

> The LTTng tracer

A Low Impact Performance and Behavior
Monitor for GNU/Linux.

Mathieu Desnoyers
Ottawa Linux Symposium 2006

> Plan

- Tracing overview
- Linux Trace Toolkit Next Generation (LTTng)
- Linux Trace Toolkit Viewer (LTTV)
- Results
- Conclusion

> Tracing overview

- Trace
 - Detailed record of steps taken by a program
- Goals
 - Low impact
 - High performance
 - Flexibility
 - Easy to use
 - Precision of timestamps
 - Extensibility

> Tracing overview

- Performance monitors
 - post-processing
 - Collect raw data in a trace for various post-mortem analysis
 - pre-processing
 - Information to calculate is known prior to execution
 - Examples
 - Number of I/O requests per seconds
 - System call per second for a specific PID
 - Can be integrated with post-processing

> Tracing overview

- Existing solutions
 - post-processing
 - LTT
 - Linux Kernel Tracer
 - K42
 - IrixView
 - Tornado

> Tracing overview

- Existing solutions
 - pre-processing
 - SystemTAP
 - Kerninst
 - dtrace
 - PEM

> Tracing overview

- Dynamic probes
 - Breakpoints
- Static instrumentation
 - Source modification
 - Tracing statements follows code revisions
 - Fast

> LTTng

- Previous Work

- LTT

- + Stable instrumentation base
 - Fixed size types
 - Monolithic tracer and viewer
 - Use NTP corrected timestamps

- Relay (formerly known as RelayFS)

- High-speed data relay from kernel to user space

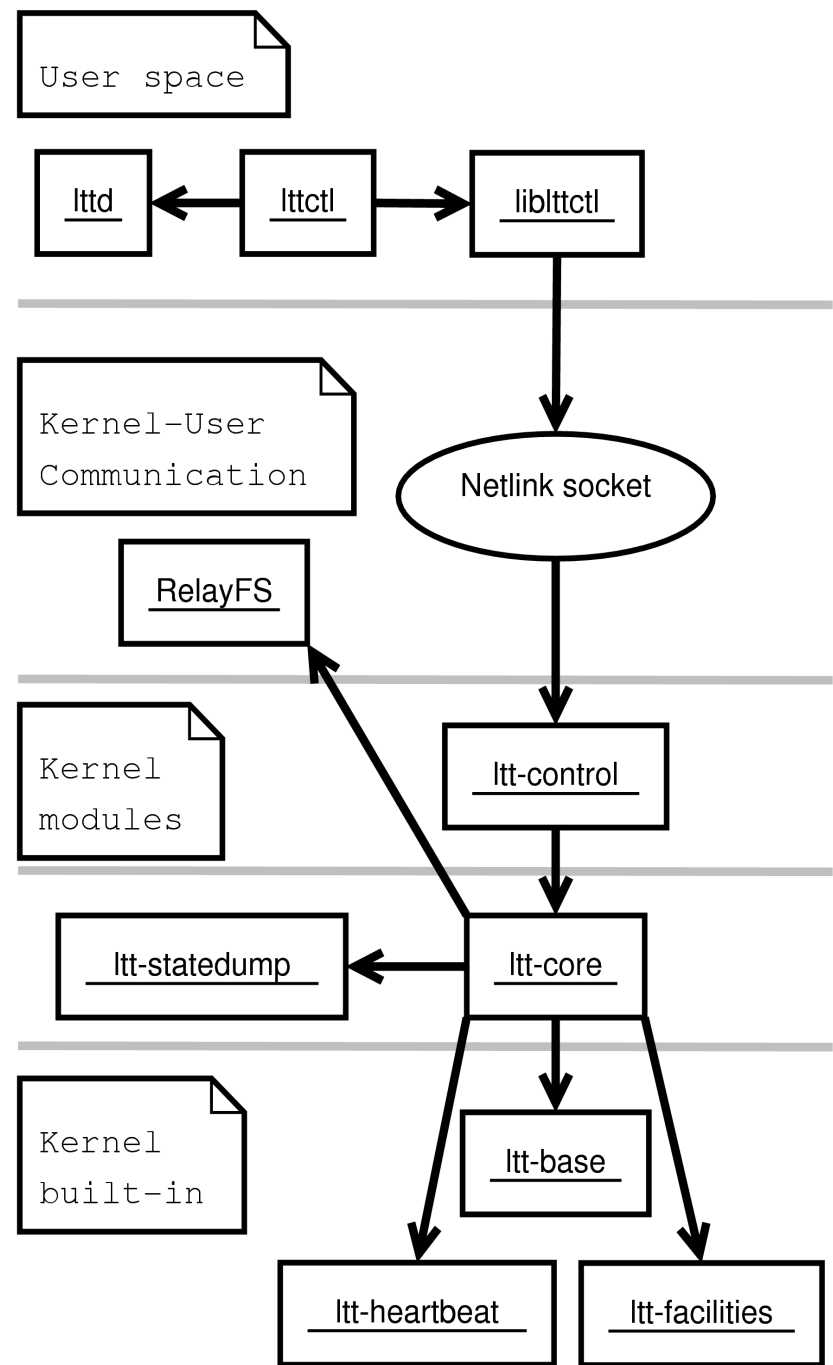
- K42

- Research OS from IBM
 - Implements scalable tracing

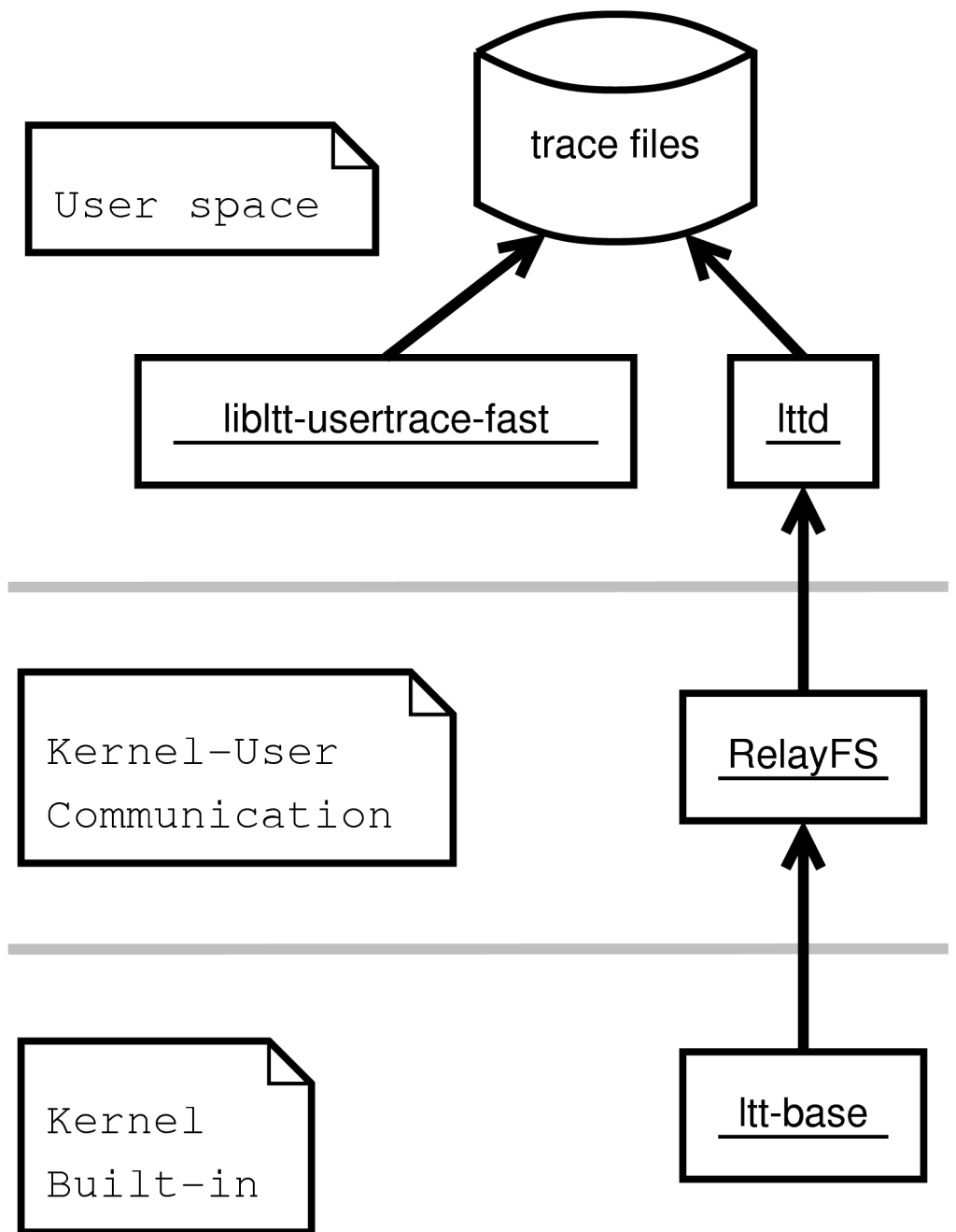
> LTTng

- Architecture
 - Control
 - Data flow
 - Instrumentation
 - Event type registration
 - Tracing
 - Reentrancy
 - Scalability
 - Time precision
 - User space tracing

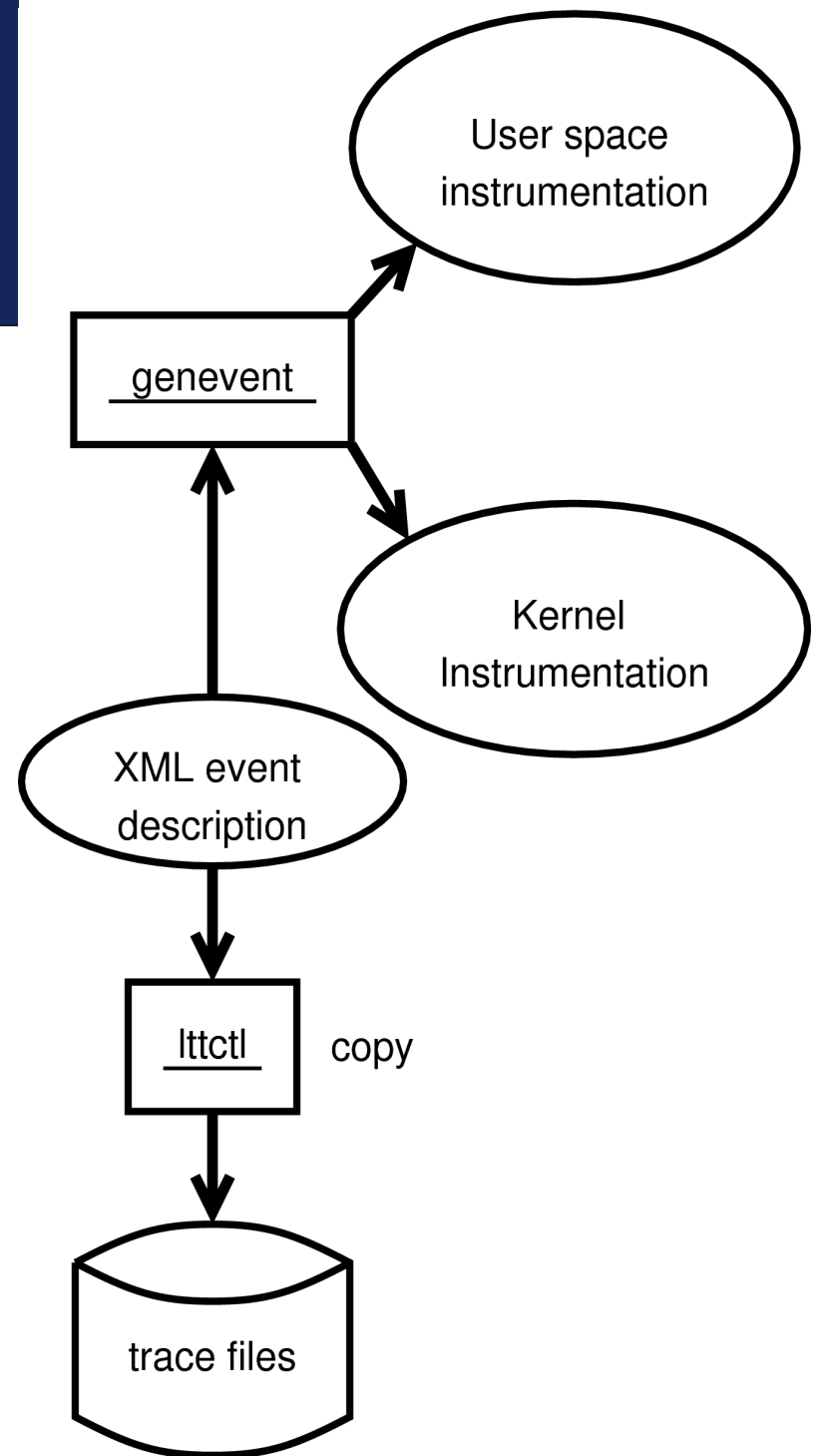
> Control



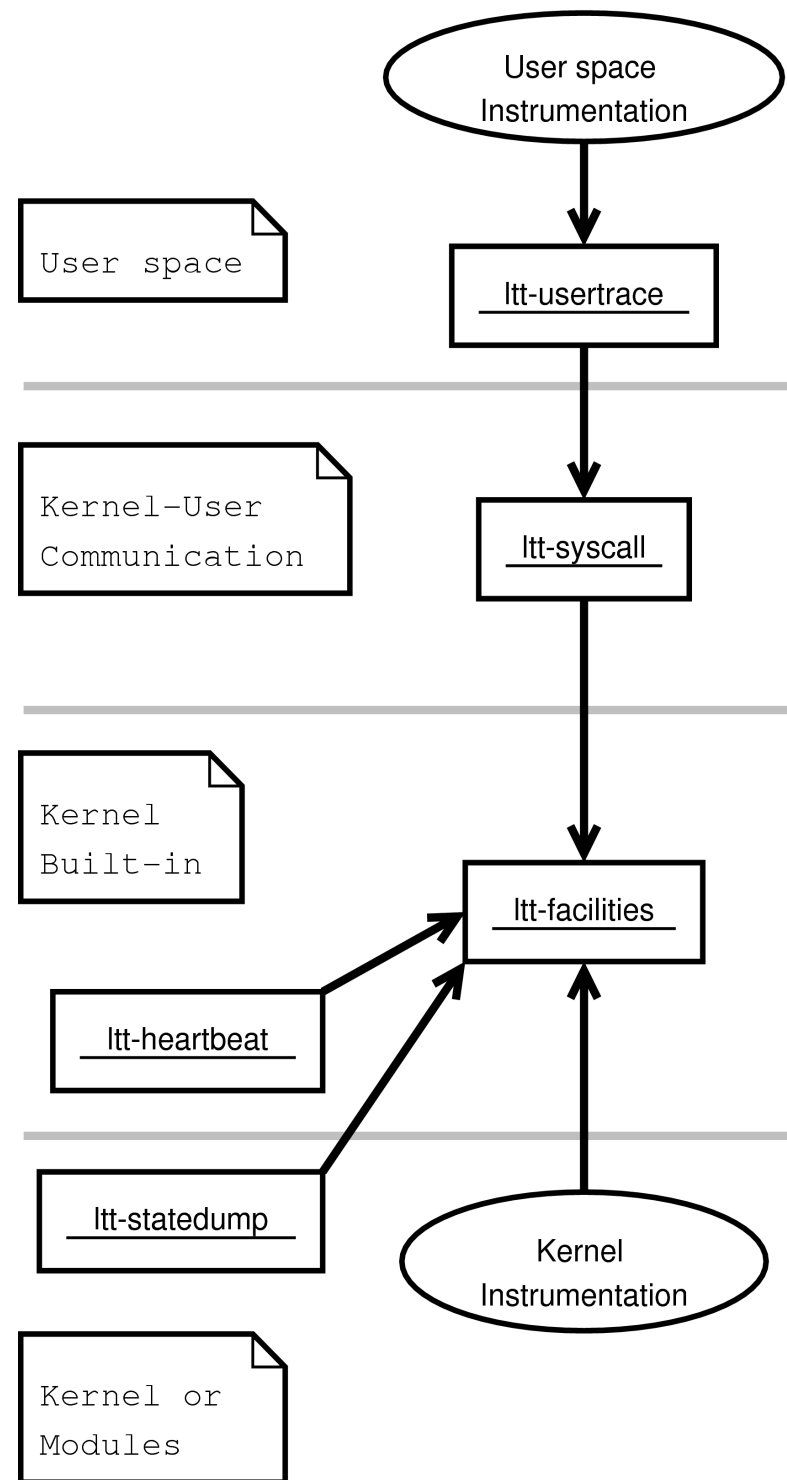
> Data Flow



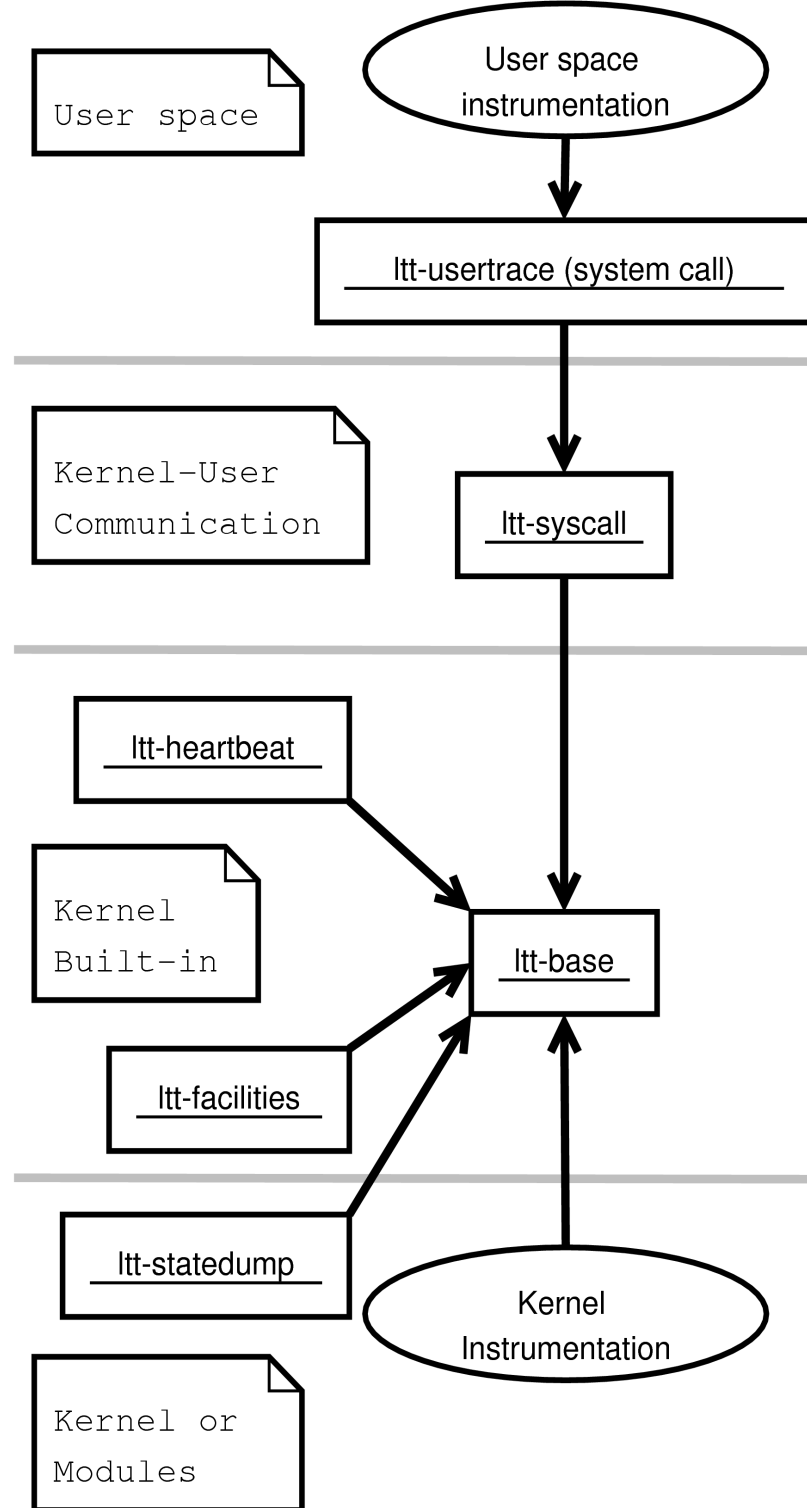
> Instrumentation



> Event Type Registration

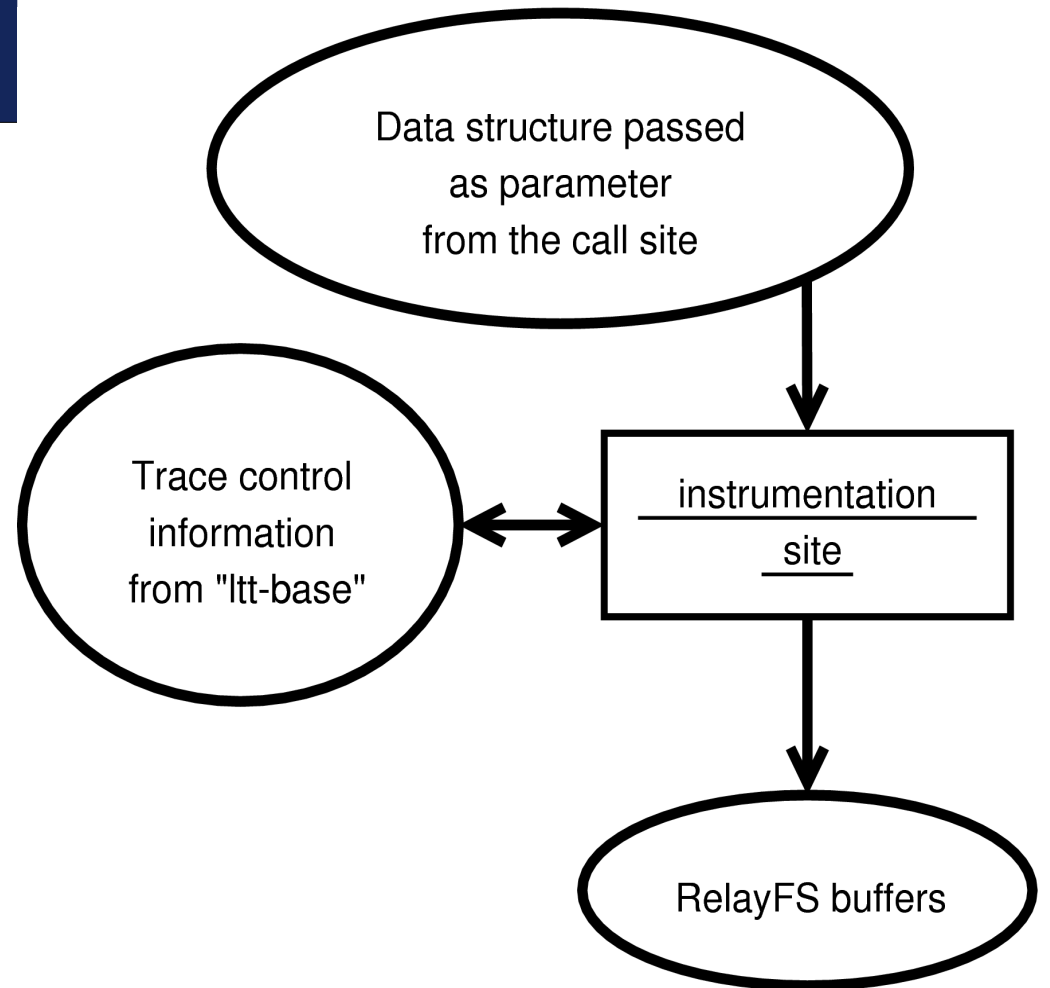


> Tracing



> Reentrancy

Inputs/Outputs



> Scalability

- Tracing input
 - Stack
 - Global, often already locked
- Per CPU
 - buffers
 - No false sharing
 - buffer index compare-and-exchange
 - Better scalability than spinlock
 - Full reentrancy (NMI)
 - 5 times faster than spinlock (43 vs 214 cycles on P4)

> Time Precision

- Monotonic timestamps
- CPU cycle counter
- Compatibility
 - Jiffies counter
 - Global logical clock
- Post-processing
 - Convert cycles to nanoseconds
 - double precision floating point

> User Space Tracing

- Slow
 - Through system call
- Fast
 - User space library
 - One disk writer process per thread
 - Anonymous shared memory map
 - Preserves buffer when thread is killed

> LTTV

- $O(1)$ time seeks in multi-GB SMP traces
- Extensible with specialized plugins
 - Text
 - Graphical
- Extensible XML event description
- Modular
- Recreates the kernel state evolving through time
- Filter

> Demo

- Instrumenting a subsystem
- Example : instrument interrupts
 - Create new facility
 - Insert instrumentation
 - Execute
 - Analyze

> Results (microbenchmarks)

- On a Pentium 4 3.0 Ghz
- Probe time
 - 95.15ns
- Interrupt latency
 - No interrupt disabling
 - Interrupt handler instrumentation
 - 192.29ns
- Scheduler latency
 - 192.29ns

> Results (macrobenchmarks)

Load size	Test Series	CPU time (%)			Data rate (MiB/s)	Events/s
		load	probes	ltd		
Small	mozilla (browsing)	1.15	0.053	0.27	0.19	5,476
Medium	find	15.38	1.150	0.39	2.28	120,282
High	find + gcc	63.79	1.720	0.56	3.24	179,255
Very high	find + gcc + ping flood	98.60	8.500	0.96	16.17	884,545

Table 2: LTTng macrobenchmarks for different loads

> Results (object size)

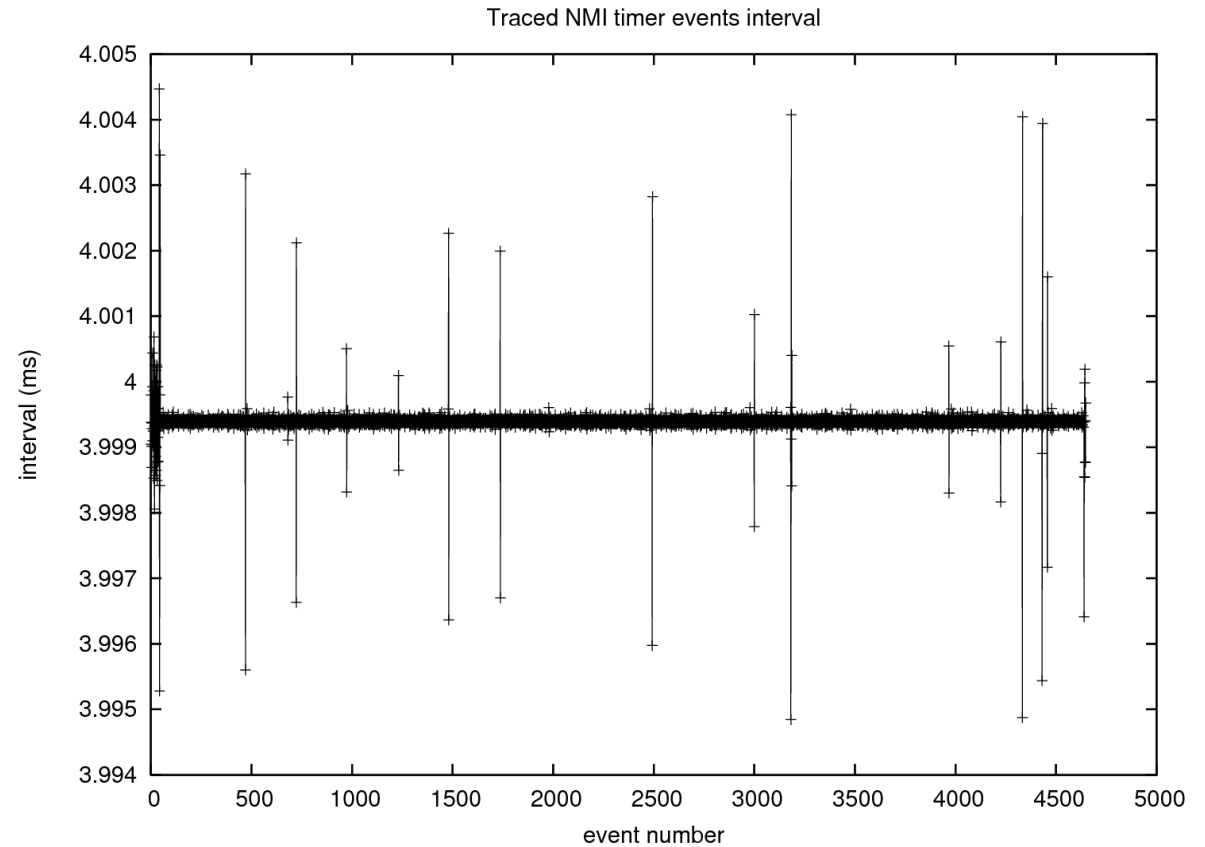
- 2.37kB per event
- Work in progress
 - Inlining vs function call
 - Slower
 - Stack usage
 - Reuse type serialization functions

> Results (time precision)

- Test on Intel D915-GAG motherboard
 - timer is driven by a TXC HC-49S crystal
 - ± 30 PPM precision

> Results (time precision)

- std. dev. 52 ns
- Each 4ms
- 13 PPM



> Conclusion

- 2% disturbance on heavy workload
- Reentrancy insured by atomic operations
- Integrated analysis
 - User space programs and libraries
 - Kernel (any execution context)
- Precise time measurement
- Facilitates instrumentation
- Ongoing work : cluster analysis in LTTV

> Questions ?

- See paper for more details
- Project website : <http://litt.polymtl.ca>
- Kernel integration of tracer core

> Tracing Core

- Serializing information in event records
- Serializing event records
- Must be trivial to use by subsystem maintainers
 - Usable from any context
- Relatively simple, minimum side effect
- Identity of the event record
- Types in the event record
- Tracing control mechanism (activation...)

> Tracing features

- Instrumentation
 - Dynamic / Static
- Aggregation
 - May be written periodically to buffers
- Dynamic filtering
- Dynamically identify new instrumentation
 - at module loading (even for static)
 - dynamic instrumentation

> Tracing features (cont.)

- Static disabling of tracing
 - Global
 - Specific
- Access buffers, dump to disk, network
- Saving information upon crash
- Early tracing