BOINC Virtual Machine Controller Infrastructure

David García Quintas

CERN, Switzerland

October 23, 2009
1 Introduction
  • What?
  • Why?

2 Development (ie, How?)
  • Our Proposed Solution
  • Getting Technical

3 Going For It

4 Conclusions
1 Introduction
   • What?
   • Why?

2 Development (ie, How?)
   • Our Proposed Solution
   • Getting Technical

3 Going For It

4 Conclusions
... are we looking for

A means to *interact* with the system running inside a VM instance.

With *code*, not a human, as the “user” behind this interaction (namely, the BOINC wrapper).
What?

... are we looking for

A means to *interact* with the system running inside a VM instance.

With *code*, not a human, as the “user” behind this interaction (namely, the BOINC wrapper).
1 Introduction
   • What?
   • Why?

2 Development (ie, How?)
   • Our Proposed Solution
   • Getting Technical

3 Going For It

4 Conclusions
For one...
The porting or adapting an application isn’t always affordable. Or even *possible*. 
Why?

But not only:

Pros

- **Ultimate control over the computation environment.**
- Isolation.
- Flexible resource allocation.
Why?

But not only:

Pros

- Ultimate control over the computation environment.
- Isolation.
- Flexible resource allocation.
Why?

But not only:

Pros

- Ultimate control over the computation environment.
- Isolation.
- Flexible resource allocation.
On the other hand:

**Cons**

- Need for a "bigger" machine (esp. RAM + HD).
- Performance penalty (≈ 3% – 7%).
Why?

On the other hand:

Cons

- Need for a "bigger" machine (esp. RAM + HD).
- Performance penalty ($\approx 3\% - 7\%$).
Index

1 Introduction
   • What?
   • Why?

2 Development (ie, How?)
   • Our Proposed Solution
   • Getting Technical

3 Going For It

4 Conclusions
Requirements

Cool stuff, I’m all in! What do we need?
To be able to...

- Start / stop the VM.
- Pause / unpause the VM.
- Snapshots (checkpoints).
- VM creation.
- File transfer to/from the VM.
- Arbitrary command execution on the VM.
Requirements

Cool stuff, I’m all in! What do we need?
To be able to...

- Start / stop the VM.
- Pause / unpause the VM.
- Snapshots (checkpoints).
- VM creation.
- File transfer to/from the VM.
- Arbitrary command execution on the VM.
Cool stuff, I’m all in! What do we need?
To be able to...

- Start / stop the VM.
- Pause / unpause the VM.
- Snapshots (checkpoints).
- VM creation.
- File transfer to/from the VM.
- Arbitrary command execution on the VM.
Cool stuff, I’m all in! What do we need?
To be able to...

- Start / stop the VM.
- Pause / unpause the VM.
- **Snapshots (checkpoints).**
- VM creation.
- File transfer to/from the VM.
- Arbitrary command execution on the VM.
Requirements

Cool stuff, I’m all in! What do we need?
To be able to...

- Start / stop the VM.
- Pause / unpause the VM.
- Snapshots (checkpoints).
- VM creation.
- File transfer to/from the VM.
- Arbitrary command execution on the VM.
Cool stuff, I’m all in! What do we need?
To be able to...

- Start / stop the VM.
- Pause / unpause the VM.
- Snapshots (checkpoints).
- VM creation.
- File transfer to/from the VM.
- Arbitrary command execution on the VM.
Cool stuff, I’m all in! What do we need?
To be able to...

- Start / stop the VM.
- Pause / unpause the VM.
- Snapshots (checkpoints).
- VM creation.
- File transfer to/from the VM.
- Arbitrary command execution on the VM.
Index

1. Introduction
   - What?
   - Why?

2. Development (ie, How?)
   - Our Proposed Solution
   - Getting Technical

3. Going For It

4. Conclusions
Three technologies to help us reach our goals:

- **Python.**
- Twisted Framework.
- **STOMP Protocol** ([http://stomp.codehaus.org/Protocol](http://stomp.codehaus.org/Protocol)).
- **Chirp Protocol** ([http://www.cse.nd.edu/~ccl/software/chirp/](http://www.cse.nd.edu/~ccl/software/chirp)).
Technologies

Three technologies to help us reach our goals:

- Python.
- Twisted Framework.
- STOMP Protocol
  (http://stomp.codehaus.org/Protocol).
- Chirp Protocol
  (http://www.cse.nd.edu/~ccl/software/chirp/).
Three technologies to help us reach our goals:

- Python.
- Twisted Framework.
- **STOMP Protocol**
  (http://stomp.codehaus.org/Protocol).
- **Chirp Protocol**
  (http://www.cse.nd.edu/~ccl/software/chirp/).
Three technologies to help us reach our goals:

- Python.
- Twisted Framework.
- Chirp Protocol (http://www.cse.nd.edu/~ccl/software/chirp/).
What we get from these choices:

- **Multiplatform** (tested on Windows, Linux and OS X).
- Powerful event-driven network framework.
- Simple yet flexible message passing protocol.
Technologies

What we get from these choices:

- Multiplatform (tested on Windows, Linux and OS X).
- Powerful event-driven network framework.
- Simple yet flexible message passing protocol.
What we get from these choices:

- Multiplatform (tested on Windows, Linux and OS X).
- Powerful event-driven network framework.
- Simple yet flexible message passing protocol.
How does all this fit into BOINC?
BOINC (wrapper) exports an API.

The Hypervisor exports an API.
All that’s left to do is to interface both APIs, providing them with what they need from each other.
And this is how this interfacing is realized:

Architecture
What we get out of all this

Main advantages of doing things this way:

- A highly decoupled system (could be “clouded”).
- Easily extensible.
- Multiplatform.
- Scalable (basically, as much as the broker).
- Hypervisor agnostic.
Main advantages of doing things this way:

- A highly decoupled system (could be “clouded”).
- Easily extensible.
- Multiplatform.
- Scalable (basically, as much as the broker).
- Hypervisor agnostic.
What we get out of all this

Main advantages of doing things this way:

- A highly decoupled system (could be “clouded”).
- Easily extensible.
- **Multiplatform**.
- Scalable (basically, as much as the broker).
- Hypervisor agnostic.
What we get out of all this

Main advantages of doing things this way:

- A highly decoupled system (could be “clouded”).
- Easily extensible.
- Multiplatform.
- **Scalable** (basically, as much as the broker).
- Hypervisor agnostic.
What we get out of all this

Main advantages of doing things this way:

- A highly decoupled system (could be “clouded”).
- Easily extensible.
- Multiplatform.
- Scalable (basically, as much as the broker).
- Hypervisor agnostic.
Enough with the marketing!
Index

1. Introduction
   - What?
   - Why?

2. Development (ie, How?)
   - Our Proposed Solution
   - Getting Technical

3. Going For It

4. Conclusions
Host (VM Controller) requests to some of the VMs the execution of a command.
A Simple Example

The addressed VMs process the incoming msg.
They stay busy for a while...
... and eventually they come up with something. At any given moment! All operations are asynchronous.
... and eventually they come up with something. At any given moment! All operations are asynchronous.
The VM controller (thus the BOINC client) eventually receives the result(s) of the computation (together with some extra data).
1 Introduction
   - What?
   - Why?

2 Development (ie, How?)
   - Our Proposed Solution
   - Getting Technical

3 Going For It

4 Conclusions
How much is it going to cost me?

On the VM image:

- Python Runtime (≥ 2.4. 2.6 recommended).
- Twisted Framework.
- Zope Interfaces.
- Stomper.
- Netifaces.
- Chirp.
Going For It

How much is it going to cost me?

On the VM image:

- Python Runtime ($\geq 2.4$. 2.6 recommended).
- Twisted Framework.
- Zope Interfaces.
- Stomper.
- Netifaces.
- Chirp.
How much is it going to cost me?

On the host side (ie, BOINC client):

- Python Runtime (\(\geq 2.4.2.6\) recommended).
- Twisted Framework.
- Stomper.
- Chirp.
- Your hypervisor of choice (we’ve been using VirtualBox).
How much is it going to cost me?

On the host side (ie, BOINC client):

- Python Runtime (≥ 2.4. 2.6 recommended).
- Twisted Framework.
- Stomper.
- Chirp.
- Your hypervisor of choice (we’ve been using VirtualBox).
No vaporware!
Put to the test in the context of the ALICE experiment at CERN.
(successfully so)
Trials at CERN

No vaporware!

Put to the test in the context of the ALICE experiment at CERN. (successfully so)
1 Introduction
   • What?
   • Why?

2 Development (ie, How?)
   • Our Proposed Solution
   • Getting Technical

3 Going For It

4 Conclusions
Wrapping up

Something to take away from this talk:

- Virtualization eases the deployment of certain computations. In some cases, making them possible altogether.
- It is now possible to BOINC-ify applications strongly tied to a certain environment (by virtualizing it).
- You’re not limited by your choice of hypervisor.
Wrapping up

Something to take away from this talk:

- Virtualization eases the deployment of certain computations. In some cases, making them possible altogether.
- It is now possible to BOINC-ify applications strongly tied to a certain environment (by virtualizing it).
- You’re not limited by your choice of hypervisor.
Wrapping up

Something to take away from this talk:

- Virtualization eases the deployment of certain computations. In some cases, making them possible altogether.
- It is now possible to BOINC-ify applications strongly tied to a certain environment (by virtualizing it).
- You’re not limited by your choice of hypervisor.
Wrapping up

It’s even documented!

http://boinc.berkeley.edu/trac/wiki/VmApps

Overall architecture and VirtualBox specific details:
http://boinc.berkeley.edu/trac/wiki/VirtualBox
Wrapping up

It’s even documented!
http://boinc.berkeley.edu/trac/wiki/VmApps

Overall architecture and VirtualBox specific details:
http://boinc.berkeley.edu/trac/wiki/VirtualBox
What’s next?

- Support for more hypervisors.
- Automate the installation process, on both sides.
- Testing, testing, testing!
What’s next?

- Support for more hypervisors.
- Automate the installation process, on both sides.
- Testing, testing, testing!
What’s next?

- Support for more hypervisors.
- Automate the installation process, on both sides.
- Testing, testing, testing!
Thank you!