CS601: Software Development for Scientific Computing

Autumn 2023

Week7: Tools for debugging and profiling and more..

Valgrind

- Suite of tools for debugging and profiling
 - memcheck and cachegrind are popular ones
 - cachegrind is cache and branch-prediction profiler.
 - memcheck is a memory error detector.
- Demo of cachegrind tool with matmul
 - https://valgrind.org/docs/manual/cg-manual.html
- Demo of memcheck with matmul

Steps to use cachegrind

- Example: matmul.cpp
 - Compile with -g and create a target.
 - Run as: valgrind --tool=cachegrind ./matmul 2048
 - Out of cachegrind is dumped in a file that has the format cachegrind.out.xxxxxx where xxxxx is the process ID
 - Use cg_annotate to get annotated output
 - E.g. cg_annotate cachegrind.out.12345

cachegrind

Visualizing cache transactions

```
I1 cache:
                                                     32768 B, 64 B, 8-way associative
 L1 Instruction
                                                     32768 B, 64 B, 8-way associative
                                   D1 cache:
                                   LL cache:
                                                      37748736 B, 64 B, 18-way associative
 L1 Data -
                                   Command:
                                                      ./matmul ijk 2048
                                   Data file:
                                                     cachegrind.out.1395356
 Last layer
                                                     Ir I1mr ILmr Dr D1mr DLmr Dw D1mw DLmw
                                   Events recorded:
                                   Events shown:
                                                     Ir I1mr ILmr Dr D1mr DLmr Dw D1mw DLmw
Instructions read
                                   Event sort order: Ir I1mr ILmr Dr D1mr DLmr Dw D1mw DLmw
                                   Thresholds:
                                                      0.1 100 100 100 100 100 100 100 100
L1 Instruction read misses
                                   Include dirs:
Last layer instruction read misses
                                   User annotated:
Data reads (total memory reads)
                                   Auto-annotation:
                                                     on
```

- L1 data read misses
- Last layer data read misses
- Data writes (total memory writes)
- L1 data write misses
- Last layer data write misses

Total last layer misses = ILmr + DLmr + DLmw

cachegrind

Visualizing cache transactions (ijk loop ordering of matmul)

```
Itmr (L1 read miss)

LLmr (LL instruction read miss)

Dr (Data read == number of memory reads)

438,893,764,234 (100.0%) 2,267 (100.0%) 2,157 (100.0%) 189,231,226,549 (100.0%)

Dlmr (L1 Data read miss)

DLmr (LL data read misses)

10,740,872,902 (100.0%) 7,827,585,951 (100.0%)

Dw (Data write = number of memory writes)

Dlmw (L1 data cache write miss)

Dlmw (LL data write miss)

8,674,338,548 (100.0%) 1,586,278 (100.0%) 1,582,786 (100.0%)
```

Nikhil Hegde

cachegrind

Visualizing cache transactions (ikj loop ordering of matmul)

```
438,803,764,251 (100.0%) 2,267 (100.0%) 2,157 (100.0%) 189,231,226,544 (100.0%)

D1mr (L1 Data read miss)

DLmr (LL data read misses)

1,223,946,667 (100.0%) 1,004,088,043 (100.0%)

Dw (Data write = number of memory writes)

D1mw (L1 data cache write miss)

DLmw (LL data write miss)
```

8,674,338,550 (100.0%) 1,586,278 (100.0%) 1,582,786 (100.0%)

Total last layer misses are much lesser than that in ijk loop!