Instruction Cache Miss Rate Ratio**

Code: likwid-perfctr -C 0-4 -g ICACHE ./main

This provides us with the number of requests, misses and stalls made by the L1 Cache for fetching instruction

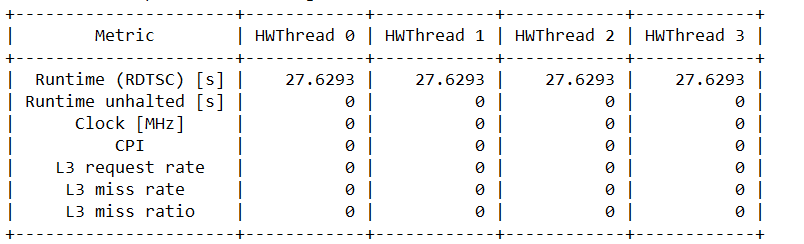
**L2 Cache Miss Rate Ratio**

Code: likwid-perfctr -C 0-4 -g L2CACHE ./main



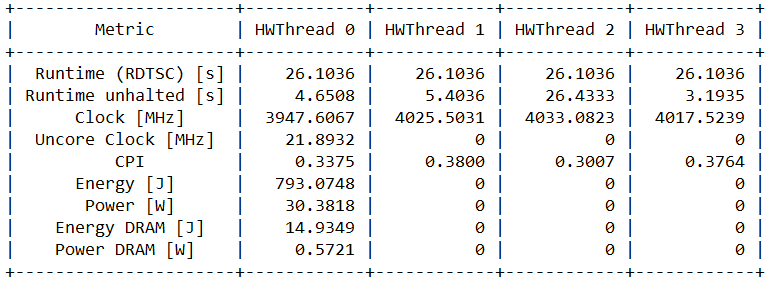
**L3 Cache Miss Rate Ratio**

Code: likwid-perfctr -C 0-4 -g L3CACHE ./main



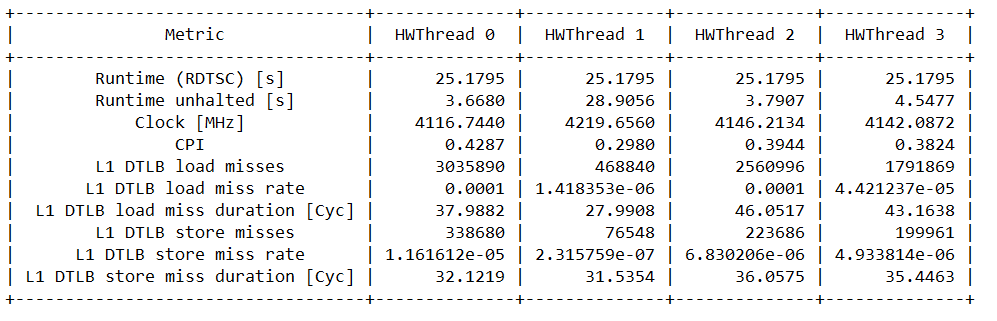
**Clock, Energy and Power Consumption**

Code: likwid-perfctr -C 0-4 -g CLOCK ./main



**Data TLB - Translation Look aside Buffer for Data**

Code: likwid-perfctr -C 0-4 -g TLB\_DATA ./main



**Instruction TLB - Translation Look aside Buffer for Instructions**

Code: likwid-perfctr -C 0-4 -g TLB\_INSTR ./main

