Discovering Hypervisor Overheads using Micro and Macrobenchmarks

Andrea Bastoni Paolo Palana
Daniel P. Bovet Marco Cesati

University of Rome "Tor Vergata"
System Programming Research Group
{bastoni,palana,bovet,cesati}@sprg.uniroma2.it

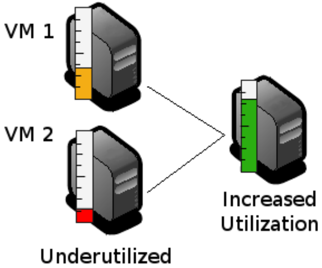
Computer Architecture and Operating System co-design
Pisa – January 23, 2010

An overview of Virtualization

Virtualization refers to the separation of a *service request* from the *physical delivery* that service

Virtualization is not a new concept:

- was introduced by IBM (IBM M44, IBM VM/370) in the 1960's
- but has gained renewed interest in recent years, especially for *server consolidation*



Server consolidation allows to *reduce* waste of resources by consolidating *group* of servers into *one* physical machine

Full Virtualization

In full virtualization solutions:

- *unmodified* guest kernels run on top of a virtualization layer
- guest VMs are unaware of the virtualization layer

The Virtual Machine Monitor (*hypervisor*) deceives guest kernels:

- all guest kernels *sensitive* instructions are *trapped* or *binary-translated* to safely execute on the physical CPU
- the guest VM believes to run directly on the real hardware

Using virtualization to perform server consolidation allows for an higher level of *isolation* among consolidated servers

On *x86* architecture:

- guest kernels and applications run at a lower privilege level (*ring*) than the hypervisor
- *ring deprivileging* is the major source of architectural problems in supporting x86 full virtualization

Hardware support to virtualization

Supporting unmodified x86 guest without ring deprivileging is possible:

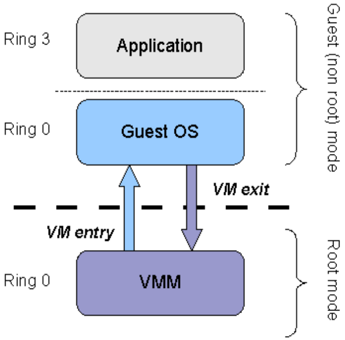
- hardware *modifications and extensions* are needed
- *AMD* (AMD-V) and *Intel* (Intel VT-x/VT-i) started integrating such modifications in 2005

Two new operating modes:

- *guest mode* for VM
- *root mode* for hypervisor

Guest OSes run in their original privilege levels

Hypervisor controls guest execution through *control bits* and *hardware-defined* structures (*VMCS*)



Motivation

Virtual Machines execution is *slowed down* by several overheads

Previous studies that analyzed these overheads suffer some drawbacks:

- studies focusing on the impact of virtualization on server consolidation use entirely different workloads or stress different hardware components
  - may prevent the discovery of overheads and scalability problems
- studies using microbenchmarks focus on a single virtualized component of the system
  - cannot register interactions among virtualized components

Furthermore:

- few studies targeted overheads of hardware support
- few studies explored virtualization of 64-bit guest on 64-bit host

The proposed approach

Evaluate performance and scalability for three open source virtualization technologies with hardware support

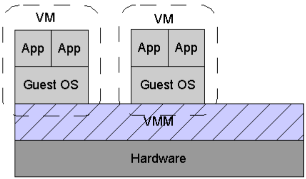
- Global performance evaluation using *SPECweb2005* macrobenchmark
  - Some behaviours are difficult to explain
- Integrate macrobenchmark results using microbenchmark
  - Some behaviours are unexpected and difficult to explain

A more detailed analysis is needed, but...

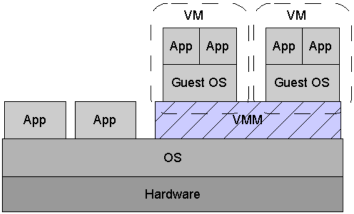
- Currently available profiling and monitoring techniques provide inadequate support to full virtualization
- We propose some *architectural changes* that may help in overcoming these limitations

Tested *full virtualization* solutions

Xen	**KVM**	**VirtualBox**
Type 1 Hypervisor	Hybrid type 1 Hypervisor (The OS *is* the VMM)	Type 2 Hypervisor
Modified QEMU device model	QEMU device model	Own device model (Originally based on QEMU)



Type 1 Hypervisor



Type 2 Hypervisor

Benchmarks and Performance Metrics

Macrobenchmark (SPECweb2005):

- uniform workload on all virtual machines concurrently executing on the hypervisor
- evaluate overheads due to interactions between system components (CPU, network, disk)
- is the de-facto standard for web server performance evaluation
  - Simulates a real web server workload in a real environment

SPECweb performance metric:

- SPEC simultaneous sessions: number of sessions the SUT supports while meeting a pre-defined Quality-of-Service level
  - QoS requirements are defined by two parameters (*Time_Good*, *Time_Tolerable*)
  - these parameters identify the maximum aggregate response time allowed for each page request

Benchmarks and Performance Metrics

Microbenchmarks:

- evaluate overheads of a single component of the system
- `Bzip2`: CPU overheads
- `Netperf`: Network overheads
- `dd`: disk overheads

Performance metrics are defined by each microbenchmark

- Bzip: seconds
- Netperf: Mb/s
- dd: MB/s

Specific test of 64 bit VMs on 64 bit Hypervisor:

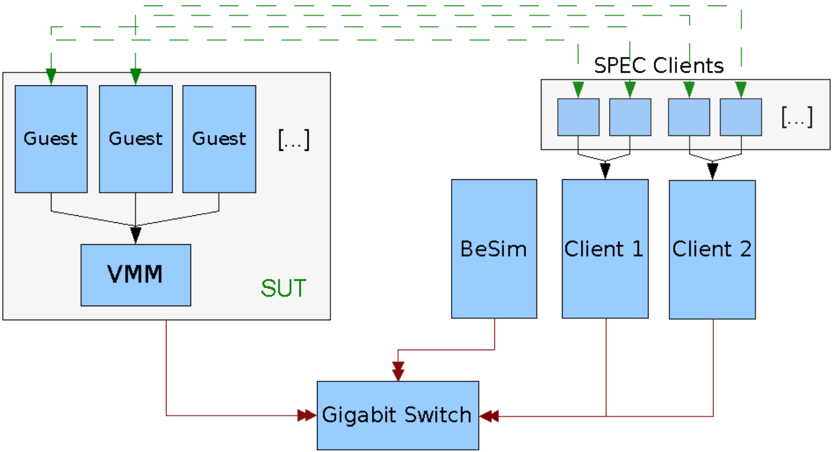
- compare performance with previous studies on 32 bit guest over 32 bit host

Experimental setup

- Virtualizators version:
  - Xen: 3.3.0, KVM: 75, Virtual Box: 2.0.6
- Hypervisor and Guests Linux kernel: 2.6.21.7
- System Under Testing:
  - AMD Opteron: 4 dual-core NUMA, 2GHz, 16 GB Ram
  - HTTP Apache Web Server 2.2.9
- Each VM has 1.5 GB Ram and 1 or 2 VCPUs
- 1 to 10 VMs concurrently executing
- Tests setup:
  - *E-Commerce* SPECweb workload (3 iterations = 1 complete run)
  - *Netperf* TCP test (standard 10 second test)
  - *dd* raw copy of *Gentoo livecd image* (742 MB)
  - *bzip* compression of the same file
- Systems rebooted after each run

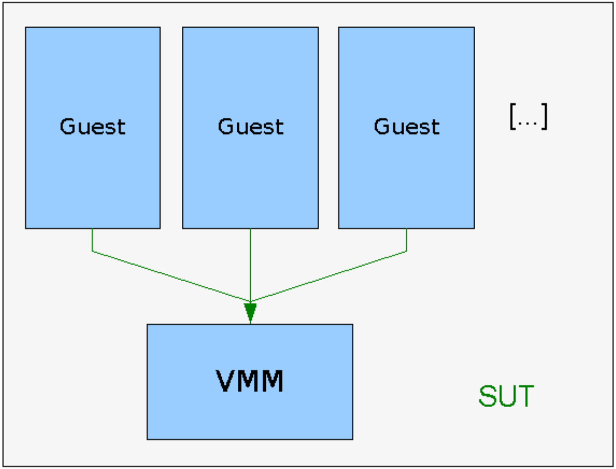
Experimental Setup

SPECweb test setup



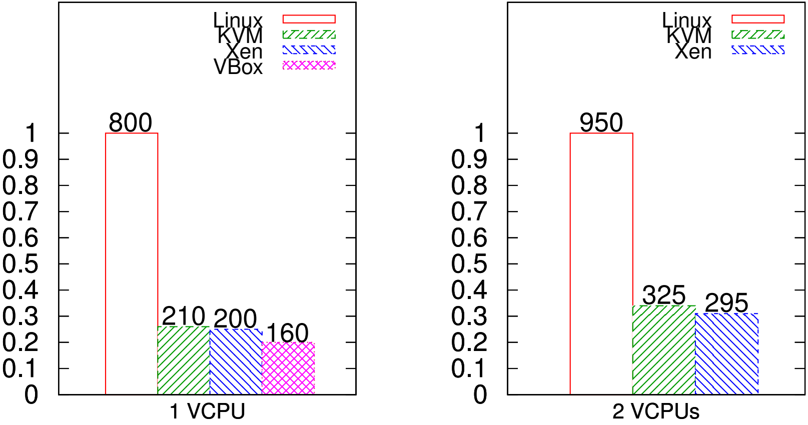
Experimental Setup

Microbenchmark test setup



Macrobenchmark performance

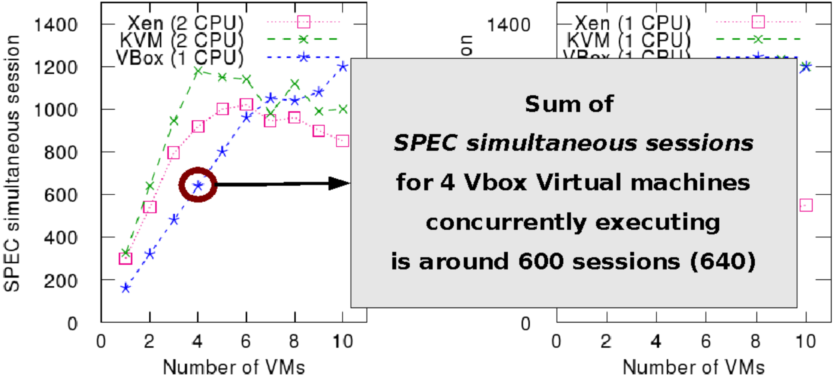
SPEC simultaneous sessions normalized to Linux (“Higher is better”)



- Performance far below non-virtualized Linux
- Adding a virtual CPU introduces additional overheads (NUMA)

Macrobenchmark scalability

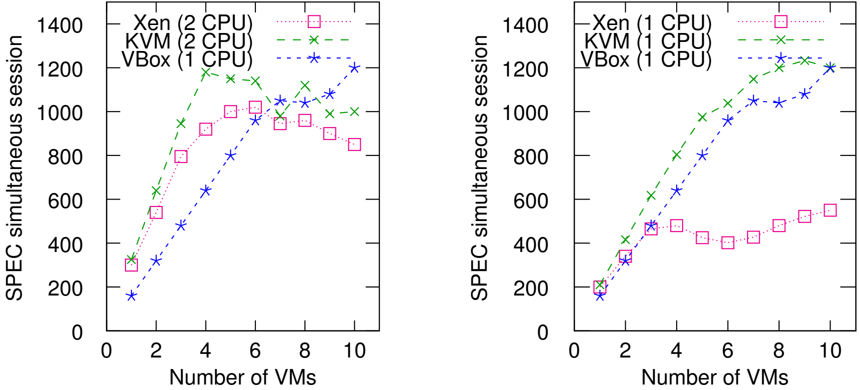
Cumulative SPEC simultaneous sessions as number of VM increases



Linux performance not shown: 2750 sessions on average

Macrobenchmark scalability

Cumulative SPEC simultaneous sessions as number of VM increases

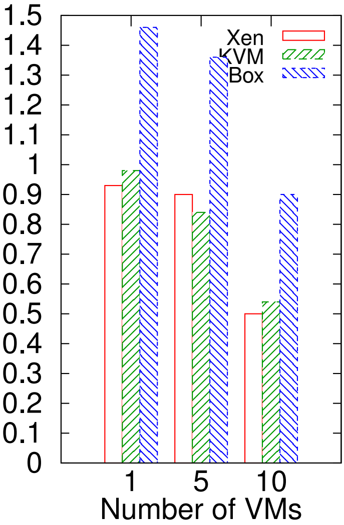


Linux performance not shown: 2750 sessions on average

- KVM and Xen performance drops as number of VMs increases
- Xen (1 VCPU) obtains poor performance

Microbenchmark results

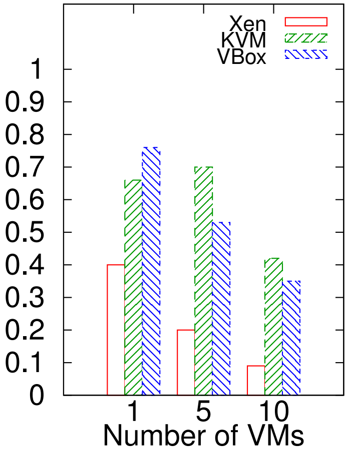
Bzip2 performance normalized to Linux (“Higher is better”, 1 VCPU)



- VBox performs better than non-virtualized Linux
- VBox runs threads at kernel privilege level
- avoids many sanity checks normally done by kernel in dealing with userspace
- Likely to influence the good VBox performance in SPEC scalability

Microbenchmark results

Netperf performance normalized to Linux (“Higher is better”, 1 VCPU)



- Xen performs very poorly
- Confirm results from Apparao et al. (2006)[1]
- Likely to influence the poor SPEC cumulative performance of Xen 1 VCPU

[1] Apparao et al. (2006), Characterization of network processing overheads in Xen, In *Proc. of the 2nd Int. Workshop on Virtualization Technology in Distributed Computing*

Limitations of current analysis techniques

Comparison of micro and macrobenchmark results:

- High overheads with respect to Linux
- 64 bit over 64 bit obtains similar results to those previously published (32 bit on 32 bit)
- unexpected behaviours (e.g., VBox CPU performance)
- behaviours difficult to explain (e.g., Xen poor cumulative performance)

Need of a deeper analysis:

- *on-line monitoring*
- *profiling*

Currently available tools are limited:

- `top`, `sar` provide limited information on VM resources usage
- Profiling and monitoring tools for virtualized solutions are available for *paravirtualized* techniques only (*Xenoprof*, *Xenmon*)

Limitations of current analysis techniques

Xenoprof[2] profiler:

- “Porting of Oprofile” to Xen
- Two level profiler:
  - *Hypervisor layer (Xenoprof)*: monitors performance counters and forwards PC interrupts to domains
  - *Domain layer*: modified Oprofile for attributing samples to routines inside VM
- Domains need to be *modified* to interact with VMM

In a virtualized environment:

- Profiling cannot be centralized:
  - Hypervisor cannot determine the process currently running in a guest domain
- Domains cannot access hardware performance counters

[2] Menon et al. (2005), Diagnosing performance overheads in the xen virtual machine environment, In *VEE '05: Proc of the 1st ACM/USENIX international conference on Virtual execution environments*

Virtualization-aware hardware performance counters

Extend the virtualization hardware support to include *virtualized performance counters*:

- guest can access hardware performance counters
- non-modified profiling and monitoring tools can directly execute on guest domains
- simple *system-wide* profiling of the machine (VMM and VMs)

Virtualization-aware Hardware Performance Counter architecture

- Expand the Virtual Machine Control Structure (VMCS)
- Save the status of physical performance counters upon Virtual Machine switch

VMCS is the main hardware support control structure:

- controls root $\rightleftarrows$ non-root VM operating modes transitions
- each VM has one VMCS per VCPU

Virtualization-aware hardware performance counters

While active and running, each non-modified VM can program hardware performance counters:

- the VMM intercepts PC interrupts and delivers them to the VM

On Virtual Machine switch:

- the current value of PCs (used by the switched-out VM) is *saved on the VMCS*
- the PC values used by the switched-in VM are restored

Furthermore:

- Performance counters programming information can also be saved on VMCS
- Hypervisor performance counters accounting can be done similarly

Virtualization-aware hardware performance counters

Each guest domain:

- has coherent access to its performance statistics
- can directly execute *non-modified* profiling and monitoring tools
  - e.g., Oprofile, in-kernel support to hardware performance counters. . .

Easy system-wide profiling of the machine:

- can be done by the hypervisor only
- *gather* per-VM performance counters information

Conclusion

Hypervisor performance analysis:

- Neither macrobenchmarks nor microbenchmarks can fully explain some behaviours of full virtualization solutions
- A more detailed analysis is needed

Profiling and monitoring tools:

- Available tools cannot be used in a full virtualized environment
- Virtualization-aware hardware performance counters integrate hardware performance counters in the hardware virtualization support
- Non-modified profiling and monitoring tools can be used on full virtualized guests

Thank you!