System Wide Tracing User Need

dominique <dot> toupin <at> ericsson <dot> com

April 2010

About me

- Developer Tool Manager at Ericsson, helping Ericsson sites to develop better software efficiently
- Background in telecommunication systems
- A standards-based communications-class server:
  - Open, standards-based common platform
  - High availability (greater than 99.999%)
  - Broad range of support for both infrastructure and value-added applications
  - Multimedia, network and application processing capabilities
  - Product life-cycle of 7 years

About me

- Improving development tools with research projects, open source tools, tool vendors and other companies
- GDB improvements, non-stop, multi-process, global breakpoint, dynamic tracepoint, core awareness, OS awareness, … with CodeSourcery
- Eclipse GDB integration, debug analysis with CDT community e.g. WindRiver
- Linux tracing research project with Ecole Polytechnique (Prof. Michel Dagenais)

About me

- Linux tracing: user space tracing, GDB integration, binary format, buffering scheme, … with EfficiOS (Mathieu Desnoyers)
- Eclipse Linux tracing integration and analysis with Red Hat
- Organizing Linux Tracing Summit:

2008: https://ltt.polymtl.ca/tracingwiki/index.php/TracingSummit2008

2009: http://www.linuxsymposium.org/2009/view_abstract.php?content_key=108

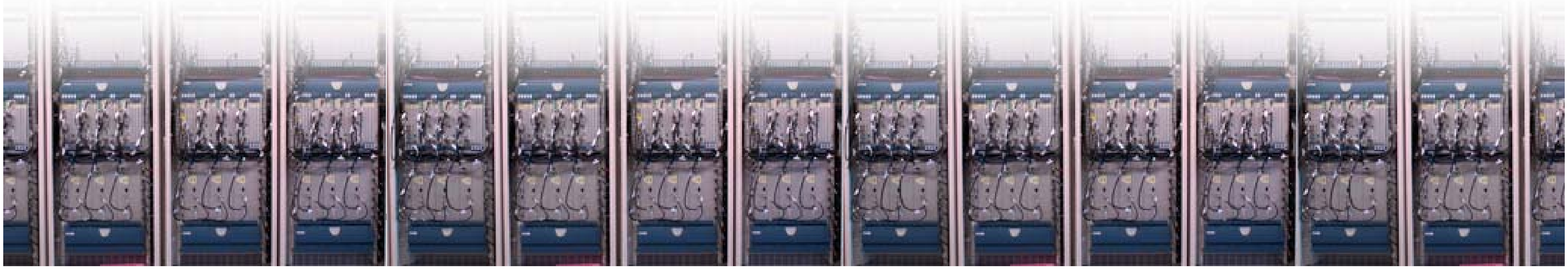
2010: http://events.linuxfoundation.org/events/linuxcon/minisummits

Some Context

- Not only enterprise use cases
- Not the amount of memory/disk like enterprise, not the small amount of data of small devices like camera
- Facilitate Linux usage in big embedded systems
- Always have host – target scenario
- Analyse trace on host without the target kernel

Some Context

- Autodesk, C2 Microsystems, Cisco, Ericsson, Freescale, Fujitsu, IBM, Mentor Graphic, MontaVista, Nokia, Siemens, Sony, ST Microelectronics, TI, WindRiver, etc.
- Linux at its best, efficient tracing solution can only benefit enterprise/IT/parallel computing



Static Tracepoint

- E.g. kernel tracepoints, trace_event APIs
- Created by designer *before compilation* at *development time*
- Static tracepoints represent *wisdom of developers* who are most familiar with the code
- Helps developers to think about tracing (using only trial-error dynamic traces is not efficient)
- The rest of the world can use them to extract a great deal of useful information *without having to know the code*

Trace Data Transport

- Trace data initially stored in shared memory buffers
- Tracing daemon then writes to the chosen trace-store:
  - circular “flight recorder” buffer
  - local disk
  - remote disk via network interface or serial port
- Streaming, i.e. live monitoring
  - CPU should be allowed to stay in sleep state in order to save energy
  - No periodic check to wake up a CPU
  - Able to analyse/view data on host while it is gathered, impacts the tracer and the analyser

Trace Data Transport

- Event compactness decreases overhead, e.g. PID, event size, etc. should be optional
- Maximum event size should be configurable
- Self describing trace format
- Generate events with arbitrary number of arguments i.e. variable event sizes

Trace Data Transport

- Trace buffers flushing in core dump when process crash, post mortem analysis
- Flight recorder mode: event backlog size should be configurable per event group e.g. IRQ, signals
- Huge traces > 10 GB
- Can be efficiently accessed based on time e.g. binary search
- Multi-node tracing

Scalability

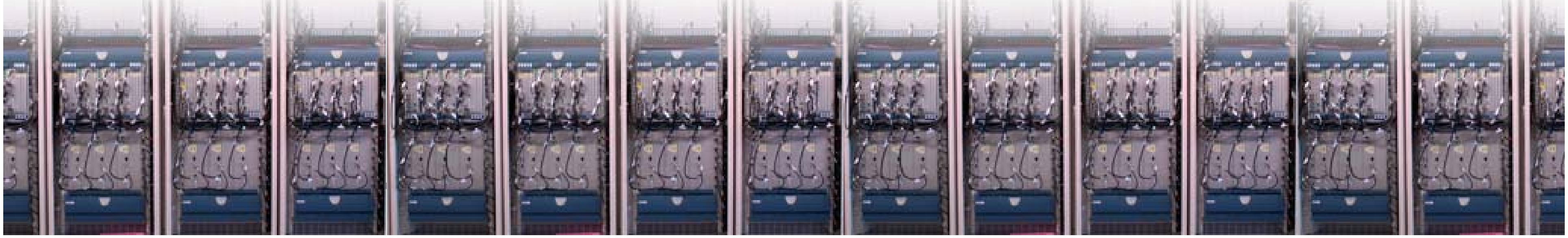
- Scalable to high core numbers
- Wait-free Read-Copy-Update mechanism
- Per-CPU buffers
- Non-blocking atomic operations
- Create and run more than one trace session in parallel at the same time, e.g.:
  - system administrator monitoring
  - field engineered to troubleshoot a specific problem

Reliability

- In production systems, no corruption of data
- Lost events must be accounted for
- Algorithms have to be robust
- Formal verification provides correctness and reliability guarantees

Low Overhead

- Low overhead is key, better tracing means more troubleshooting in field and quicker resolution of problems
- Don’t want to change behaviour of the system
- Minimal impact on network bandwidth, i.e. telecom system not a tracing system
- Very efficient probes with static jump, no trap, no system call
- Zero copy from event generation to disk write.
- Trying to keep per-CPU-core operation without un-needed synchronization



Low Overhead

- Almost zero performance impact with instrumentation points disabled
- Enable instrumentation points needs to have low performance impact
- Conditional tracing can tremendously reduce overhead

User Space Tracing

- Very low disturbance, highly scalable
- Same binary format as the kernel
- Merge kernel and user space traces, e.g. with timestamp
- Same features, (e.g. low overhead, robustness, scalability, …) as the kernel tracer
- Node-wide, i.e. multiple processes, multiple processors
- Conditional tracing in userspace

Time

- Accurate event ordering is key to enable trace synchronization or correlation of traces from
  - different CPU, cores
  - traffic exchanged between nodes
  - virtual machine, etc.

- Timestamp precision 1-100ns range, i.e. cycle counter

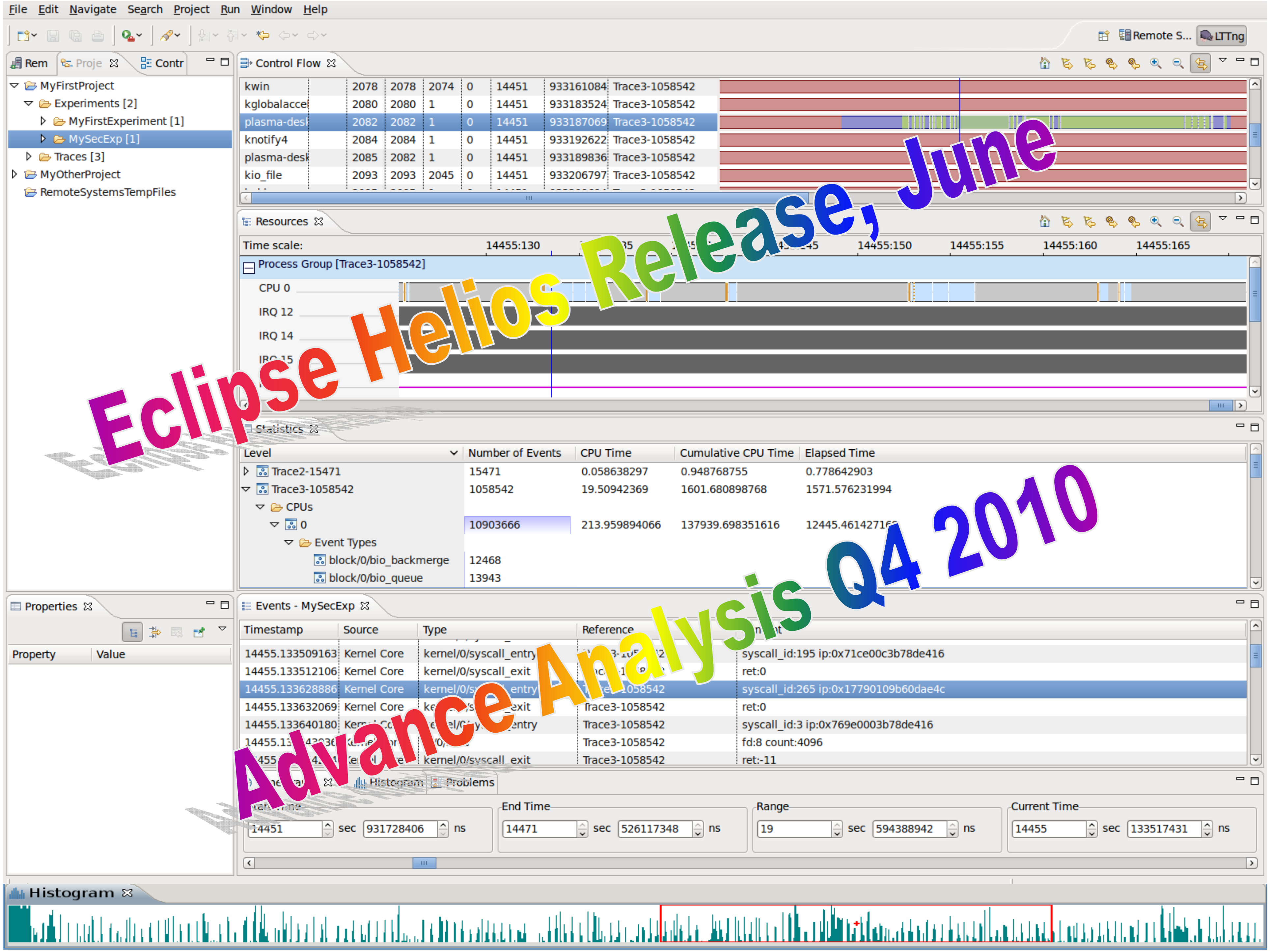
Traceable Data

- Everything should be traceable
- User space
- Kernel
- Non-Maskable Interrupt (NMI)
- Thread and signal safe
- Events may not be lost because of race conditions
- Collect large trace data > 10GB
- Static tracepoint integration with dynamic tracepoint: GDB dynamic tracepoint+LTTng UST, kernel kprobes+LTTng kernel

Analysis

- What do we do with all this data?

- Resource view
- Per thread execution state (control flow view)
- Event rate histogram
- Detailed event list, filtering
- View synchronization
- IRQ latency



Eclipse Helios Release, June

Advance Analysis Q4 2010

kwin	2078	2078	2074	0	14451	933161084	Trace3-1058542
kglobalaccel	2080	2080	1	0	14451	933183524	Trace3-1058542
plasma-desk	2082	2082	1	0	14451	933187069	Trace3-1058542
knotify4	2084	2084	1	0	14451	933192622	Trace3-1058542
plasma-desk	2085	2082	1	0	14451	933189836	Trace3-1058542
kio_file	2093	2093	2045	0	14451	933206797	Trace3-1058542

Level	Number of Events	CPU Time	Cumulative CPU Time	Elapsed Time
Trace2-15471	15471	0.058638297	0.948768755	0.778642903
Trace3-1058542	1058542	19.50942369	1601.680898768	1571.576231994
CPUs				
0	10903666	213.959894066	137939.698351616	12445.46142716
Event Types				
block/0/bio_backmerge	12468			
block/0/bio_queue	13943			

Property	Value

Timestamp	Source	Type	Reference	Content
14455.133509163	Kernel Core	kernel/0/syscall_entry	[illegible]	syscall_id:195 ip:0x71ce00c3b78de416
14455.133512106	Kernel Core	kernel/0/syscall_exit	[illegible]	ret:0
14455.133628886	Kernel Core	kernel/0/syscall_entry	Trace3-1058542	syscall_id:265 ip:0x17790109b60dae4c
14455.133632069	Kernel Core	kernel/0/syscall_exit	Trace3-1058542	ret:0
14455.133640180	Kernel Core	kernel/0/syscall_entry	Trace3-1058542	syscall_id:3 ip:0x769e0003b78de416
14455.13[illegible]	[illegible]	[illegible]	Trace3-1058542	fd:8 count:4096
[illegible]	[illegible]	kernel/0/syscall_exit	Trace3-1058542	ret:-11

Start Time: 14451 sec 931728406 ns
End Time: 14471 sec 526117348 ns
Range: 19 sec 594388942 ns
Current Time: 14455 sec 133517431 ns

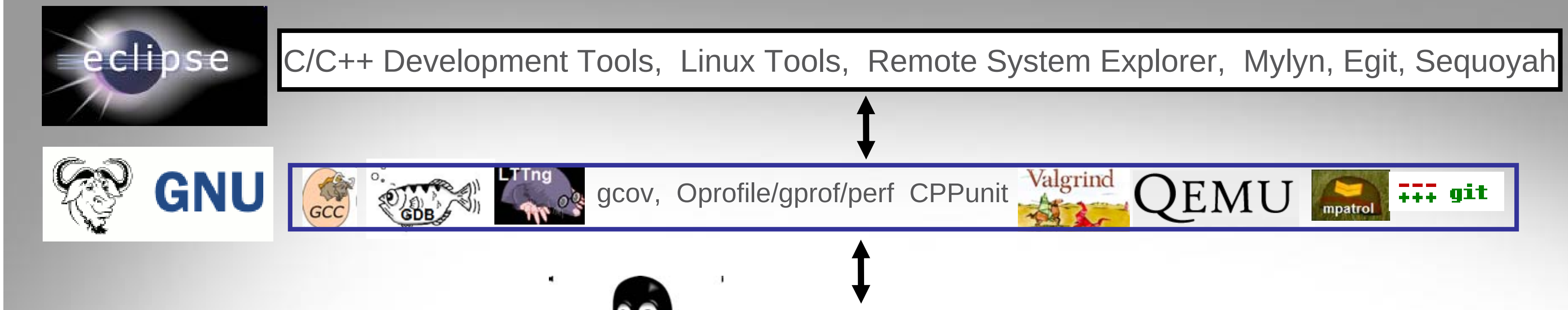
Eclipse IDE, what for?

- Debug multi-process, non-stop with cmd line?
- Performance analysis?
- What is your reason to use an IDE?

Context switching, bug, e-mail, new feature, interruptions, etc?
Code at the speed of thought? try Eclipse Mylyn

http://en.wikipedia.org/wiki/Task-focused_interface
http://www.tasktop.com/videos/mylyn/webcast-mylyn-3.0.html
http://tasktop.com/videos/w-jax/kersten-keynote.html

Linux Eclipse projects



eclipse
C/C++ Development Tools, Linux Tools, Remote System Explorer, Mylyn, Egit, Sequoyah
GNU
GCC
GDB
LTTng
gcov, Oprofile/gprof/perf CPPunit
Valgrind
QEMU
mpatrol
git

Linux

Linux Tools
http://www.eclipse.org/linuxtools

C/C++ Development Tool
http://www.eclipse.org/cdt/

Target Management
http://www.eclipse.org/dsdp/tm

Parallel Tools Platform
http://www.eclipse.org/ptp/

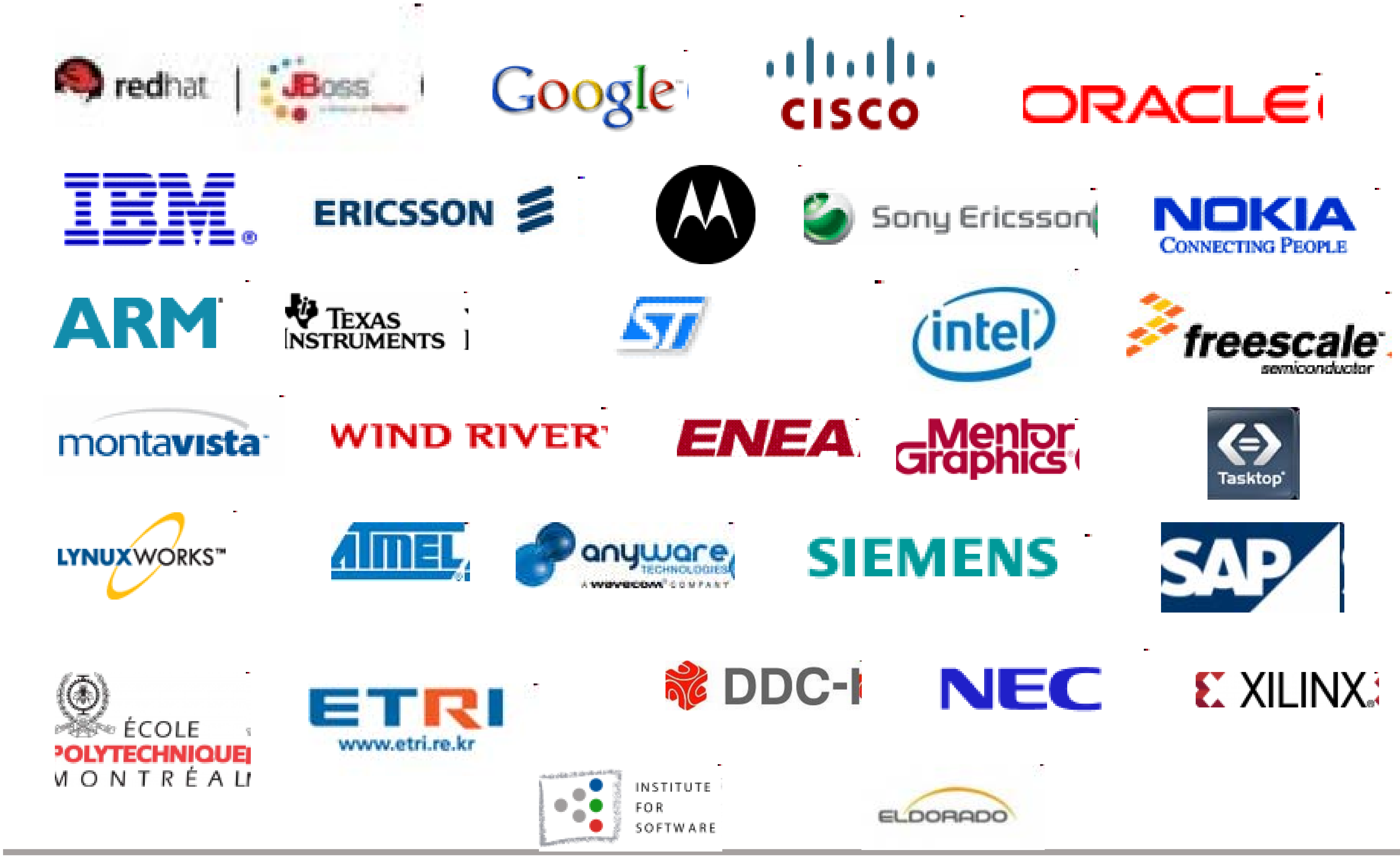
Tools for Mobile Linux / Sequoyah
http://www.eclipse.org/dsdp/tml

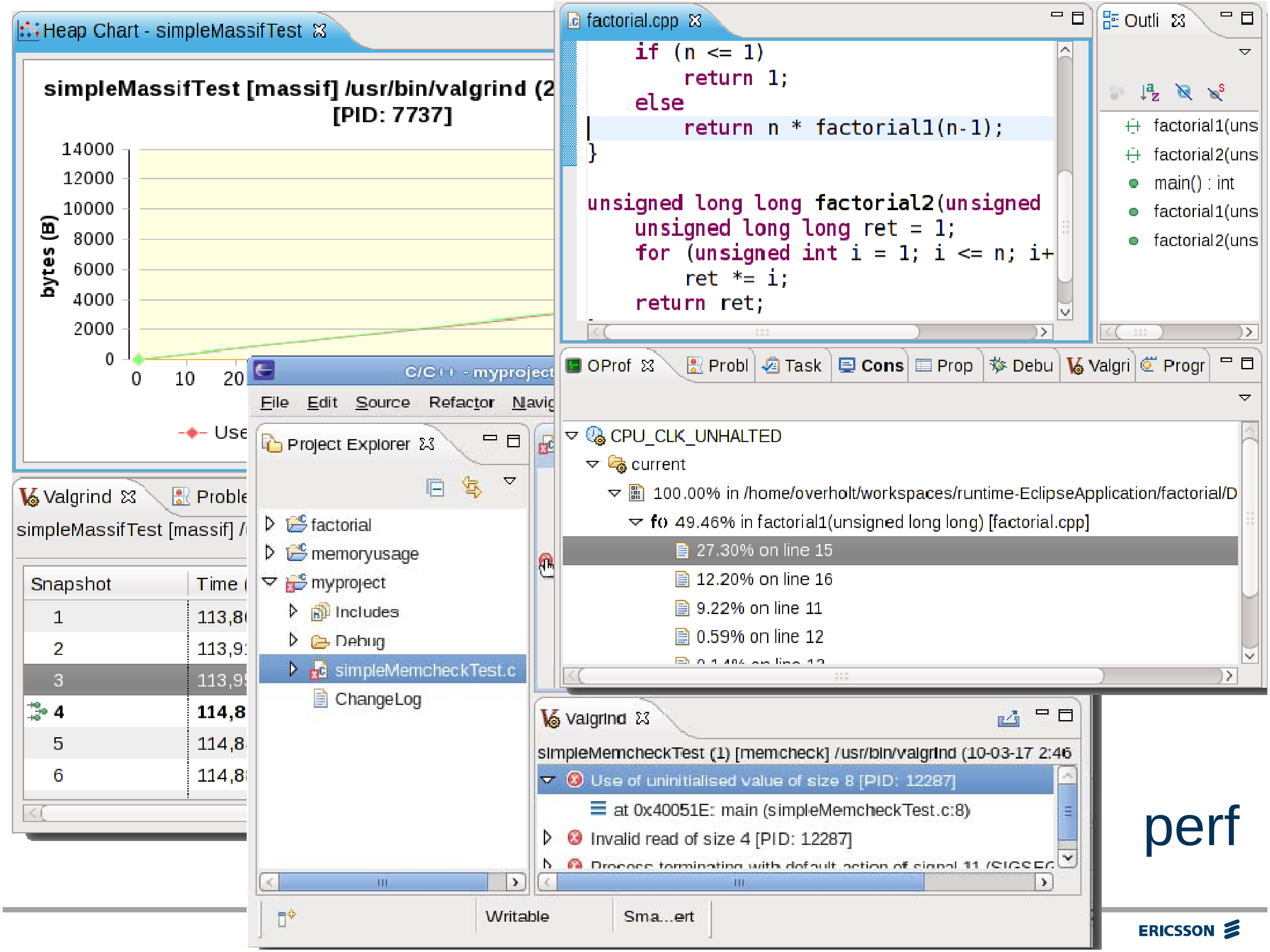
Mylyn, code at the speed of thought
http://www.eclipse.org/mylyn

EGit
http://www.eclipse.org/egit

All
http://www.eclipse.org/projects/listofprojects.php

Eclipse Foundation, 200 members





simpleMassifTest [massif] /usr/bin/valgrind (2
[PID: 7737]

simpleMassifTest [massif] /

Snapshot	Time (
1	113,8
2	113,9
3	113,9
4	114,8
5	114,8
6	114,8

```
    if (n <= 1)
        return 1;
    else
        return n * factorial1(n-1);
}

unsigned long long factorial2(unsigned
    unsigned long long ret = 1;
    for (unsigned int i = 1; i <= n; i+
        ret *= i;
    return ret;
```

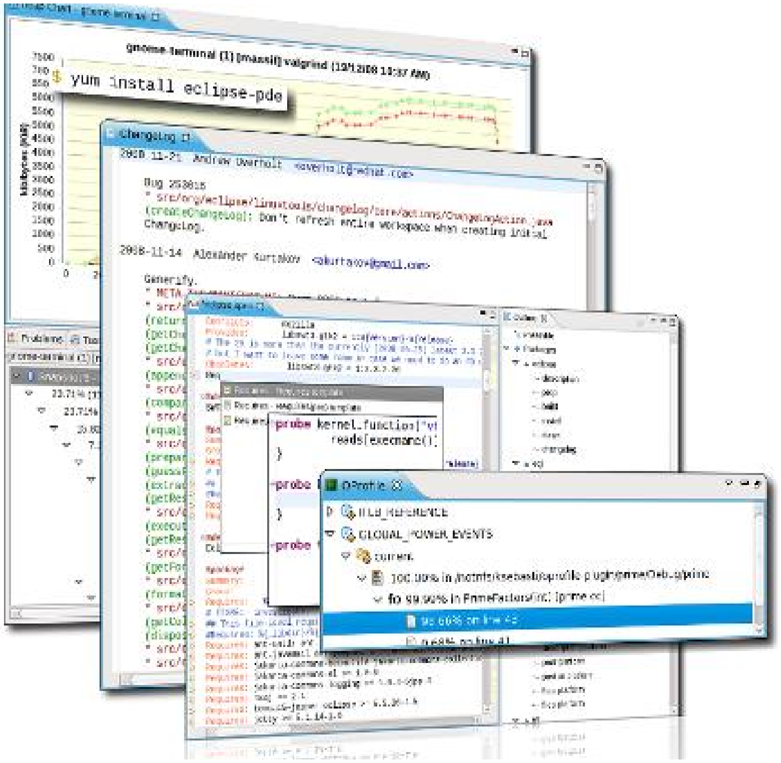
```
CPU_CLK_UNHALTED
  current
    100.00% in /home/overholt/workspaces/runtime-EclipseApplication/factorial/D
      f() 49.46% in factorial1(unsigned long long) [factorial.cpp]
          27.30% on line 15
          12.20% on line 16
          9.22% on line 11
          0.59% on line 12
```

```
simpleMemcheckTest (1) [memcheck] /usr/bin/valgrind (10-03-17 2:46
  Use of uninitialised value of size 8 [PID: 12287]
      at 0x40051E: main (simpleMemcheckTest.c:8)
  Invalid read of size 4 [PID: 12287]
  Process terminating with default action of signal 11 (SIGSEG
```

perf

Eclipse Linux Tools project

- Contribute/Get Involved
- Contact Us
- Downloads
- C/C++ Tools Projects
  - Autotools
  - Callgraph
  - ChangeLog
  - Libhover
  - Man Page
  - LTTng
  - OProfile
  - Systemtap
  - Valgrind
- Packaging/Distribution Projects
  - eclipse-build
  - RPM Stubby
  - Specfile Editor
  - Distribution Packaging Status



```
$ yum install eclipse-pde
```

```
2008-11-21  Andrew Overholt  <overholt@redhat.com>

	Bug 253816
	* src/org/eclipse/linuxtools/changelog/core/actions/ChangeLogAction.java
	(createChangeLog): Don't refresh entire workspace when creating initial
	ChangeLog.

2008-11-14  Alexander Kurtakov  <akurtakov@gmail.com>

	Generify.
```

```
probe kernel.function("vf
        reads[execname()]
}
```

The Linux Tools project aims to bring a **full-featured C and C++ IDE** to Linux developers. We build on the source editing and debugging features of the CDT and integrate popular native development tools such as the GNU Autotools, Valgrind, OProfile, RPM, SystemTap, GCov, GProf, LTTng, etc. Current projects include Autotools build integration, a Valgrind heap usage analysis tool, and an OProfile call profiling tool. We also have projects implementing LTTng trace viewers and analyzers.

The project also provides a place for Linux distributions to collaboratively overcome issues surrounding distribution packaging of Eclipse technology. The project produces both best practices and tools related to packaging. Since our 0.3.0 release, one of our features is a source archive of the Eclipse SDK that can be used by all Linux distributions building and distributing it.

Downloads
Get our latest **0.5** release (*2010-03-18*)!

Get Involved
Find out how you can get involved with the project

- Managed build for various toolchains, standard make build
- Source navigation, type hierarchy, call graph, include browser, macro definition browser, code editor with syntax highlighting, folding and hyperlink navigation,
- Source code refactoring, static analysis
- Visual debugging tools, including memory, registers, and disassembly viewers

Analysis

- Trace synchronization
  - Time correction
  - Multi-core
  - Multi-level
  - Multi-node, distributed

- Dependency analysis, delay analyzer
  - Dependencies among processes
  - How total elapsed time is divided into main components

Analysis

- Pattern matching
  - Security
  - Performance
  - Testing lock acquisitions

- Correlation
  - Other format
  - Text base logs
  - Multi-level

Multi-Core Troubleshooting

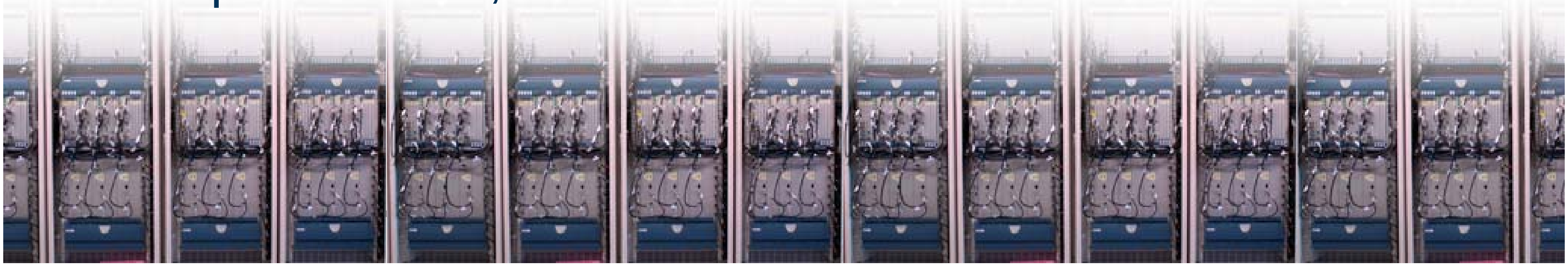
- Major software redesign is normally required to benefit from multi-core architectures
- Software development industry and individual developers are facing problems whose resolution requires to understand the interaction between all layers, including third party products e.g.
  - Hypervisor
  - Operating system
  - Virtual machines
  - System libraries
  - Applications
  - Operation and maintenance
  - Many languages: C/C++, Java, Erlang, …

Complex systems

- A typical system these days
  - SMP Linux on a few cores
  - Low-level RTOS on another core
  - DSP's, etc.
- Developed in different context
  - In-house development
  - Consultant
  - Reusable components
  - Third party products

- Domain knowledge
  - Telecom
  - Financial
  - Automotive
  - Consumer electronics
  - Industrial
  - Military
  - Medical
  - Etc.

- Understanding what is happening on the system requires compatible tools, i.e. de facto standard



Linux Tracing Systems?

- In addition to file system, memory, etc, companies switching to Linux also need a tracing infrastructure
- Distributions like MontaVista, WindRIver, etc. need to apply large patches to enable tracing
- Patching commercial kernel leads to unsupported distribution!

Linux Tool Work Group

- Open source contributions are growing exponentially, contributions can sometimes be incompatible or result in duplicated work:
  - forks of GDB
  - competing projects have emerged, e.g. frysk, EDC
  - Linux trace initiatives e.g. LTTng, ftrace, perf, utrace, SystemTap, etc.
  - Very hard to plan cross project features

- Let's take this to the next level
  - not only contribute the parts needed for one company, plan together
  - avoid incompatible data, inconsistent work, and duplicated efforts
  - e.g. Executable and Linkable Format (ELF), DWARF debug format
  - create an industry de-facto standard for tools
  - Budget cycle! Ecosystem of tool improvements, support
  - Linux foundation tool work group?

We can do better than printf