Profiling Tools for Performance Metrics

Theodoros Theodoris
Feb. 2024
Performance Monitoring

Modern CPUs can measure various performance related parameters:

Raw metrics: e.g., number of vectorized instructions

Performance events: e.g., number of cycles “wasted” waiting for memory

Branch recording: tracking which branches (“if statements, loops) are executed and how many times

...
Performance Counters

Performance counters are special registers that can be configured to track the event of interest, e.g.:

- \# scalar executed floating point operations.
- \# vectorized executed floating point operations.
- \# branch misses
- \# cache misses

For more details on performance counters look at the performance counters slides on the course website.
Profiling Tools

Profiling tools help with measuring and identifying performance related events and metrics.

This is typically done by running the target program and sampling the relevant CPU’s performance counters.

Metrics can be counted throughout a program’s execution (e.g., total number of memory transactions), or correlated with certain program locations (i.e., finding hotspots).

Hierarchical analyses, such as, the Top-down Microarchitecture Analysis Method (TMA), can be used to systematically identify bottlenecks.
(non-exhaustive) List of Profiling Tools

**Intel VTune** is probably the most feature complete tool

**AMD uProf** supports AMD specific events

**Apple Xcode Instruments** can access perf. counters on M1 and M2 CPUs

**Linux perf** is a command line that can measure most things that VTune can, but it is less intuitive and without a GUI

**toplev** implements **Top-down Microarchitecture Analysis Method** (only works on Intel+Linux based systems)
The **perf-book** is a great resource that demonstrates (lots of examples) various tools and methodologies for performance analysis. Some interesting chapters:

Performance Analysis Approaches (5) show various methods that enable performance, instrumentation, sampling, compiler optimization reports, etc.

CPU Features for Perf. Analysis (6) explains mechanisms implemented in CPUs that facilitate performance analysis

Overview of Tools (7) is a quick tour of various tools such as Vtune
Perf Example: counting events (1/2)

Given a program a.out, we can ask perf to count performance related metrics:

```
perf stat ./a.out
```

Performance counter stats for './a.out':

```
   0.79 msec task-clock:u # 0.751 CPUs utilized
   0 context-switches:u   # 0.000 /sec
   0 cpu-migrations:u    # 0.000 /sec
   79 page-faults:u      # 99.912 K/sec
  2,052,295 cycles:u     # 2.596 GHz
  2,403,057 instructions:u # 1.17 insn per cycle
  511,384 branches:u     # 646.750 M/sec
   5,022 branch-misses:u # 0.98% of all branches
 10,261,475 slots:u      # 12.978 G/sec
 2,736,393 topdown-retiring:u # 26.7% retiring
 482,892 topdown-bad-spec:u # 4.7% bad speculation
 965,785 topdown-fe-bound:u # 9.4% frontend bound
 6,076,402 topdown-be-bound:u # 59.2% backend bound
```

Around 1% of the branches were mispredicted.
Perf Example: counting events (2/2)

To measure a specific metric:

```bash
perf stat -e LLC-load-misses ./a.out
Performance counter stats for './a.out':
   291      LLC-load-misses:u
```

Multiple events can be measured:

```bash
perf stat -e fp_arith_inst_retired.256b_packed_single,LLC-load-misses ./a.out
Performance counter stats for './a.out':
   240,000      fp_arith_inst_retired.256b_packed_single:u
   285      LLC-load-misses:u
```

The exact name event (metric) name depends on the CPU. `perf list` shows the supported events.

240,000 vectorized arithmetic instructions were executed
Perf: sampling

The previous examples showed how to count specific events (e.g., number of cache misses).

perf can also use sampling to identify which parts of a program trigger those events (perf record).

Linux perf Examples (brendangregg.com) shows various examples.