Real-Time Micro-Benchmark for Real-Time Linux

Vernon Mauery

IBM Linux Technology Center
Real-Time Linux

Disclaimer

This is not my code, but I know who wrote it

- Mike Fulton <fultonm@ca.ibm.com>
- Ajith Ramanath <ajith.ramanath@in.ibm.com>
- Radhakrishnan Thangamuthu <rthangam@in.ibm.com>

Recently released into the wild

- http://sourceforge.net/projects/rtmicrobench/
- for those who just can't wait:
- svn co https://rtmicrobench.svn.sourceforge.net/svnroot/rtmicrobench/rtmb

Benchmarking

Lies, damn lies, and benchmarks
Fabricated workloads that give a skewed picture
Can offer a small window into specific workloads
Pick your favorite benchmark to tell you the lies that sooth you the most

Yet Another Benchmark?

- No ideal RT benchmark
  - Cross platform
  - Cross language
  - Testing RT APIs (with micro benchmarks)
- Customer Java acceptance test
  - Does it fail in C too?
  - A small pile of disembodied tests
  - Still testing against hard-coded standards
- Starting from the bottom up
  - Use an automatic calibration
  - Use a test harness
  - Use proper statistical models

How does it work?

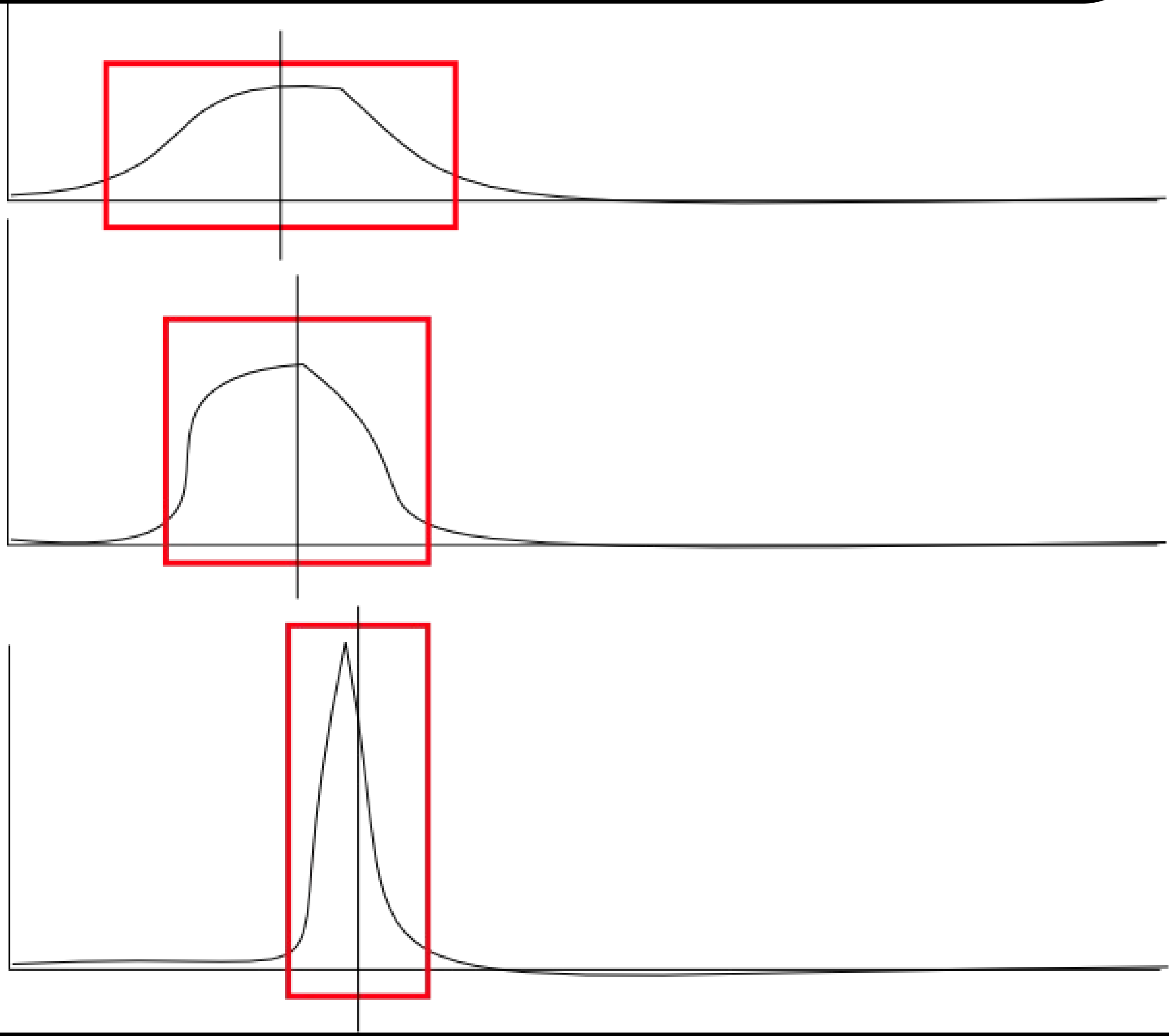
```
Establish the clock resolution
For each remaining test
    For iteration count
        Get current time
        Perform work (test some API or other small task)
        Get time
    Did the test run for longer than the clock resolution?
        If not, increase work size
    Is determinism within user constraints? (soft, hard, guaranteed)
    re-run with larger workload (longer time) or quit
    Give the test a latency score based on how deterministic it is
    Give a throughput score at that level of determinism
```

Determinism Scores

Soft RT
- 30% deviation
- 99th percentile

Hard RT
- 20% deviation
- 99.9th percentile

Guaranteed RT
- 10% deviation
- 99.999th percentile



What does it test?

	C	C++	Java
Bytecode Consistency			X
Clock	X	X	X
Compilation			X
Disk I/O read	X	X	X
Disk I/O write	X	X	X
Event dispatch	X	X	X
Float rate	X	X	X
Garbage generation			X
Int rate	X	X	X
Lock consistency	X	X	X
Mcache perf	X	X	X
Multi CPU		X	X
Multi thread	X	X	X
Net I/O	X	X	X
NHRT support			X
Periodic event	X	X	X
Scoped memory			X
Thread priority	X	X	X
Time accuracy	X	X	X
Timer	X	X	X

Tests

- Bytecode Consistency (Java)
  - Verifies that each bytecode instructions perform correctly
- Clock (All)
  - Determines the granularity of the clock
- Compilation (Java)
  - Verifies that the bytecode compiler does not interfere with the performance of the application
- Disk I/O Read and Write (All)
  - Determines the time to read or write a block of data on the disk
- Event Dispatch (All)
  - Measures time to dispatch work between two running threads
- Float Rate (All)
  - Measures the rate of a mix of floating point operations
- Garbage Generation (Java)
  - Measures effectiveness and impact of GC on running threads

Tests Continued

Int Rate (All)

Measures the rate of a mix of integer instructions

Lock Consistency (All)

Measures uncontended lock acquisition at various priorities

Mcache perf (All)

Measures the performance of the memory cache

Multi-CPU (C++, Java)

Creates objects and does work in a periodic scheduling setup

Multi-Thread (All)

Throughput of matrix multiplication between 1 and NR_CPUS

Net I/O (All)

Measures time to read a block of data from a network connection

NHRT No Heap Real Time (RTSJ)

Measures consistency of NHRT threads running on a loaded system

More Tests

Periodic Event (All)

- Measures how consistently periodic events are dispatched

Scoped Memory (RTSJ)

- Measures the cost of accessing objects within various scopes

Thread Priority (All)

- Verifies that higher-priority threads preempt lower-priority threads

Time Accuracy (All)

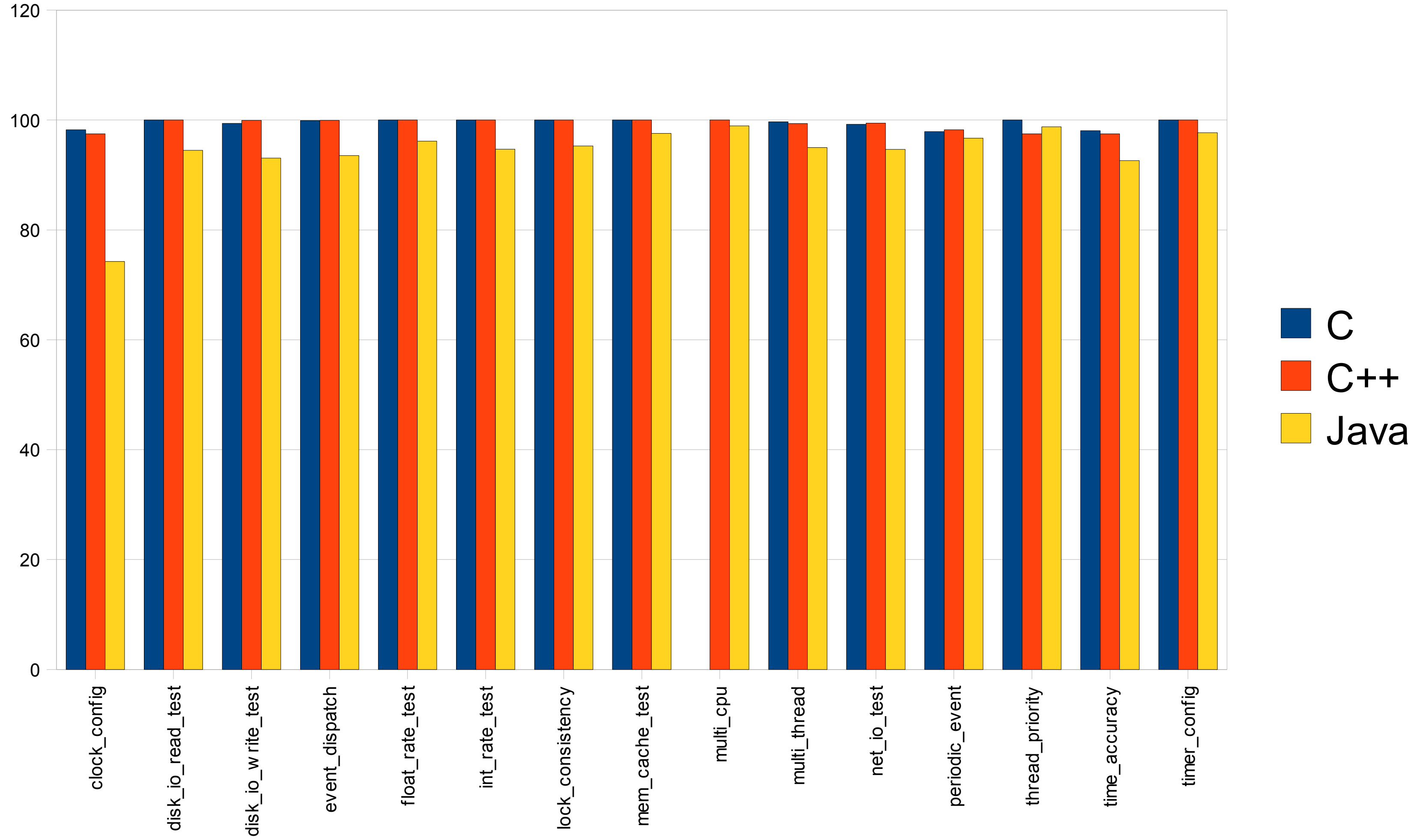
- Evaluates the accuracy of the time subsystem by sleeping with the high-resolution timer and then comparing with the clock

Timer (All)

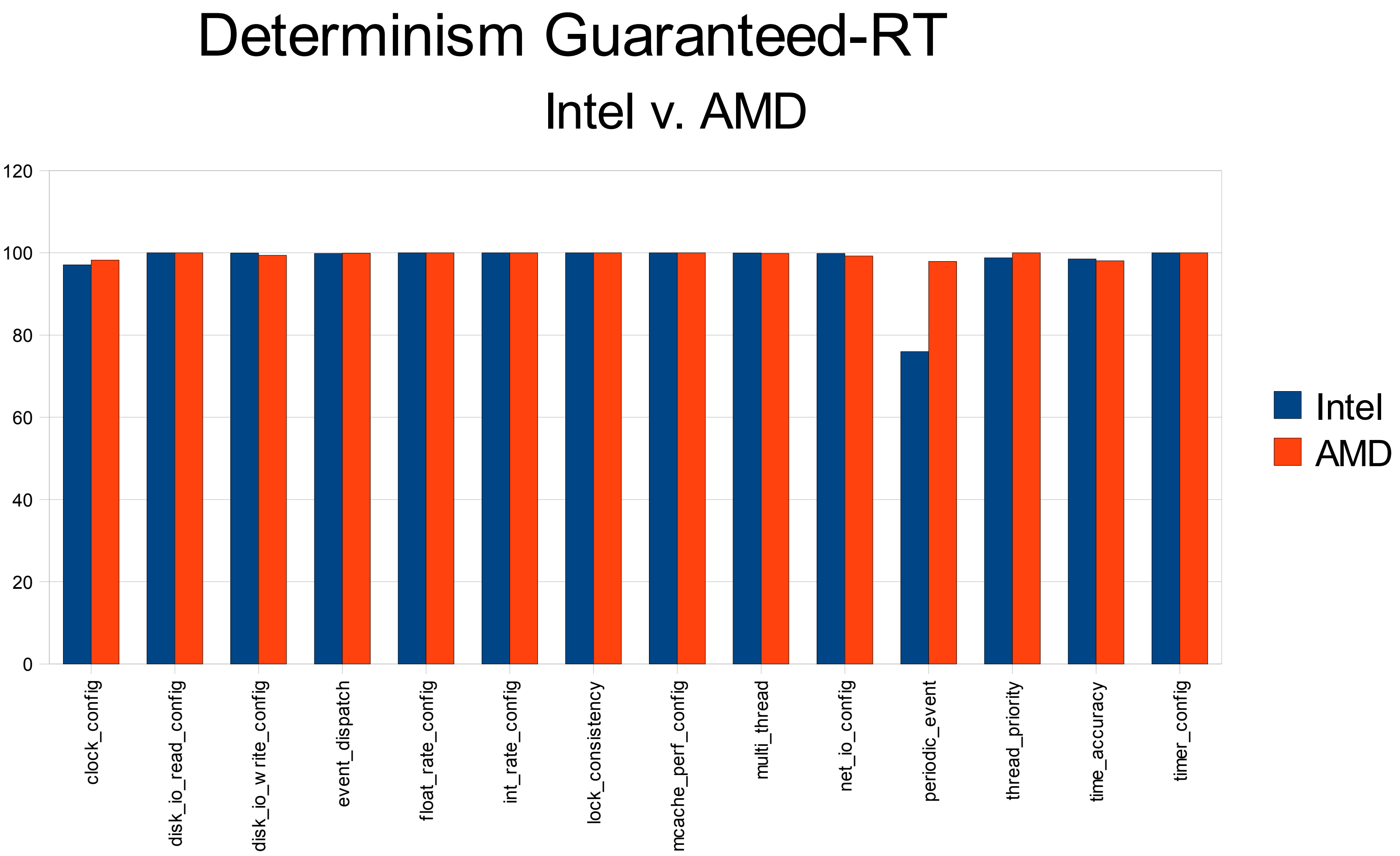
- Measures accuracy of the high-resolution timer by checking for the minimum time a thread can sleep and deterministically wake up

Cross Language

Determinism Guaranteed-RT

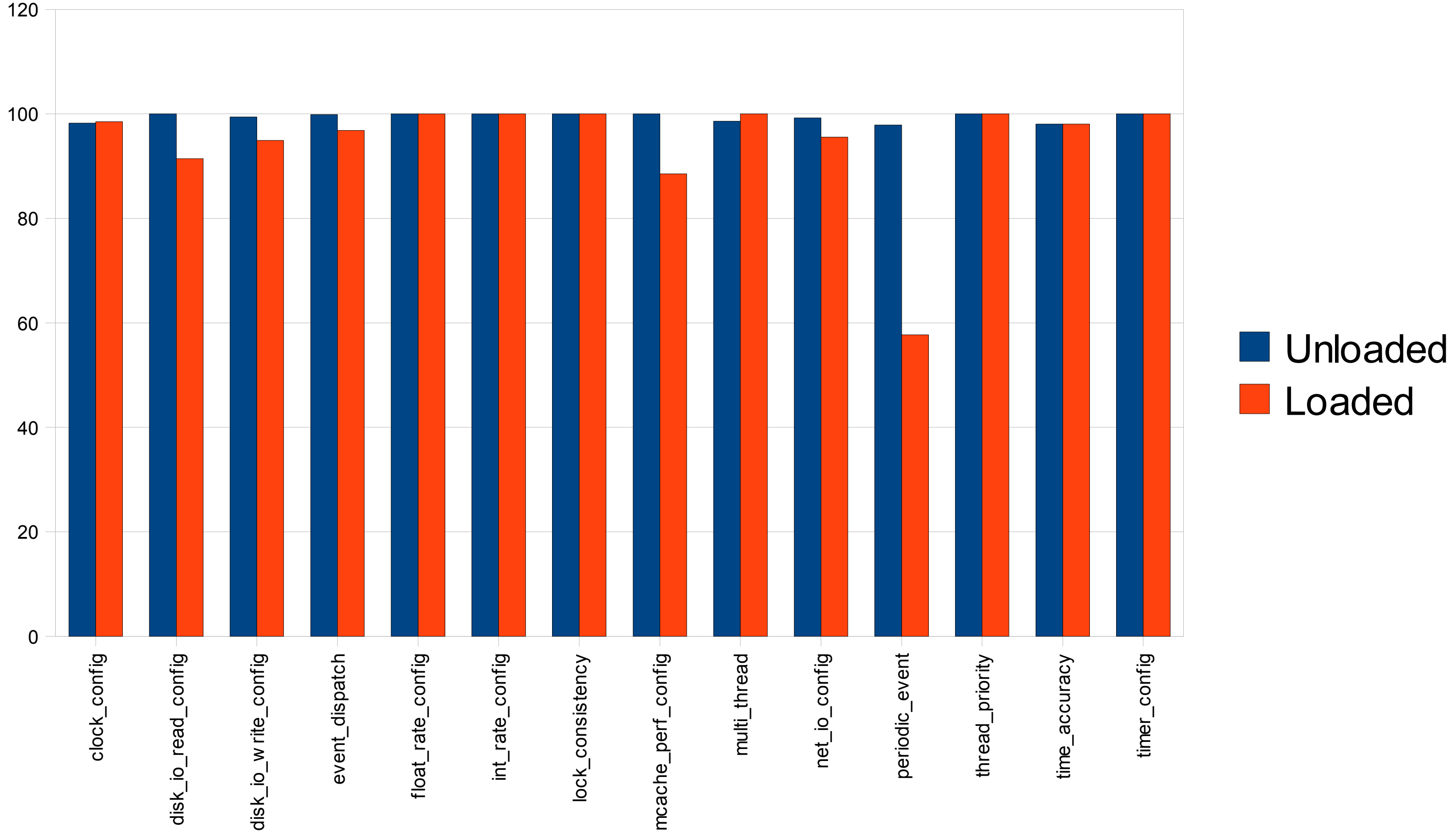


Cross Platform



Loaded vs. Unloaded

Determinism Loaded v. Unloaded



Uses for RTMB

A solid, across-the-board real-time comparison (system, OS, language)

Regressions

- Daily test.kernel.org RTMB runs...

So you can brag about how awesome your box is

So you [your company] can sell more software

So you [your company] can sell more services

So you [your company] can sell more hardware

Just in case /dev/null is looking a little empty

Legal Statement

This work represents the view of the author and does not necessarily represent the view of IBM.

IBM, IBM (logo), e-business (logo), pSeries, e (logo) server, and xSeries are trademarks or registered trademarks of International Business Machines Corporation in the United States and/or other countries.

Linux is a registered trademark of Linus Torvalds.

Other company, product, and service names may be trademarks or service marks of others.