

System Performance Tracing and Analysis

Michel Dagenais
Mathieu Desnoyers
Pierre-Marc Fournier
Gabriel Matni

Department of Computer and Software Engineering
Ecole Polytechnique, Montreal

Summary

- Introduction
- Data providers
- Data collectors
- Data analysis
- Frameworks
- Challenges



Introduction

- Embedded real-time systems with detailed timing data.
- General purpose computers with application level tools and some operating system statistics.
- High performance parallel computing with library level (MPI) tracing.



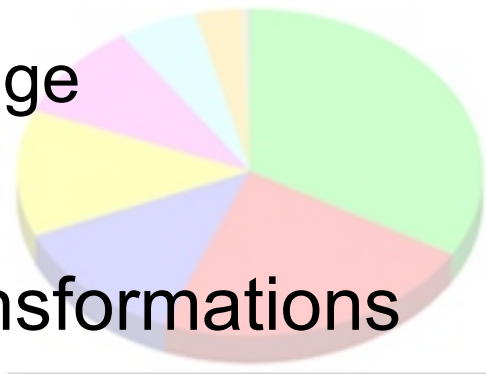
But!?

- Real-time multi-core high performance systems, with virtualization?
- Rapidly evolving heterogeneous systems?



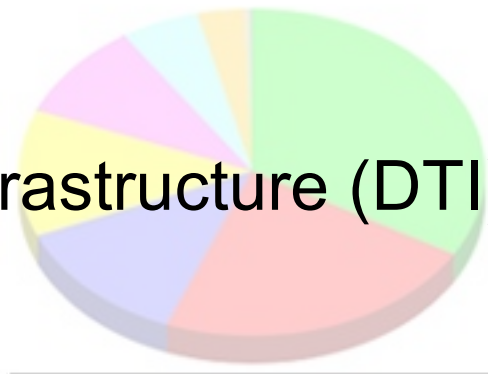
Data providers

- Source-level, auto-generated
 - ♦ gcc -finstrument-functions
 - ♦ gcc -fprofile-arcs
 - ♦ gcc -ftest-coverage
 - ♦ gcc -pg
- Source-level, transformations
 - ♦ javacc
 - ♦ TXL



Data providers

- Source-level, manual insertion
 - ♦ Logging API, Java, log4cpp, .NET...
 - ♦ printk
 - ♦ evlog
 - ♦ Driver tracing infrastructure (DTI)
 - ♦ Kernel markers



Data providers

- Static binary instrumentation
 - ATOM, EEL
- Dynamic binary instrumentation
 - DTrace
 - SystemTap
 - DynInst
 - GDB



Data providers

- Sampling
 - GProf
 - OProfile
- Simulators
 - Valgrind
 - QEMU



Data collectors

- Simple counter
- Code hook (interpreted or precompiled)
- Unbuffered write
- Buffered write
- Buffered write with lock or atomic ops
- Flight recorder mode (in memory)
- Trace written to disk or network



Data collectors

- Per CPU buffers
- Atomic operations
- Count of lost events
- Zero copy of buffers in memory
- Init time tracing



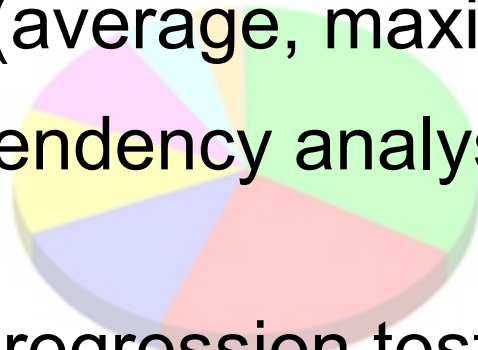
Data analysis

- Total number of events
 - ♦ code coverage
 - ♦ number of read/write, pf, irq, packets...
 - ♦ number of execution samples and calls
- Elapsed time
 - ♦ Events with timestamps...
 - ♦ “time”
 - ♦ Kernel Function Tracer
 - ♦ LTTng



Data analysis

- Check for conditions (filters, assertions).
- Check for visual patterns.
- Compute delays (average, maximum).
- Critical path (dependency analysis) between command and answer.
- Compare traces (regression test, health monitoring...).



Example: LTT

User Mode:

CPU 5.894726

Elapsed 15.677299

WaitCPU 2.358053

WaitFork 0.000002

Syscall Mode:

Elapsed/Calls 0.0003861

CPU 0.308954

Elapsed 5.119347

WaitCPU 0.599823

WaitFile-001.png 0.000453

WaitFile-002.png 0.000346

WaitFile-003.png 0.000213

...



Overhead

- test unmodified gzip: 28.16s
- sampling: 28.30
- sampling and function entry: 29.88
- basic block entry: 31.36s



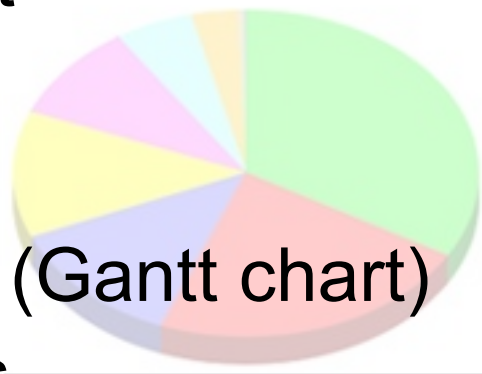
Overhead

- Java logging API: 7.7us/event in memory, 208us/event to file
- DTrace: 1.18us/event, 3.1us/read event
- SystemTap: 1.3us/event
- Kernel marker inactive: 0.0005us/event
- Kernel marker active: 0.198us/event
- LTTng flight recorder: 0.746us/event



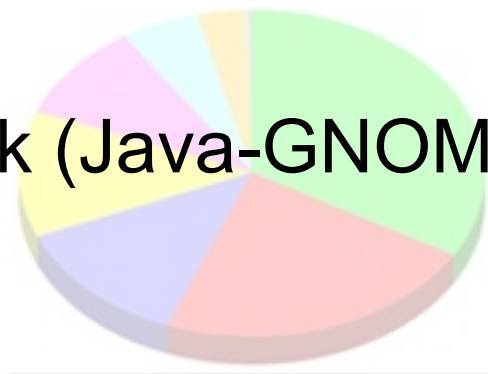
Frameworks

- Programming environment
- Views, plugins, resources, scripting
- Detailed event list
- Statistics
- Graphs
- Control flow view (Gantt chart)
- Filters, assertions
- Trace bookmarks
- Profiling, coverage, memory analysis, disk analysis, critical path, client/server requests...



Frameworks samples

- QNX, WindRiver, ZealCore... Eclipse!
- Intel VTune, Eclipse.
- DTrace, Chime.
- SystemTap, Frysk (Java-GNOME, Eclipse).
- LTTng, LTTV.



QNX Momentics

The screenshot displays the QNX System Profiler interface. The main window is titled "QNX System Profiler - test.kev - QNX Momentics IDE". The interface includes a menu bar (File, Edit, Navigate, Search, Project, Run, System Profiler, Window, Help) and a toolbar. On the left, there is a "Navigator" pane showing a file tree with folders like "tmp" and "x86", and files like "hello.kev", "test.kev", "trace_128.k", "trace_all_ev", ".cdtproject", and ".project". Below the navigator is a "Target ..." pane showing a list of processes on "localhost (localh...)", including "procnto (1)", "tinit (2)", "pci-bios (4099)", "slogger (4100)", "io-usb (4101)", "io-hid (4102)", "devc-con-hid", "devb-eide (82)", "umass-enum (16394)", "pipe (16394)", "mqueue (2049)", "phfont (20481)", "devb-fdc (901)", "devc-ser8250", and "io-net (90127)".

The main area is divided into several sections:

- Summary**: Contains a "System Activity" section with a pie chart and a table. The table shows the following data:

Name	%	Time
Idle	85.0%	19sec 51ms
User	13.6%	3sec 125ms
System	1.1%	249ms
Interru...	0.3%	71ms

 To the right of the table, summary statistics are listed: Total Log File Time: 22sec 960ms, CPUs: 1, User Time: 3sec 125ms, and System Time: 19sec 834ms.
- Process & Thread Activity**: Features a bar chart showing "Event Count" (black bars) and "CPU 1 Usage" (green bars) over time. Below the chart is a table with the following data:

Process Name	Running Time	Ready Time	Blocked Time	Num Kernel Calls	Num Messa...
devb-eide	521ms	743ms	3min 51sec	38132	11001
io-graphics	162ms	19ms	1min 8sec	5266	1887
mozserver	139ms	117ms	1min 10sec	4836	4250
- Trace Event L...**: A pane showing "Data of test.kev" with a table of events:

Event	Time	Owner	Type	Data
0013	4us		Create Thread	pid 1 tid 6
0014	5us	procnto-instr Thread 6	Receive	pid 1 tid 6
0015	5us		Create Thread	pid 1 tid 7
0016	5us	procnto-instr Thread 7	Receive	pid 1 tid 7

WindRiver Workbench

The screenshot displays the WindRiver Workbench interface for a project named 'ball/main.c'. The main window is titled 'Device Debug - ball/main.c - Wind River Workbench'. The interface is divided into several panes:

- Code Editor:** Shows the source code for 'main.c'. The code includes a function signature 'BALL * p;', a call to 'ballNew (HARD, hardBounce, hardCollide, ballMove, hardShow);', a call to 'ballPlaceMovable (p);', and a 'return p;' statement. A comment below the code reads: '/* * hardBounce - bounce incoming ball by 180% (back where it came from) *'.
- Debug Console:** Shows the execution stack with 'main() - main.c:' selected. The console also displays 'tMain : 0x616fe5e8'.
- Breakpoints:** A breakpoint is set at '/ball/main.c:88 (*Planted*') and is currently active.
- Memory Renderings:** A table of memory addresses and their contents is shown. The first row is highlighted:

Address	Value	Comment
0x60369F10	0x60369F10	<Traditional>
0x60369F20	2020207C 20202020 202A2020 7C202040	* @
0x60369F30	2020207C 20202020 20202020 7C202020	
0x60369F40	2020207C 20202020 2020204F 7C202020	0
0x60369F50	2A20207C 20202020 20202020 7C202020	*
0x60369F60	2020207C 20202020 20202020 7C202020	
0x60369F70	2020207C 20202020 20202020 7C202020	
0x60369F80	2020207C 20202020 2020204F 7C202020	* @

The bottom status bar indicates '55M of 119M' memory usage.

Zealcore SystemDebugger

System Debugger 1.1 (c) ZealCore Embedded Solutions AB

File Navigate Search Window Help

System Navigator

- Example
 - logset
 - example1.log
 - printouts.txt

Overview Log... Logmarks

Example

Example/logset

Properties

Property	Value
TaskSwitch	
Date	2007-03-28 11:56
LogFile	example1.log
LogSet	Example/logset
Resource1	GenericTask[idle,L
Resource1	GenericTask[worke
Timestamp	117508656107600

logset

Format: Y-M-D h:m:s ns

88 -98952126 -68980689 -14999994 +15040371 +57010431 +88016901 +12

2007-03-28 11:56:01 076005250

- idle
- interr_A
- interr_B
- monitor
- worker1
- worker2

logset

Match: Format: Y-M-D h:m:s ns

A	Time	Type	Logfile	Properties
	2007-03-28 11:56:00 900004561	TaskSwitch	example1.log	GenericTask[worker2,LogSession=Example/logset]
	2007-03-28 11:56:00 915001023	TaskSwitch	example1.log	GenericTask[idle,LogSession=Example/logset]
	2007-03-28 11:56:00 930001562	TaskSwitch	example1.log	GenericTask[monitor,LogSession=Example/logset]
	2007-03-28 11:56:00 935058962	MeasureEvent	example1.log	20.3456
	2007-03-28 11:56:00 940030550	TaskSwitch	example1.log	GenericTask[interr_A,LogSession=Example/logset]

logset

Format: Y-M-D h:m:s ns

MeasureEvent TaskSwitch

2007-03-28 11:56:01 076005250

003688 -98952126 -68980689 -14999994 +20995873 +58007273 +99105952 +

Intel VTune

VTune(TM) Performance Tools - Call Graph Results [localhost] - Thu Jan 24 19:05:26 2008 - Intel(R) Software Development P...

File Edit Navigate Search Project Tuning Run Window Help

Tuning Activities:

- First Use - Run this first.** Find where your program is spending its time.
- Call Graph -** Determine program flow and critical call sequences. High overhead ~8X slower. [Learn more...](#)
- Sampling -** Identify the elements of code that use the most processor time and configure the monitored events.

Call Graph Results [localhost] - Thu Jan 24 19:05:26 2008

Process: /opt/intel/vtune/samples/gsexample/gsexample2a; PID:19457; Size:3

Function	Calls	Self Time	Total Time	Self Wait...	Total Wait...	Class	Module Path
GenDenormals	32,031	131,792	163,159	0	0		/opt/intel/vtune/samples/gs
__intel_new_proc_init.H	1	3	3	0	0		/opt/intel/vtune/samples/gs
__intel_new_proc_init	2	0	0	0	0		/opt/intel/vtune/samples/gs
__get_cpu_indicator	1	0	0	0	0		/opt/intel/vtune/samples/gs
__libc_csu_init	1	0	0	0	0		/opt/intel/vtune/samples/gs

Show Top: 50 % Recalculate Highlight: None

Thread_0(B7D81...) → libc_start_main → main → { feof, fclose, GenDenormals, fopen, time, call_gmon_start }

libc_csu_init → call_gmon_start

Graph Call list

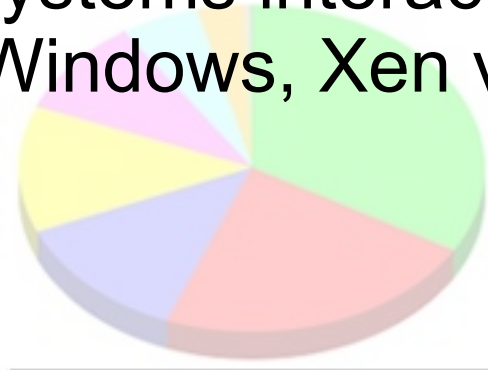
Run 1

Process	Time	Count	Self Time	Total Time
12	Thu Jan 17 16:29:32 2008 - Sampling Results [gabpc]	14524	10.084	0.08%

Processes

Challenges

- Variety of data providers (kernel markers, SystemTap, DTI, printk, logging API, gcc coverage and profiling...).
- Heterogeneous systems interacting (Linux kernel, QNX, VxWorks, Windows, Xen virtualization, JVM...).



Challenges

- Multi-core systems
- Distributed systems
- Real-time systems
- High-performance systems
- Low overhead tracing for all of the above
- Multi gigabytes traces



Challenges

- Integrate with different views or paradigms (UML diagrams, VM, RPC).
- Build more advanced analysis plugins (cache, RAM, virtual memory, garbage collector, real-time response, critical path, load leveling across CPUs, disks, computers).
- Mix and match data providers, data collectors and data analysis within the same framework.



Conclusion

- In depth study of the existing systems.
- Clear picture of the unmet needs of demanding users.
- Find common and complementary functionality among different tools.
- Common framework for tool interoperability.
- Needs for new tools.

