A CME/Chicago Board of Trade/NYMEX Company

Enterprise Context and Thoughts on What We Want in our Toolkit

Vinod Kutty
Senior Director
CME Group

Tracing Summit @ Linux Plumber's Conference
2012-08-30

Context

- Mission-critical Derivatives Exchange
- Linux on x86 2-processor systems
- “Linux” means RHEL at the moment
- Custom kernels in lab only for testing
- Few maintenance windows
- Lots of Java applications, mostly transaction processing
- Lots of multicast UDP
- Low latency == single-digit microseconds for some key apps
- Not easy to reproduce some problems outside production

Trends

- **Busy-spin model**
  - Fewer threads spinning / polling
  - CPU util not a good indicator of issues
- **Kernel-bypass + 10GigE to reduce network latency**
- **Disable power management features**
- **Disable hyper-threading**
- **Getting rid of SMIs (or at least trying to) and buggy BIOS features that may cause issues (e.g. BIOS grabbing perf counters)**
- **Virtualization (KVM, possibly Linux Containers later)**

Sample problems / scenarios

- **Scheduling latencies – too many threads fighting for too few cores**
  - Addressed with fewer threads and busy spin model
- **Measure transaction timestamp going into a server and response going out. What happened during that time?**
- **Regular periodic latencies – what's causing them?**
  - Would be useful to have more consolidated docs on underlying system cyclic events
- **Latencies after idle periods**
- **Packet loss**
- **Lack of visibility into third-party binary only middleware solution libraries / daemons**

Types of Tools We Use

- **Longer term system metrics for capacity planning / trending**
  - Traditionally satisfied with tools that are essentially replacements for sar/vmstat/etc.
  - The less expensive a metric is to collect, the more likely we will collect it because we can
  - Good starting point for troubleshooting

Types of Tools We Use (cont.)

- **Functional issues – app stability, memory leaks, etc.**
  - Use tools closer to app itself
- **Performance issues**
  - Applications tend to use custom code, such as timestamps in transaction payloads
  - Packet sniffing for wire timestamps
  - Often unable to correlate to OS events
  - Hard to detect hardware issues like SMIs

User Requirements for Tracing / Introspection Capabilities (some going back ~ 4 years)

- **Tracing must extend into user space - Java, Perl, Python, etc.**
  - Includes mapping entities such as Java threads / locks to external library / kernel entities
- **Fast + low-impact start up**
- **Overall low impact - performance-wise and production safety-wise**
- **Low footprint (tens of Mbytes ?) install requirements**

Requirements (cont.)

- Process arg matching (e.g. MY_APP instead of 'java')
- Global variables (if needed) must perform well and be safe
- Safety: multiple users/scripts run concurrently
- Flight recorder mode with programmable event triggers for start/stop
- Per process network + disk I/O in real-time and for process accounting
- Latency tracing

Requirements (cont.)

- **Better visibility into network socket buffers, including:**
  - ability to look at packets that were discarded because an app could not keep up
  - ability to overflow same into separate ring buffer for later analysis (possibly with pseudo-fs mapping as pcap file?)

Requirements (cont.)

- **Non-root access/Fine Grained User roles for scripts - with no restriction on location/pre-compilation of scripts**
- **Page cache analysis:**
  - what pages were swapped out from which PIDs?
  - what data from what files is in caches/buffers?
- **Resettable system counters (especially for errors). Business cycles don't match reboot cycles. Clearing counters without rebooting, before start of next biz cycle, is important. Is this for user space or kernel?**

CME Group
A CME/Chicago Board of Trade/NYMEX Company

Requirements (cont.)

- **At-a-glance major subsystem capacity summary (CPU, Disk, Network, Mem, etc.) with configurable thresholds for state change**
  - could be user space tool or pseudo-filesystem entry.
- **Per process accounting of system vs. idle vs. iowait%, etc.**
- **Better async I/O visibility**
- **Model transaction flow through multiple machines (physical or virtual)**

Requirements (cont.)

- Better dm-multipathing stats with user space tools based on user-defined names rather than dm names
- Uniform rates for low-level network stats – issues in the past with device driver specific counter update interval causing probs with user space tools (stats-block-usecs related).

Requirements (cont.)

- Ability to profile the whole system
- Linux Toolset integration – command line tools for common tasks
- Visibility into network traffic when using kernel bypass techniques (RDMA, OpenOnload, etc.)

Summary

- Need low-impact robust tools in Enterprise distros
- Great progress over the years in tracing tools!
- User space tracing importance increasing, especially with kernel-bypass techniques
- Higher level tools for common metrics are needed. Need to parse information out of the increasing amount of data
- Even higher-level consolidated stats for 10000ft view would be useful. Remember Virtual Adrien in Solaris?
  - Show “hot” subsystems (CPU, Mem, I/O, etc.) then drill down from there.