Low overhead virtual machines tracing in a cloud infrastructure

Mohamad Gebai
Michel Dagenais

Dec 7, 2012

École Polytechnique de Montreal

Content

- Area of research
- Current tracing: LTTng vs ftrace / virtio-trace
- Preliminary work
- Main project proposal
- References

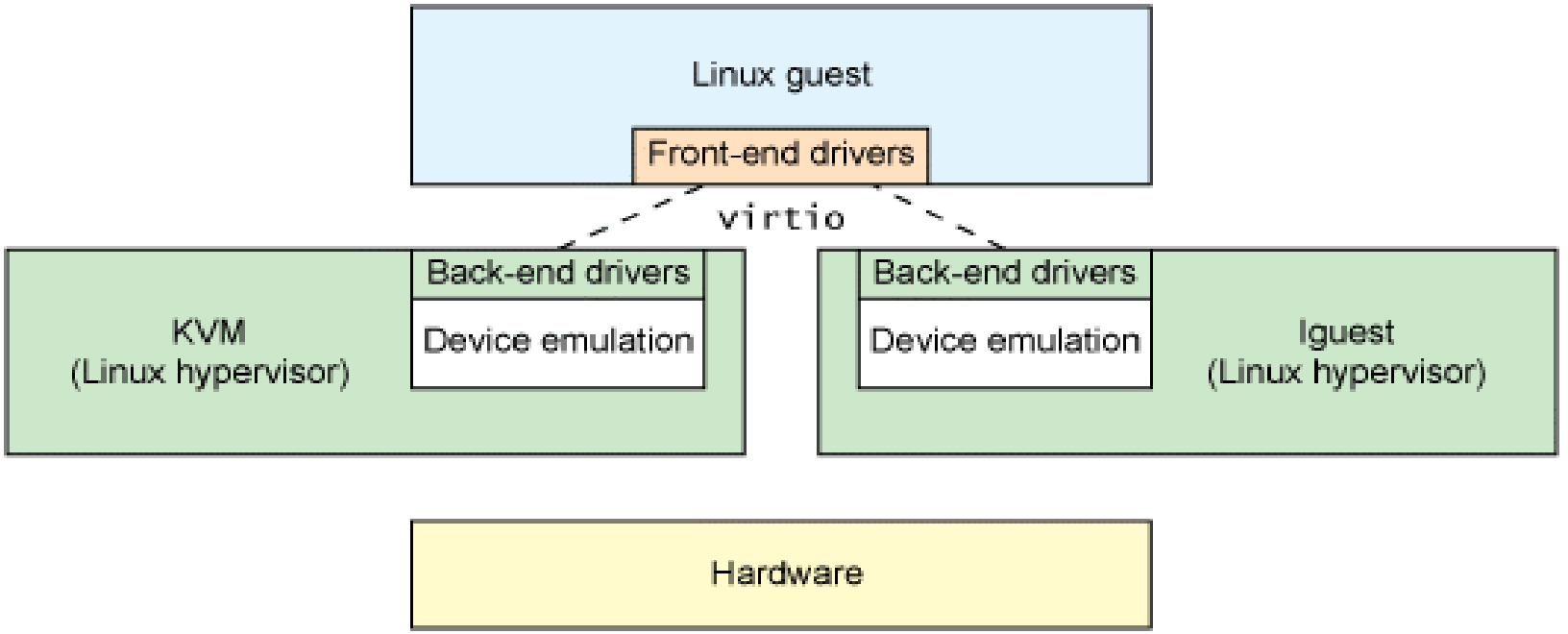
Area of research

- Tracing virtualized machines in a cloud context

Current VM tracing

- LTTng-tools 2.1: support for network streaming
- Sending trace data between the guest and the host over the network
- Might not be the best suited approach for the guest and the host to communicate trace data
  - Usage of network bandwidth
  - IP-based communications may be inadvertently blocked by the virtual machine administrator

Virtio

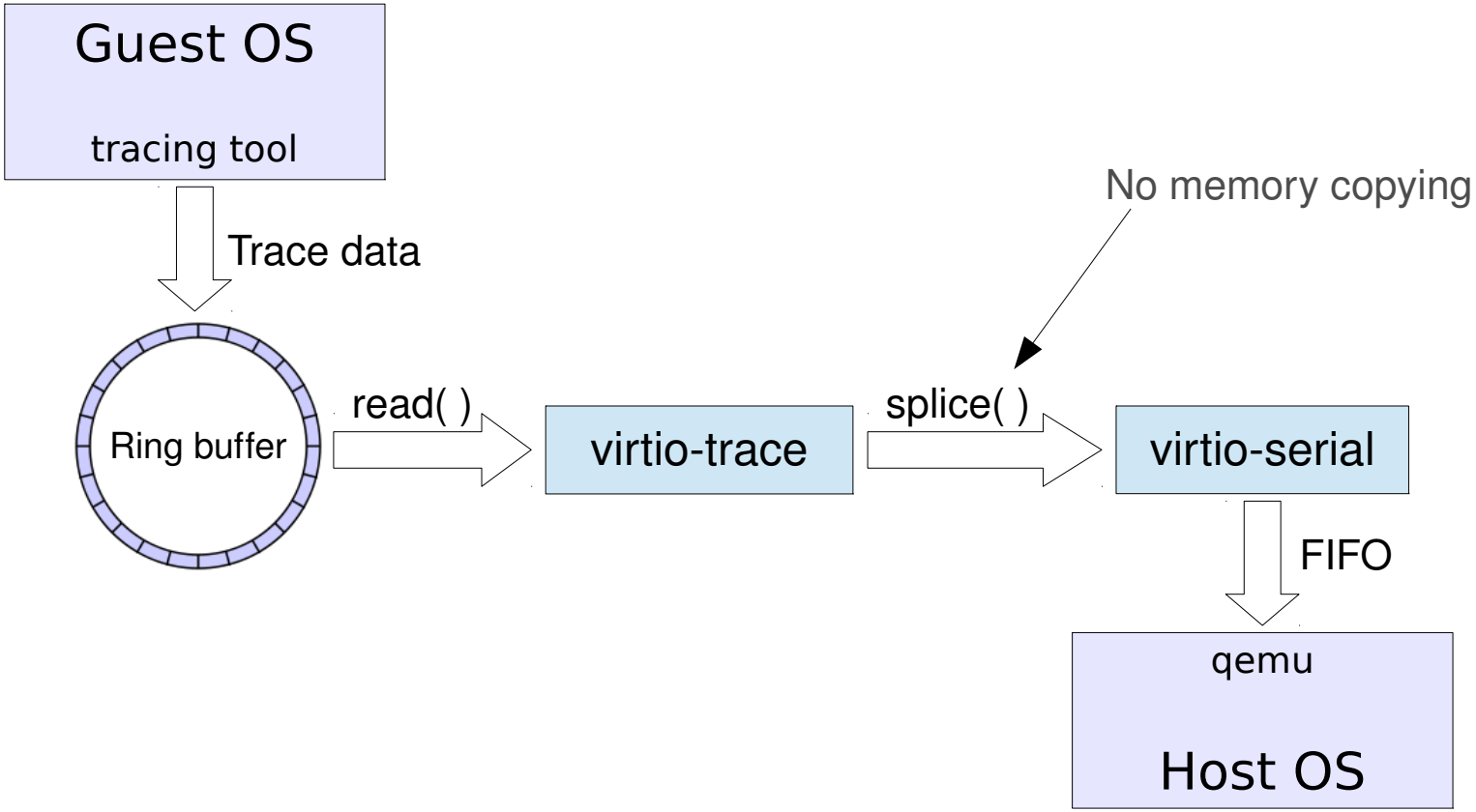


http://www.ibm.com/developerworks/linux/library/l-virtio/figure2.gif

- Based on paravirtualization : the operating system knows it is running in a VM
- Virtio: a set of device drivers for virtual machines
  - virtio-net (network adapter)
  - virtio-balloon (dynamic allocation and disallocation of the guest's memory)
  - virtio-serial (char device – communication channel)

Virtio-trace

- Device driver for tracing since Linux kernel version 3.7
- Sending traces from guest to host without copying
- Low overhead



Preliminary work

- Running benchmarks on native VM, with ftrace (using virtio-trace), with LTTng (using streaming)
- Compare the guest to host communication overhead between ftrace and LTTng

Main project proposal

- Improve the performance and extent of virtual machine tracing
  - Adapt the best features of existing kernel ring buffers for LTTng and ftrace
  - Insure uniform clock sources between kernel and user-space applications in physical and virtual machines
  - Trace and analyze VM-specific behavior such as page sharing with KSM (kernel same page merging)

References

- http://www.virtualbox.org/manual/ch06.html
- http://www.ibm.com/developerworks/linux/library/l-virtio/index.html
- http://log.amitshah.net/tag/virtio-serial/
- https://events.linuxfoundation.org/images/stories/pdf/lcjp2012_yunomae.pdf
- https://lkml.org/lkml/2012/7/23/511
- https://lkml.org/lkml/2012/6/5/143

Large Scale Data Center Monitoring and Debugging Infrastructure

Julien Desfossez
Michel Dagenais

Dec 7, 2012

École Polytechnique de Montreal

Content

- Problem definition
- State of the art
- Proposed approach
- Early work
- References

Problem Definition

- Large scale data centers : hosting, cloud computing, scientific computing, etc
- Virtualized data-centers
- Monitoring inefficient and costly (/proc reading, locks and system calls)
- Mostly high-level metrics (CPU, memory, load, etc) or application-level monitoring without correlation to the OS or physical layer
- No virtualisation/hypervisor metrics

Monitoring

- Multiple aspects:
  - Availability testing
  - Load measurement
  - Trending
  - Reporting

State of the Art

- Commercial products add “Cloud” to their softwares titles
- Most of commercial systems focus on the interface and a predefined set of tests
- Reuse the same technology as monitoring a single machine and the associated services
- Focus mainly on capacity planning with high-level information

State of the Art

- Swift Recon : middle-man metrics collector
  - Common metrics : Load average, sockets stats, /proc/meminfo, etc
  - Object storage dedicated metrics :
    - MD5 of each ring file
    - most recent object replication time
    - count of each type of quarantined files: account, container, or object.
    - Count of “async_pendings” (deferred container updates) on disk

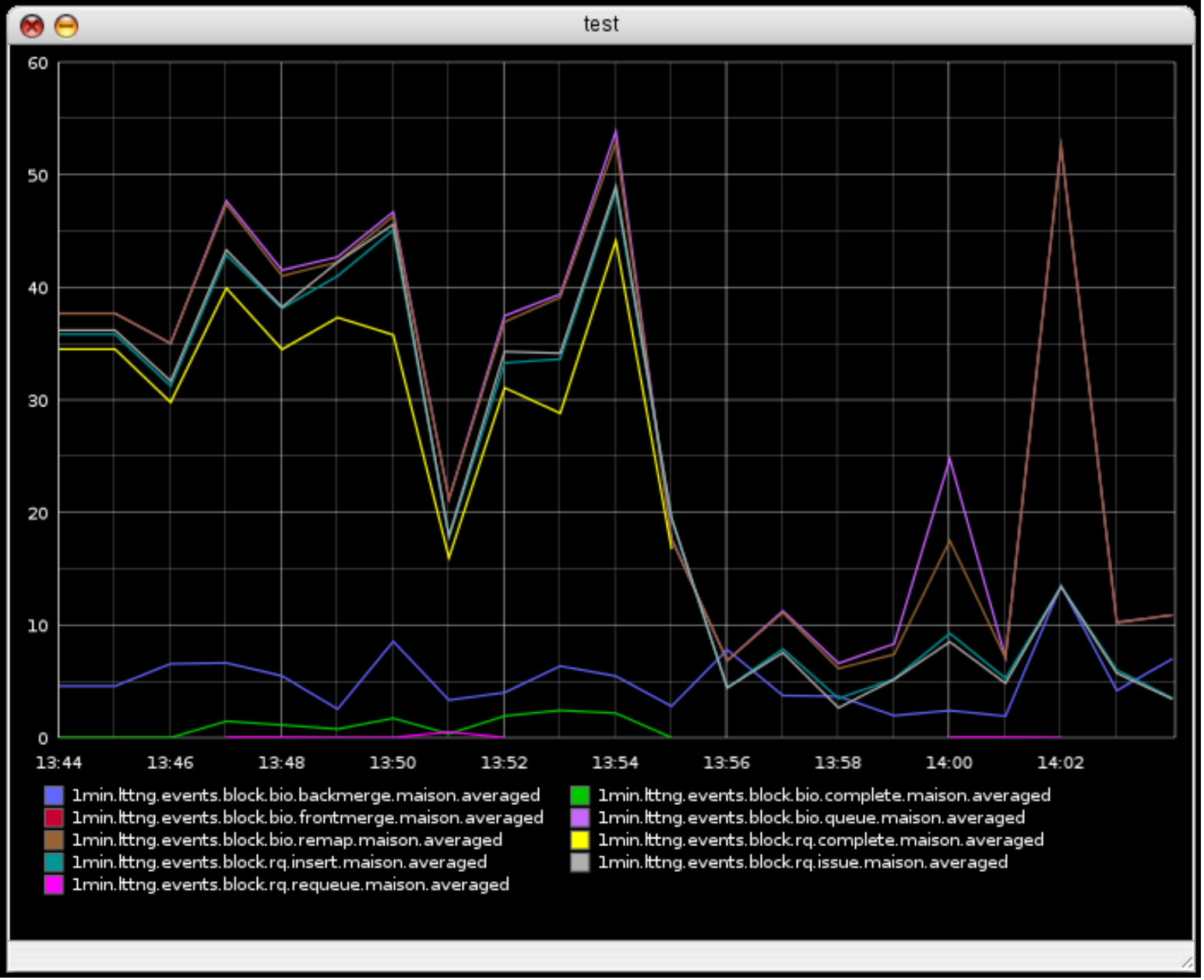
State of the Art

- Rackspace Cloud Monitoring
  - API to access monitoring data
  - High-level remote tests
    - HTTP requests to pages
    - TCP connect
    - Banner check

State of the Art

- RockSteady
  - Metric analysis and correlation engine
  - User-defined metrics
  - Sending/receiving done with message queueing (RabbitMQ)
  - Optional graphing engine (Graphite)
  - Complex Event Processing to extract root-cause from event by integrating multiple data sources

```
graphite>create test
graphite>draw 1min.lttng.events.block.*.*.*.* from -20min in test
graphite>
```



Proposed Approach

- Tracing can be used as an efficient monitoring backend (Master's results)
- LTTng + Perf PMU counters
- Extract metrics from trace data
- Define metrics relevant for large scale data centers and virtualized environments
- Generate statistics and detailled usage reports

Proposed Approach

- Dynamically adjustable level of information (high-level metrics down to detailled in application behavior)
- Streaming and live analysis
- Aggregation on intermediate nodes (depending on topology)

Target environment

- OpenStack
- KVM
- LXC
- cgroups

Early Work

- Basic metrics computation and reporting (LTTngTop during Master)
- Streaming traces (lttng-tools 2.1)
- Index generation
- Live analysis (in progress)

References

- OpenStack Manual – chapter 6, OpenStack Object Storage Monitoring
- Proven Practice: Metrics for Virtualization Management – http://communities.vmware.com/docs/DOC-11375?decorator
- Intel Virtualization Performance Metrics - http://software.intel.com/en-us/articles/virtualization-perform
- Google Rocksteady - https://code.google.com/p/rocksteady/