Cyber-surveillance and Protection of Critical Computing Infrastructures

AHLS DND-NSERC Project

Scalable Observation infrastructure

Michel Dagenais

Department of Computer and Software Engineering

POLYTECHNIQUE MONTRÉAL
AFFILIÉE À L'UNIVERSITÉ DE MONTRÉAL

Ottawa, February 6, 2014

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.

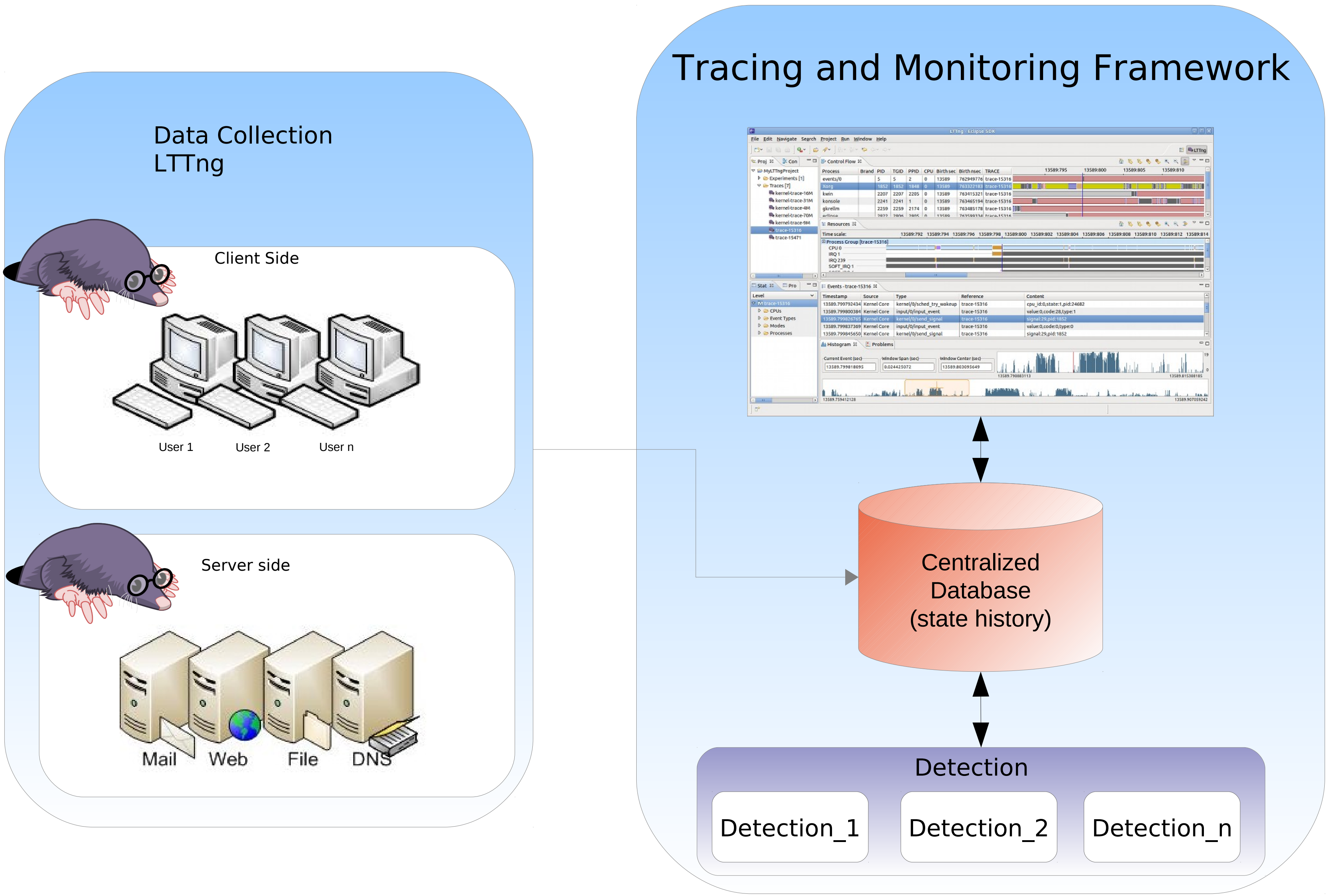
Complexity

- 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?

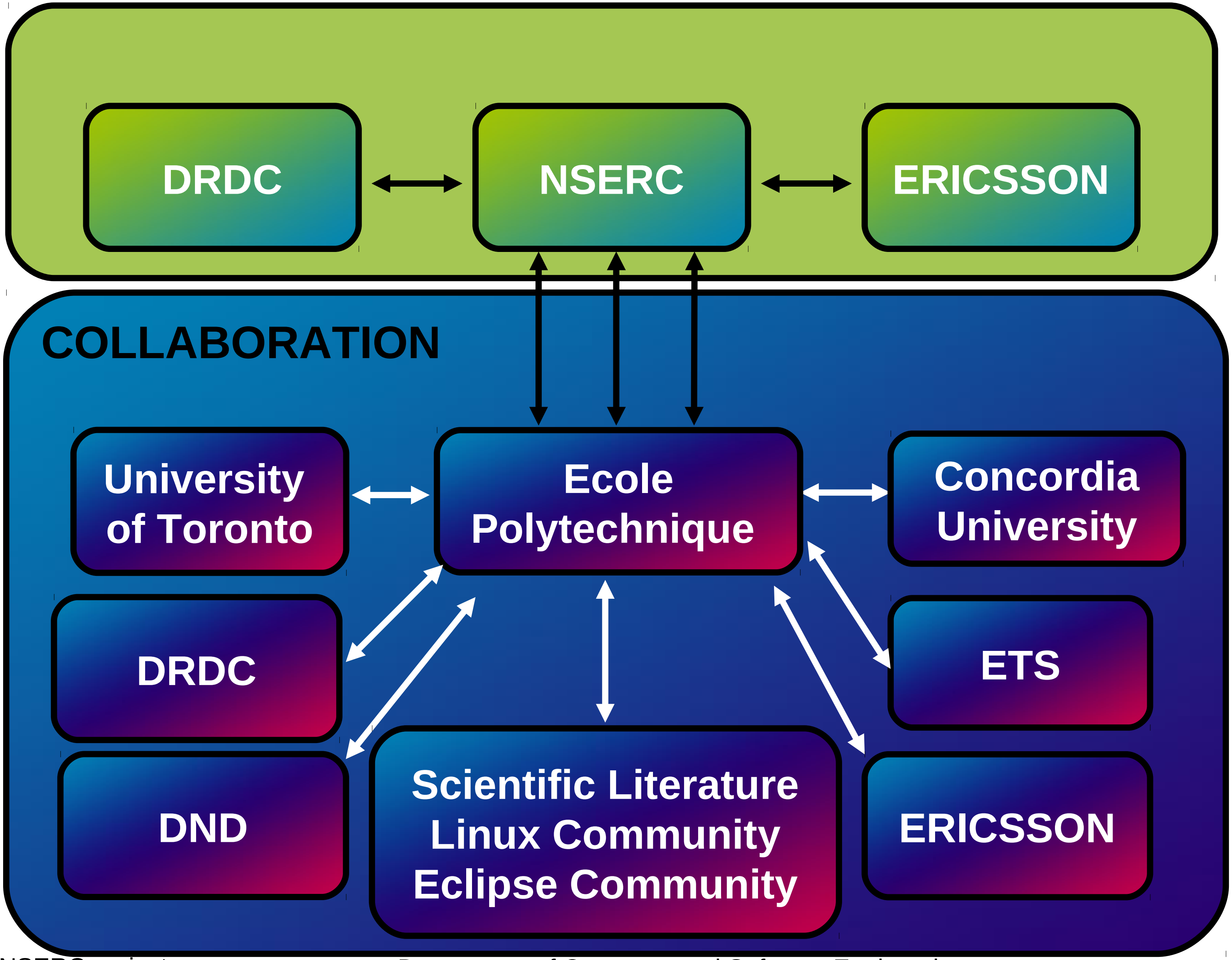
Objectives

- 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


Data Collection
LTTng
Client Side
User 1
User 2
User n
Server side
Mail
Web
File
DNS
Tracing and Monitoring Framework
Centralized
Database
(state history)
Detection
Detection_1
Detection_2
Detection_n

AHLS Project Structure



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.
- New track: Embedded Computers and Cyber-Security. Ecole de Technologie Supérieure, Chamseddine Talhi.

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).
- Intrusion Detection Systems (SNORT, OSSEC, AppArmor).
- 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] (+0.000090453) 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 = 0x7FFF2A055D
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 }
```

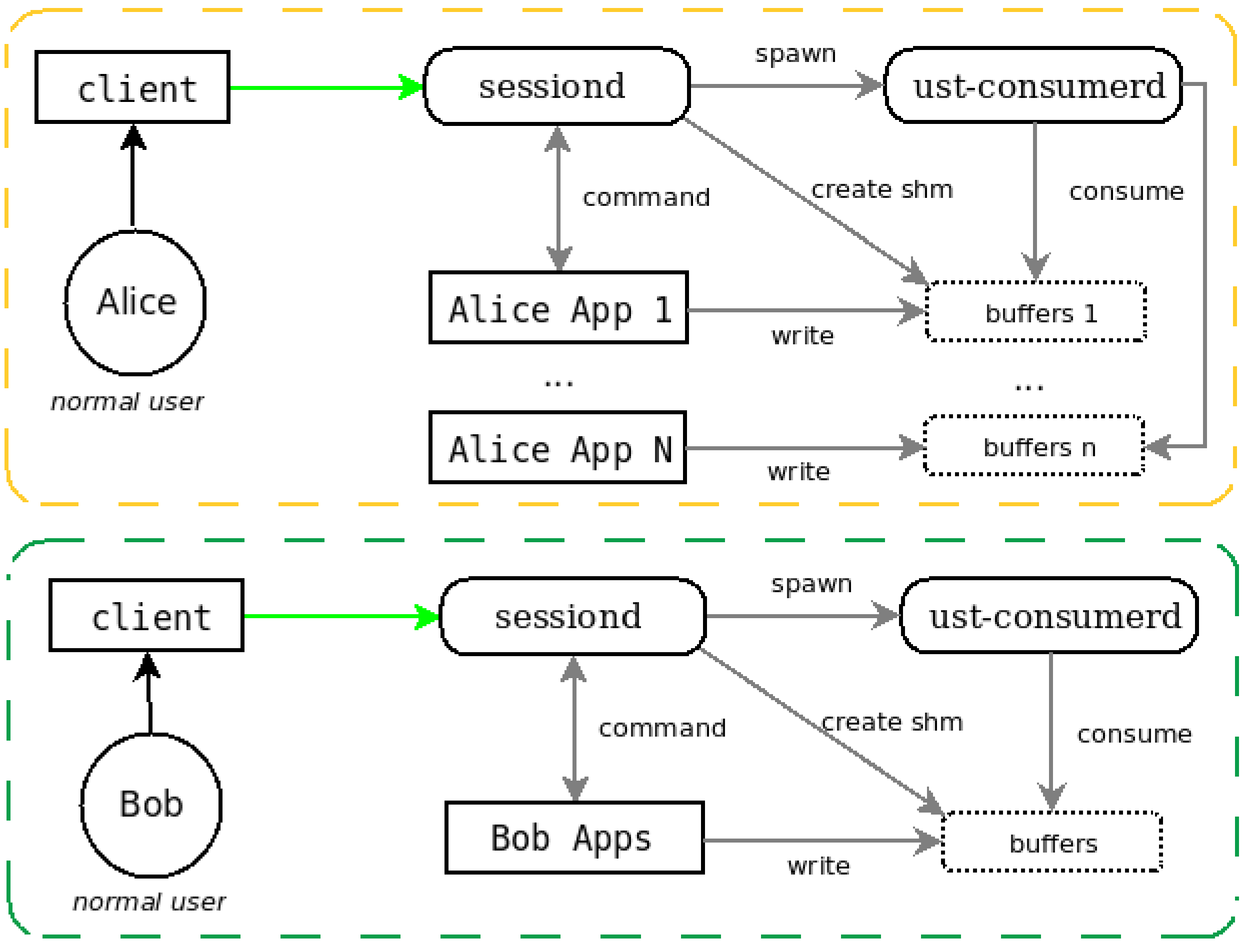
Trace collection performance

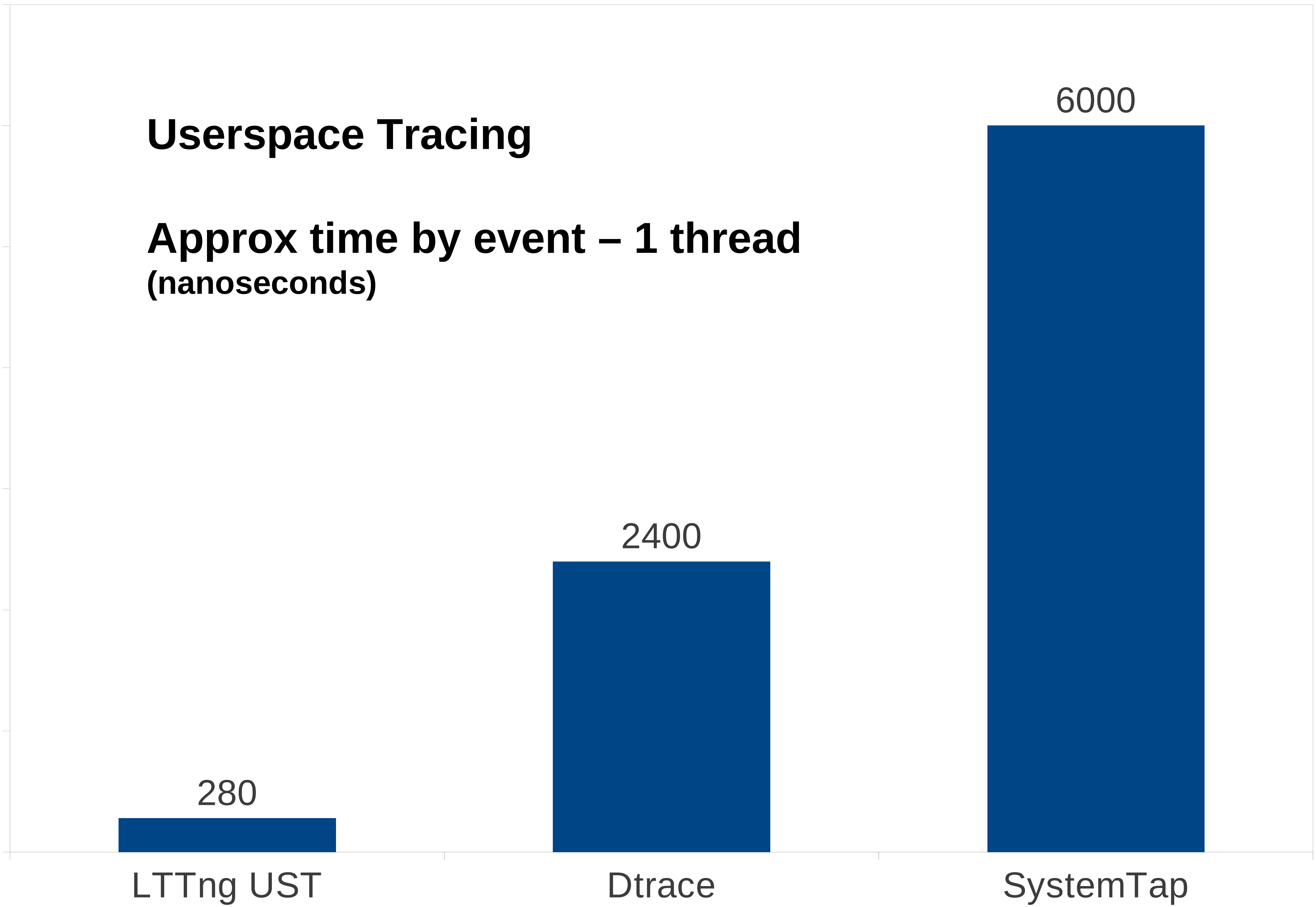
- User-space tracing in user space, no system call.
- “Unlikely if” for tracepoint activation.
- Efficient binary format being optimised and standardized as the Multi-Core Association Common Trace Format. No formatting!
- Per CPU buffers with local lockless atomic operations.
- Read Copy Update (RCU) 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!

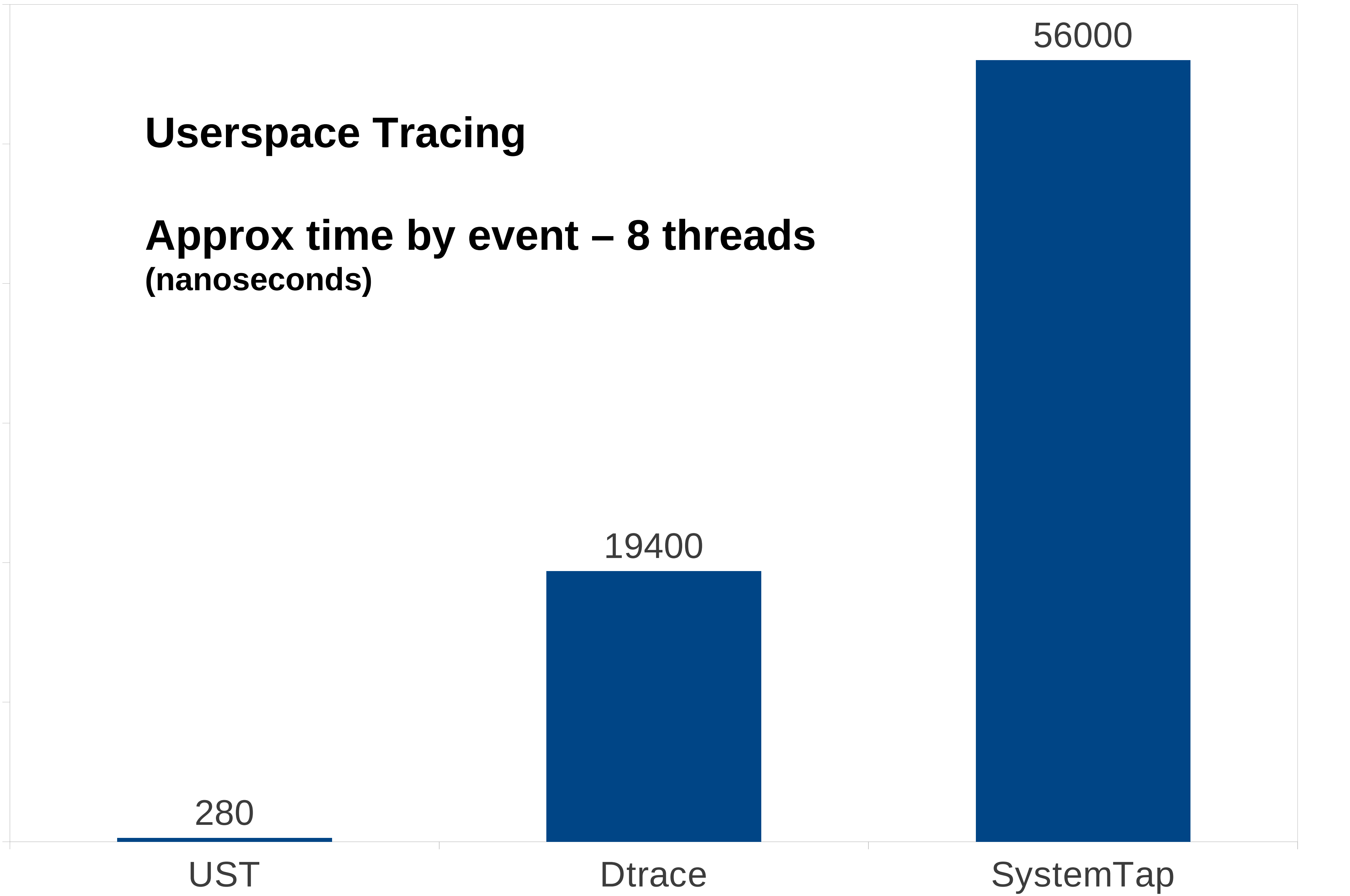
Capture all observations

- 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




Userspace Tracing
Approx time by event – 1 thread
(nanoseconds)
6000
2400
280
LTTng UST
Dtrace
SystemTap



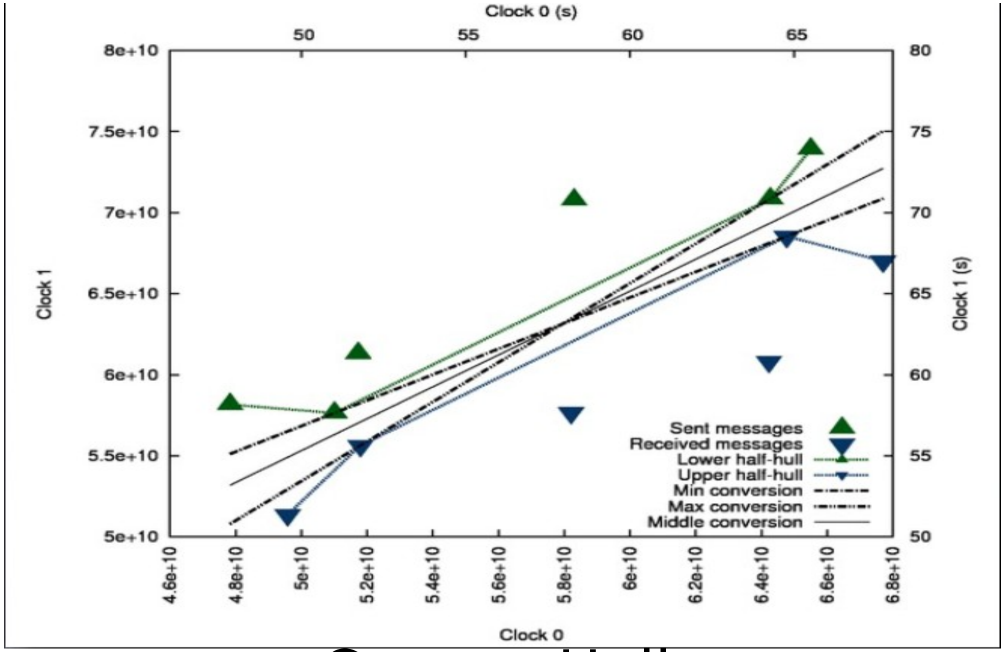
Userspace Tracing
Approx time by event – 8 threads
(nanoseconds)
56000
19400
280
UST
Dtrace
SystemTap

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.



Convex Hull

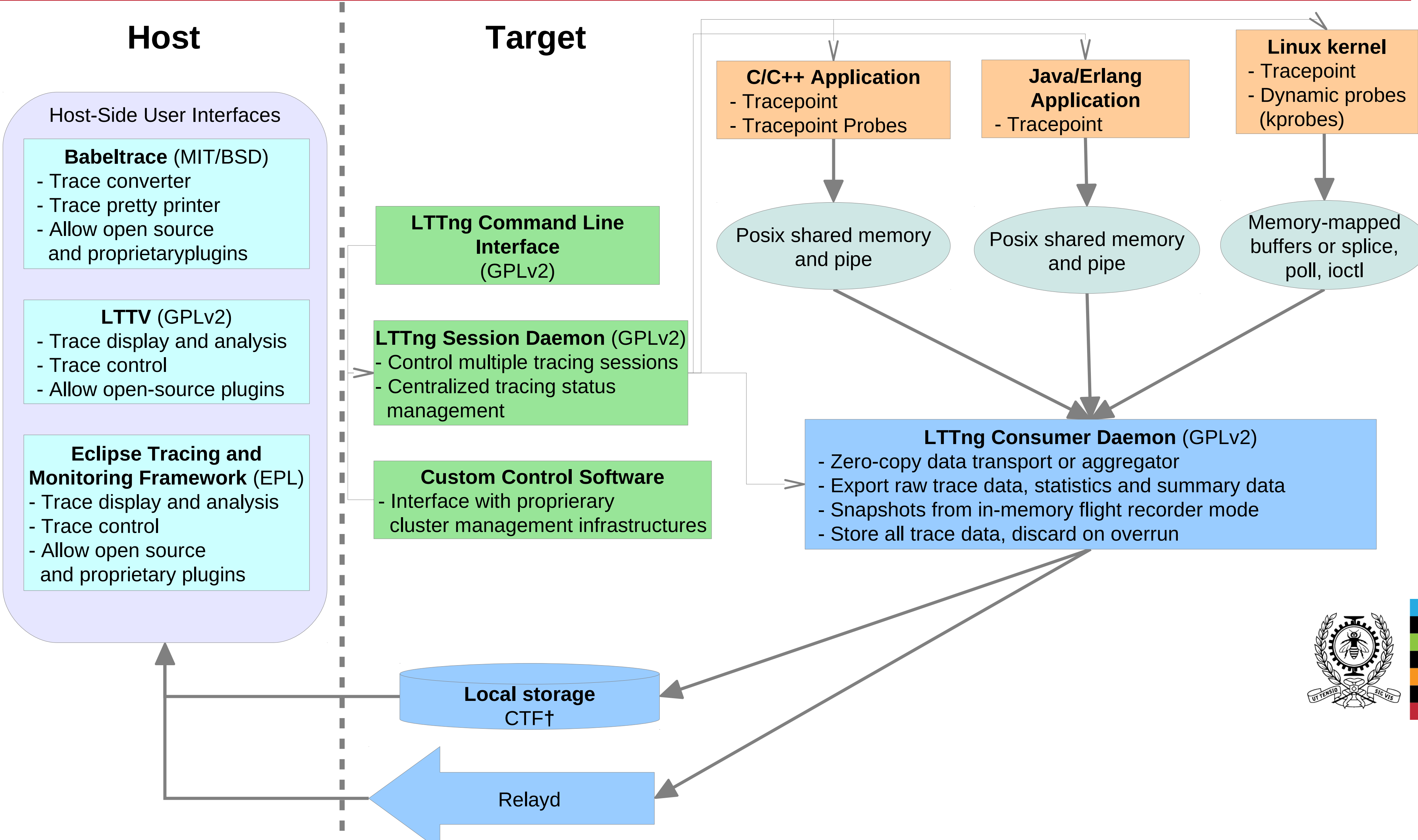
Results

- 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.

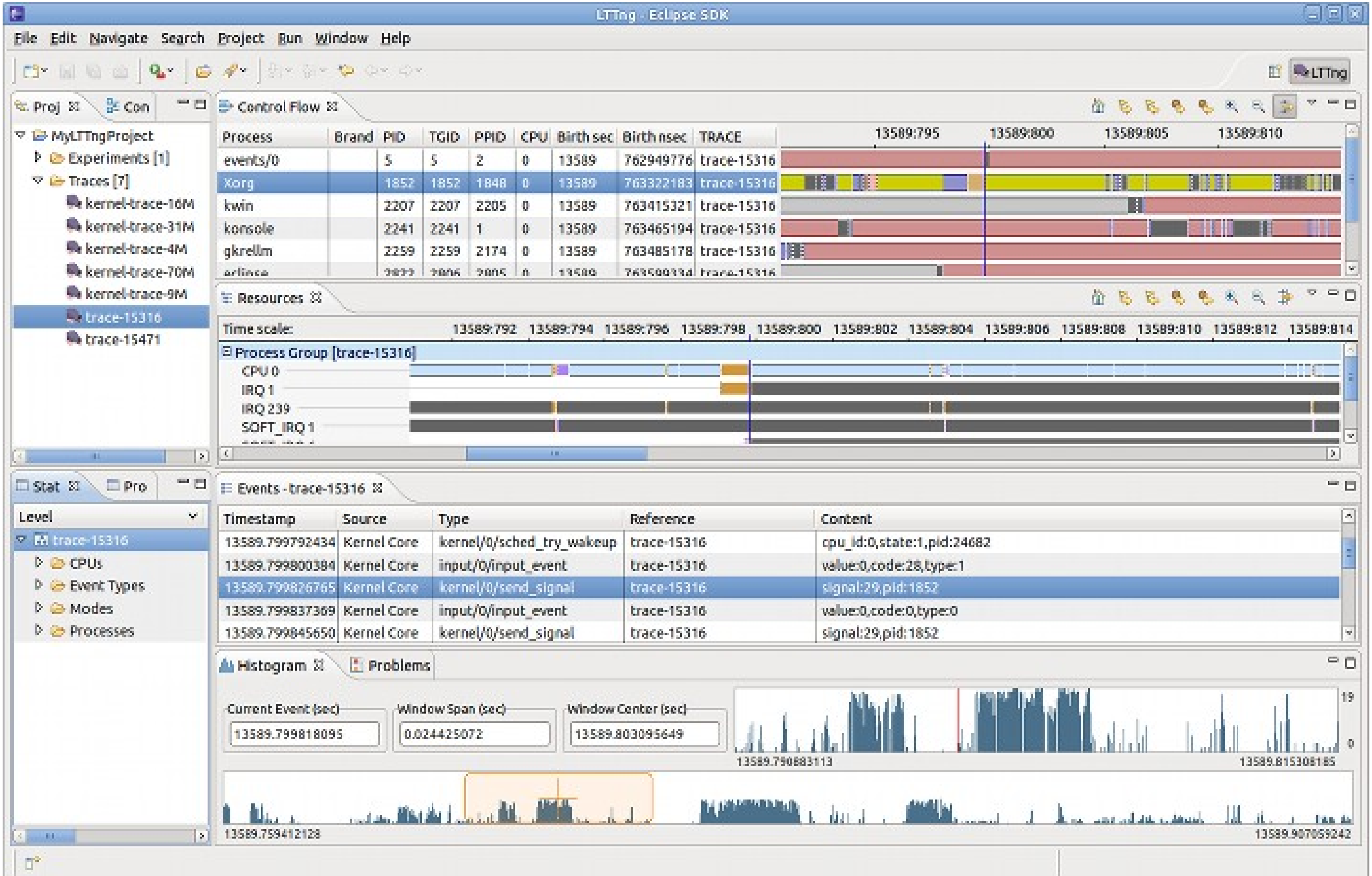
Centralized data store and software pattern identification

- 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.

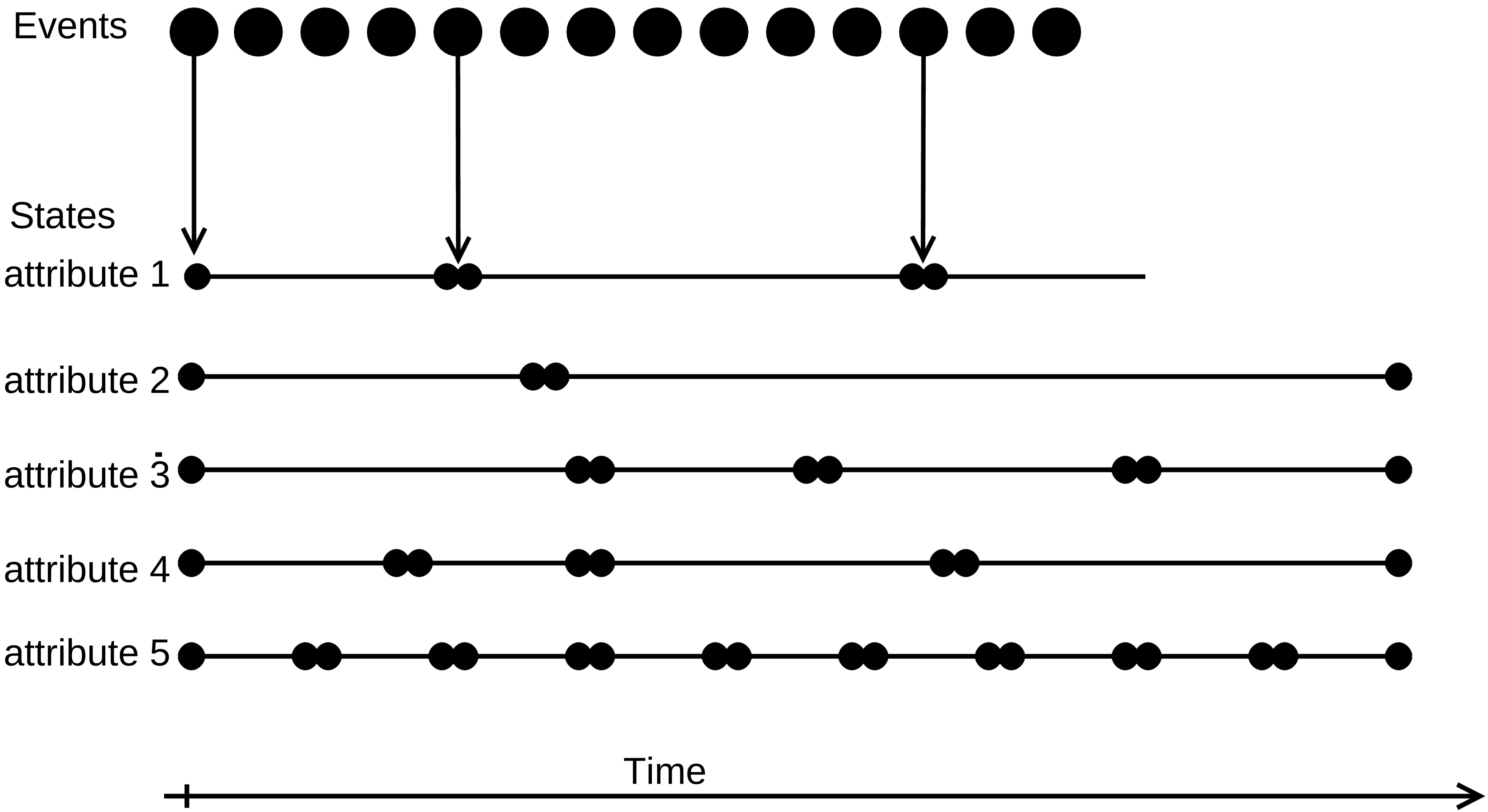
LTTng 2.x Low-Overhead Tracing Architecture



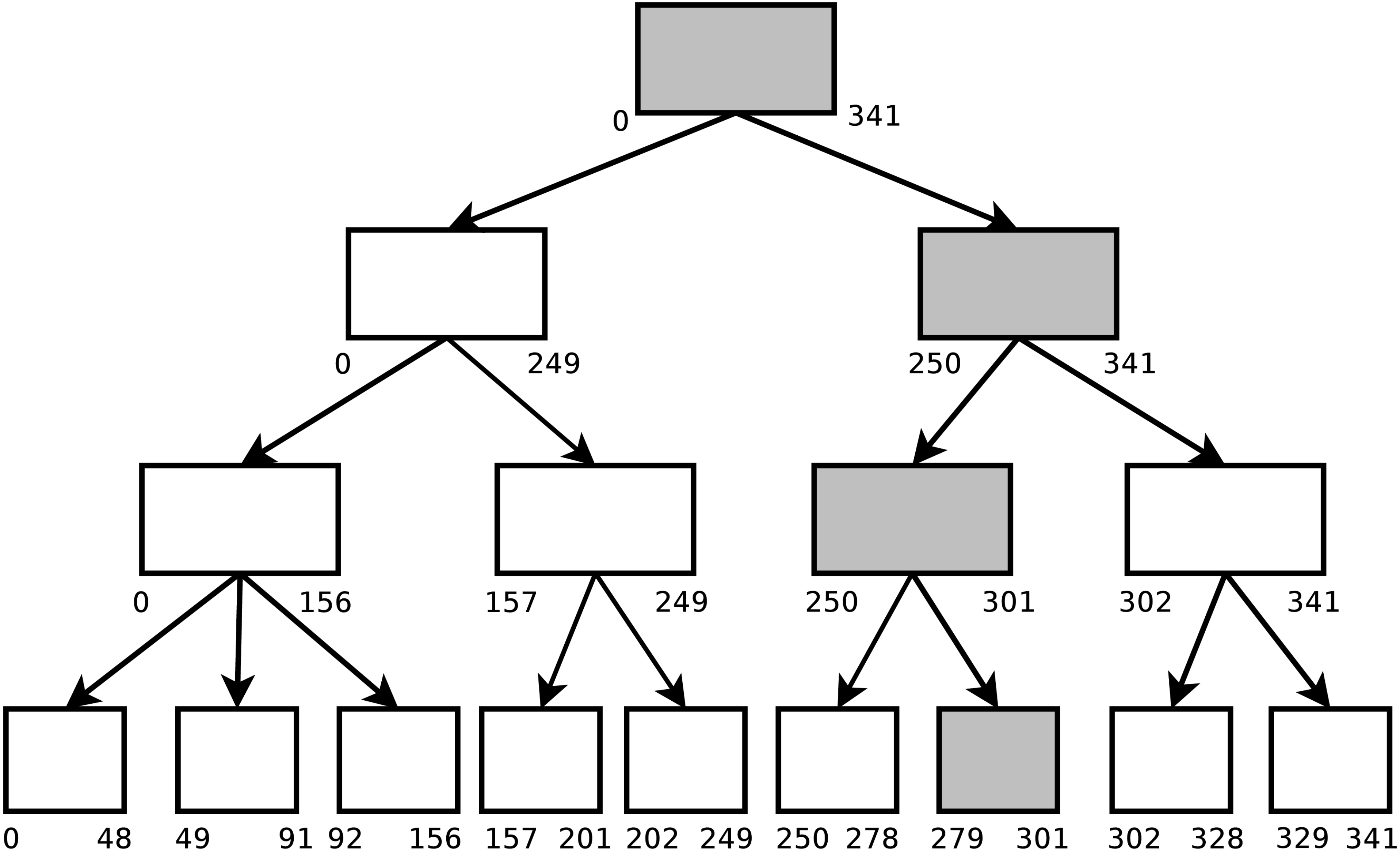
Eclipse Tracing and Monitoring Framework



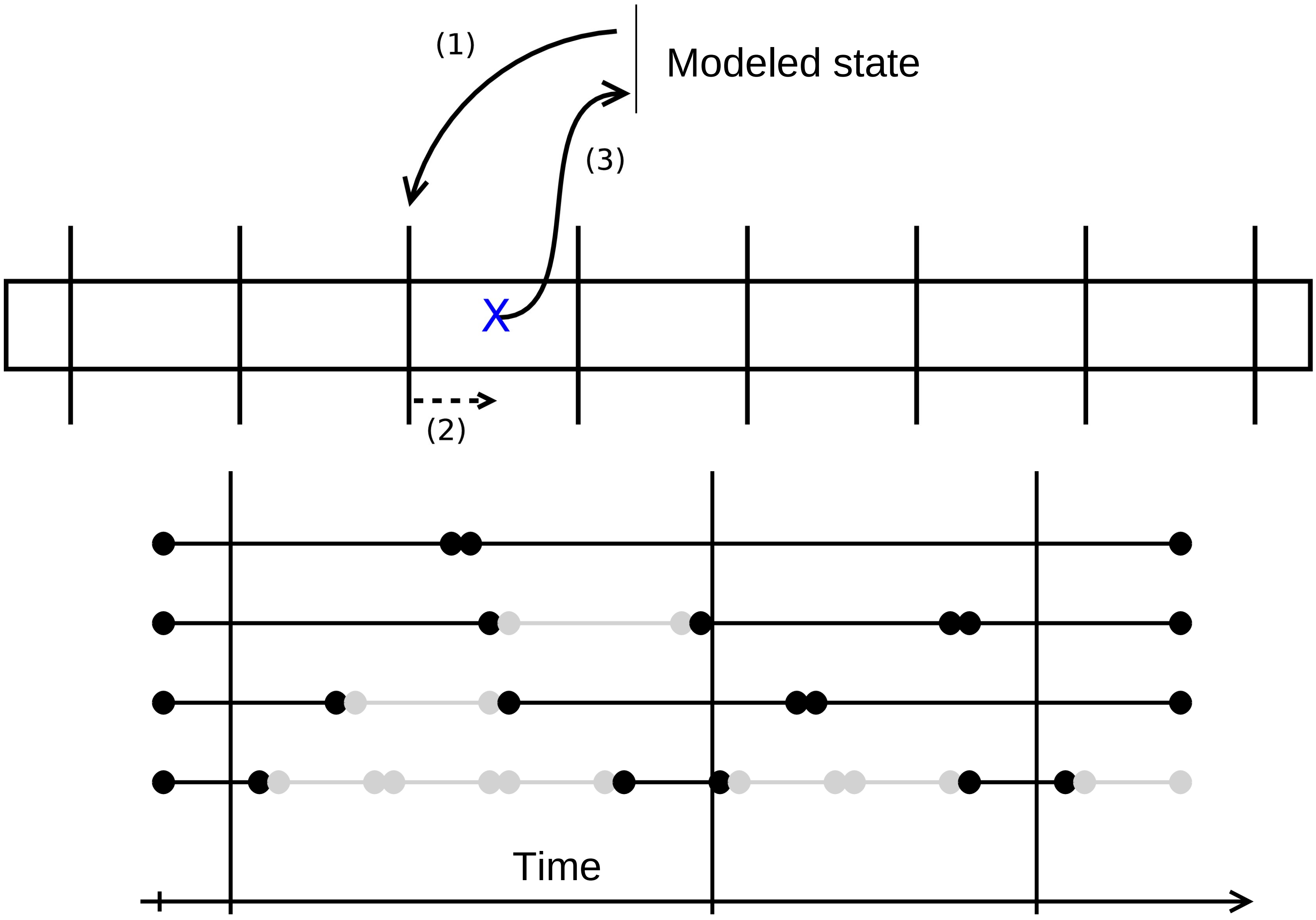
Modelled State



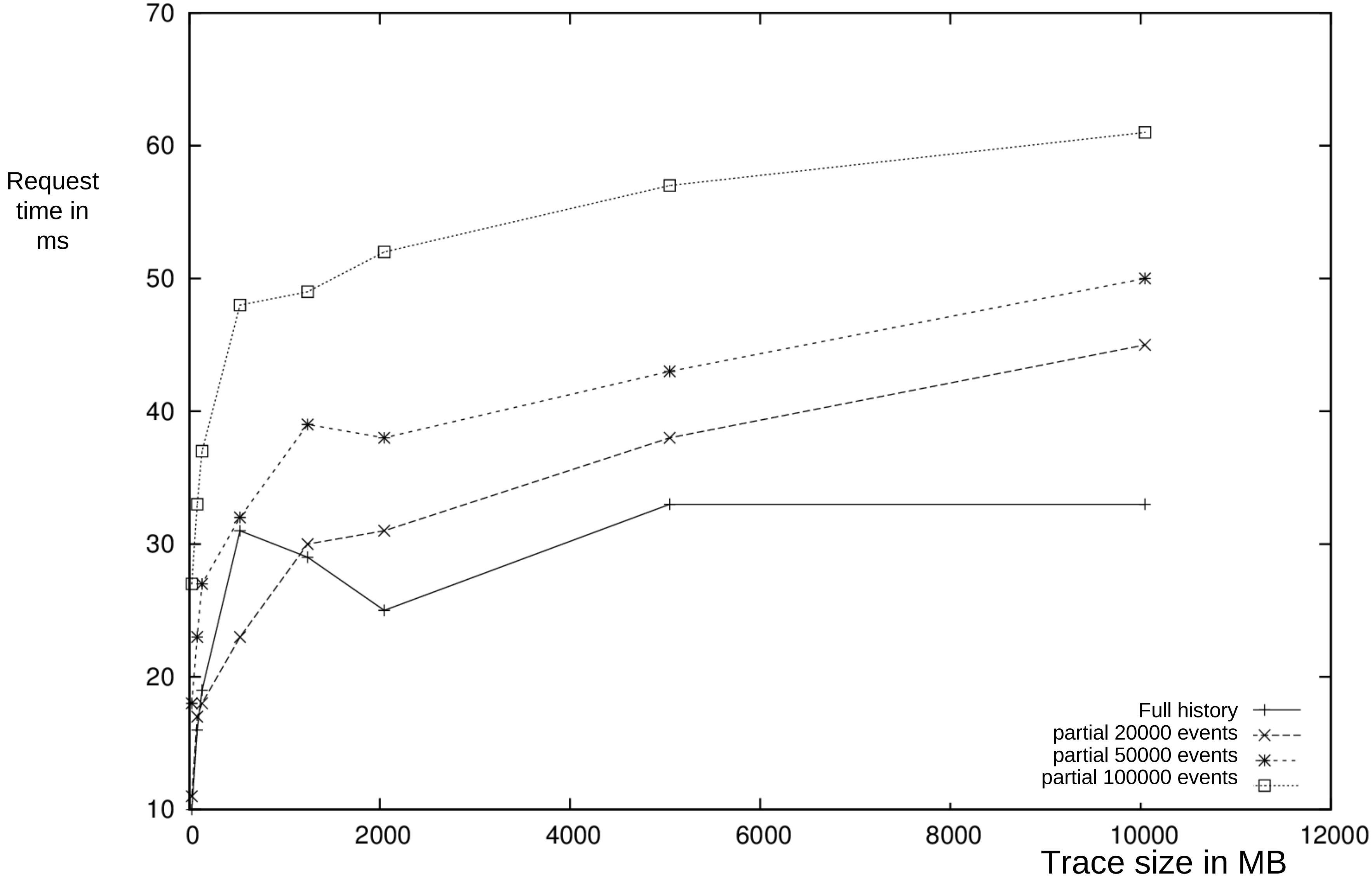
Modelled State

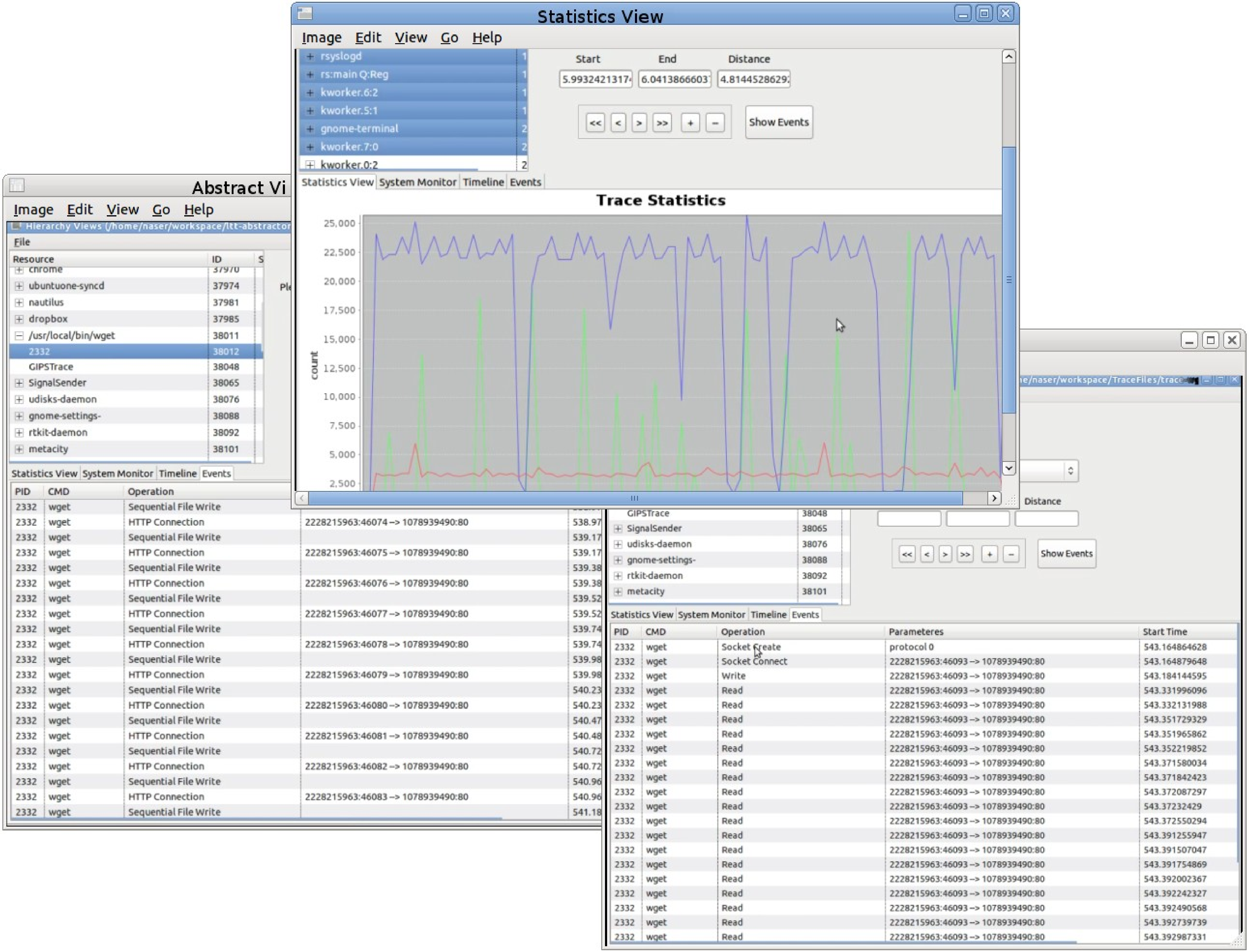


Modelled State



Modelled State




Statistics View
Image Edit View Go Help
rsyslogd
rs:main Q:Reg
kworker.6:2
kworker.5:1
gnome-terminal
kworker.7:0
kworker.0:2
Start
End
Distance
5.9932421317
6.0413866603
4.8144528629
Show Events
Statistics View System Monitor Timeline Events
Trace Statistics
count
25,000
22,500
20,000
17,500
15,000
12,500
10,000
7,500
5,000
2,500
Abstract Vi
Image Edit View Go Help
Hierarchy Views (/home/naser/workspace/ltt-abstractor
File
Resource ID
chrome 37970
ubuntuone-syncd 37974
nautilus 37981
dropbox 37985
/usr/local/bin/wget 38011
2332 38012
GIPSTrace 38048
SignalSender 38065
udisks-daemon 38076
gnome-settings- 38088
rtkit-daemon 38092
metacity 38101
Statistics View System Monitor Timeline Events
PID CMD Operation
2332 wget Sequential File Write
2332 wget HTTP Connection 2228215963:46074 --> 1078939490:80 538.97
2332 wget Sequential File Write 539.17
2332 wget HTTP Connection 2228215963:46075 --> 1078939490:80 539.17
2332 wget Sequential File Write 539.38
2332 wget HTTP Connection 2228215963:46076 --> 1078939490:80 539.38
2332 wget Sequential File Write 539.52
2332 wget HTTP Connection 2228215963:46077 --> 1078939490:80 539.52
2332 wget Sequential File Write 539.74
2332 wget HTTP Connection 2228215963:46078 --> 1078939490:80 539.74
2332 wget Sequential File Write 539.98
2332 wget HTTP Connection 2228215963:46079 --> 1078939490:80 539.98
2332 wget Sequential File Write 540.23
2332 wget HTTP Connection 2228215963:46080 --> 1078939490:80 540.23
2332 wget Sequential File Write 540.47
2332 wget HTTP Connection 2228215963:46081 --> 1078939490:80 540.48
2332 wget Sequential File Write 540.72
2332 wget HTTP Connection 2228215963:46082 --> 1078939490:80 540.72
2332 wget Sequential File Write 540.96
2332 wget HTTP Connection 2228215963:46083 --> 1078939490:80 540.96
2332 wget Sequential File Write 541.18
e/naser/workspace/TraceFiles/trac
GIPSTrace 38048
SignalSender 38065
udisks-daemon 38076
gnome-settings- 38088
rtkit-daemon 38092
metacity 38101
Distance
Show Events
Statistics View System Monitor Timeline Events
PID CMD Operation Parameteres Start Time
2332 wget Socket Create protocol 0 543.164864628
2332 wget Socket Connect 2228215963:46093 --> 1078939490:80 543.164879648
2332 wget Write 2228215963:46093 --> 1078939490:80 543.184144595
2332 wget Read 2228215963:46093 --> 1078939490:80 543.331996096
2332 wget Read 2228215963:46093 --> 1078939490:80 543.332131988
2332 wget Read 2228215963:46093 --> 1078939490:80 543.351729329
2332 wget Read 2228215963:46093 --> 1078939490:80 543.351965862
2332 wget Read 2228215963:46093 --> 1078939490:80 543.352219852
2332 wget Read 2228215963:46093 --> 1078939490:80 543.371580034
2332 wget Read 2228215963:46093 --> 1078939490:80 543.371842423
2332 wget Read 2228215963:46093 --> 1078939490:80 543.372087297
2332 wget Read 2228215963:46093 --> 1078939490:80 543.37232429
2332 wget Read 2228215963:46093 --> 1078939490:80 543.372550294
2332 wget Read 2228215963:46093 --> 1078939490:80 543.391255947
2332 wget Read 2228215963:46093 --> 1078939490:80 543.391507047
2332 wget Read 2228215963:46093 --> 1078939490:80 543.391754869
2332 wget Read 2228215963:46093 --> 1078939490:80 543.392002367
2332 wget Read 2228215963:46093 --> 1078939490:80 543.392242327
2332 wget Read 2228215963:46093 --> 1078939490:80 543.392490568
2332 wget Read 2228215963:46093 --> 1078939490:80 543.392739739
2332 wget Read 2228215963:46093 --> 1078939490:80 543.392987331

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).

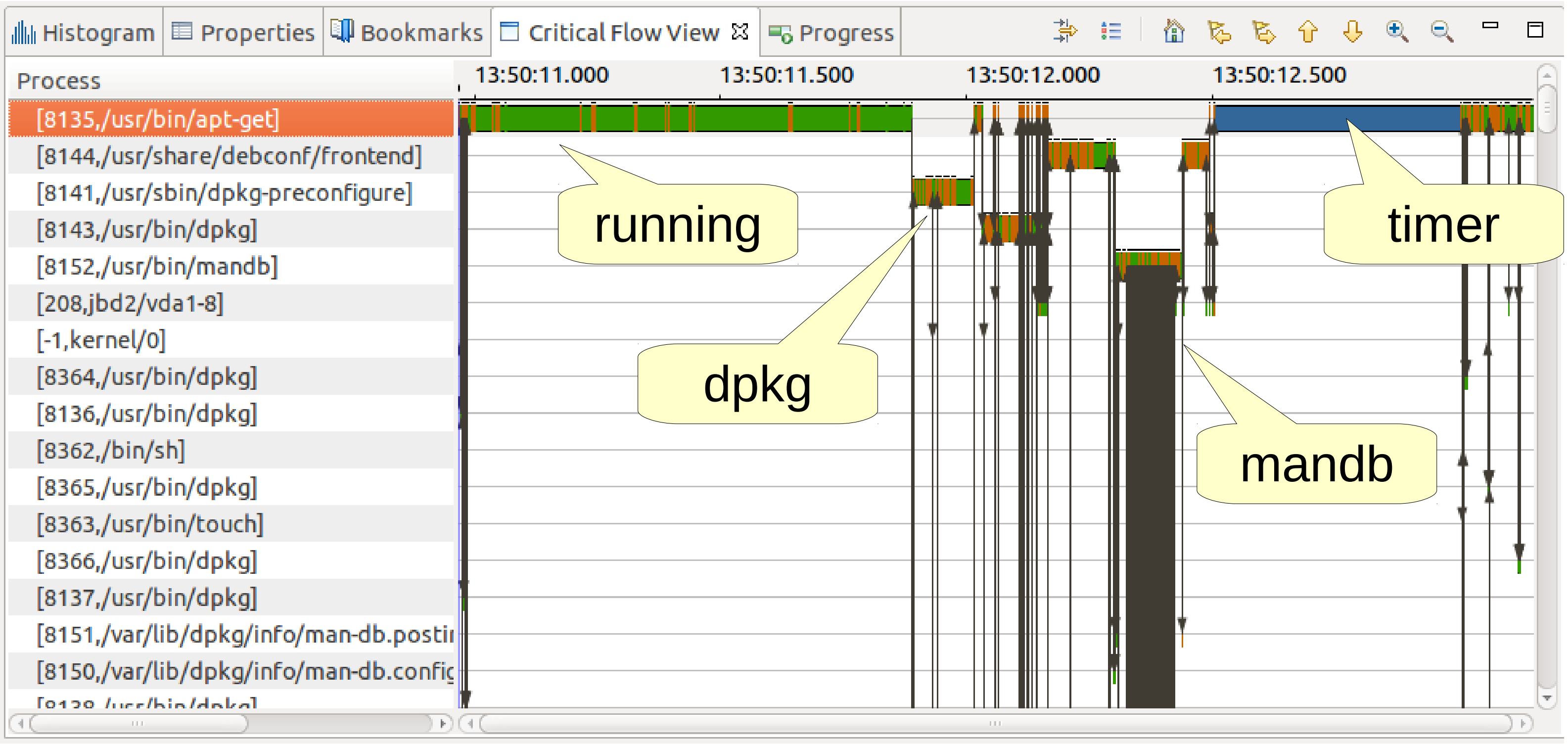
Resulting Toolchain (2)

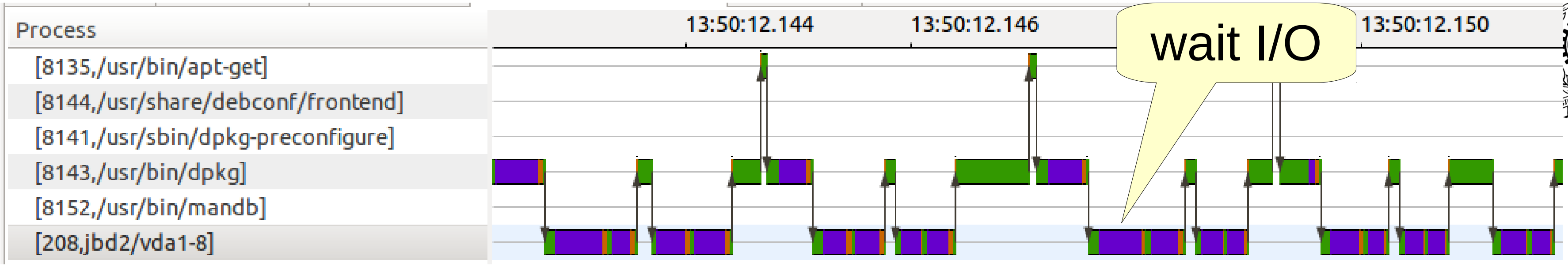
- Eclipse Tracing and Monitoring Famework viewer (TMF).
- State History Tree database to quickly navigate and display the state of huge traces.
- Possibility to use TMF as a leaner rich client, outside of Eclipse.
- Automatic synchronization of traces using independent clocks, in beta (ML-4).
- Possibility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

Related Work

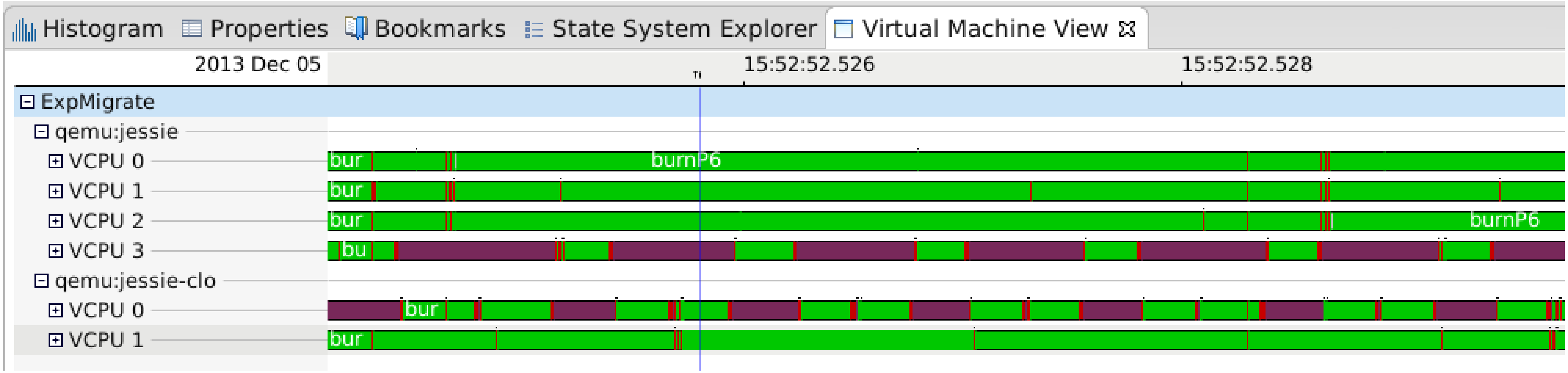
- Dependency analysis in TMF. Follow events and interactions between processes.
  - Display the critical path for performance, pending review (ML-3).
  - Identify the source of latency problems in real-time systems.
- Display the true state for Virtual Machines (really running versus suspended to run another VM), (ML-3).
- Lightweight online monitoring with LTTngTop, (ML-3).
- More efficient dynamic insertion of new tracepoints.
- GPU tracing.

Critical Flow View





Real Performance of Virtual Machines



lttngtop

Statistics for interval [1330053201794942051, 1330053202795131720[

CPUs	4	(max/cpu : 25.00%)	
Processes	N/A	(0, 0)	
Threads	N/A	(0, 0)	
Files	N/A	(0, 0)	N/A kbytes/sec
Network	N/A	(0, 0)	N/A Mbytes/sec

CPU Top

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

Status

Starting display
Pause

F2:CPUtop F3:PerfTop F6:IOTop Enter:Details q:Quit | P:Perf Pref p: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.