HEP Applications in Virtual Machines

Jakob Blomer

April 15th 2008
(Benchmarks of Summer 2007)

Delivering Software

The classical approach: source package + install script

```
tar xfvz root_v5.15.08.source.tar.gz
```
$\Longrightarrow$ `export ROOTSYS=/opt/root`
$\Longrightarrow$ `./configure --enable-`$\cdots$ ($\approx 5\,000$ LOC)
$\Longrightarrow$ `make; make install` ($\approx 6\,100$ LOC)
$\Longrightarrow$ Libraries and binaries with dependencies

```
$ ldd root
linux-gate.so.1 => (0xb7f7a000)
libXext.so.6 => /usr/X11R6/lib/libXext.so.6 (0xb7f64000)
libX11.so.6 => /usr/X11R6/lib/libX11.so.6 (0xb7ea3000)
libXft.so.2 => /usr/X11R6/lib/libXft.so.2 (0xb7e92000)
libstdc++.so.6 => /usr/lib/libstdc++.so.6 (0xb7dce000)
...
```

The VM approach: Disk image + VM description (< 50 LOC)
Installation by copy (assumed a VMM is installed)

Delivering Software

The classical approach: source package + install script

`tar xfvz root_v5.15.08.source.tar.gz`

$\Longrightarrow$ `export ROOTSYS=/opt/root`

$\Longrightarrow$ `./configure --enable-···` ($\approx 5\,000$ LOC)

$\Longrightarrow$ `make; make install` ($\approx 6\,100$ LOC)

$\Longrightarrow$ Libraries and binaries with dependencies

```
$ ldd root
linux-gate.so.1 => (0xb7f7a000)
libXext.so.6 => /usr/X11R6/lib/libXext.so.6 (0xb7f64000)
libX11.so.6 => /usr/X11R6/lib/libX11.so.6 (0xb7ea3000)
libXft.so.2 => /usr/X11R6/lib/libXft.so.2 (0xb7e92000)
libstdc++.so.6 => /usr/lib/libstdc++.so.6 (0xb7dce000)
...
```

The VM approach: Disk image + VM description (< 50 LOC)

Installation by copy (assumed a VMM is installed)

Delivering Software

The classical approach: source package + install script

```
tar xfvz root_v5.15.08.source.tar.gz
```

$\Longrightarrow$ `export ROOTSYS=/opt/root`

$\Longrightarrow$ `./configure --enable-···` ($\approx 5\,000$ LOC)

$\Longrightarrow$ `make; make install` ($\approx 6\,100$ LOC)

$\Longrightarrow$ Libraries and binaries with dependencies

```
$ ldd root
linux-gate.so.1 => (0xb7f7a000)
libXext.so.6 => /usr/X11R6/lib/libXext.so.6 (0xb7f64000)
libX11.so.6 => /usr/X11R6/lib/libX11.so.6 (0xb7ea3000)
libXft.so.2 => /usr/X11R6/lib/libXft.so.2 (0xb7e92000)
libstdc++.so.6 => /usr/lib/libstdc++.so.6 (0xb7dce000)
...
```

The VM approach: Disk image + VM description (< 50 LOC)

Installation by copy (assumed a VMM is installed)

Delivering Software

The classical approach: source package + install script

```
tar xfvz root_v5.15.08.source.tar.gz
```

$\Longrightarrow$ `export ROOTSYS=/opt/root`

$\Longrightarrow$ `./configure --enable-...` ($\approx 5\,000$ LOC)

$\Longrightarrow$ `make; make install` ($\approx 6\,100$ LOC)

$\Longrightarrow$ Libraries and binaries with dependencies

```
$ ldd root
linux-gate.so.1 => (0xb7f7a000)
libXext.so.6 => /usr/X11R6/lib/libXext.so.6 (0xb7f64000)
libX11.so.6 => /usr/X11R6/lib/libX11.so.6 (0xb7ea3000)
libXft.so.2 => /usr/X11R6/lib/libXft.so.2 (0xb7e92000)
libstdc++.so.6 => /usr/lib/libstdc++.so.6 (0xb7dce000)
...
```

The VM approach: Disk image + VM description (< 50 LOC)

Installation by copy (assumed a VMM is installed)

Delivering Software

The classical approach: source package + install script

`tar xfvz root_v5.15.08.source.tar.gz`

$\Longrightarrow$ `export ROOTSYS=/opt/root`

$\Longrightarrow$ `./configure --enable-`$\cdots$ ($\approx 5\,000$ LOC)

$\Longrightarrow$ `make; make install` ($\approx 6\,100$ LOC)

$\Longrightarrow$ Libraries and binaries with dependencies

```
$ ldd root
linux-gate.so.1 => (0xb7f7a000)
libXext.so.6 => /usr/X11R6/lib/libXext.so.6 (0xb7f64000)
libX11.so.6 => /usr/X11R6/lib/libX11.so.6 (0xb7ea3000)
libXft.so.2 => /usr/X11R6/lib/libXft.so.2 (0xb7e92000)
libstdc++.so.6 => /usr/lib/libstdc++.so.6 (0xb7dce000)
...
```

The VM approach: Disk image + VM description (< 50 LOC)

Installation by copy (assumed a VMM is installed)

Delivering Software

The classical approach: source package + install script

`tar xfvz root_v5.15.08.source.tar.gz`

$\Longrightarrow$ `export ROOTSYS=/opt/root`

$\Longrightarrow$ `./configure --enable-`$\cdots$ ($\approx 5\,000$ LOC)

$\Longrightarrow$ `make; make install` ($\approx 6\,100$ LOC)

$\Longrightarrow$ Libraries and binaries with dependencies

```
$ ldd root
linux-gate.so.1 => (0xb7f7a000)
libXext.so.6 => /usr/X11R6/lib/libXext.so.6 (0xb7f64000)
libX11.so.6 => /usr/X11R6/lib/libX11.so.6 (0xb7ea3000)
libXft.so.2 => /usr/X11R6/lib/libXft.so.2 (0xb7e92000)
libstdc++.so.6 => /usr/lib/libstdc++.so.6 (0xb7dce000)
...
```

The VM approach: Disk image + VM description (< 50 LOC)

Installation by copy (assumed a VMM is installed)

Delivering Software

The classical approach: source package + install script

`tar xfvz root_v5.15.08.source.tar.gz`

$\Longrightarrow$ `export ROOTSYS=/opt/root`

$\Longrightarrow$ `./configure --enable-`$\cdots$ ($\approx 5\,000$ LOC)

$\Longrightarrow$ `make; make install` ($\approx 6\,100$ LOC)

$\Longrightarrow$ Libraries and binaries with dependencies

```
$ ldd root
linux-gate.so.1 => (0xb7f7a000)
libXext.so.6 => /usr/X11R6/lib/libXext.so.6 (0xb7f64000)
libX11.so.6 => /usr/X11R6/lib/libX11.so.6 (0xb7ea3000)
libXft.so.2 => /usr/X11R6/lib/libXft.so.2 (0xb7e92000)
libstdc++.so.6 => /usr/lib/libstdc++.so.6 (0xb7dce000)
...
```

The VM approach: Disk image + VM description (< 50 LOC)

Installation by copy (assumed a VMM is installed)

Delivering Software

The classical approach: source package + install script

`tar xfvz root_v5.15.08.source.tar.gz`

$\Longrightarrow$ `export ROOTSYS=/opt/root`

$\Longrightarrow$ `./configure --enable-`$\cdots$ ($\approx 5\,000$ LOC)

$\Longrightarrow$ `make; make install` ($\approx 6\,100$ LOC)

$\Longrightarrow$ Libraries and binaries with dependencies

```
$ ldd root
linux-gate.so.1 => (0xb7f7a000)
libXext.so.6 => /usr/X11R6/lib/libXext.so.6 (0xb7f64000)
libX11.so.6 => /usr/X11R6/lib/libX11.so.6 (0xb7ea3000)
libXft.so.2 => /usr/X11R6/lib/libXft.so.2 (0xb7e92000)
libstdc++.so.6 => /usr/lib/libstdc++.so.6 (0xb7dce000)
...
```

The VM approach: Disk image + VM description (< 50 LOC)

Installation by copy (assumed a VMM is installed)

Delivering Software

The classical approach: source package + install script

`tar xfvz root_v5.15.08.source.tar.gz`

$\Longrightarrow$ `export ROOTSYS=/opt/root`

$\Longrightarrow$ `./configure --enable-`$\cdots$ ($\approx 5\,000$ LOC)

$\Longrightarrow$ `make; make install` ($\approx 6\,100$ LOC)

$\Longrightarrow$ Libraries and binaries with dependencies

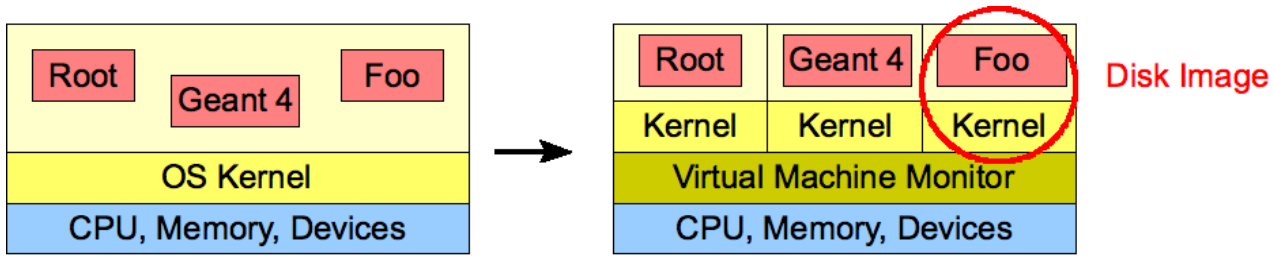
```
$ ldd root
linux-gate.so.1 => (0xb7f7a000)
libXext.so.6 => /usr/X11R6/lib/libXext.so.6 (0xb7f64000)
libX11.so.6 => /usr/X11R6/lib/libX11.so.6 (0xb7ea3000)
libXft.so.2 => /usr/X11R6/lib/libXft.so.2 (0xb7e92000)
libstdc++.so.6 => /usr/lib/libstdc++.so.6 (0xb7dce000)
...
```

The VM approach: Disk image + VM description (< 50 LOC)

Installation by copy (assumed a VMM is installed)

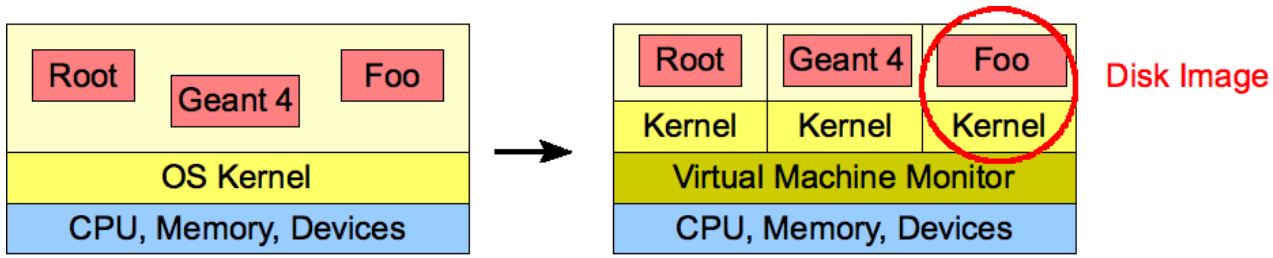
Virtualization Techniques

Principle:

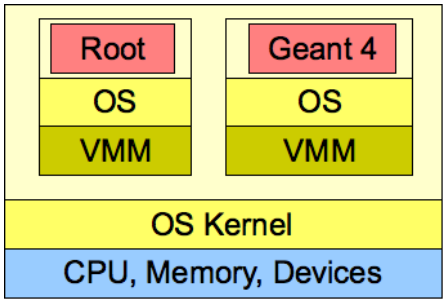


Virtualization Techniques

Principle:



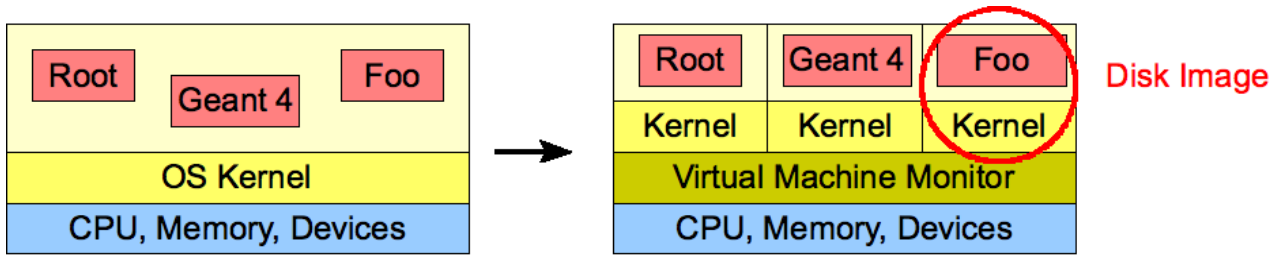
Emulation:



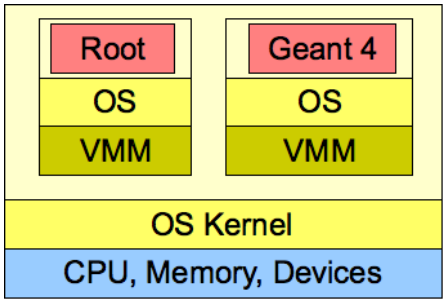
Pure QEmu:	Interpretation
VMware Server (GSX) / WS:	Dynamic Code Rewriting
MS Virtual PC:	Dynamic Recompilation

Virtualization Techniques

Principle:

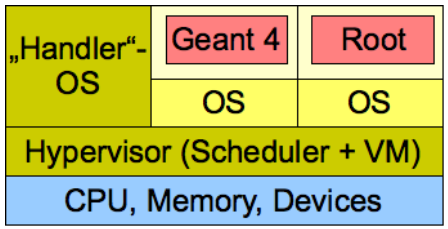


Emulation:



Pure QEmu:	Interpretation
VMware Server (GSX) / WS:	Dynamic Code Rewriting
MS Virtual PC:	Dynamic Recompilation

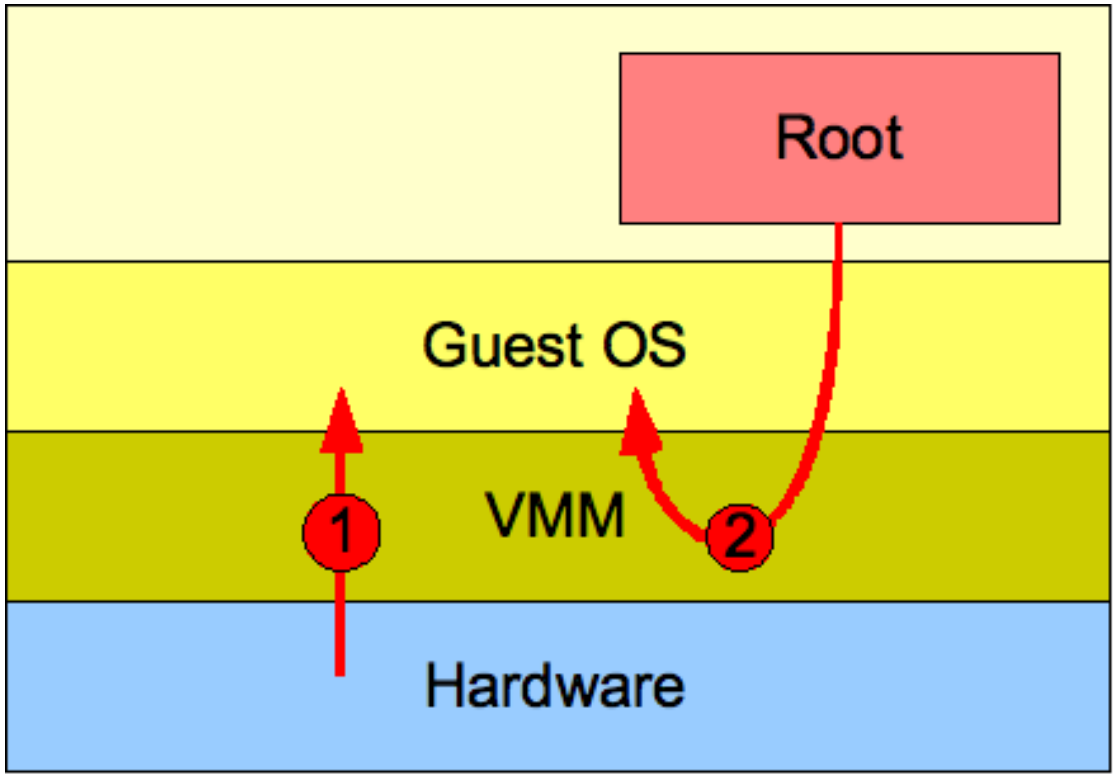
Hypervisor:



Xen, VMware ESX:	Hypervisor + Mgmt OS
KVM:	Hypervisor = Mgmt OS

Critical & Expensive Operations (x86)

- Kernel Trap (general)



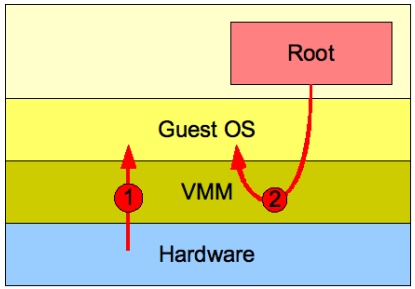
- Page Fault

```
malloc (1024*sizeof(int));
```

  - Virtual memory address $\mapsto$ Physical memory address
  - 3-level address lookup
  - Has to be done by the host OS as well as by the guest OS

Critical & Expensive Operations (x86)

- Kernel Trap (general)



1. Interrupt
2. Exception (System Call)

- Page Fault

```
malloc (1024*sizeof(int));
```

  - Virtual memory address $\mapsto$ Physical memory address
  - 3-level address lookup
  - Has to be done by the host OS as well as by the guest OS

- Sensitive Operations

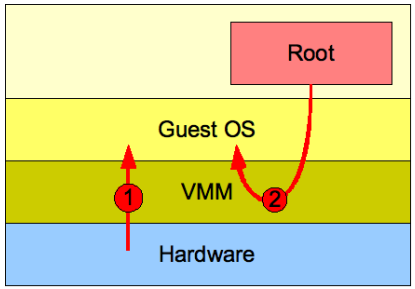
  Do not raise kernel trap, but interfere with underlying host kernel

  Solution: Para- / Pre-virtualization, IA-32 Extensions

See also Robin, Irvine: *Analysis of the Intel Pentium's Ability to Support a Secure Virtual Machine Monitor* Proc. 9th USENIX Security Symposium

Critical & Expensive Operations (x86)

- Kernel Trap (general)



1. Interrupt
2. Exception (System Call)

- Page Fault

```
malloc (1024*sizeof(int));
```

  - Virtual memory address $\mapsto$ Physical memory address
  - 3-level address lookup
  - Has to be done by the host OS as well as by the guest OS

- Sensitive Operations

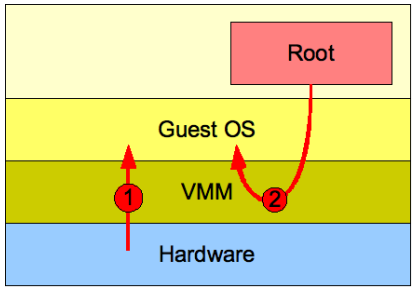
  Do not raise kernel trap, but interfere with underlying host kernel

  Solution: Para- / Pre-virtualization, IA-32 Extensions

See also Robin, Irvine: *Analysis of the Intel Pentium's Ability to Support a Secure Virtual Machine Monitor* Proc. 9th USENIX Security Symposium

Critical & Expensive Operations (x86)

- Kernel Trap (general)



1. Interrupt
2. Exception (System Call)

- Page Fault

```
malloc (1024*sizeof(int));
```

  - Virtual memory address $\mapsto$ Physical memory address
  - 3-level address lookup
  - Has to be done by the host OS as well as by the guest OS

- Sensitive Operations

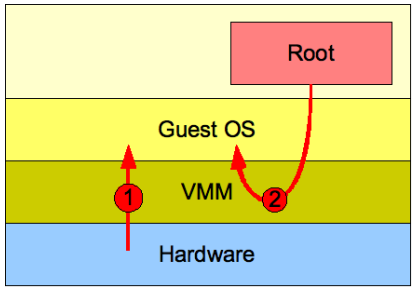
  Do not raise kernel trap, but interfere with underlying host kernel

  Solution: Para- / Pre-virtualization, IA-32 Extensions

See also Robin, Irvine: *Analysis of the Intel Pentium's Ability to Support a Secure Virtual Machine Monitor* Proc. 9th USENIX Security Symposium

Critical & Expensive Operations (x86)

- Kernel Trap (general)



1. Interrupt
2. Exception (System Call)

- Page Fault

```
malloc (1024*sizeof(int));
```

  - Virtual memory address $\mapsto$ Physical memory address
  - 3-level address lookup
  - Has to be done by the host OS as well as by the guest OS

- Sensitive Operations

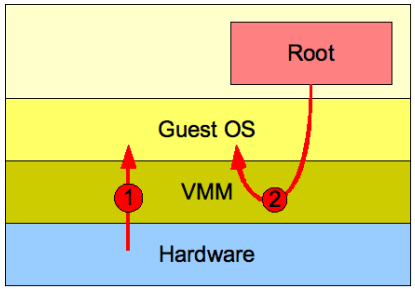
  Do not raise kernel trap, but interfere with underlying host kernel

  Solution: Para- / Pre-virtualization, IA-32 Extensions

See also Robin, Irvine: *Analysis of the Intel Pentium's Ability to Support a Secure Virtual Machine Monitor* Proc. 9th USENIX Security Symposium

Critical & Expensive Operations (x86)

- Kernel Trap (general)



1. Interrupt
2. Exception (System Call)

- Page Fault

```
malloc (1024*sizeof(int));
```

  - Virtual memory address $\mapsto$ Physical memory address
  - 3-level address lookup
  - Has to be done by the host OS as well as by the guest OS

- Sensitive Operations

  Do not raise kernel trap, but interfere with underlying host kernel

  Solution: Para- / Pre-virtualization, IA-32 Extensions

See also Robin, Irvine: *Analysis of the Intel Pentium's Ability to Support a Secure Virtual Machine Monitor* Proc. 9th USENIX Security Symposium.

Benchmark Suite

Application Benchmarks

- Root v5-15-08 stressHepix
- AliRoot v4-05-Rev-05 simulation & reconstruction
- Geant 4.9.0: simple geometry, ATLAS barrel calorimeter, CMS crystal calorimeter

Explaining the Numbers—Synthetic Benchmarks

- nbench (common algorithms)
- pagefaults
- bonnie (harddisk IO, cache)
- ttcp (network IO)

Benchmark Suite

Application Benchmarks

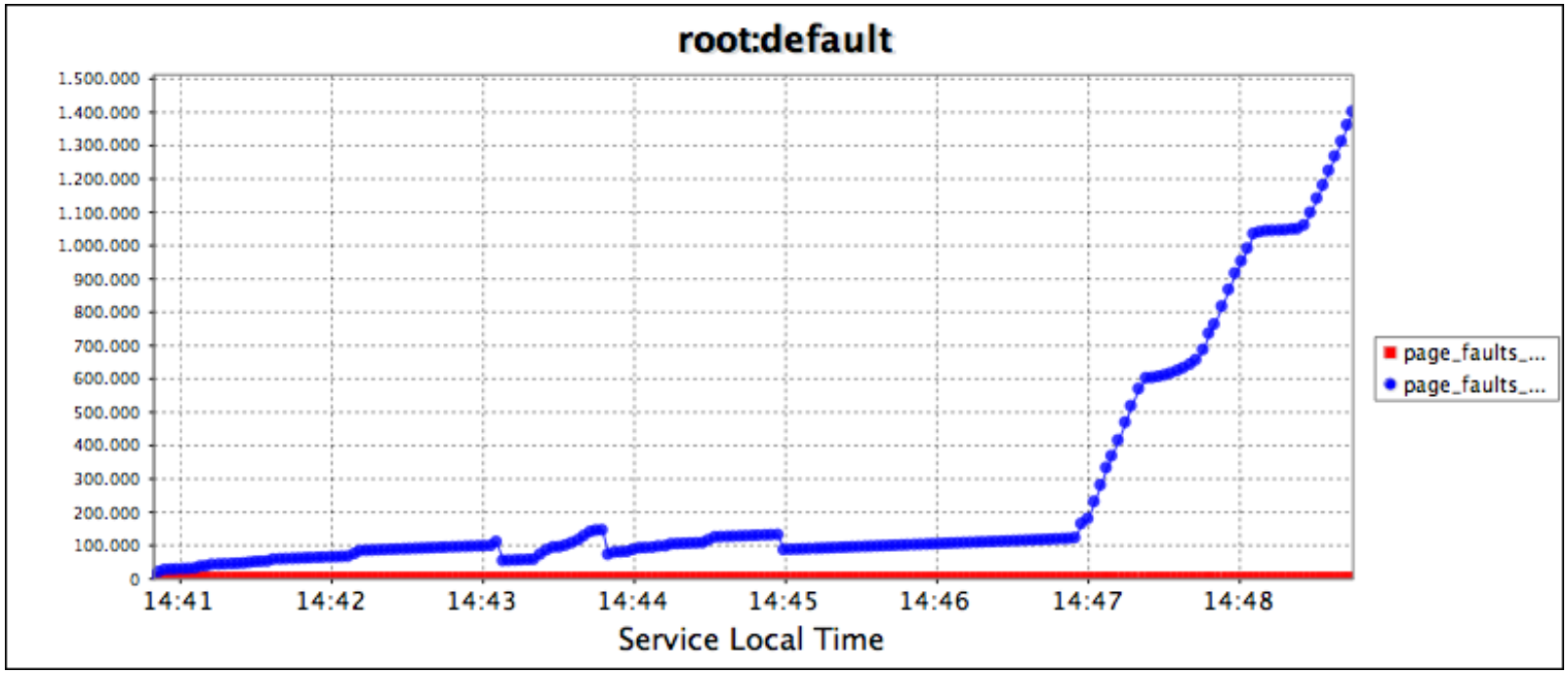
- Root v5-15-08 stressHepix
- AliRoot v4-05-Rev-05 simulation & reconstruction
- Geant 4.9.0: simple geometry, ATLAS barrel calorimeter, CMS crystal calorimeter

Explaining the Numbers—Synthetic Benchmarks

- nbench (common algorithms)
- pagefaults
- bonnie (harddisk IO, cache)
- ttcp (network IO)

Footprints (by MonALISA)

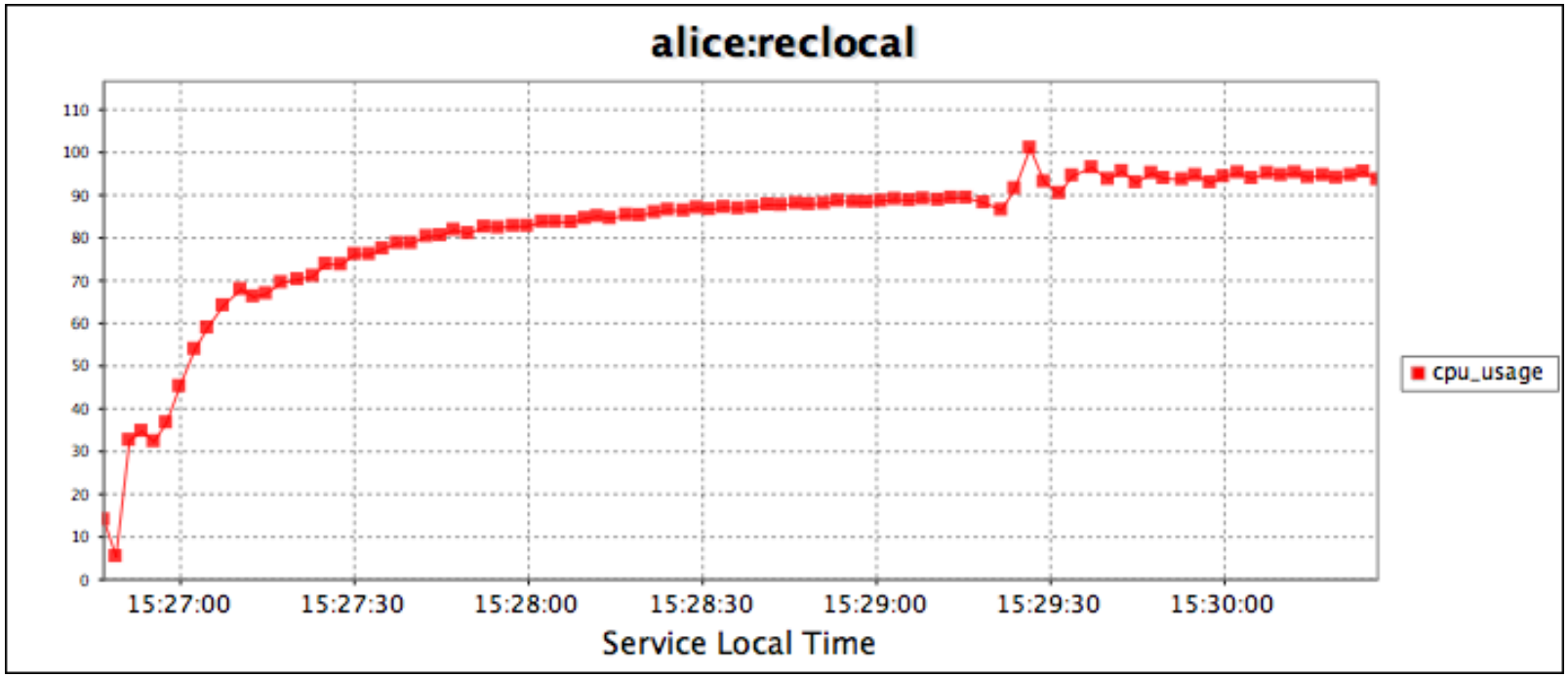
Root stressHepix Pagefaults



$\approx$ 5GB page faults, 100 – 200MB memory consumption

Footprints (by MonALISA)

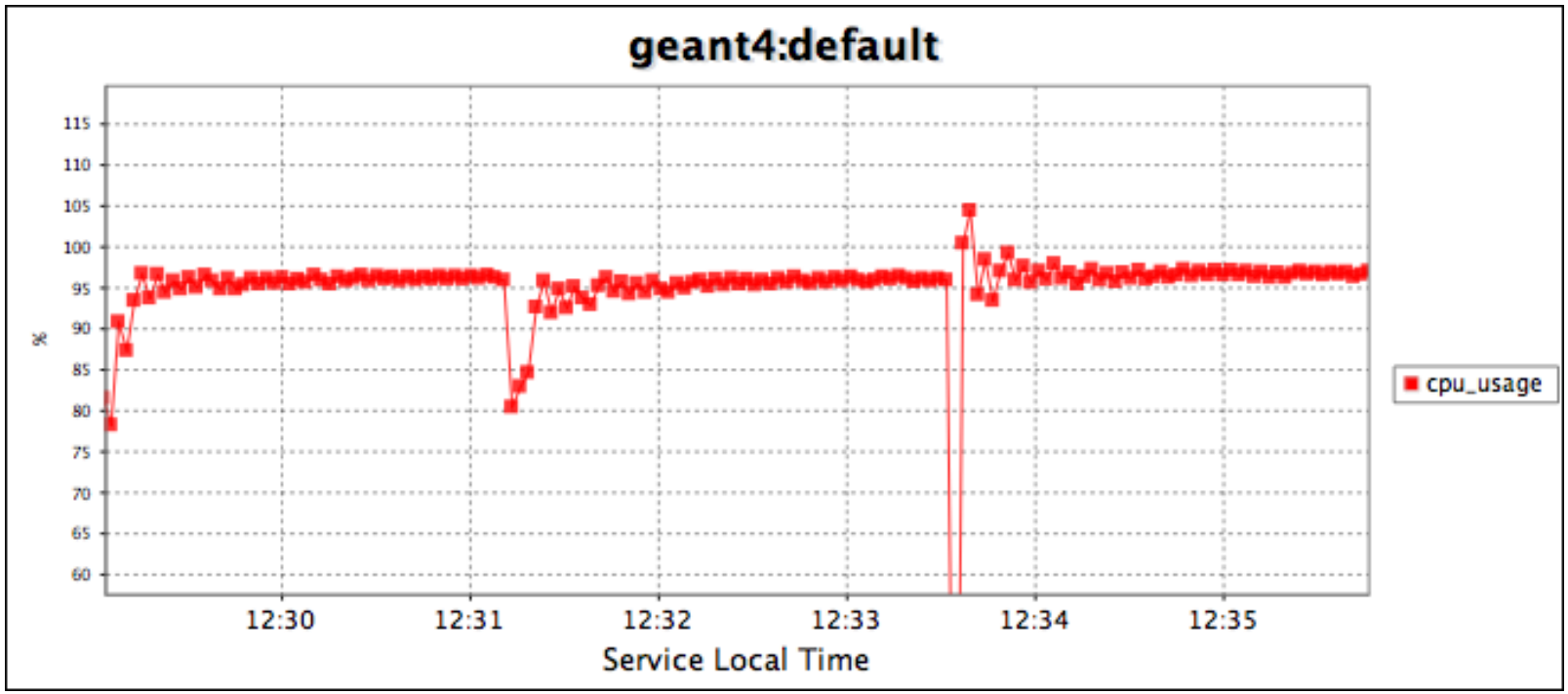
AliRoot simulation and reconstruction CPU usage



$\approx$ 12GB page faults, 500MB – 1GB memory consumption

Footprints (by MonALISA)

Geant4 CPU usage



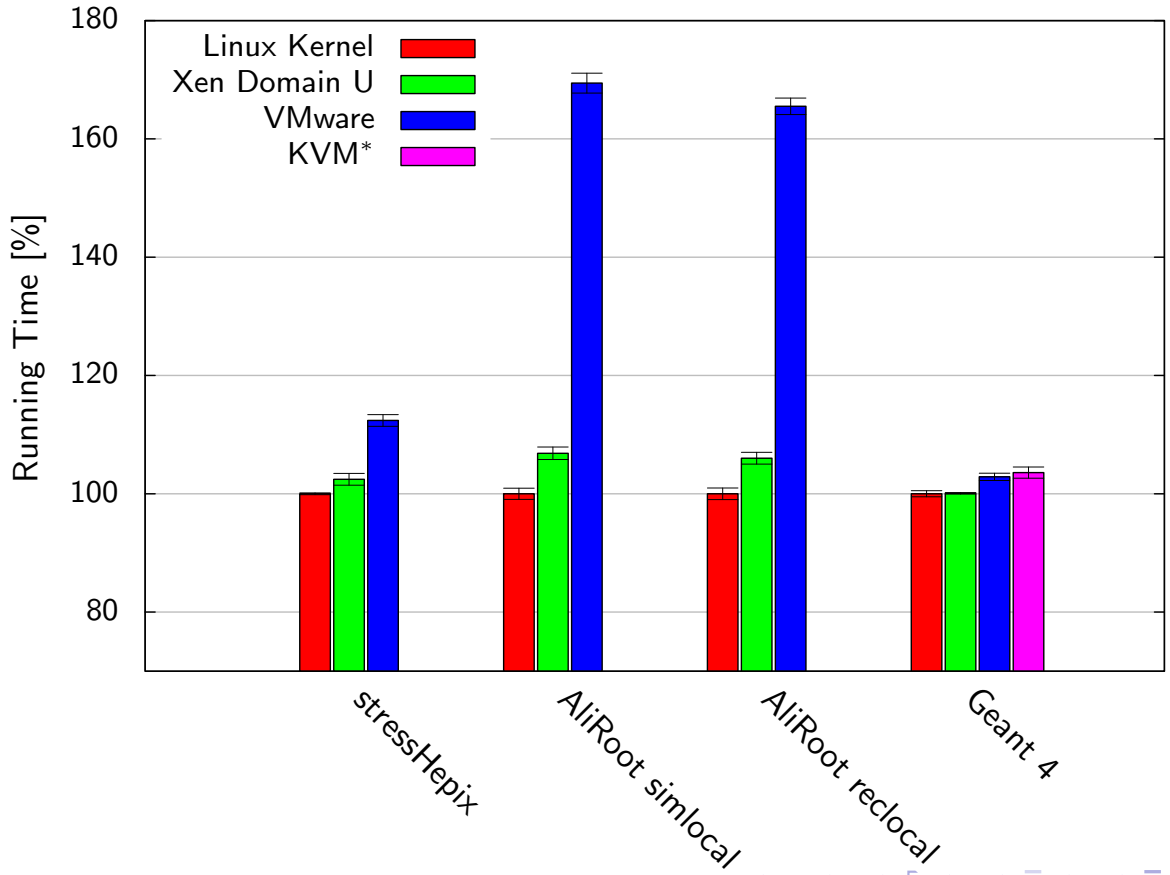
$\approx$ 500MB page faults, $<$ 100MB memory consumption

Results PC-Linux I

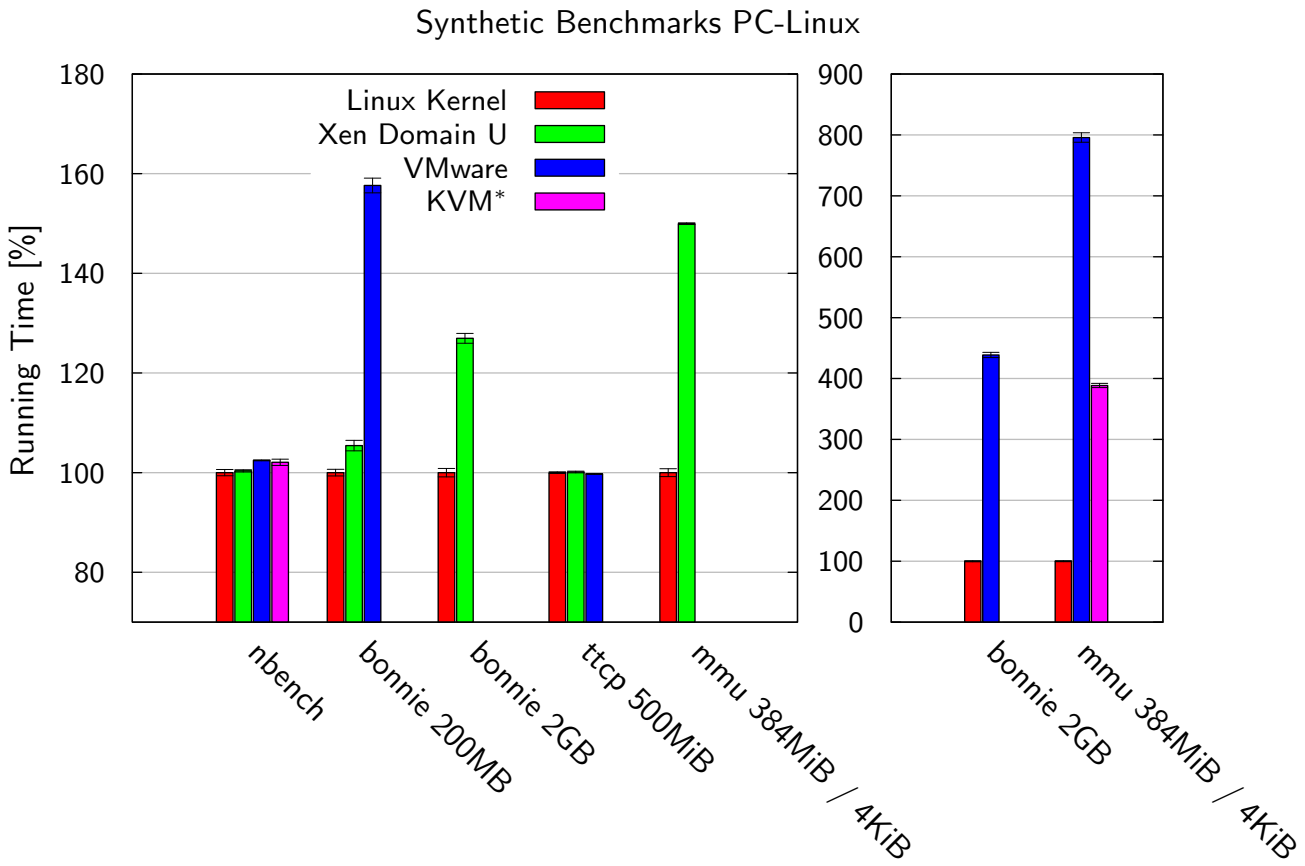
Machine: Pentium D 3.4 GHz, 2GiB RAM, 7200RPM SATA Harddrive, 100MBit NIC

Compiler is gcc in the same version for the host system and for the guest system, except for Darwin (see below)

*) There are some stability problems with KVM

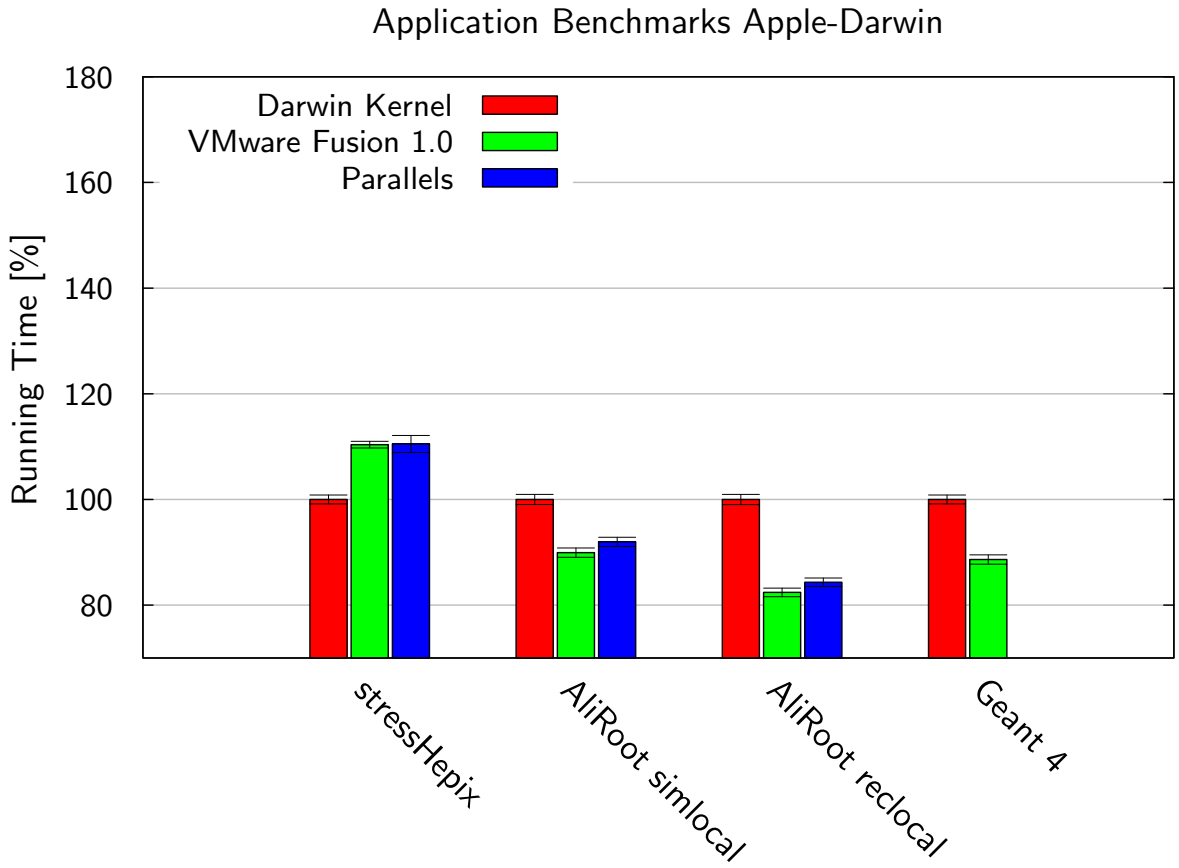


Results PC-Linux II

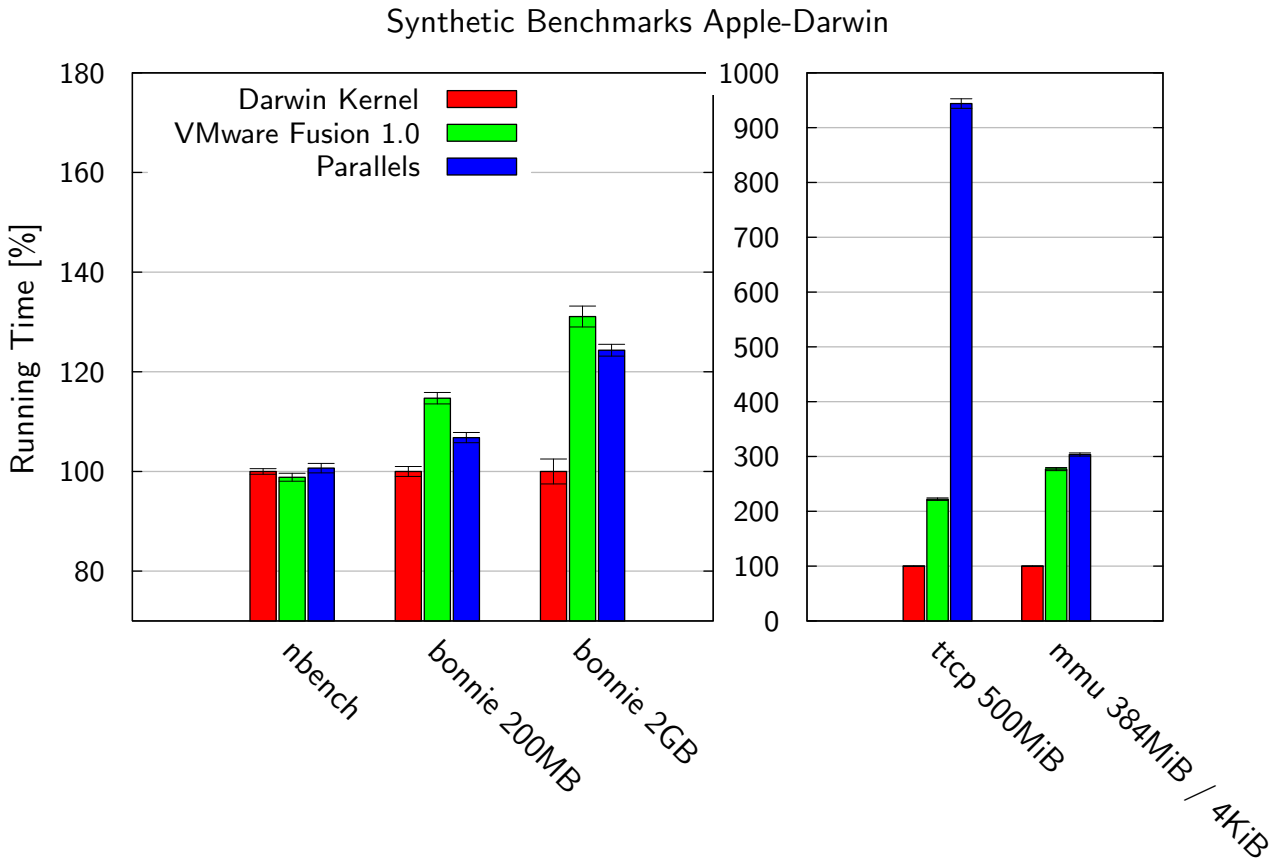


Results Apple-Darwin I

Machine: Intel Xeon 2GHz, 2GiB RAM, 7200RPM SATA Harddrive, GBit NIC



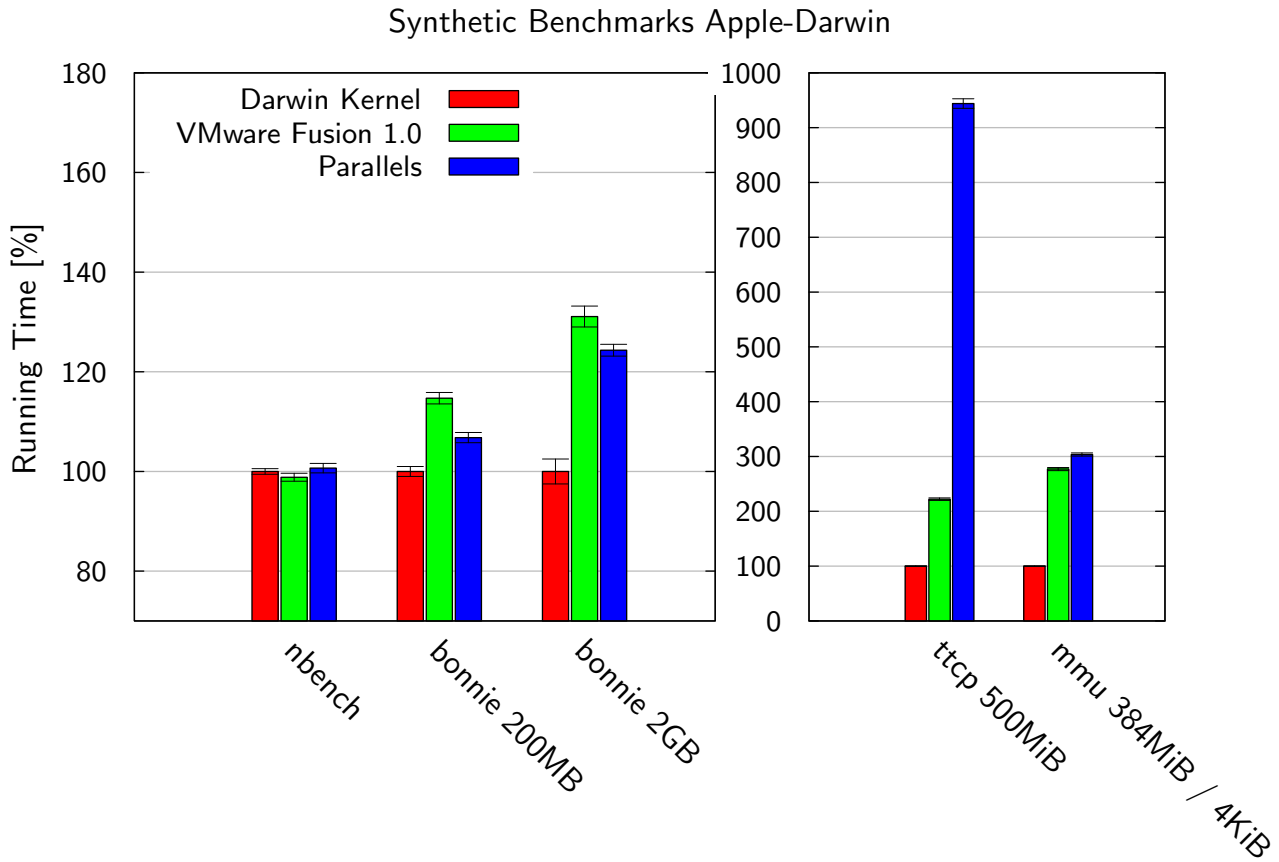
Results Apple-Darwin II



Note: By default there is only a gcc 4.0 for Darwin

Performance loss gcc 4.1.1 $\Rightarrow$ gcc 3.4.4: $\approx$ 20% nbench $\approx$ 10% geant4

Results Apple-Darwin II

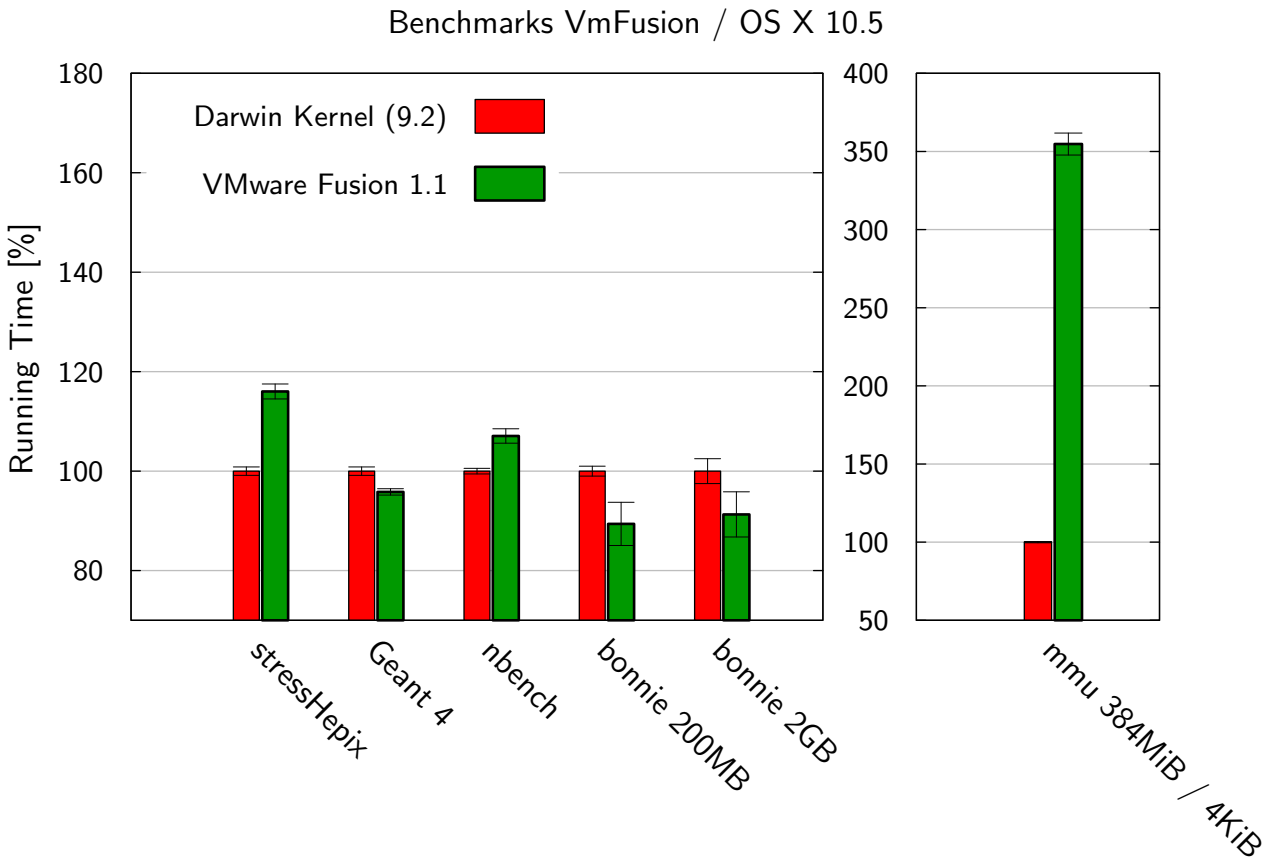


Note: By default there is only a gcc 4.0 for Darwin

Performance loss gcc 4.1.1 $\Rightarrow$ gcc 3.4.4: $\approx$ 20% nbench $\approx$ 10% geant4

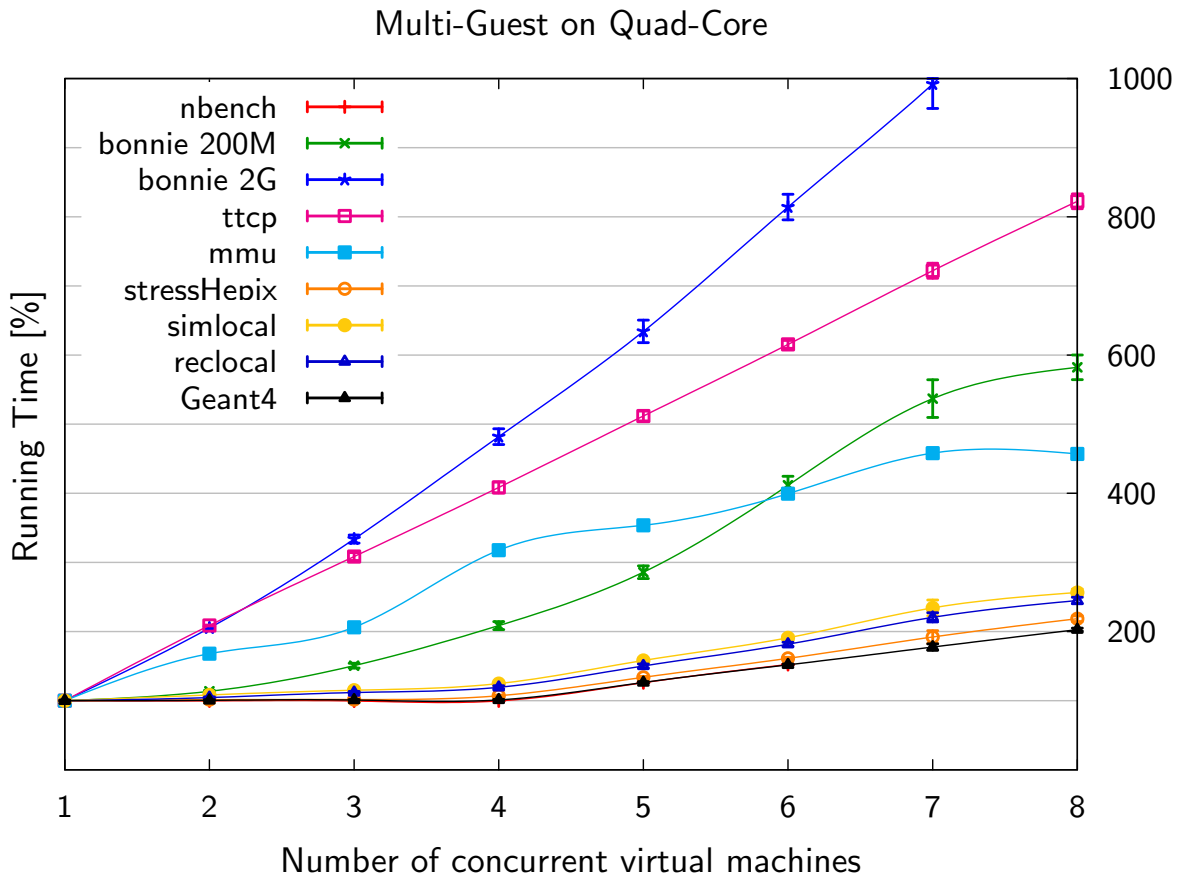
Recent Results Apple-Darwin III

Machine: Mac OS X 10.5, VmFusion 1.1, IA-32 Binaries



Multi-Guest Results

Machine: Core 2 Quad, 8GiB RAM, 4MiB 2nd Level Cache, 7200RPM SATA Harddrive



Summary: Expectations & Difficulties

User's viewpoint: No installation, robust application, encapsulated tasks

Developer's viewpoint: focus on *one* architecture, fine-tuned (modified) OS

VM Property	How to deal with
Sandbox	ssh, web application, mountable harddisk image
Multiple VMMs around	Building tools, e. g. *rBuilder*
Harddisk consumption	Lightweight Linux $\approx$ 100MB (compared to 900MB AliRoot)
Memory consumption	64Bit Host, Leightweight Linux $\approx$ 100MB resident in memory
Timeshift	NTP client

Summary: Expectations & Difficulties

User's viewpoint: No installation, robust application, encapsulated tasks
Developer's viewpoint: focus on *one* architecture, fine-tuned (modified) OS

VM Property	How to deal with
Sandbox	ssh, web application, mountable harddisk image
Multiple VMMs around	Building tools, e. g. *rBuilder*
Harddisk consumption	Lightweight Linux $\approx$ 100MB (compared to 900MB AliRoot)
Memory consumption	64Bit Host, Leightweight Linux $\approx$ 100MB resident in memory
Timeshift	NTP client

Summary: Expectations & Difficulties

User's viewpoint: No installation, robust application, encapsulated tasks
Developer's viewpoint: focus on *one* architecture, fine-tuned (modified) OS

VM Property	**How to deal with**
Sandbox	ssh, web application, mountable harddisk image
Multiple VMMs around	Building tools, e. g. *rBuilder*
Harddisk consumption	Lightweight Linux $\approx$ 100MB (compared to 900MB AliRoot)
Memory consumption	64Bit Host, Leightweight Linux $\approx$ 100MB resident in memory
Timeshift	NTP client

Virtualization Pitfalls

- Xen and VMware destroy each others bridging
- VMware allows to set only MAC addresses of their own [00:50:56:00:00:00] - [00:50:56:FF:FF:FF]
- Linux on Parallels
- Moving disk images between VMMs
- VMware license restricts the publishing of benchmark results
- 1 000 seconds inside virtual machines. . .

Xen DomU	1 000 seconds
VMware Server	1 003 seconds
KVM	1 004 seconds
VMware Fusion	1 000 seconds
Parallels	1 041 seconds

Virtualization Pitfalls

- Xen and VMware destroy each others bridging
- VMware allows to set only MAC addresses of their own [00:50:56:00:00:00] - [00:50:56:FF:FF:FF]
- Linux on Parallels
- Moving disk images between VMMs
- VMware license restricts the publishing of benchmark results
- 1 000 seconds inside virtual machines. . .

Xen DomU	1 000 seconds
VMware Server	1 003 seconds
KVM	1 004 seconds
VMware Fusion	1 000 seconds
Parallels	1 041 seconds

Virtualization Pitfalls

- Xen and VMware destroy each others bridging
- VMware allows to set only MAC addresses of their own [00:50:56:00:00:00] - [00:50:56:FF:FF:FF]
- Linux on Parallels
- Moving disk images between VMMs
- VMware license restricts the publishing of benchmark results
- 1 000 seconds inside virtual machines. . .

Xen DomU	1 000 seconds
VMware Server	1 003 seconds
KVM	1 004 seconds
VMware Fusion	1 000 seconds
Parallels	1 041 seconds

Virtualization Pitfalls

- Xen and VMware destroy each others bridging
- VMware allows to set only MAC addresses of their own [00:50:56:00:00:00] - [00:50:56:FF:FF:FF]
- Linux on Parallels
- Moving disk images between VMMs
- VMware license restricts the publishing of benchmark results
- 1 000 seconds inside virtual machines. . .

Xen DomU	1 000 seconds
VMware Server	1 003 seconds
KVM	1 004 seconds
VMware Fusion	1 000 seconds
Parallels	1 041 seconds

Virtualization Pitfalls

- Xen and VMware destroy each others bridging
- VMware allows to set only MAC addresses of their own [00:50:56:00:00:00] - [00:50:56:FF:FF:FF]
- Linux on Parallels
- Moving disk images between VMMs
- VMware license restricts the publishing of benchmark results
- 1 000 seconds inside virtual machines. . .

Xen DomU	1 000 seconds
VMware Server	1 003 seconds
KVM	1 004 seconds
VMware Fusion	1 000 seconds
Parallels	1 041 seconds

Virtualization Pitfalls

- Xen and VMware destroy each others bridging
- VMware allows to set only MAC addresses of their own [00:50:56:00:00:00] - [00:50:56:FF:FF:FF]
- Linux on Parallels
- Moving disk images between VMMs
- VMware license restricts the publishing of benchmark results
- 1 000 seconds inside virtual machines. . .

Xen DomU	1 000 seconds
VMware Server	1 003 seconds
KVM	1 004 seconds
VMware Fusion	1 000 seconds
Parallels	1 041 seconds

HEP Applications in Virtual Machines

Jakob Blomer

April 15th 2008

(Benchmarks of Summer 2007)