Efficient Tracing Infrastructure

David Goulet
david.goulet@polymtl.ca

December 8, 2010
Mid Project Meeting

Content

1. Tracing Architecture
2. State of UST
3. Experimentation
4. Challenges
5. Conclusion

Tracing Architecture

State of UST

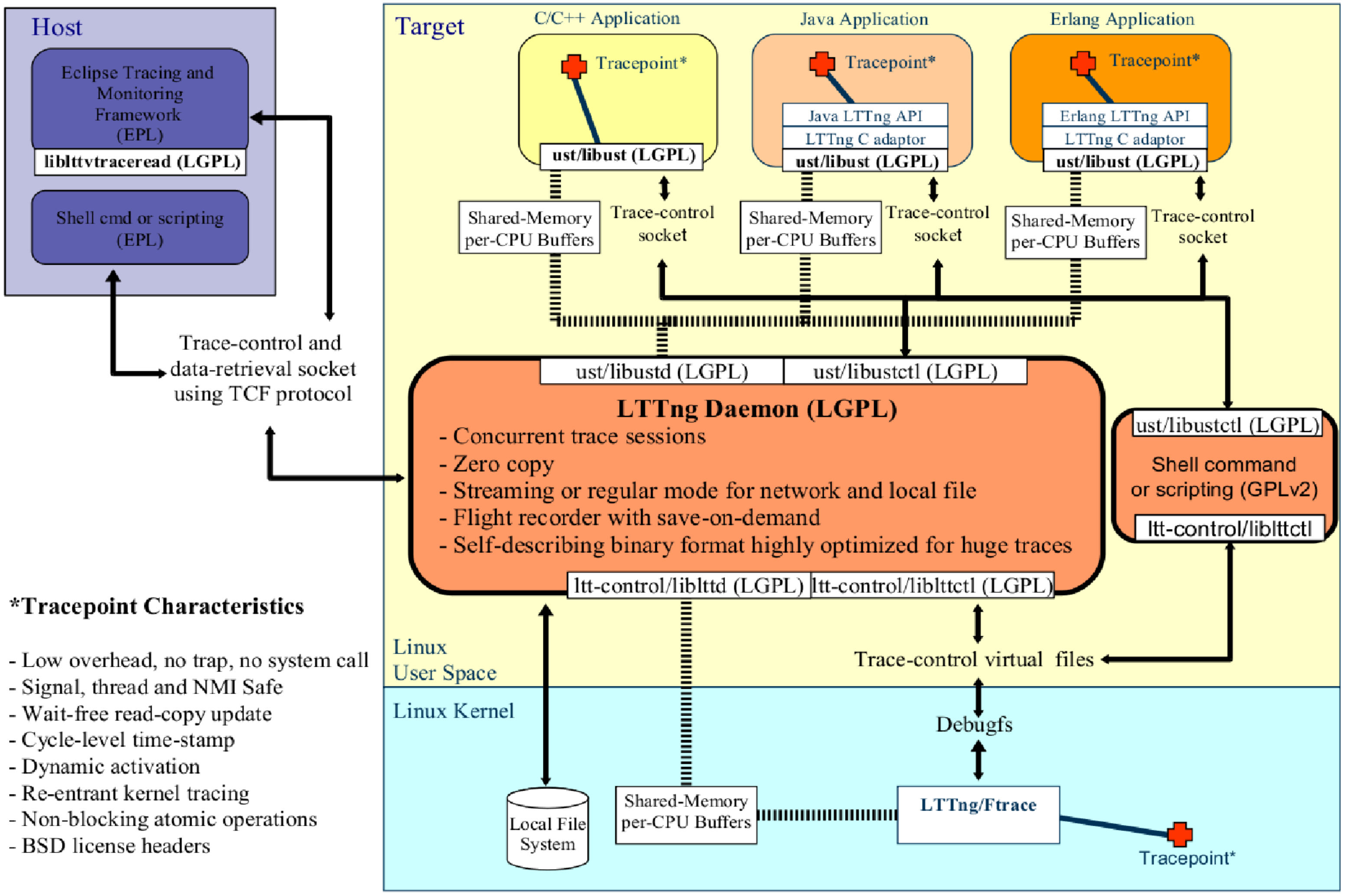
Experimentation

Challenges

Conclusion

1. Tracing Architecture

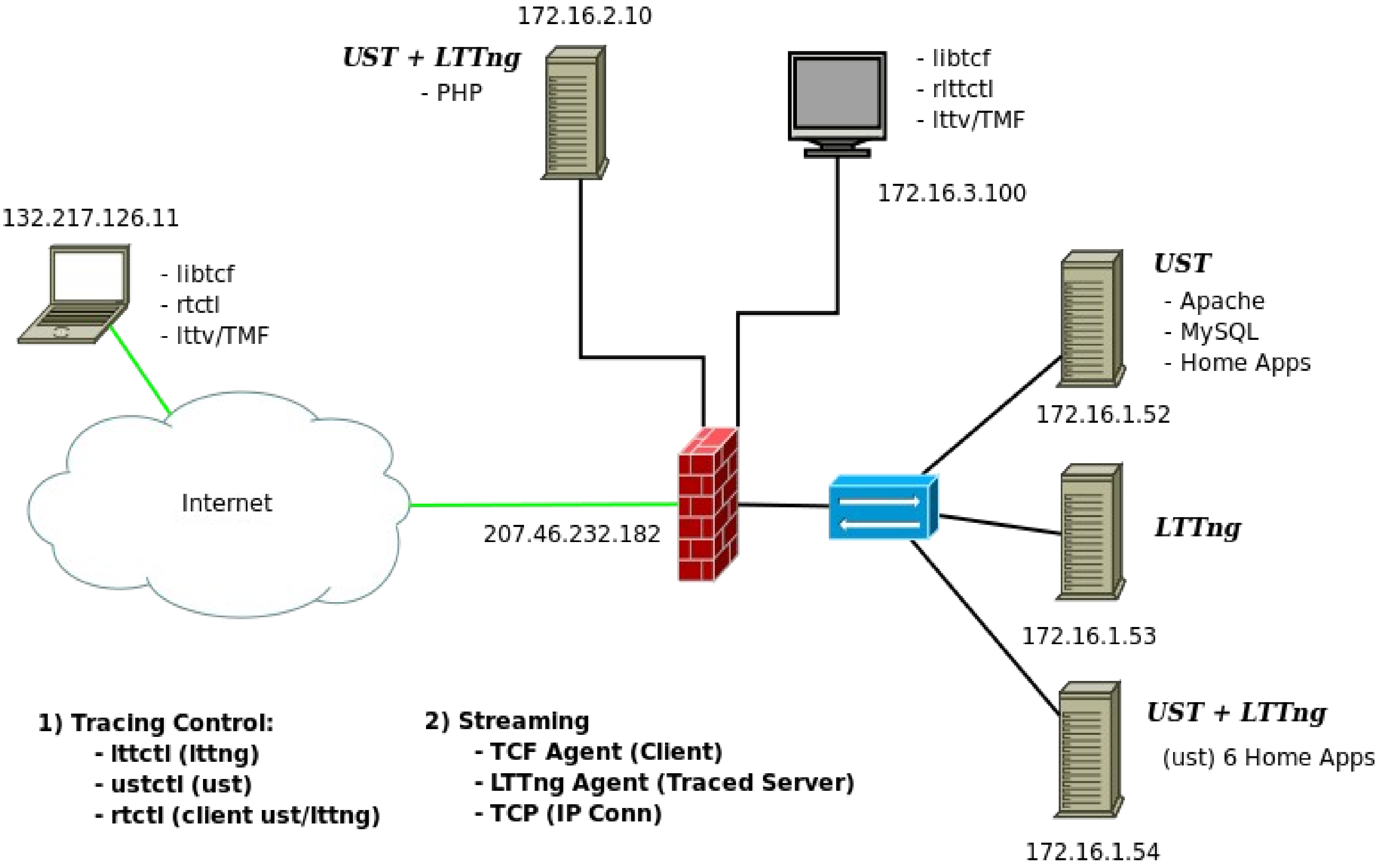
LTTng Low-Overhead Tracing Architecture


Host
Eclipse Tracing and Monitoring Framework (EPL)
liblttvtraceread (LGPL)
Shell cmd or scripting (EPL)
Trace-control and data-retrieval socket using TCF protocol
Target
C/C++ Application
Tracepoint*
ust/libust (LGPL)
Java Application
Tracepoint*
Java LTTng API
LTTng C adaptor
ust/libust (LGPL)
Erlang Application
Tracepoint*
Erlang LTTng API
LTTng C adaptor
ust/libust (LGPL)
Shared-Memory per-CPU Buffers
Trace-control socket
Shared-Memory per-CPU Buffers
Trace-control socket
Shared-Memory per-CPU Buffers
Trace-control socket
ust/libustd (LGPL)
ust/libustctl (LGPL)
LTTng Daemon (LGPL)
- Concurrent trace sessions
- Zero copy
- Streaming or regular mode for network and local file
- Flight recorder with save-on-demand
- Self-describing binary format highly optimized for huge traces
ltt-control/liblttd (LGPL)
ltt-control/liblttctl (LGPL)
ust/libustctl (LGPL)
Shell command or scripting (GPLv2)
ltt-control/liblttctl
Linux User Space
Trace-control virtual files
Linux Kernel
Debugfs
Local File System
Shared-Memory per-CPU Buffers
LTTng/Ftrace
Tracepoint*

***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

1. Tracing Architecture (2)



172.16.2.10

UST + LTTng

- PHP

- libtcf
- rlttctl
- lttv/TMF

172.16.3.100

132.217.126.11

- libtcf
- rtctl
- lttv/TMF

Internet

207.46.232.182

UST

- Apache
- MySQL
- Home Apps

172.16.1.52

LTTng

172.16.1.53

UST + LTTng

(ust) 6 Home Apps

172.16.1.54

1) Tracing Control:

- **lttctl (lttng)**
- **ustctl (ust)**
- **rtctl (client ust/lttng)**

2) Streaming

- **TCF Agent (Client)**
- **LTTng Agent (Traced Server)**
- **TCP (IP Conn)**

1. Tracing Architecture (4) : Research Goal

- Address production use cases
- Very low impact on the host and infrastrucutre
- Security
  - UST Daemon
  - Traces over the network
- Streaming
- Event Filtering

Tracing Architecture

State of UST

Experimentation

Challenges

Conclusion

2. State of UST : Recent developments

- Multi Core Scaling
- Added data pointer to Tracepoints. First step for event filtering
- *InProcess* Library communication redone
- Userspace use cases studied
- Four releases in the last 5 months

2. State of UST : TRACE_EVENTS

- Linux Kernel mechanism for tracing
- Dynamic conditions *(Rafik Fahem)*
- Basic structure implemented *(Nils Carlson, Ericsson)*
- Evolution follows current developments in LTTng and TRACE_EVENTS

2. State of UST : *InProcess* Library Comm.

- Internal communication completely re-engineered *(Nils Carlson, Ericsson)*
  - More compact protocol
  - No memory allocation for incoming packets
  - Systematic error code reporting
- Does NOT depend on any other external library

Tracing Architecture

State of UST

Experimentation

Challenges

Conclusion

3. Experimentation

- September 2010, QEMU instrumented with UST **upstream**
- MariaDB (MySQL fork) contains 46 UST tracepoints and is under revision for **upstream**
- Works is being done on `lttngtrace` tool for UST integration and being able to give the user a **one** command action for tracing and results.
- Closely working with Revolution Linux infrastructure needs and use cases

3. Experimentation : Performance

- Performance data
  - `trace_mark` :
    - → ~ 247 ns / per event
  - `tracepoint + trace_mark` :
    - → ~ 271 ns / per event
  - `tracepoint + custom_probe` :
    - → ~ 189 ns / per event

Tracing Architecture

State of UST

Experimentation

Challenges

Conclusion

4. Challenges

- The UST daemon, for production use, must be *refactored* on two levels :
  - Security
  - Threading Model : efficient, non-intrusive and compatible with the security model.
- LTTv quick *human readable* text dump for non developer usage (Ex: Sysadmins) *(Vincent Attard)*
- Tools surrounding UST are being analyze for every possible **real** use cases and modified accordingly.

Tracing Architecture

State of UST

Experimentation

Challenges

Conclusion

6. Conclusion

- UST is getting more and more attention
- Combining traces with LTTng and viewing them with the same tool is a **key** feature
- Packaging Debian *(Alexandre Montplaisir)*
- User base is growing so :
  - Stability is getting better
  - Features are done for the needs of users out there