Linux Static Tracing
System Wide!
More than Kernel Tracing

• Many layers
  – O&M
  – Application software
  – Middleware
  – Operating system
  – Virtualization

• Developed in different context, i.e. de facto standard needed
  – In house development
  – Consultant
  – Reusable components
  – Third party products

• Many Languages: C/C++, Java, Erlang

• Node-wide, i.e. multiple processes, multiple processors
Trace storage

- Each traced process write in a shared memory buffers.

- Shared memory buffers are accessed from the LTTng daemon process and data is written to the chosen trace-store:
  - circular “flight recorder” buffer
  - local disk
  - remote disk
  - remote stream (to be completed)
Three Interfaces

1) Creation
   - software developer at development time
   - prepare software with information to enable future tracing

2) Activation
   - field-engineer, system administrator, tester, developer
   - activate, de-activate, listing
   - Same interface for all component, language, layer

3) Analysis
   - field-engineer, system administrator, tester, developer
   - Typically done offline
High Level Architecture

C/C++ Linux Application Environment

Java Application Environment

Erlang Application Environment

Application TracePoint

Java Virtual Machine

Erlang Virtual Machine

Application Marker

Java LTTng API

Erlang LTTng API

LTTng JNI adaptor

LTTng adaptor

Shared Memory Buffer

Local FS

flight recorder buffer

Legend

Control Path

Output Path

LTTng Daemon process

Linux Operating System

Linux Kernel

Kernel Trace Facilities

Remote Disk

Remote Stream
Node-Wide Tracing

• Very low overhead
• Streaming daemon for multi-process/processors, multi-node trace control and retrieval
• Tracing at process start for user-space i.e. detection of active trace sessions and automatic enablement of relevant trace-points
• Tracing at system start for kernel
• Conditional tracing in userspace
• Trace buffers flushing in core dump when process crash to allow post mortem analysis
• Access control, e.g. limit some tracepoint to particular group or role
Node-Wide Tracing

- Quota per tracepoint, subsystem or globally to avoid unacceptable performance degradation
- Name space division in order to guarantee uniqueness of trace-point names and avoid name-collisions
- Structure of trace-points into “layers” in order to give system insight in a certain level (system/function) e.g. com.<company>.<component>.<layer>.<function>.
- Node-relevant condition data, e.g. subscriber id, call id, SIP session id, etc.
Activation

- Activating one trace points or groups of trace points

- Trace session can be controlled by a trace script: activation, deactivation, selecting trace store, etc.

- Create and run more than one trace session in parallel at the same time
Analysis

• Typically done off-line after tracing session ended.

• Tool should have a plug-in architecture to facilitate different kind of analyses and merging different kind of traces.

• Eclipse was chosen because of its wide use and plug-in architecture.