Inferring Scheduling Behavior with Hourglass

John Regehr

School of Computing, University of Utah
http://www.cs.utah.edu/~regehr

6/14/2002

1

What is Hourglass?

- Synthetic real-time application
  - Only purpose is to take measurements
  - Timing constraints
  - No kernel modules or patches

2

Goal

- Provide fast, accurate answers to CPU scheduling questions
  - Microbenchmarks: Dispatch latency? Timer accuracy? Context switch time?
  - Single-application: Would it help to use a different timer? To reduce compute time by 15%?
  - Multi-application: Will X, Y, and Z work together?

3

Why Answer Scheduling Questions?

- Identify / solve application timing problems
- Make predictions about application performance
- Compare OSs e.g.
  - Linux 2.2 vs. 2.4
  - Preemptible vs. low latency Linux
  - Linux vs. Windows XP vs. FreeBSD
- Debug schedulers

4

Other Ways to Answer Scheduling Questions

- Add instrumentation to a non-synthetic real-time application
  - E.g. Game, DVD player, mp3 player, software modem, …
- Use an instrumented kernel
  - E.g. Linux Trace Toolkit
- More detail in paper…

5

Key Capabilities

- Create accurate execution trace
- Support multiple thread models
- Provide portable access to scheduling functionality

6

Execution Trace

- **Precise map of when each Hourglass thread runs**
  - Threads poll timestamp counter
  - Log "gaps" to memory buffer
- **Important details: need to**
  - Know CPU speed
  - Select minimum gap size appropriately
  - Avoid spurious page faults

7

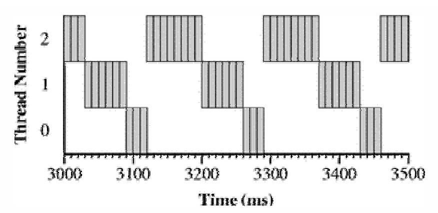
Raw Execution Trace

Thread	Duration	Gap
2	9.976	0.006
2	9.976	0.005
0	9.972	0.009
0	9.976	0.005
1	7.574	0.006
1	1.242	0.009
1	1.139	0.009
1	0.122	0.005

8

Graphical Execution Trace

- **Postprocess with Perl, jgraph, etc. to get:**



9

Supported Thread Models

- **Periodic with blocking**
  - Most non-game real-time apps
- **Periodic non-blocking**
  - Most games and other rendering loops
- **Also: CPU-bound, latency test, scanning**
- **Easy to extend…**

10

Portability

- **Uniform command-line access to**
  - Thread models
  - Timers
  - Priorities
  - CPU reservations

11

How It Works

- **main()**
  - Spawns worker threads
  - Sleeps
  - Prints results
- **Worker threads**
  - Run gap-detection loop
  - At appropriate times:
    - Schedule wakeup and go to sleep
    - Register deadline hit / miss
    - Touch memory

12

Using Hourglass

- First: Map the scheduling question onto a concrete scenario
- Second: Create an Hourglass command line that implements the scenario
  - Use other apps to supply contention
- Third: Run Hourglass, interpret the results

13

Example

- Question 1: Can a demanding digital audio app reliably meet its deadlines on Linux?
  - App requires 4ms CPU during every 5ms period
- Command line:

```
hourglass -d 20s -n 1 -t 0     \
          -p RTHIGH            \
          -w PERIODIC 4ms 5ms  \
          -i RTC
```

14

Example Cont'd

- Answer: YES
  - No deadlines missed on a variety of Linux kernels
  - (On an otherwise quiet machine)

15

Example Cont'd

- Question 2: How about during network receive processing?
  - Same Hourglass command line
  - Use Netperf to receive full-bandwidth data over 100Mbps Ethernet
- Answer: Sometimes…

16

Example Cont'd

- Numerical answer:

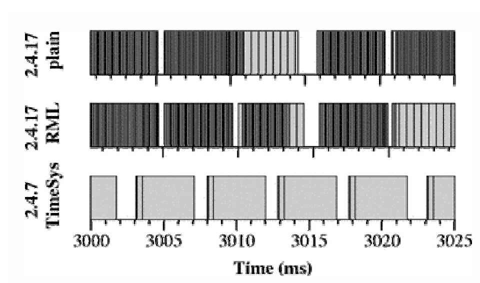
Experiment	Missed deadlines	Throughput
2.4.17 plain	33.0%	68 Mbps
2.4.17 RML	0.4%	66 Mbps
2.4.7 TimeSys	0.0%	59 Mbps

- RML == preemptible kernel + lock breaking patches
- Netperf baseline: 94 Mbps

17

Example Cont'd

- Visual answer:



18

Related Work

- LMBench, HBench, Latencytest
- Linux Trace Toolkit
- *Gscope*
- txofy, mptxofy

19

Availability

- Runs on
  - Pentium-class x86
  - Linux, FreeBSD, Win32
- BSD style license
- Home page
  - www.cs.utah.edu/~regehr/hourglass
  - Or Google for "regehr hourglass"

20

Conclusion

- Can learn a lot using a synthetic real-time application:
  - Execution trace is surprisingly useful for making inferences about scheduling behavior
  - A few thread models cover most interesting applications

21

The End

- More info at www.cs.utah.edu/~regehr/hourglass

- Let's talk…

22