Highly-Scalable Wait-Free Buffering Scheme for Multi-Core System Tracing

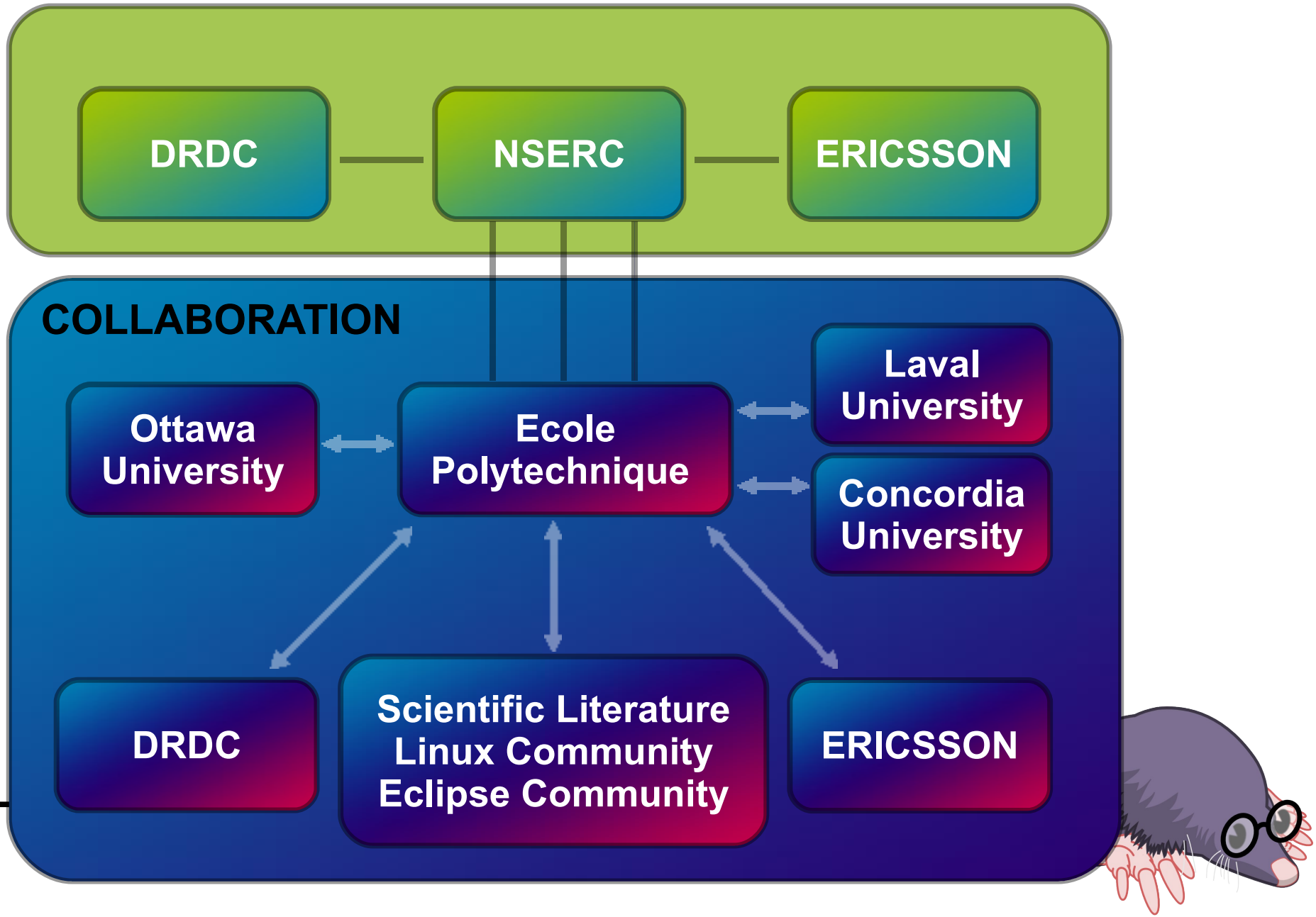
Mathieu Desnoyers, Michel Dagenais,
Polytechnique Montreal
Dominique Toupin
Ericsson

Workshop On Multi-Core Computing, MCC09
November 2009

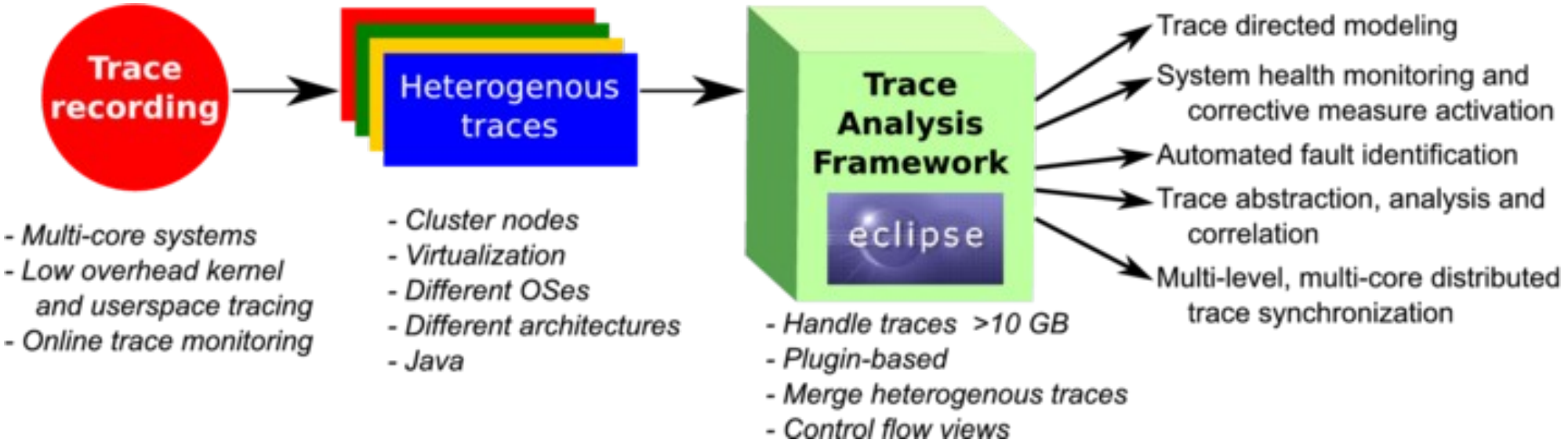
Summary

- System tracing for Multi-Core
- Tracer architecture
- Synchronization primitives and wait-free buffering scheme
- Results
- Discussion

Project Structure



Research Tracks



- Adaptative Fault Probing, Prof. Michel Dagenais, Ecole Polytechnique
- Multi-level, multi-core distributed traces synchronization, Prof. Michel Dagenais and Robert Roy, Ecole Polytechnique
- Trace abstraction, analysis and correlation, Prof. Abdelwahab Hamou-Lhadj, Concordia University
- Automated fault identification, Prof. Bechir Ktari, Laval University
- Trace directed modeling, Prof. Tim Lethbridge, Ottawa University
- System health monitoring and corrective measure activation, Prof. Michel Dagenais, Ecole Polytechnique

System Tracing

- DTrace
- SystemTap
- FTrace
- K42
- LTTng

DTrace

- Part of Sun Solaris since 2005.
- Static and dynamic tracepoints.
- Dynamically activate tracepoints.
- Execute a script or log an event when a tracepoint is encountered.
- Disable interrupts and take a sequence lock for logging an event.

SystemTap

- Red Hat released version 1.0 in September 2009.
- Linux kernel static Tracepoints or dynamic Kprobes.
- Dynamically activate probes.
- Execute a script or log an event when a probe is encountered.
- Disable interrupts and take a spin lock for logging an event.

FTrace

- Developed in the Linux kernel in 2009 from a subset of LTTng, plus Function Tracer, plus some reimplementation and extensions.
- Linux kernel static Tracepoints or dynamic Kprobes.
- Dynamically activate probes.
- Log an event when a probe is encountered.
- Disable interrupts and take per-cpu spinlock for logging en event. Evolving to lockless.

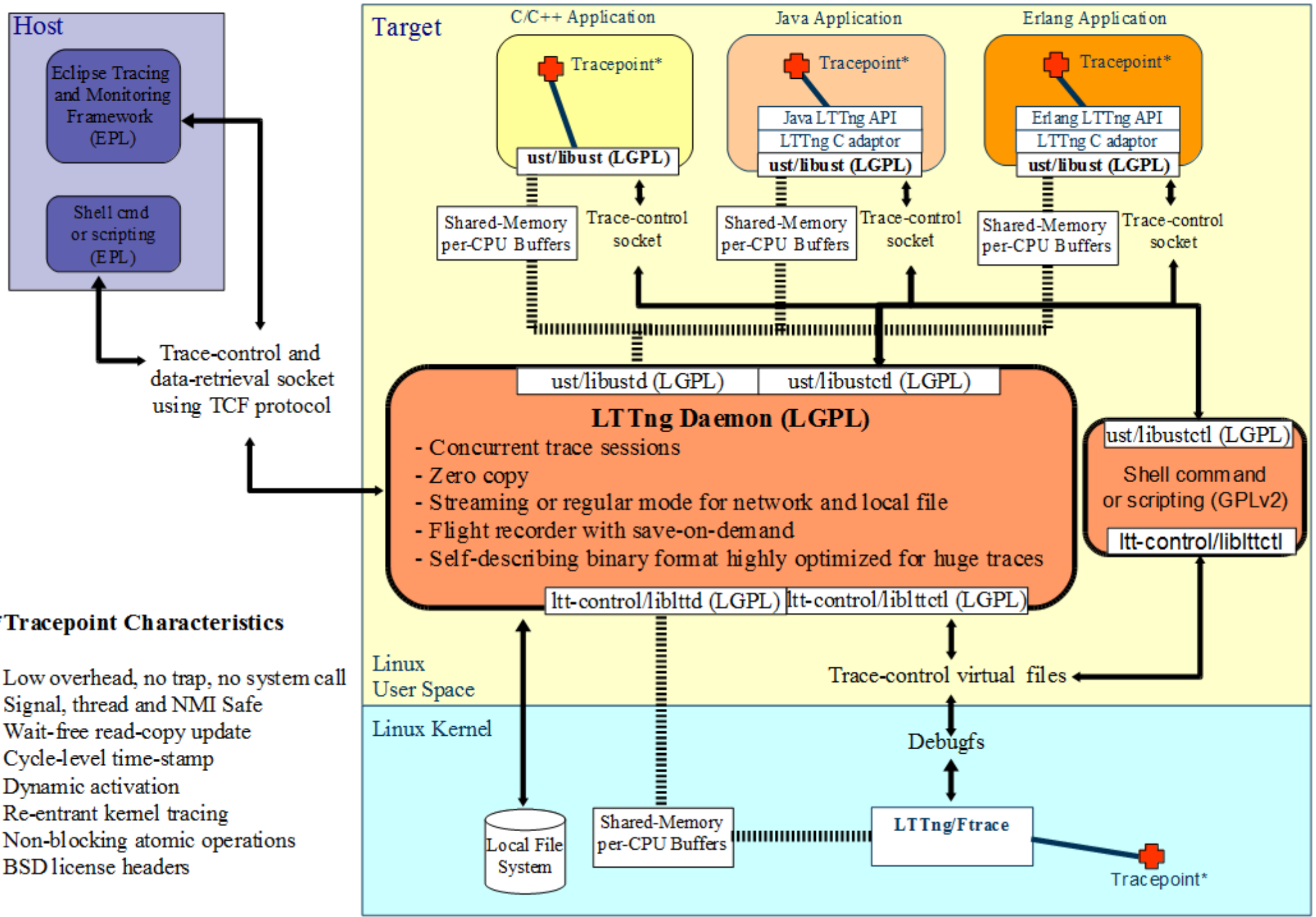
K42

- Developed at IBM Research between 1999 and 2006, highly scalable OS for multi-core, with tracer.
- Lockless per-cpu buffering scheme.
- One trace retrieval thread for each CPU to simplify synchronization.
- No separation between kernel and processes tracing buffers.

LTTng

- Developed at Ecole Polytechnique since 2005, as a rewrite of LTT which started in 1999, with the collaboration of Ericsson, Fujitsu, Google, IBM, Red Hat, Sony, Wind River...
- Linux kernel static Tracepoints or dynamic Kprobes.
- Dynamically activate probes.
- Log an event when a probe is encountered.
- Wait-free buffering scheme suitable even for events in NMI context.
- Optimized for high volume, low overhead.

LTTng Low-Overhead Tracing Architecture



***Tracepoint Characteristics**

- Low overhead, no trap, no system call
- Signal, thread and NMI Safe
- Wait-free read-copy update
- Cycle-level time-stamp
- Dynamic activation
- Re-entrant kernel tracing
- Non-blocking atomic operations
- BSD license headers

Synchronization primitives

- Writing an event:
  - Reserve space by increasing the write count
  - Write the data
  - Update the commit count
- Synchronization
  - Compare and Swap (CAS), atomic increment
  - Disable/enable interrupts
  - CPU local CAS, atomic increment

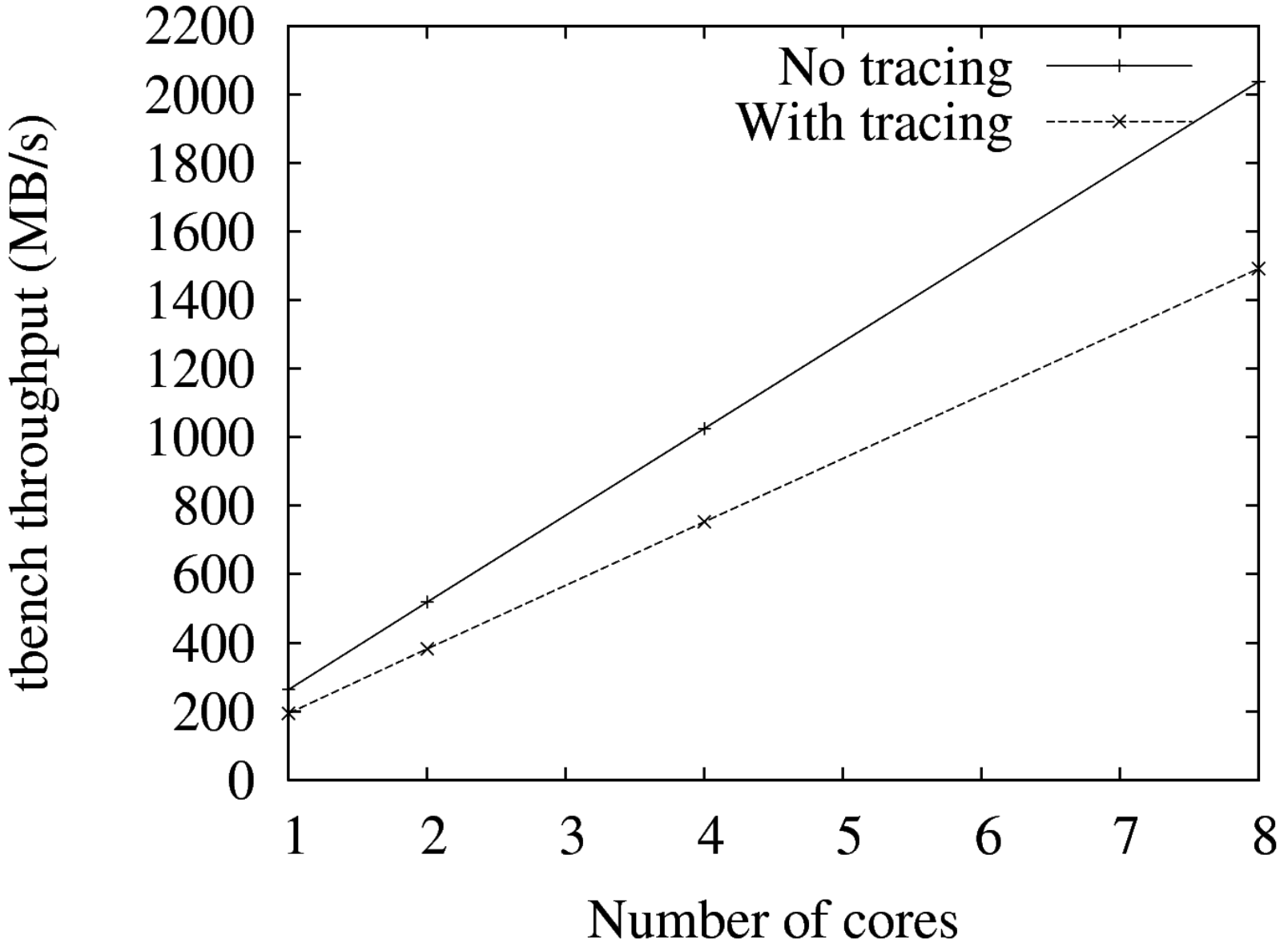
Synchronization Performance

Architecture	CAS cycles	Local CAS	Cli + sti
Intel Pentium 4	81	25	131
AMD Athlon 64X2	24	6	23
Intel Core 2	24	8	43
Intel Xeon E5405	24	8	42
PowerPC G5	2	1	4
PowerPC Power6	17	9	16
ARM v7 OMAP3	71	11	45
Itanium 2	3	3	4
UltraSparc IIIi	.394	.394	.159

Tracing performance

- Intel Pentium 4, tracing to buffer:
  - DTrace 1180ns
  - LTTng 182ns
- Intel Xeon
  - LTTng 119ns

Scalability of TBench Throughput



Discussion

- Supporting NMI contexts requires atomic operations.
- Read-Copy Update (RCU) synchronization allows wait-free reads of complex structures.
- Per-CPU resources reduce costly global memory barriers.
- Any shared resource in contention will severely impact scalability at the 64 processors or more level.

Availability

- LTTng.org (Ecole Polytechnique de Montreal)
  - LTTng: patch for Linux kernel tracing
  - LTTV: efficient trace viewer
  - Urcu: user-space RCU
  - Ust: user-space tracing
- Eclipse.org
  - Tracing and Monitoring Framework contributed by Ericsson to Linux Tools project