Performance Evaluation of Computer Systems

Timers & Counters

Vojtěch Horký    Petr Tůma

Department of Distributed and Dependable Systems
Faculty of Mathematics and Physics
Charles University in Prague
Czech Republic

March 29, 2017
Task

**Timers**
Explore precision of available Linux timers.

**Counters**
Write code that emits as many events of given type as possible.

**Implementation**
Either Java or C++, PAPI library.
# Source Code & Submission

## Harness and examples

http://d3s.mff.cuni.cz/teaching/peva/files/lab02.tar.gz
(Includes compilation and run scripts.)

## Submission

By e-mail to horky@d3s.mff.cuni.cz.

Send TAR (or ZIP) with (commented) sources, data, run scripts and a brief README.
Timers Task
Available Timers

- `clock_gettime()` (see man 2 clock_gettime)
  - `CLOCK_REALTIME`
  - `CLOCK_REALTIME_COARSE`
  - `CLOCK_MONOTONIC`
  - ...
- `System.nanoTime()` (Java)
- `std::chrono::steady_clock` (C++11)

For Java, we have an agent prepared for you to access the `clock_gettime()` function from JVM.
The Task

Compare the precision of the aforementioned timers.

What is the shortest time interval that can be measured with each of them?

You do not need to compare all the timers, but at least:

- **CLOCK_MONOTONIC**
- one of the **_*COARSE* clocks**
- one of the **_*CPUTIME_ID* clocks**
- `System.nanoTime()` or `std::chrono::steady_clock`
Example Source Code

Prints the current clock value and also the reported resolution (see man 2 clock_getres()).

Use the example as the starting point for your solution.

C++

Direct calls to clock_gettime() and clock_getres().

Java

Java native methods (use -Djava.library.path=. ) for accessing the C library.
Hints

Data gathering

What is the shortest operation?
Zero is wrong result ...
Do not forget about warm-up (especially for Java) ...

Data plotting

clock,end,start
MONOTONIC,133.075341805,133.075341952
MONOTONIC,134.051579635,134.068485495
...
nanoTime,350.139651503,350.139651559

\[ x \leftarrow \text{read.csv("data.csv")}; \text{boxplot}(x$end - x$start \sim x$clock) \]

\text{Rscript plot-compare-timers/plot.r data.csv timers.png}
Counters Task
The Task

Write a code that exercises a specific counter a lot.

The score for your code is computed as a ratio of fired events and the total number of instructions. (Thus, inserting a loop inside your implementation will typically not increase the score.)

Select at least 3 counters from the list (see next slide).

Description of each counter can be obtained through papi_avail and papi_native_avail.
Counters

- PAPI_FDV_INS
- PAPI_BR_PRC
- MISPREDICTED_BRANCH_RETIRED
- LAST_LEVEL_CACHE_MISSES
- PAGE_WALKS
- MUL
- BR_CALL_EXEC
- MACRO_INSTS
- perf::PAGE-FAULTS
**Example Source Code**

Stresses the load instructions (PAPI_LD_INS).

Your code would inherit from the base workload class and would override the `execute()` method.

`execute()` returns a value to prevent aggressive compilation optimizations (that could remove all your code in the worst case).

The actual measurement runs in the following manner:

```python
for i in 1..10000:
    start-counters()
    for j in 1..1000
        workload.execute()
    done
stop-and-store-counters()
done
```
C++
Add harness_run() calls to stress-counters.cpp.

Java
Add your classes to run.sh and make.sh.
The make.sh scripts downloads and compiles a Java agent for accessing the PAPI library.

R
Script for plotting your data.
Running

Hints

volatile variables and randomization can kill a lot of optimizations ...
Do not forget about warm-up (we are interested in long-time behaviour) ...

C++

./make.sh
./stress-counters >out.csv

Java

./make.sh (add your sources!)
./run.sh (add your classes!)

Plotting with R

Rscript ../plot-stress-counters/plot.r out.csv out.pdf
Example Output Graph

COUNTER / INSTRUCTIONS
- Empty: 0.25257
- Workload: 0.40880
- IMPROVEMENT: 1.61853

Graph showing the ratio of events to instructions, ratio of ratios, event count, and instruction count.
Q & A