Content

• Problem faced
• Project objectives and organization
• Low disturbance multi-level observation
• Centralized data store and pattern identification
• Summary of results and applications
• Future work
Sophistication

- Powerful hardware with multiple parallel cores, specialized processors, caching, non-uniform memory access, tracing and debugging hardware. Chips with more than 1G transistors.
- Everything is computerized and connected, from vehicles to communication devices to earphones and thermostats.
- Multiple sources of data, sensors, wireless networks, intranet, Internet, GPS satellites.
- Online sophisticated applications with multiple threads, virtualization, real-time constraints...
- Software operating system, applications, libraries with tens of millions of lines of source code.
Complexity

• Any complex software system contains errors and vulnerabilities!
• How to verify if the system is working as intended?
• Why is the system slow? Where is the bottleneck?
• Why do we get this incorrect answer once in a billion times?
• Are there intrusion attempts? Did they succeed?
• Are we leaking information?
Objectives

• Observe the system behaviour with minimal disturbance, in the laboratory or in production.
  – Do not change the problem with observation.
  – Keep the attacker unaware that he is observed.

• Organize the observation data for efficient access and problem identification.

• Provide multiple anomaly detection techniques.

• Develop operating system level anomaly detection and protection.
Overview

Tracing and Monitoring Framework

Data Collection
LTTng

Client Side
User 1  User 2  User n

Server side
Mail  Web  File  DNS

Centralized Database (state history)

Detection
Detection_1  Detection_2  Detection_n
AHLS Project Structure

Collaboration with:
- University of Toronto
- Ecole Polytechnique
- Concordia University
- DRDC
- DND
- Scientific Literature
- Linux Community
- Eclipse Community
- ETS
- ERICSSON

Sponsors:
- AHLS
- DND-NSERC
- NSERC
- ERICSSON

Department of Computer and Software Engineering
Ecole Polytechnique de Montréal
AHLS Tracks

• Track 1: Scalable Observation infrastructure - Low disturbance multi-level observation and production of enhanced data. Polytechnique, Michel Dagenais.

• Track 2: Scalable Observation infrastructure - Advanced host-based Centralized data store and software pattern identification. Polytechnique, Michel Dagenais.

• Track 3: Scalable Detection infrastructure - Harmonized Anomaly Detection Techniques. Concordia University, Abdelwahab Hamou-Lhadj.

• Track 4: Scalable Detection infrastructure - Knowledge base for the Linux kernel. University of Toronto, Ashvin Goel.

Related efforts: Cloud Tracing Profiling and Debugging (CTPD)

• Industrial partners: Ericsson, EfficiOS.
• Financing: NSERC, Prompt.
• Academic participants: Ecole Polytechnique, Ecole de Technologie Superieure.
• Tracks:
  • Tracing the whole hardware infrastructure.
  • Cluster/cloud level monitoring.
  • Cluster level modeling and analysis
  • Integration of Tracer in Cloud Computing Environment
Related efforts: Real-Time Tracing (RTT)

- Industrial partners: CAE, Opal-RT.
- Financing: NSERC, CRIAQ.
- Academic participants: Ecole Polytechnique, Concordia University.
- Tracks:
  - Tracing Real-Time Avionics Systems.
  - Analysis of Real-Time Avionics Systems.
  - Trace Abstraction for Real-Time Avionics Systems
  - Visualization of Avionics Systems Traces
Software Tools: LTTng

• Open source project started at Polytechnique.
• More than 90 contributors from over 20 different organizations.
• Available in most major Linux distributions (Red Hat, Ubuntu, Suse, Debian...).
• Used in several commercial products.
• Commercial support available from EfficiOS.
• High performance, low overhead, industrial strength tracing system.
• New algorithms developed and tested at Polytechnique, integrated at EfficiOS and subsequently available to Ericsson, Red Hat...
Software Tools: Tracing and Monitoring Framework (TMF)

• Open source project started at Ericsson based on LTTV from Polytechnique. Part of the Eclipse project.

• Used in several commercial products at Ericsson, Mentor Graphics, Intel...

• Flexible and efficient trace analysis and viewing tool for huge traces.

• New algorithms developed and tested at Polytechnique and Concordia University, merged and integrated by the TMF team at Ericsson.

• Significant outside contributions are starting to appear from Google and others.
Software Tools: Polarsys

• Eclipse Industry Working Group for the creation and long term support of Open Source tools for the development of embedded systems.
• Members: Airbus, Astrium, CEA, Ericsson, Thales...
• Very long term support, on the scale of several decades, to support airplanes, helicopters, ships...
• For aerospace, defence and security, energy, health care, telecommunications, transportation...
• LTTng and TMF among the technologies included in Polarsys.
Open Source / Open Innovation

• Custom development: full control but very high costs.

• Commercial Off The Shelf (COTS): very rapid development of popular features, low initial cost, little control on roadmap or future cost (vendor lock-in).

• Open Innovation, the best of both:
  – Rapid development of popular features.
  – Benefit for free from features developed by others.
  – Commercial support and custom development available in a competitive market.
  – Cost proportional to the support level and custom features requested.
Modus operandi

• Problems identification and prioritisation with DRDC/DND and industrial partners.
• M.Sc., Ph.D. and PostDoc students work on these difficult problems with input from partners.
• Research associates helps the graduate students to integrate their new proposed algorithms in the toolchain for validation and optimisation.
• The best algorithms are added to the LTTng toolchain with support from industrial partner's R&D engineers (e.g. Ericsson TMF group and EfficiOS).
Maturity levels

• ML-1: Conceptual. Concepts were defined, likely feasible.
• ML-2: Early prototype. Works in some cases, in a very limited environment.
• ML-3: Prototype. Works in many cases, in a limited environment.
• ML-4: Early product. Works in most cases.
• ML-5: Industrial strength product. Works in almost all cases. Well tested.
Low disturbance multi-level observation

- High level system and network monitoring (OpenNMS, Nagios).
- Linux Kernel and user-space level tracing (Perf, Ftrace, SystemTap, LTTng).
- Windows tracing (Event Tracing for Windows / ETW).
Raw observations: trace events

```
[13:58:29.128909723] (+0.000002475) sys_read: { 0 }, { "firefox-bin", 3363 }, { fd = 5, buf = count = 16 }
[13:58:29.128911513] (+0.000001790) exit_syscall: { 0 }, { "firefox-bin", 3363 }, { ret = -11
[13:58:29.128919672] (+0.000008159) sys_write: { 0 }, { "firefox-bin", 3363 }, { fd = 5, buf count = 8 }
[13:58:29.128921404] (+0.000001732) exit_syscall: { 0 }, { "firefox-bin", 3363 }, { ret = 8 }
[13:58:29.128922884] (+0.000001480) sys_read: { 0 }, { "firefox-bin", 3363 }, { fd = 19, buf count = 1 }
[13:58:29.128925765] (+0.000002881) exit_syscall: { 0 }, { "firefox-bin", 3363 }, { ret = 1 }
[13:58:29.128928120] (+0.000002355) sys_write: { 0 }, { "firefox-bin", 3363 }, { fd = 5, buf count = 8 }
[13:58:29.128929552] (+0.000001432) exit_syscall: { 0 }, { "firefox-bin", 3363 }, { ret = 8 }
[13:58:29.129020005] (+0.0000090453) exit_syscall: { 0 }, { "acpid", 1536 }, { ret = 1 }
[13:58:29.129025587] (+0.000005582) sys_rt_sigprocmask: { 0 }, { "acpid", 1536 }, { how = 0, oset = 0x0, sigsetsize = 8 }
[13:58:29.129027993] (+0.000002406) exit_syscall: { 0 }, { "acpid", 1536 }, { ret = 0 }
[13:58:29.129030188] (+0.000002195) sys_poll: { 0 }, { "acpid", 1536 }, { ufds = 0x7FF2A055D meout_msecs = 0 }
[13:58:29.129032570] (+0.000002382) exit_syscall: { 0 }, { "acpid", 1536 }, { ret = 0 }
[13:58:29.129033929] (+0.000001359) sys_rt_sigprocmask: { 0 }, { "acpid", 1536 }, { how = 1, oset = 0x0, sigsetsize = 8 }
[13:58:29.129035144] (+0.000001215) exit_syscall: { 0 }, { "acpid", 1536 }, { ret = 0 }
[13:58:29.129037520] (+0.000002376) sys_read: { 0 }, { "acpid", 1536 }, { fd = 4, buf = 0x7FF = 24 }
```
Trace collection performance

• User-space tracing in user space, no system call.
• “Unlikely if” for tracepoint activation.
• Efficient binary format being optimised and standardized as the Multi-Core Association Common Trace Format. No formatting!
• Per CPU buffers with local lockless atomic operations.
• Read Copy Update (RCU) synchronisation for configuration information (tracepoint activation, multiple sessions...).
• Zero-copy trace recording on disk.
• Tracepoints may be inserted even in interrupt and NMI contexts.
• Efficient timestamping (rdtsc) even from virtual machines.
• Most efficient and flexible tracer available!
Capture all observations

- SNORT, AppArmor and other systems studied send their results to Syslog.
- Syslog was modified to include LTTng as one possible output channel. Provides a bridge from all these applications to LTTng.
- New converter, in collaboration with Google, from Windows ETW traces to LTTng Common Trace Format (Multi-Core Association standard).
- From other projects:
  - support for hardware assisted tracing, instrumentation of KVM virtual machines.
Multi-user concurrent sessions

[Diagram showing interactions between clients, sessiond, and ust-consumerd for Alice and Bob, with arrows indicating spawn, command, create shm, consume, write, and buffer connections.]
Userspace Tracing

Approx time by event – 1 thread (nanoseconds)

- LTTng UST: 280 nanoseconds
- Dtrace: 2400 nanoseconds
- SystemTap: 6000 nanoseconds
Userspace Tracing

Approx time by event – 8 threads (nanoseconds)

UST: 280
Dtrace: 19400
SystemTap: 56000
Merge Online Observations

- All observations on Linux systems are efficiently collected through LTTng with precise timestamps.
- New algorithm to wait until buffered data is flushed to insure an ordered merge of all events.
- New algorithm to incrementally compute drift and offset, between independent clocks, used for different traces (virtual versus physical machine, co-processors, different nodes...).
Synchronisation

- New linear incremental algorithm to compute the clock offset and drift between two traces.
- Send and receive events are matched by TCP sequence number in a hash table and are incrementally added to build the upper and lower convex hull bounds.
Results

• The work of each student is documented in publications and thesis, and the best algorithms are integrated into the toolchain.

• Extension to Syslog to integrate the observations from many other tools (SNORT, AppArmor...). Integrated into the official version and publicly available, ML-4.

• Converter from ETW to Common Trace Format, ML-3.

• LTTng base functionality, ML-5.

• LTTng live mode, merging traces online for immediate analysis, ML-4.

• Online synchronization of traces, ML-3.
Centralized data store and software pattern identification

• Available trace viewers limited to small traces (KernelShark, Chrome browser trace viewer...).

• Tracers for supercomputers start to offer some scalability (JumpShot, Paraver...).

• Enhanced algorithms for a special purpose database to store the modeled state history computed from traces.

• New architecture with modelled state databases at several levels (e.g. VM and physical machine).

• Declarative specification of modelled state from events and of patterns.

• Advanced work on multi-level views.
LTTng 2.x Low-Overhead Tracing Architecture

Host

Host-Side User Interfaces

- **Babeltrace (MIT/BSD)**
  - Trace converter
  - Trace pretty printer
  - Allow open source and proprietary plugins

- **LTTV (GPLv2)**
  - Trace display and analysis
  - Trace control
  - Allow open-source plugins

- **Eclipse Tracing and Monitoring Framework (EPL)**
  - Trace display and analysis
  - Trace control
  - Allow open source and proprietary plugins

Target

C/C++ Application
- Tracepoint
- Tracepoint Probes

Java/Erlang Application
- Tracepoint

Linux kernel
- Tracepoint
- Dynamic probes (kprobes)

Local storage
CTF†

Relayd

- **LTTng Command Line Interface (GPLv2)**

- **LTTng Session Daemon (GPLv2)**
  - Control multiple tracing sessions
  - Centralized tracing status management

- **LTTng Consumer Daemon (GPLv2)**
  - Zero-copy data transport or aggregator
  - Export raw trace data, statistics and summary data
  - Snapshots from in-memory flight recorder mode
  - Store all trace data, discard on overrun

- **Custom Control Software**
  - Interface with proprietary cluster management infrastructures

- **Memory-mapped buffers or splice, poll, ioctl**

AHLS DND-NSERC project
February 6, 2014

Department of Computer and Software Engineering
Ecole Polytechnique de Montréal
Eclipse Tracing and Monitoring Framework
<table>
<thead>
<tr>
<th>Events</th>
<th>States</th>
<th>attribute 1</th>
<th>attribute 2</th>
<th>attribute 3</th>
<th>attribute 4</th>
<th>attribute 5</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Modelled State

Time
Modelled State

Diagram showing a tree structure with various nodes and numbers.
Modelled State

(1) Modeled state

(2) Time

(3)
Modelled State

![Graph showing modelled state with different trace sizes and request times.]

- Full history
- Partial 20000 events
- Partial 50000 events
- Partial 100000 events

Trace size in MB vs. Request time in ms.
Results

• The work of each student is documented in publications and thesis, and the best algorithms are integrated into the toolchain.

• TMF base functionality, ML-4 to ML-5

• TMF State History Tree enhancements, ML-4.

• New architecture with modelled state databases at several levels, ML-4.

• Declarative specification of modelled state and patterns, ML-3 to ML-4.

• Advanced work on multi-level views, ML-2.
Resulting Toolchain

- Tracepoints already inserted throughout the Linux kernel and available to different tracers, including LTTng.
- User-space tracepoints available in several important applications such as Syslog, databases and KVM.
- LTTng for low overhead, high performance, tracing of local or remote systems, with concurrent sessions and per-uid shared buffers.
- Live tracing (access trace data while tracing) in beta (ML-4).
- Tracepoints in Java code in beta (ML-4).
Resulting Toolchain (2)

- Eclipse Tracing and Monitoring Framework viewer (TMF).
- State History Tree database to quickly navigate and display the state of huge traces.
- Possibility to use TMF as a leaner rich client, outside of Eclipse.
- Automatic synchronization of traces using independent clocks, in beta (ML-4).
- Possibility to easily define custom modelled state and associated views (ML-3 to ML-4).
Related Work

• Dependency analysis in TMF. Follow events and interactions between processes.
  – Display the critical path for performance, pending review (ML-3).
  – Identify the source of latency problems in real-time systems.

• Display the true state for Virtual Machines (really running versus suspended to run another VM), (ML-3).

• Lightweight online monitoring with LTTngTop, (ML-3).

• More efficient dynamic insertion of new tracepoints.

• GPU tracing.
Critical Flow View

- **Running**: dpkg
- **Timer**: mandb
- **Wait I/O**: Waiting for input/output operations to complete.

The diagram shows a timeline of processes with various status indicators.
# Real Performance of Virtual Machines

<table>
<thead>
<tr>
<th>ExpMigrate</th>
<th>qemu:Jessie</th>
</tr>
</thead>
<tbody>
<tr>
<td>VCPU 0</td>
<td>bur</td>
</tr>
<tr>
<td>VCPU 1</td>
<td>bur</td>
</tr>
<tr>
<td>VCPU 2</td>
<td>burnP6</td>
</tr>
<tr>
<td>VCPU 3</td>
<td>burnP6</td>
</tr>
<tr>
<td>qemu:jessie-clo</td>
<td></td>
</tr>
<tr>
<td>VCPU 0</td>
<td>bur</td>
</tr>
<tr>
<td>VCPU 1</td>
<td>burnP6</td>
</tr>
</tbody>
</table>

AHLS DND-NSERC project  
February 6, 2014  
Department of Computer and Software Engineering  
Ecole Polytechnique de Montréal
Statistics for interval [1330053201794942051, 1330053202795131720]

<table>
<thead>
<tr>
<th>CPUs</th>
<th>4 (max/cpu : 25.00%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Processes</td>
<td>N/A (0, 0)</td>
</tr>
<tr>
<td>Threads</td>
<td>N/A (0, 0)</td>
</tr>
<tr>
<td>Files</td>
<td>N/A (0, 0) N/A kbytes/sec</td>
</tr>
<tr>
<td>Network</td>
<td>N/A (0, 0) N/A Mbytes/sec</td>
</tr>
</tbody>
</table>

**CPU Top**

<table>
<thead>
<tr>
<th>CPU(%)</th>
<th>TGID</th>
<th>PID</th>
<th>NAME</th>
</tr>
</thead>
<tbody>
<tr>
<td>10.00</td>
<td>23844</td>
<td>23844</td>
<td>gnome-shell</td>
</tr>
<tr>
<td>5.50</td>
<td>20627</td>
<td>20627</td>
<td>firefox-bin</td>
</tr>
<tr>
<td>0.93</td>
<td>23653</td>
<td>23653</td>
<td>Xorg</td>
</tr>
<tr>
<td>0.29</td>
<td>4788</td>
<td>4788</td>
<td>epiphany-browser</td>
</tr>
<tr>
<td>0.05</td>
<td>11223</td>
<td>11223</td>
<td>kworker/2:2</td>
</tr>
<tr>
<td>0.05</td>
<td>11173</td>
<td>11173</td>
<td>kworker/0:0</td>
</tr>
<tr>
<td>0.05</td>
<td>11222</td>
<td>11222</td>
<td>kworker/1:1</td>
</tr>
<tr>
<td>0.05</td>
<td>10843</td>
<td>10843</td>
<td>kworker/3:1</td>
</tr>
<tr>
<td>0.04</td>
<td>14809</td>
<td>14809</td>
<td>hald</td>
</tr>
<tr>
<td>0.04</td>
<td>24103</td>
<td>24103</td>
<td>xchat</td>
</tr>
<tr>
<td>0.02</td>
<td>31261</td>
<td>31261</td>
<td>synergyc</td>
</tr>
<tr>
<td>0.02</td>
<td>20247</td>
<td>20247</td>
<td>emacs</td>
</tr>
<tr>
<td>0.02</td>
<td>6251</td>
<td>6251</td>
<td>emacs</td>
</tr>
<tr>
<td>0.02</td>
<td>2403</td>
<td>2403</td>
<td>soffice.bin</td>
</tr>
<tr>
<td>0.01</td>
<td>25701</td>
<td>25701</td>
<td>emacs</td>
</tr>
<tr>
<td>0.01</td>
<td>2719</td>
<td>2719</td>
<td>nmbd</td>
</tr>
<tr>
<td>0.01</td>
<td>13085</td>
<td>13085</td>
<td>icedove-bin</td>
</tr>
<tr>
<td>0.01</td>
<td>1534</td>
<td>1534</td>
<td>dbus-daemon</td>
</tr>
<tr>
<td>0.00</td>
<td>11193</td>
<td>11193</td>
<td>kworker/u:1</td>
</tr>
<tr>
<td>0.00</td>
<td>10985</td>
<td>10985</td>
<td>kworker/u:2</td>
</tr>
<tr>
<td>0.00</td>
<td>577</td>
<td>577</td>
<td>ips-monitor</td>
</tr>
<tr>
<td>0.00</td>
<td>9750</td>
<td>9750</td>
<td>ksoftirqd/3</td>
</tr>
<tr>
<td>0.00</td>
<td>17301</td>
<td>17301</td>
<td>kworker/1:2</td>
</tr>
<tr>
<td>0.00</td>
<td>23813</td>
<td>23813</td>
<td>gnome-settings-</td>
</tr>
</tbody>
</table>

**Status**

Starting display
Pause
Future Work

• Very solid foundation for tracing and monitoring
• Exploit the power of the Tracing and Monitoring Framework for more advanced analysis.
• Add optimised dynamic tracing, hardware assisted tracing, co-processor (GPU, DSP) tracing.
• Instrument other important applications and runtime environments.
• Insure scalability to K-core systems.
• Convergence of tracing, debugging, profiling and other analysis tools.
• Link tracing, monitoring and debugging activities to higher level models.
Conclusion

• Many problems can only be studied live, in production.

• LTTng and TMF are now industrial-strength and a solid foundation for future work.

• This is an excellent platform to build advanced analysis modules on top of LTTng and TMF.

• The user community is growing quickly. The interaction may be time-consuming but the benefits are significant in the long run.

• The right mix of resources is required, for fruitful collaborative research and development projects. It is an effective way to develop real solutions to real problems.