Large-scale performance monitoring framework

Julien Desfossez
Michel Dagenais

May 2013
École Polytechnique de Montreal

Summary

- Introduction
- Research question
- Objectives
- Litterature review
- Detailled objectives
- Future work
- Conclusion

Introduction

- Large-scale infrastructure (cloud computing)
- Massive use of virtualization
- High level monitoring
- Targetted monitoring (per-application)
- Fined-grained monitoring is expensive

Example of interesting performance data

- Perf counters
- Scheduling events
- Page faults
- Parameters and/or frequency of syscalls

High-level problematic

- Determine the best way to collect and analyze accurate and detailled metrics from the servers in large-scale data-centers
- Production environment
- Minimum impact of monitored systems
- Real Time

Objectives

- Collect in real time, high resolution performance data
- Monitor in high performance production environments
- Adjustable level of details
- Framework to collect and detect performance problems

Litterature review : cloud monitoring

- Distributed architectures
- High-level metrics
- XML, SOAP, etc
- Attempt to standardize on AppFlow
- Algorithms to select the best cloud provider

Litterature review : virtualization monitoring

- Hypervisor level monitoring
- VM preemption for monitoring syscalls
- Virtualization of perf counters
- Scheduler optimization

Litterature review : cloud applications

- Twitter – Zipkin
- Google – Dapper
- Google – Rocksteady

Litterature review : summary

- Lots of papers focus on application-specific monitoring
- Simulations or limited test machines
- Lack of efficient methods and algorithms for low level measurements
- Lack of methods to collection execution flow
- Across multiple layers (applications, kernel, hypervisor, VM kernel and user-space)

Detailled objectives

- Extract traces on the network
- Analyze in real time trace data
- Develop algorithms and methodologies to aggregate traces at high throughput
- Automatic and manual control facilites

Extract traces

- Large volume
- Minimum delay between production and availability
- Take into account routing and security constraints

Real-time analysis

- Synchronize all trace streams
- Send metadata before data
- Minimum resources usage (disk, network, CPU)
- Take into account execution modes (energy saving)

Traces aggregation

- Extract metrics from traces
- High throughput and real time
- Distributed analysis depending on topology, ressources and data availability

Control

- Manual, SSH
- Automation of tracepoint activation/deactivation
- Automatic snapshot recording in flight recorder mode
- Inspired from algorithmic trading for automated reaction on events and state

Future work

- Standard analysis depending on environments and applications
- Optimization of VM placement in data-centers
- Rules, filters, triggers

Conclusion

- Determine the best way to transport and analyse performance data in large-scale data-centers
- Control and automate trace recording and collecting
- Production environment
- Framework for a distributed low-level performance measurement

Virtual machine monitoring using trace analysis

Mohamad Gebai
Michel Dagenais

2 May, 2013
École Polytechnique de Montreal

Content

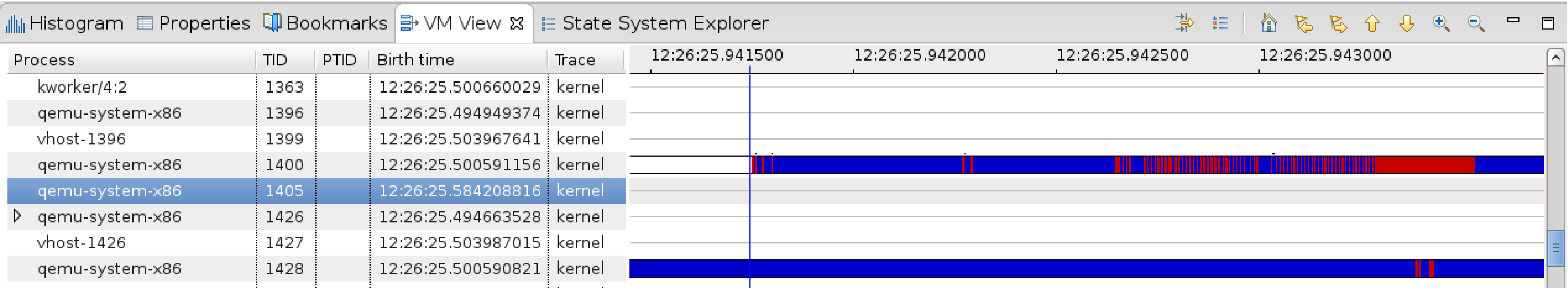
- General objectives
- TMF – Virtual Machine View
- Simultaneous tracing
- Trace synchronization
- Future work

General objectives

- Getting the state of a virtual machine at a certain point in time
- Quantifying the overhead added for virtualization
- Monitoring multiple VM on a single host OS
- Finding performance setback due to resource sharing among VMs
- Building a state system in TMF specific to virtualization

TMF Virtual Machine View

- Shows the state of the VM through time
- Based on kvm tracepoints
- Gives the exit reason upon kvm_exit events



- 2 Virtual machines with 1 virtual CPU
- Blue: VM running
  Red: Hypervisor running (overhead)
  White: VM is scheduled out

Simultaneous tracing

- Trace the host to monitor the VM state through time
- Trace the VM for regular process analysis
- Launch workloads in VM (CPU, memory benchmarks)
- Correlate workloads in the VM to its behavior on the host

Trace synchronization

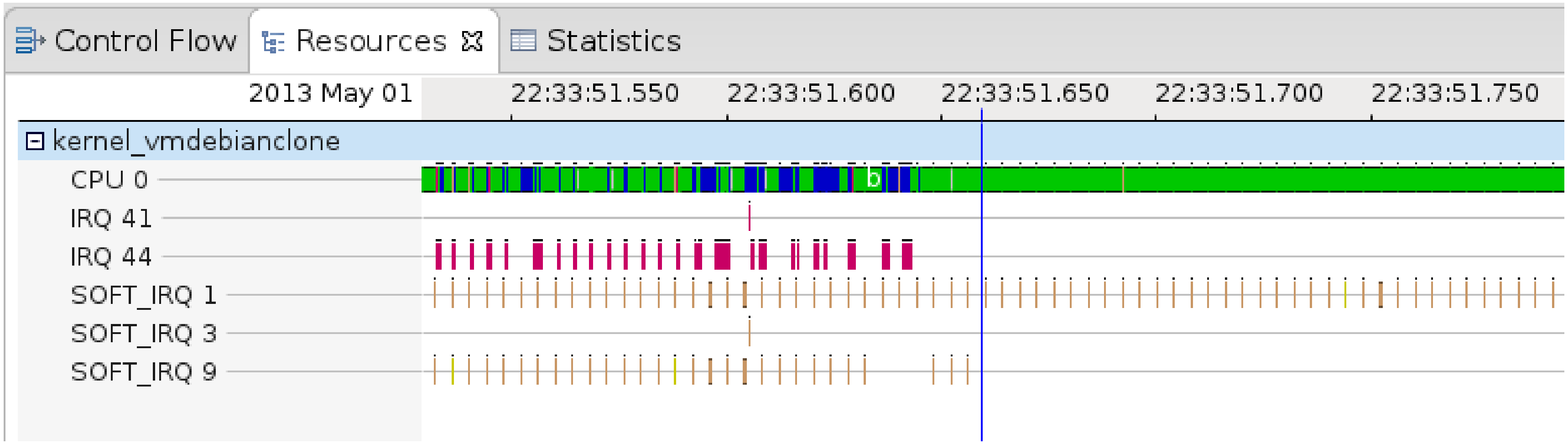
- Clocks in VM and host are not synchronized
- Getting the offset at any point in time
- Applying the time offset on the VM events

Future work

- Further investigation for more accurate delay calculation (considering the hypercall overhead)
- Applying the delay in the VM for time synchronization
- TMF view: integrating the exit reason within the state system to give more information on the VM status
- Build a state system for VM that can be adapted to Java Virtual Machines

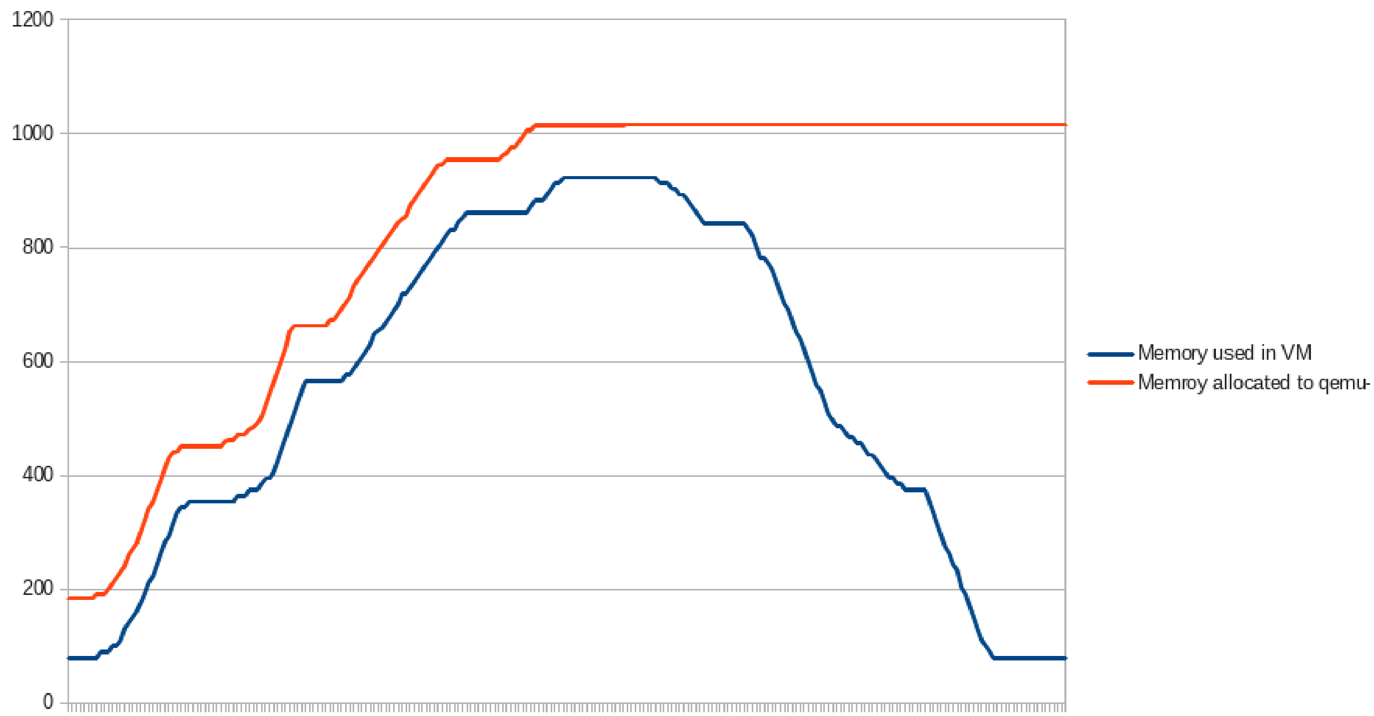
Future work (2)

- TMF View - vCPU usage
- Highlight the competition between multiple VMs over CPU time
- Highlight when a VM is preempted by another VM
- Highlight if a VM is denied CPU time because of preemption or because no workload is to be executed
- Highlight requested vCPU time vs allocated CPU time



Future work (3)

- TMF View - Memory usage
- Keep track of allocated and freed memory by the processes inside the VM
- Keep track of touched memory pages by the VM in the host
- Point out memory pages that can be freed by the hypervisor for memory overcommitment



Final objectives

- Highlight status information specific to VMs
- Point out resource sharing among multiple VMs on a single host
- Point out potential optimizations such as memory overcommitment
- Provide information useful for VMs migration in order to avoid competition over the same resources

References

- [1] D. Bueso, E. Heymann, and M. A. Senar, “Towards Efficient Working Set Estimations in Virtual Machines.”
- [2] D. Marinescu and R. Kröger, “State of the art in autonomic computing and virtualization,” Distributed Systems Lab, Wiesbaden University of Applied Sciences, 2007.
- [3] K. Anshumali, T. Chappell, and W. Gomes, “Intel 64 and ia-32 software developer's manual.pdf,” Intel Technology Journal, vol. 14, pp. 104–127, 2010.
- [4] D. Marinescu and R. Kröger, “State of the art in autonomic computing and virtualization,” Distributed Systems Lab, Wiesbaden University of Applied Sciences, 2007.