Dynamic scheduling of virtual machines, scalability and fault tolerance are still the issues!

Adrien Lèbre, Flavien Quesnel
ASCOLA Research Group
Ecole des Mines de Nantes

How Virtualization Changed The Grid Perspective

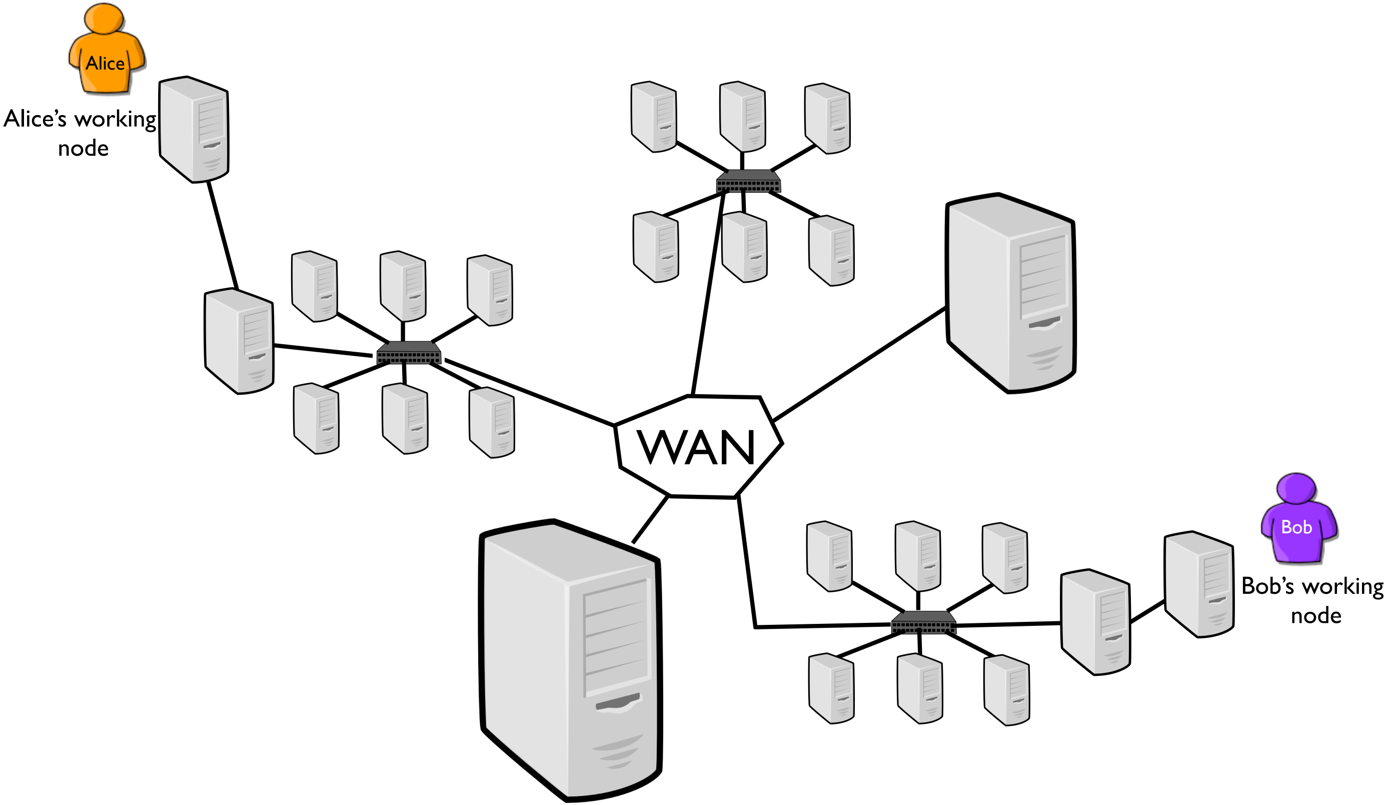
xxx Computing

- xxx as Distributed
  (Cluster / Grid / Desktop / "Hive" / Cloud / Sky / ...)

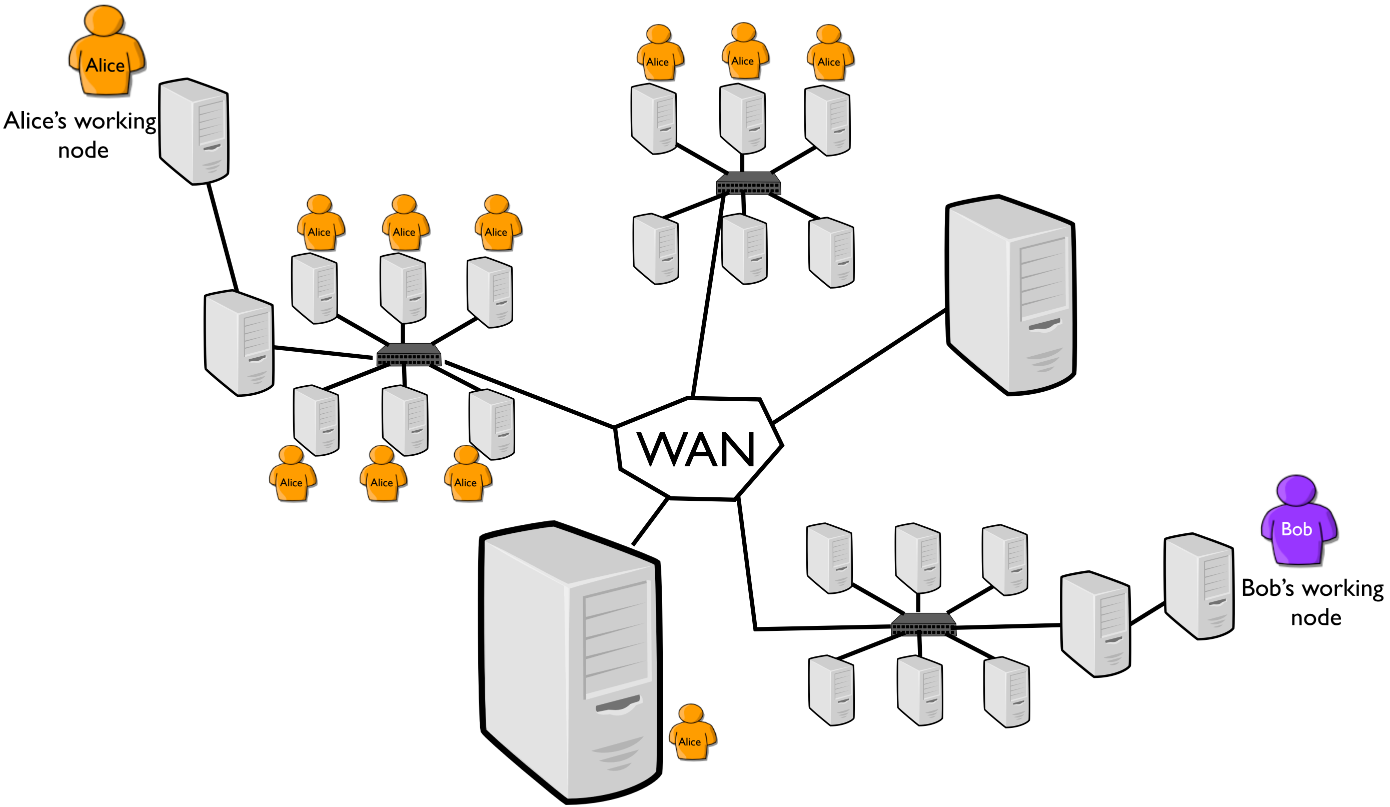
- A common objective

  provide computing resources (both hardware and software)
  in a flexible, transparent, secure, ... way

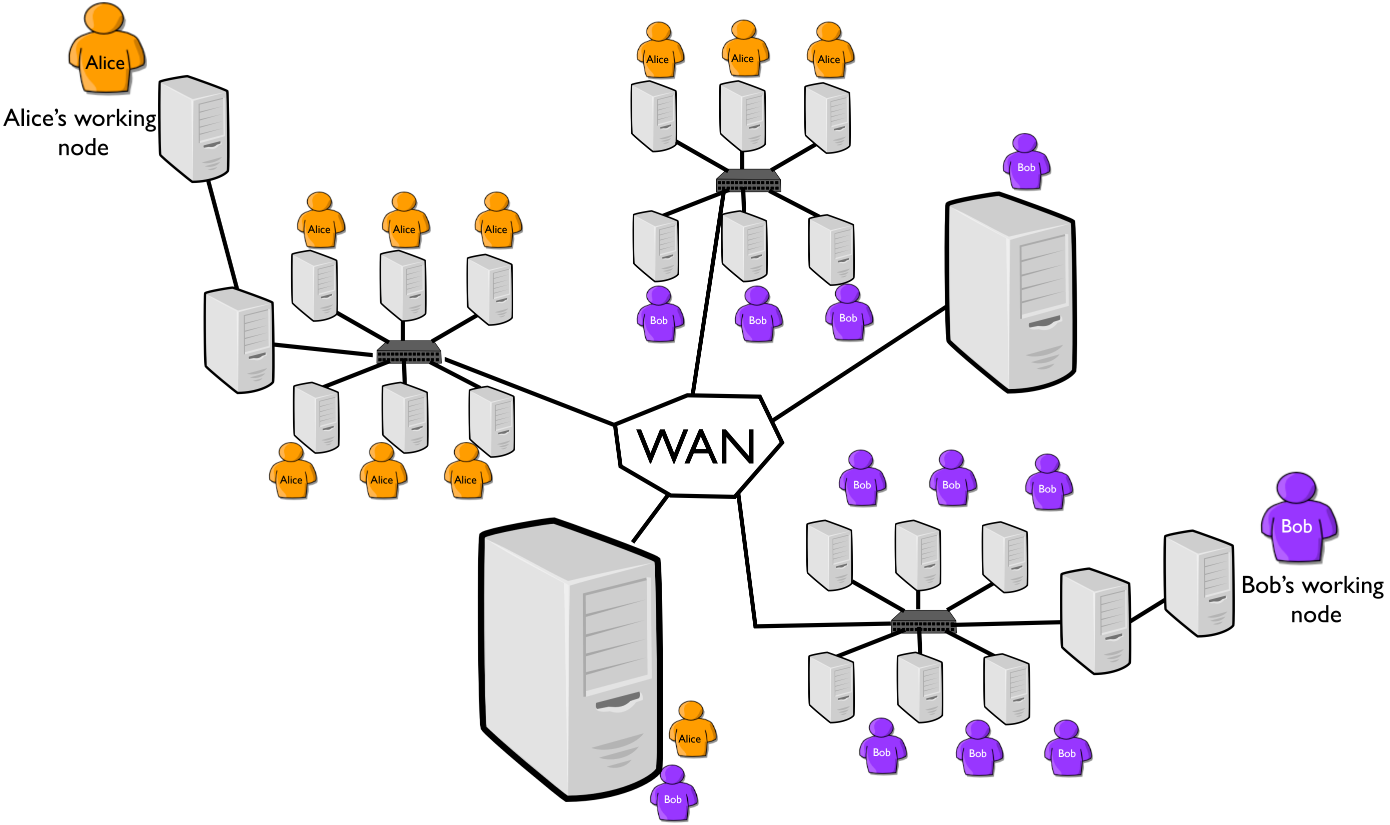
The Alice/Bob Example



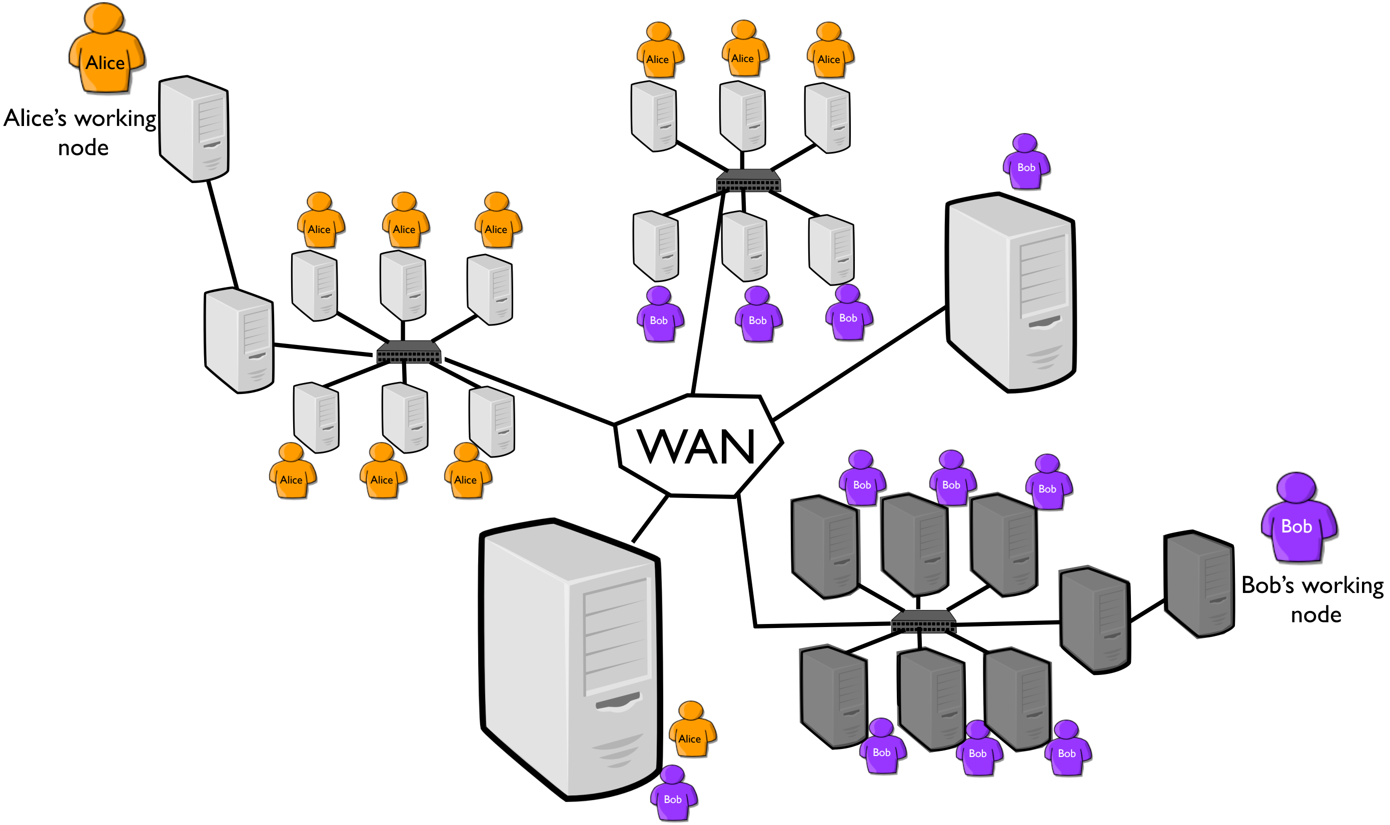
The Alice/Bob Example



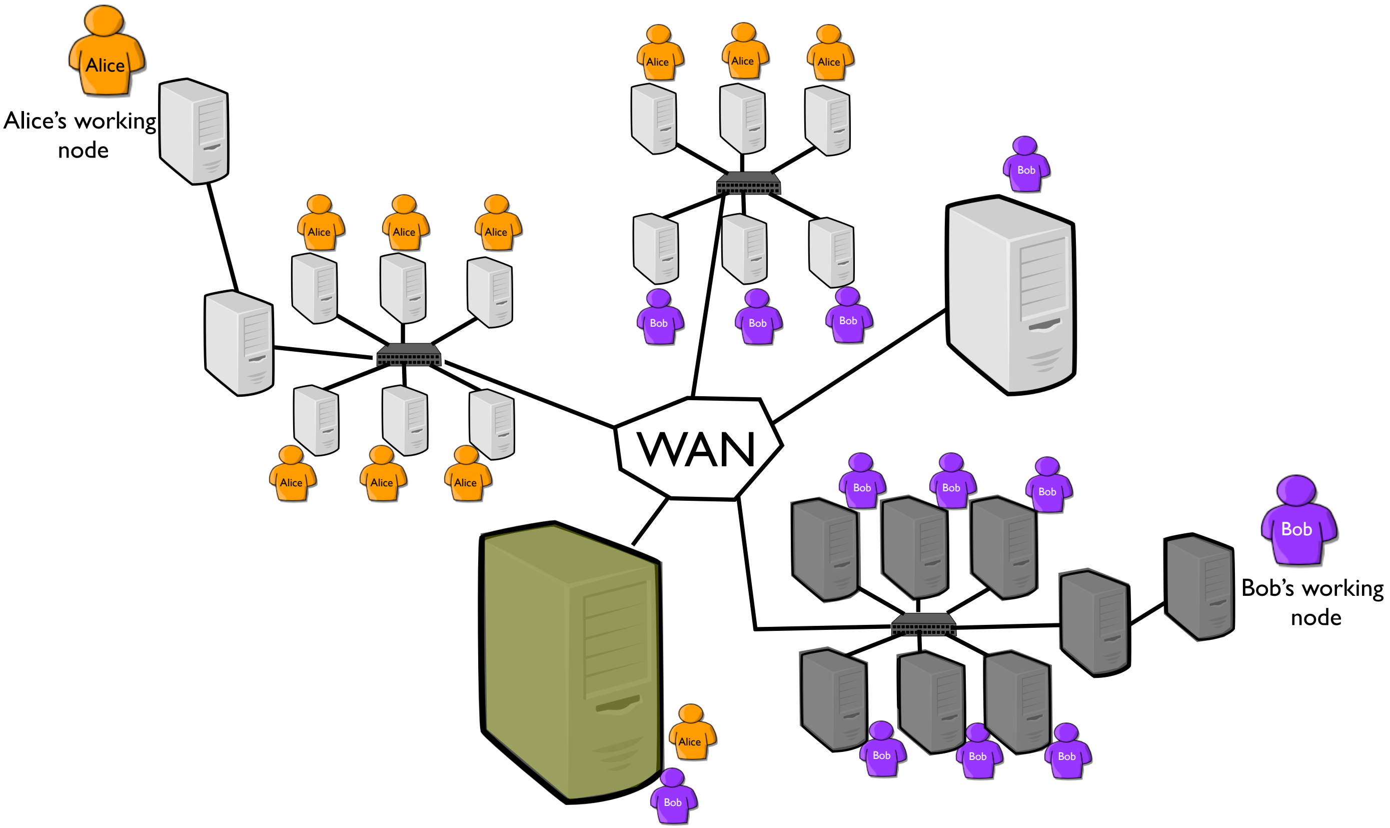
The Alice/Bob Example



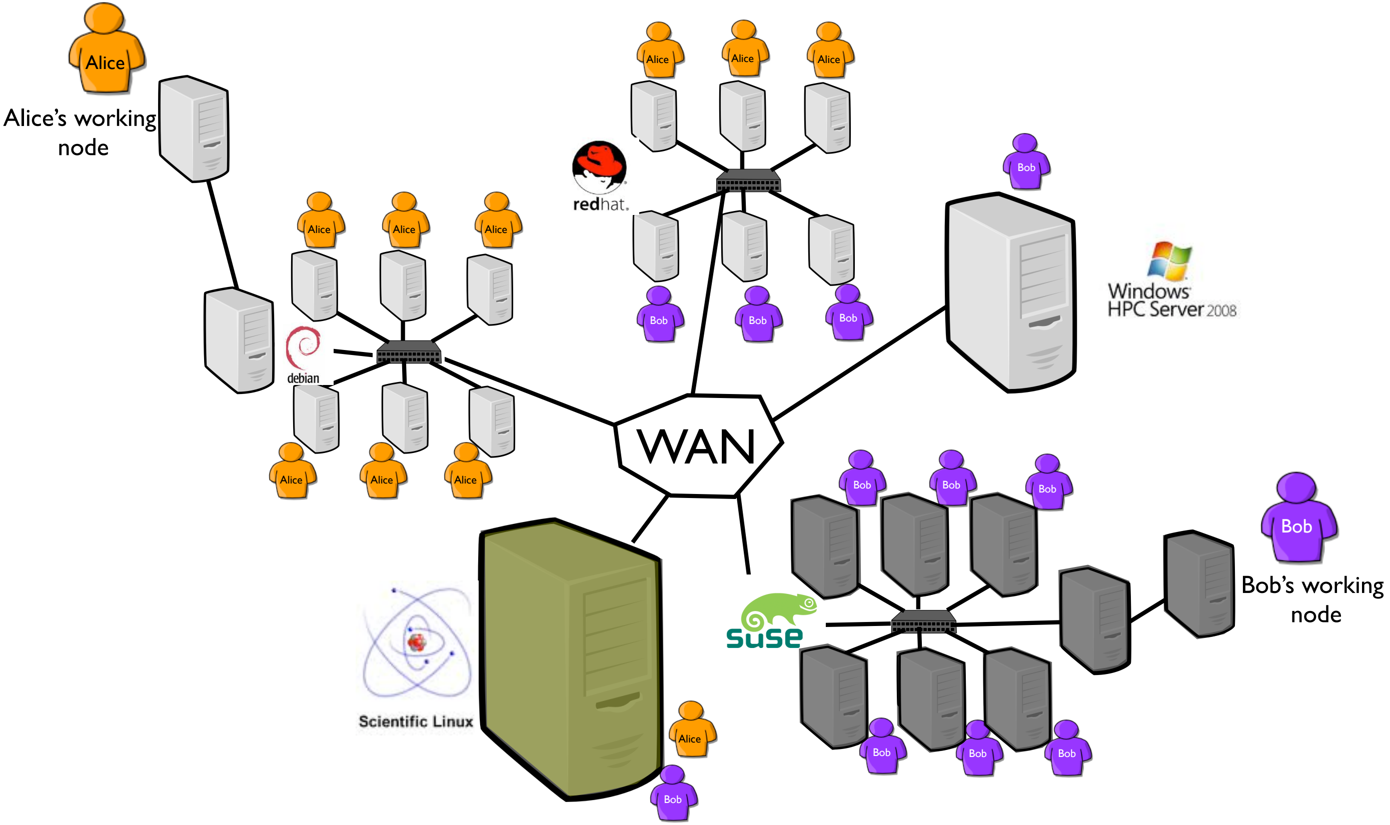
The Alice/Bob Example



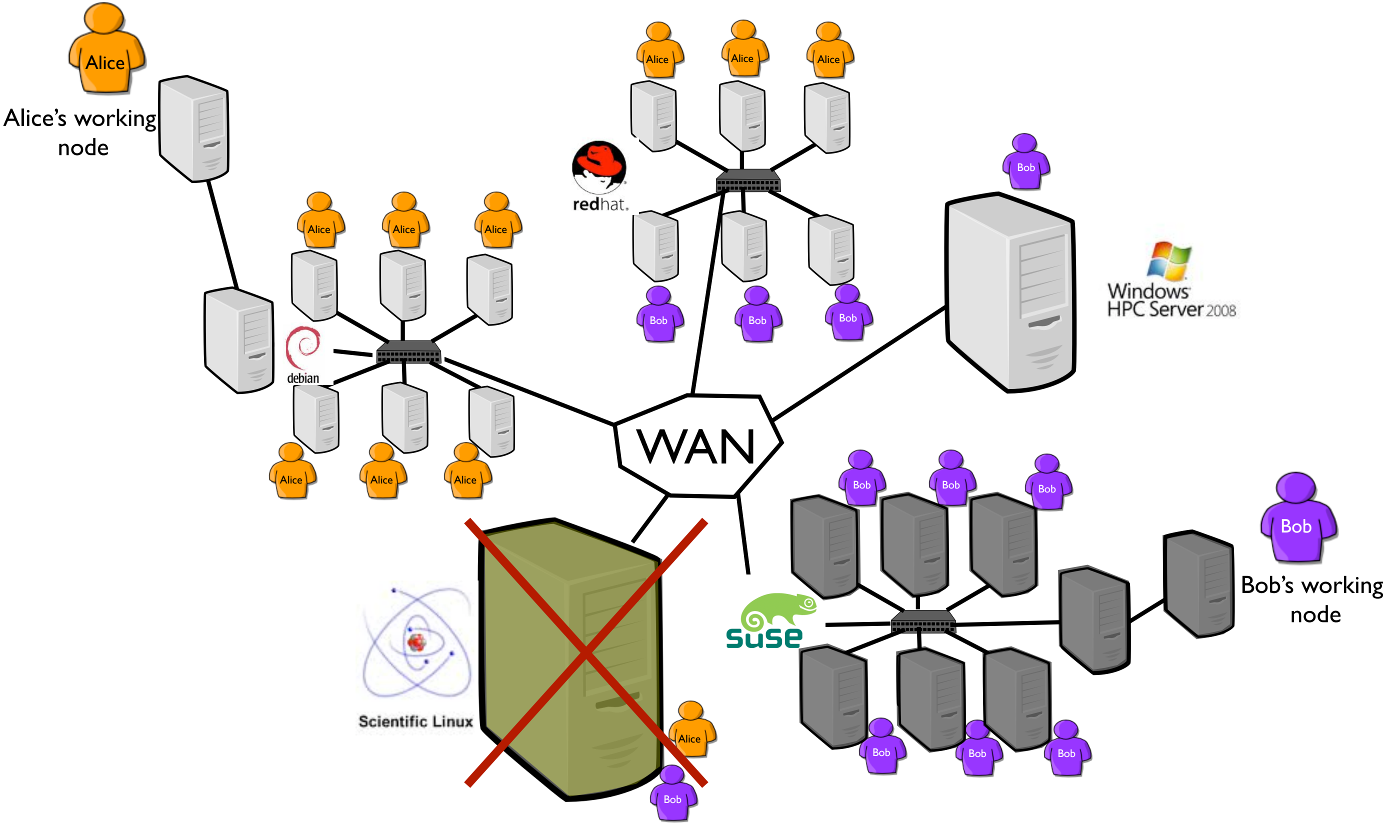
The Alice/Bob Example



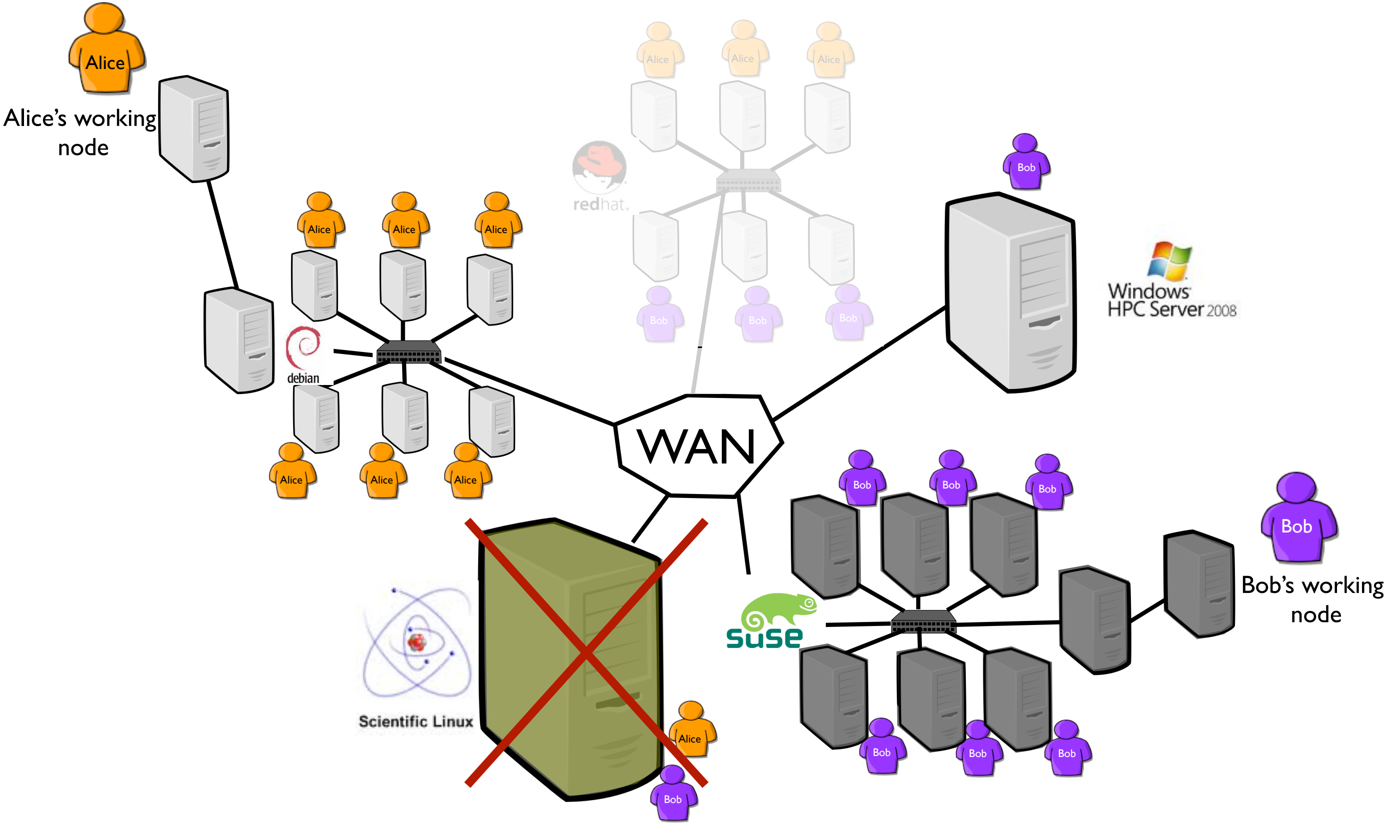
The Alice/Bob Example



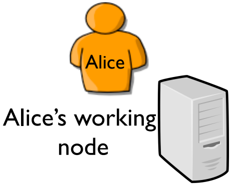
The Alice/Bob Example



The Alice/Bob Example

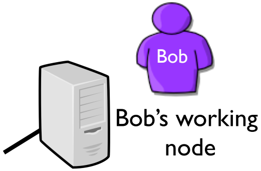


What a Grid!?!



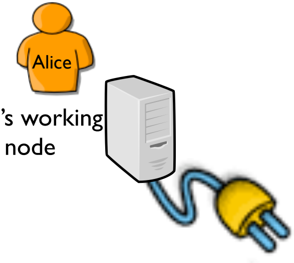
Alice's working node

Resource booking (based on user's estimates)
Security concerns (job isolation)
Heterogeneity concerns (hardware and software)
Scheduling limitations (a job cannot be easily relocated)
Fault tolerance issues
…



Bob's working node

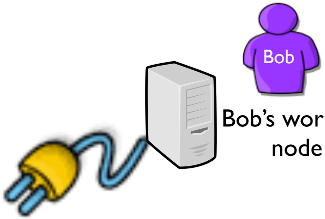
What a Grid!?!



Alice's working node

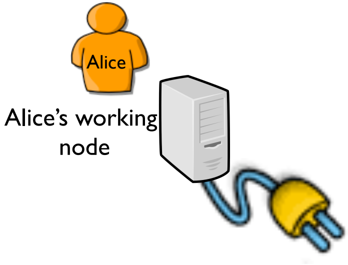
Resource booking (based on user's estimates)
Security concerns (job isolation)
Heterogeneity concerns (hardware and software)
Scheduling limitations (a job cannot be easily relocated)
Fault tolerance issues

...



Bob's working node

What a Grid!?!



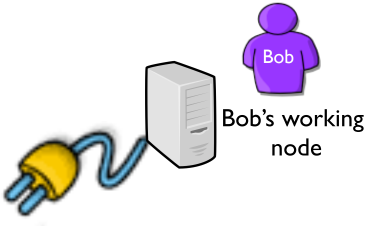
Resource

A lot of progress has been done since the 90's and several proposals partially addressed these concerns.

ated)

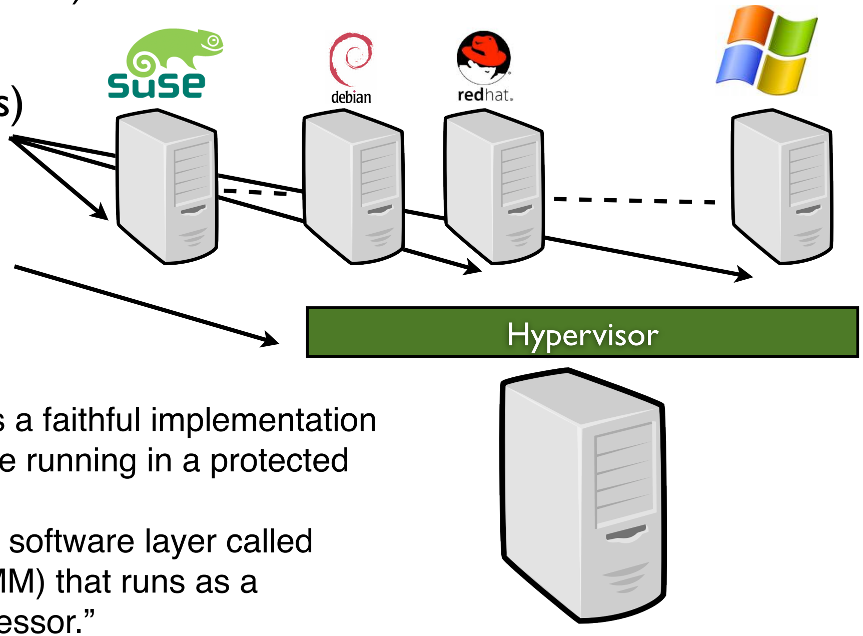
However none of them is mature enough and Strong limitations still persist !

…



Here Comes *System Virtualization*

- One to multiple OSes on a physical node thanks to a hypervisor (an operating system of OSes)



Virtual Machines (VMs)

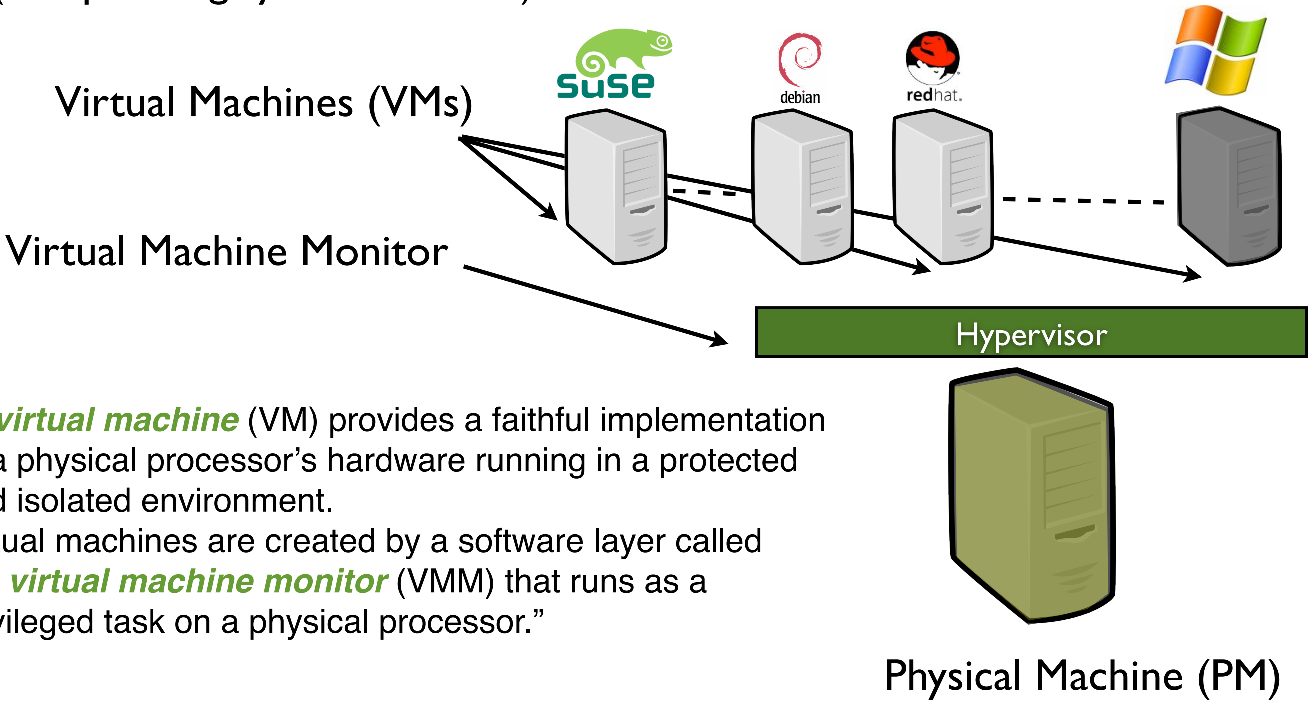
Virtual Machine Monitor

Physical Machine (PM)

“A ***virtual machine*** (VM) provides a faithful implementation of a physical processor’s hardware running in a protected and isolated environment.
Virtual machines are created by a software layer called the ***virtual machine monitor*** (VMM) that runs as a privileged task on a physical processor.”

Here Comes *System Virtualization*

- One to multiple OSes on a physical node thanks to a hypervisor (an operating system of OSes)



“A ***virtual machine*** (VM) provides a faithful implementation of a physical processor’s hardware running in a protected and isolated environment.
Virtual machines are created by a software layer called the ***virtual machine monitor*** (VMM) that runs as a privileged task on a physical processor.”

Virtualization History

- Proposed in the 60's by IBM

  More than 70 publications between 66 and 73

  *"Virtual Machines have finally arrived. Dismissed for a number of years as merely academic curiosities, **they are now seen as cost-effective techniques for organizing computer systems resources to provide extraordinary system flexibility** and support for certain unique applications" .*

  Goldberg, Survey of Virtual Machine Research, 1974

Virtualization History

- The 80's

  No real improvements
  Virtualization seems given up

- End of the 90's:

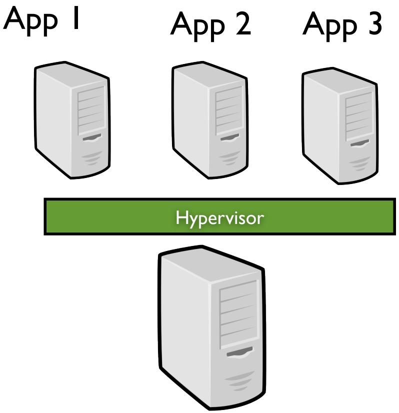
  HLL-VM : High-Level Language VM
  Java and its famous JVM!

  Virtual Server: Exploit for Web hosting
  (Linux `chroot` / containers)

  Revival of System Virtualization approach (VmWare/Xen)

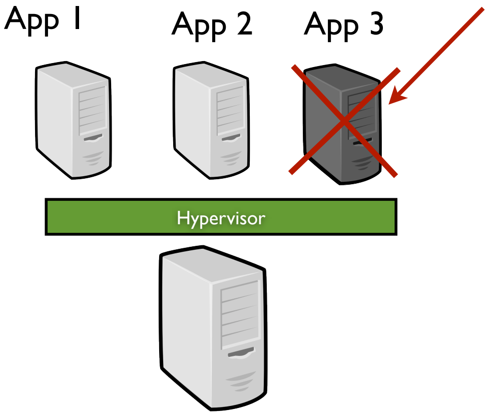
  Hard or soft partitioning of SMP/Numa Server

VM Capabilities



- Isolation (“security” between each VM)

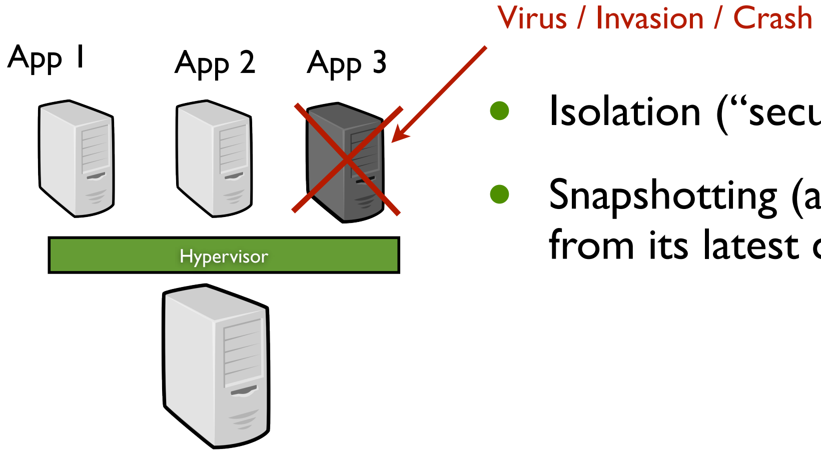
VM Capabilities



Virus / Invasion / Crash

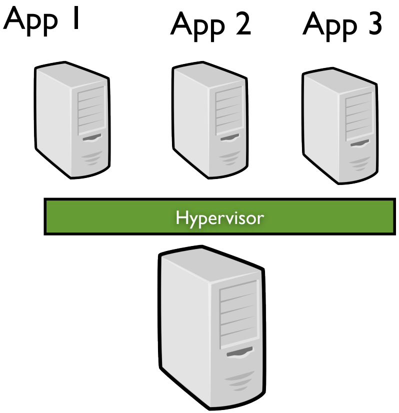
- Isolation ("security" between each VM)

VM Capabilities



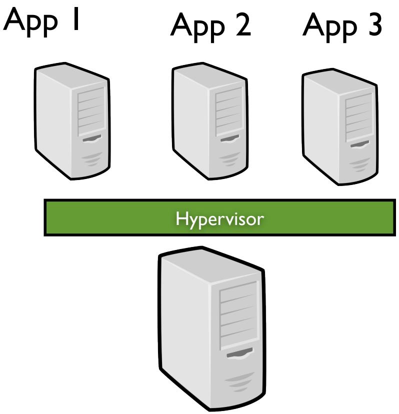
- Isolation ("security" between each VM)
- Snapshotting (a VM can be easily resumed from its latest consistent state)

VM Capabilities



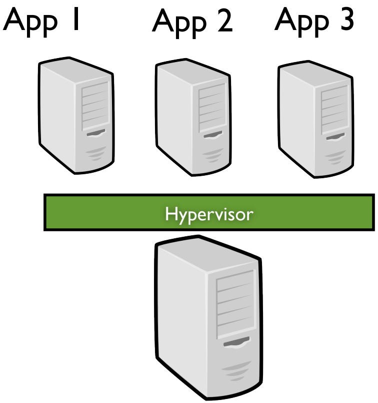
- Isolation (“security” between each VM)
- Snapshotting (a VM can be easily resumed from its latest consistent state)

VM Capabilities

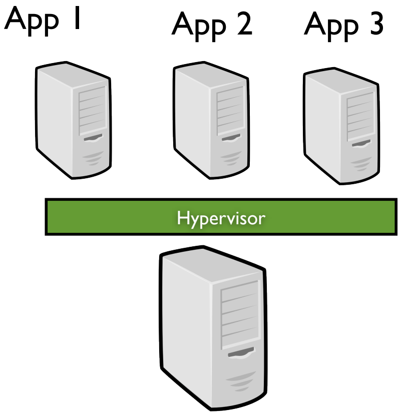


- Isolation (“security” between each VM)
- Snapshotting (a VM can be easily resumed from its latest consistent state)

- Suspend/Resume

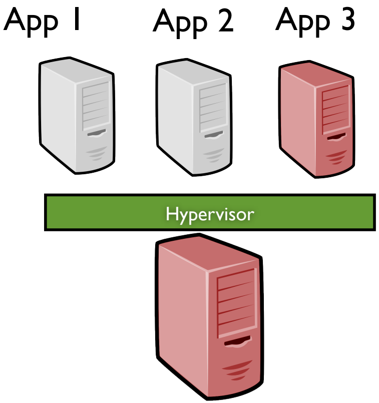


VM Capabilities

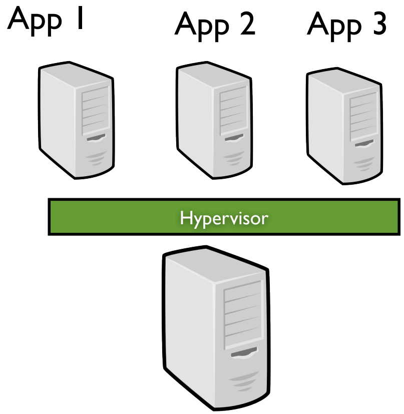


- Isolation ("security" between each VM)
- Snapshotting (a VM can be easily resumed from its latest consistent state)

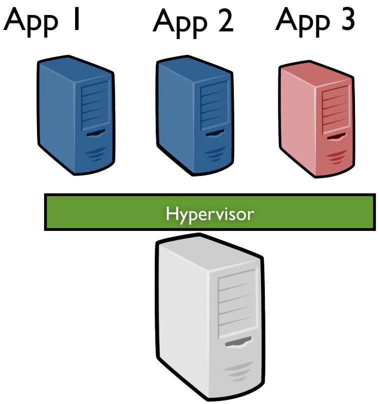
- Suspend/Resume



VM Capabilities

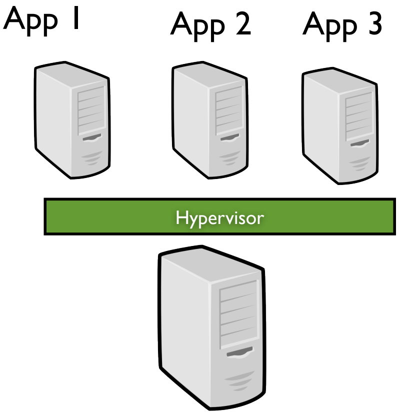


- Isolation ("security" between each VM)
- Snapshotting (a VM can be easily resumed from its latest consistent state)

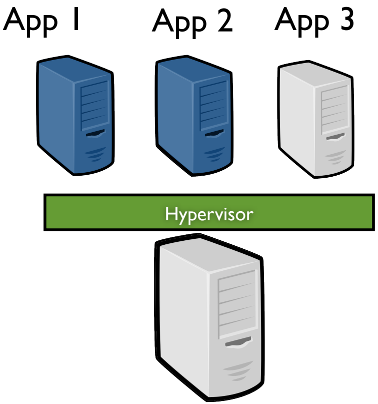


- Suspend/Resume

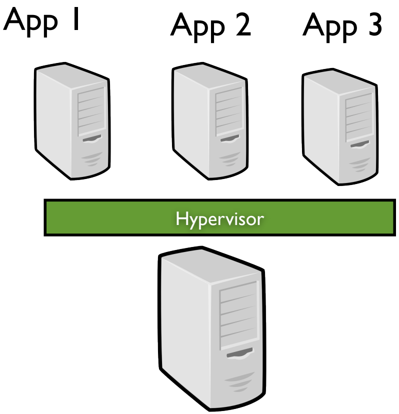
VM Capabilities



- Isolation (“security” between each VM)
- Snapshotting (a VM can be easily resumed from its latest consistent state)
- Suspend/Resume

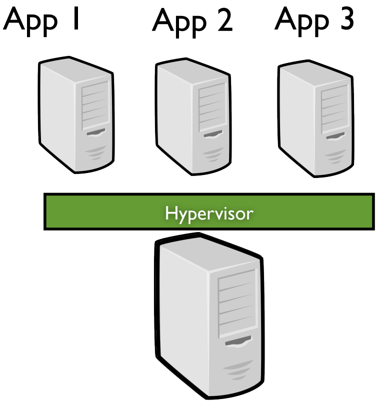


VM Capabilities

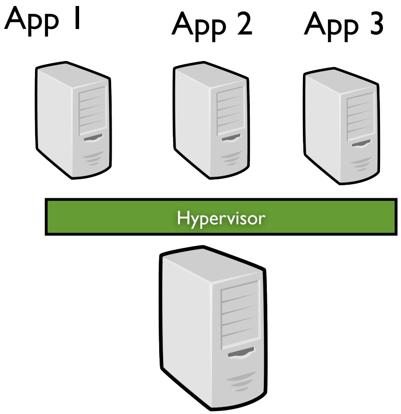


- Isolation ("security" between each VM)
- Snapshotting (a VM can be easily resumed from its latest consistent state)

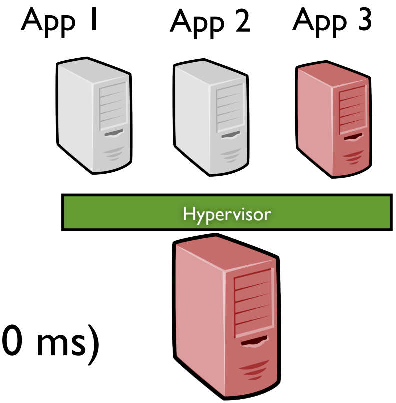
- Suspend/Resume



VM Capabilities

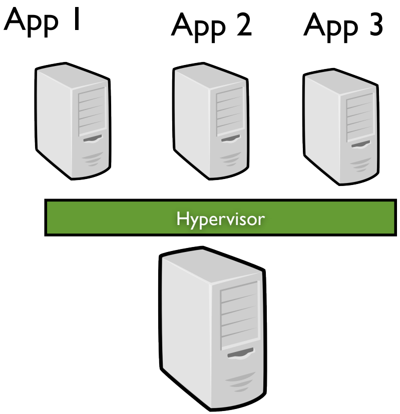


- Isolation (“security” between each VM)
- Snapshotting (a VM can be easily resumed from its latest consistent state)



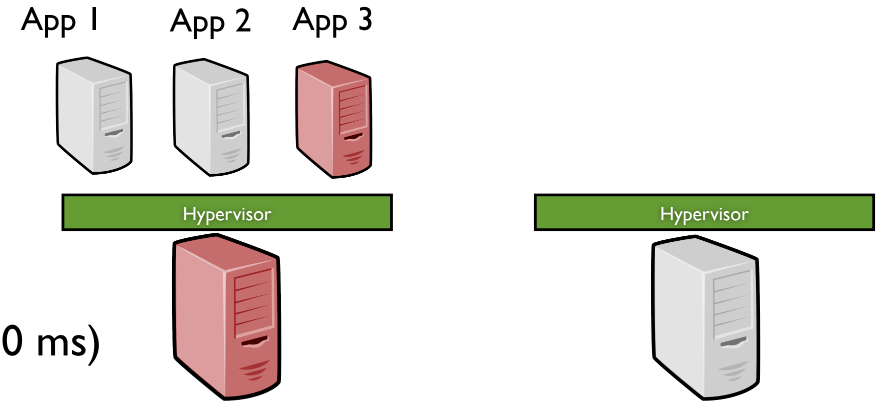
- Suspend/Resume
- Live migration
(negligible downtime ~ 60 ms)

VM Capabilities

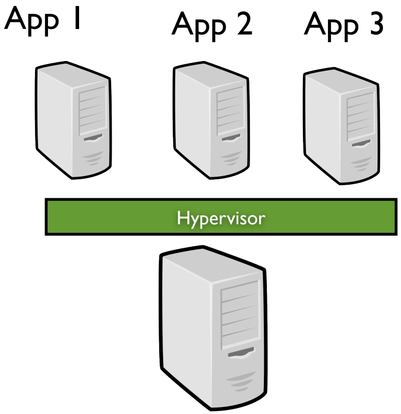


- Isolation (“security” between each VM)
- Snapshotting (a VM can be easily resumed from its latest consistent state)

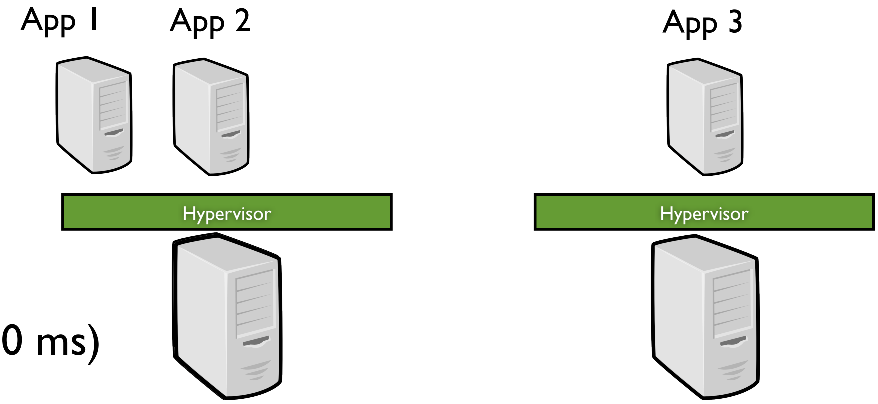
- Suspend/Resume
- Live migration (negligible downtime ~ 60 ms)



VM Capabilities

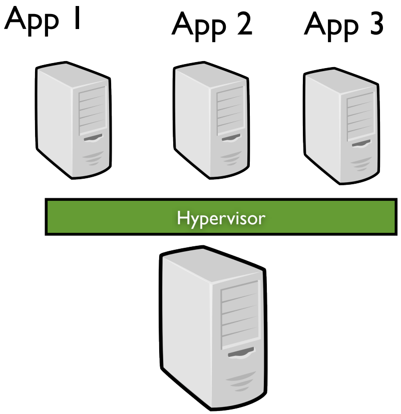


- Isolation (“security” between each VM)
- Snapshotting (a VM can be easily resumed from its latest consistent state)

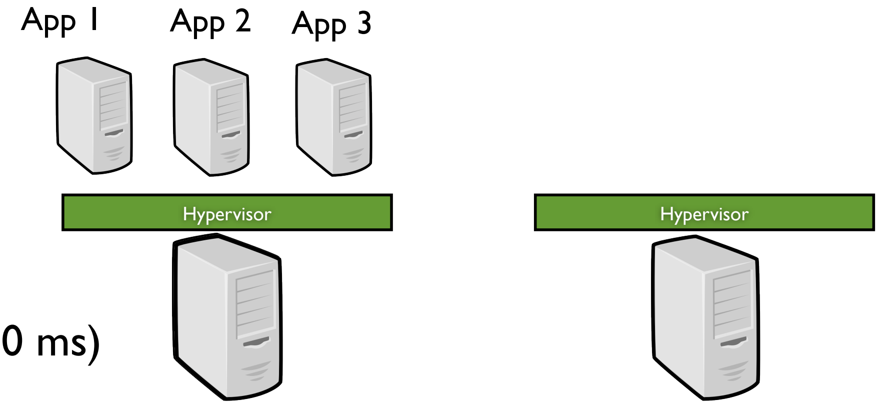


- Suspend/Resume
- Live migration
  (negligible downtime ~ 60 ms)

VM Capabilities

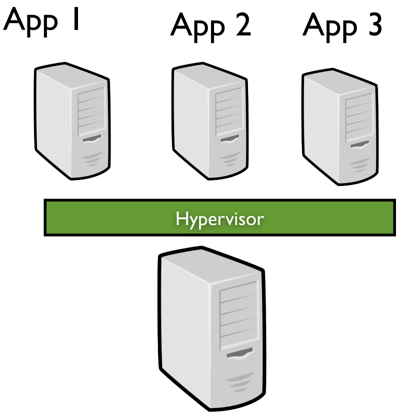


- Isolation (“security” between each VM)
- Snapshotting (a VM can be easily resumed from its latest consistent state)

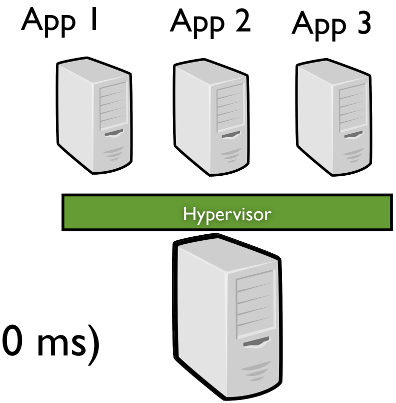


- Suspend/Resume
- Live migration
(negligible downtime ~ 60 ms)

VM Capabilities

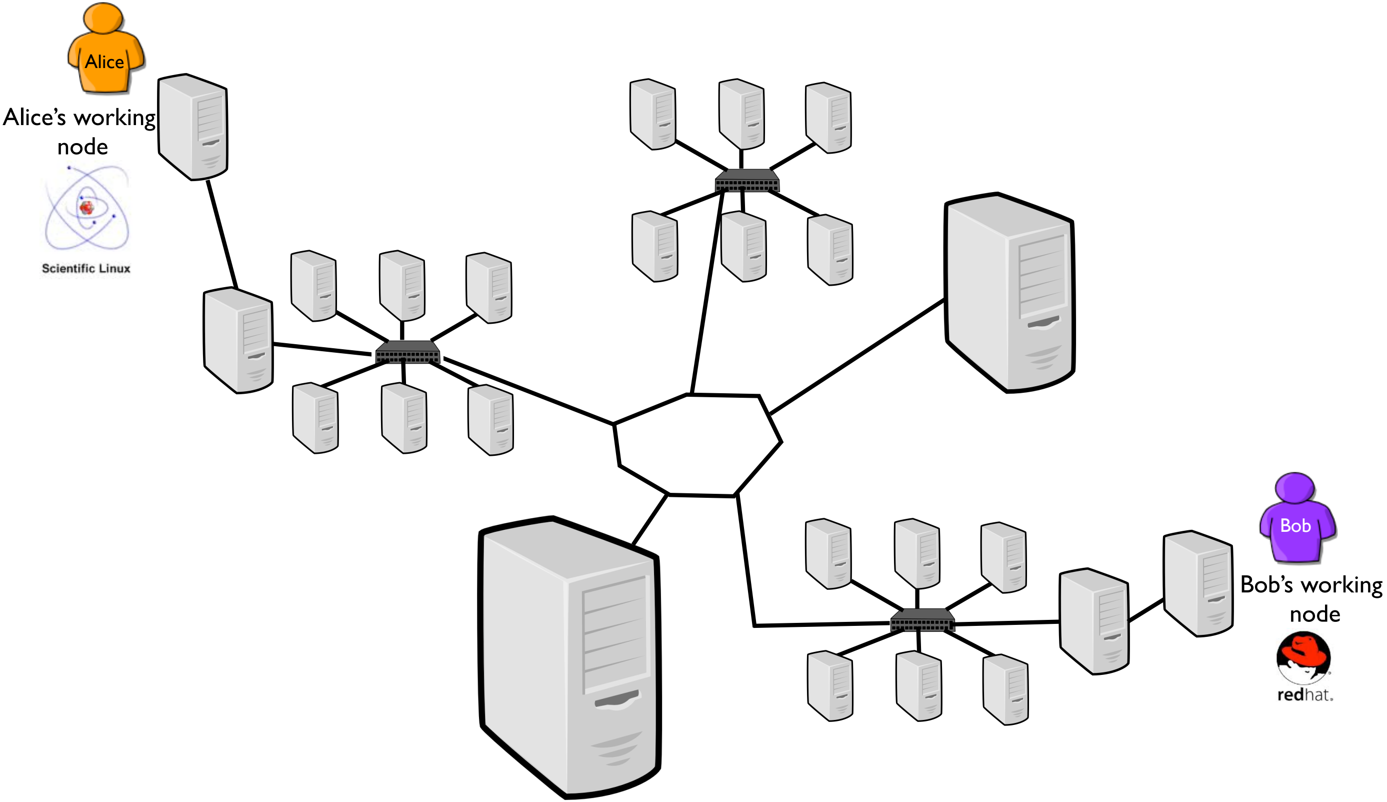


- Isolation (“security” between each VM)
- Snapshotting (a VM can be easily resumed from its latest consistent state)



- Suspend/Resume
- Live migration
  (negligible downtime ~ 60 ms)

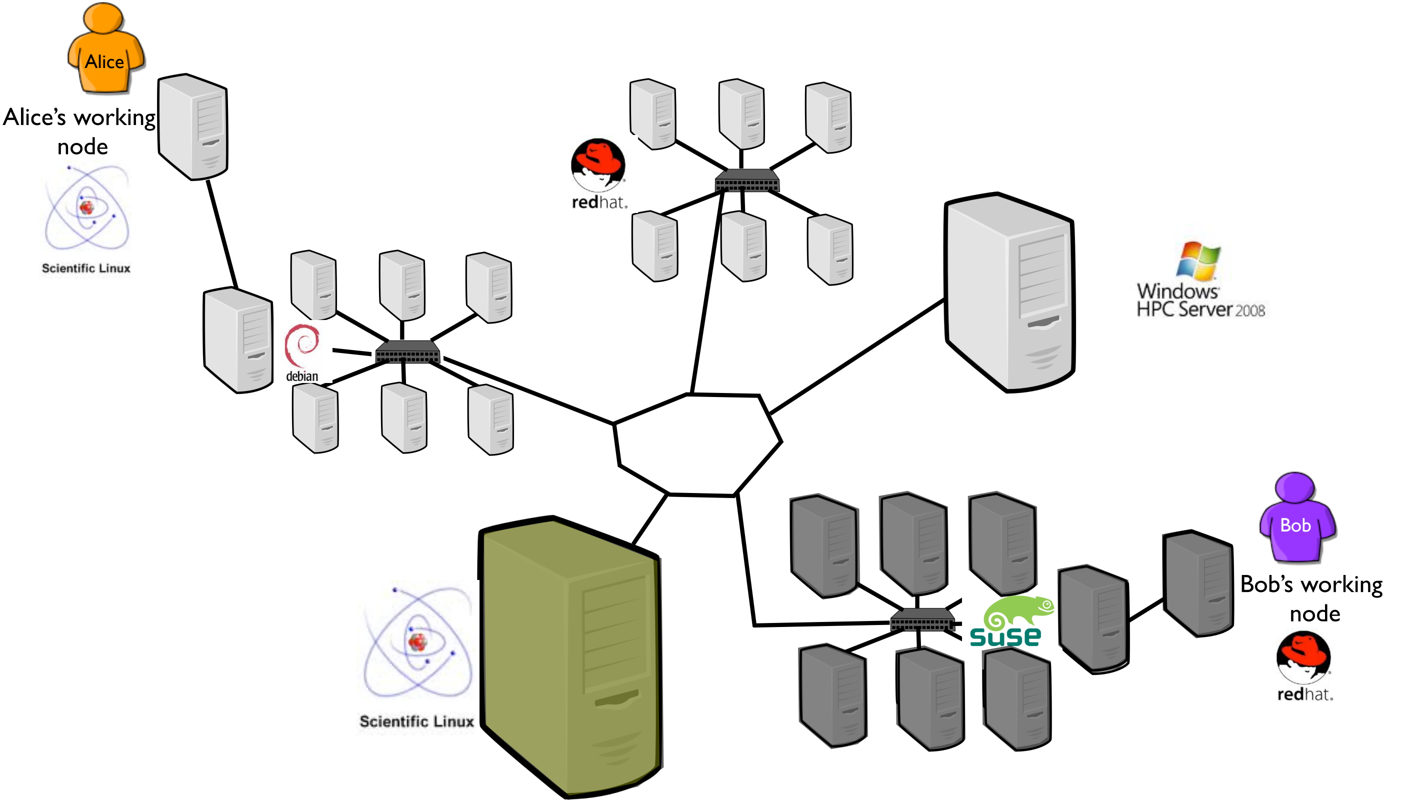
Back To The Alice/Bob Example



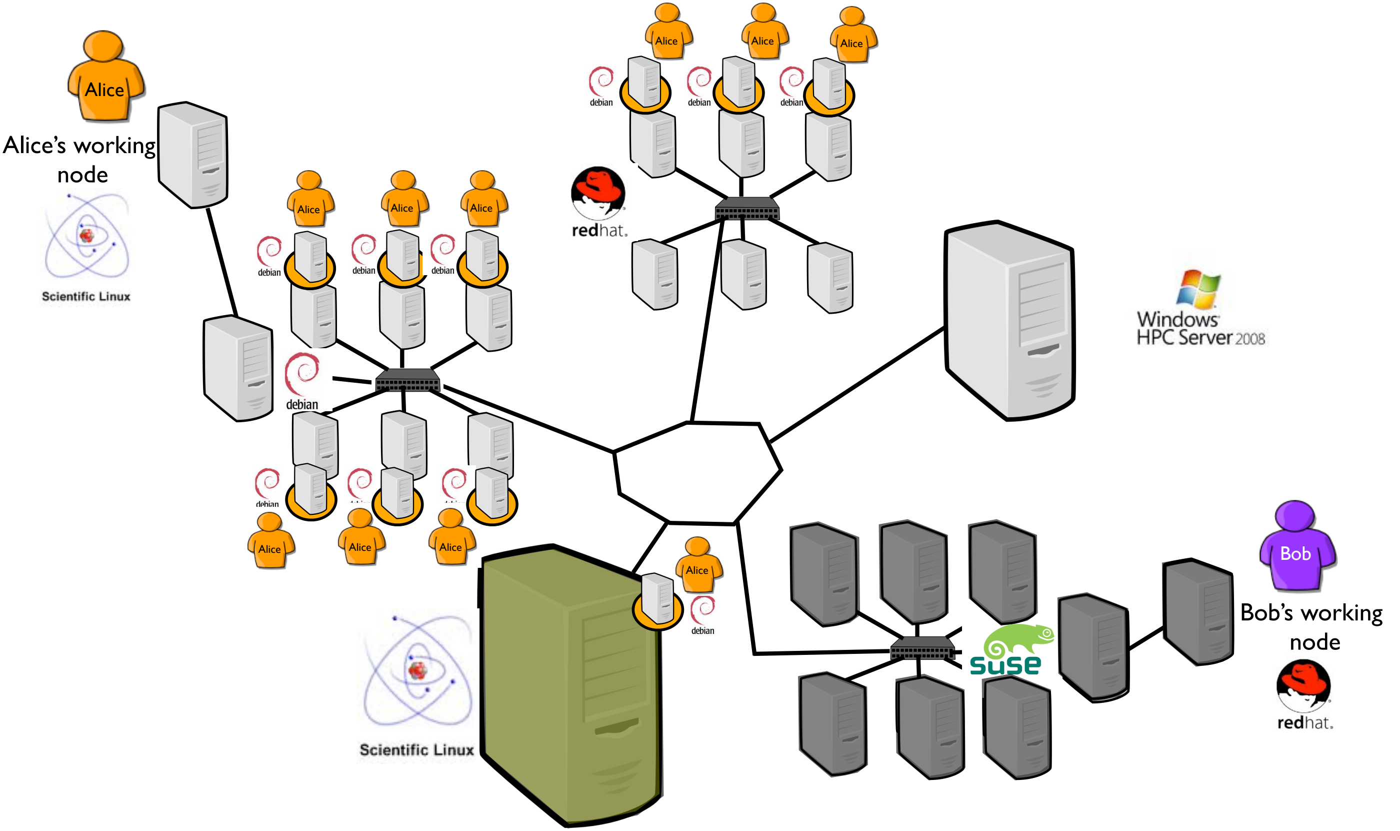
Alice's working node

Bob's working node

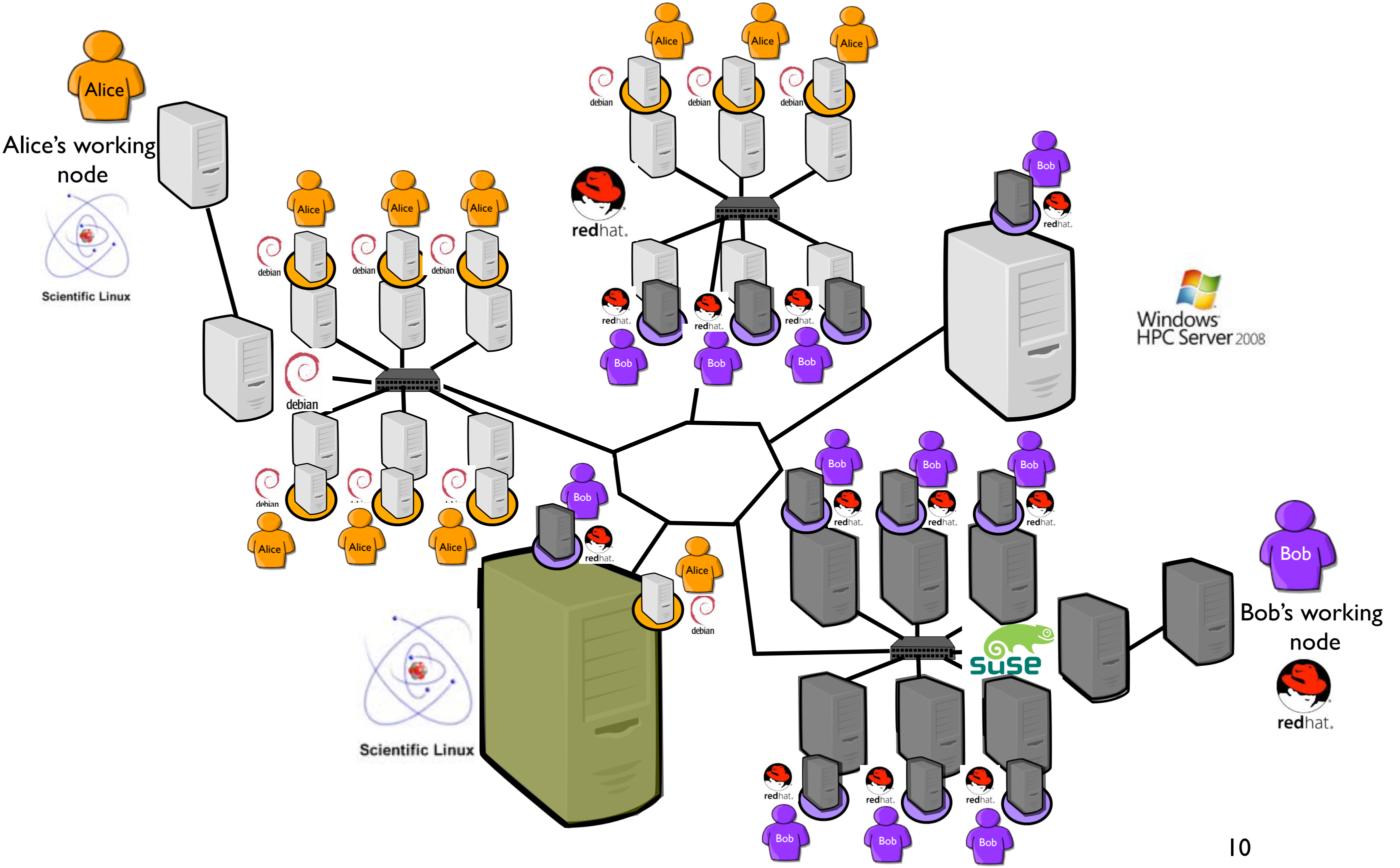
Back To The Alice/Bob Example



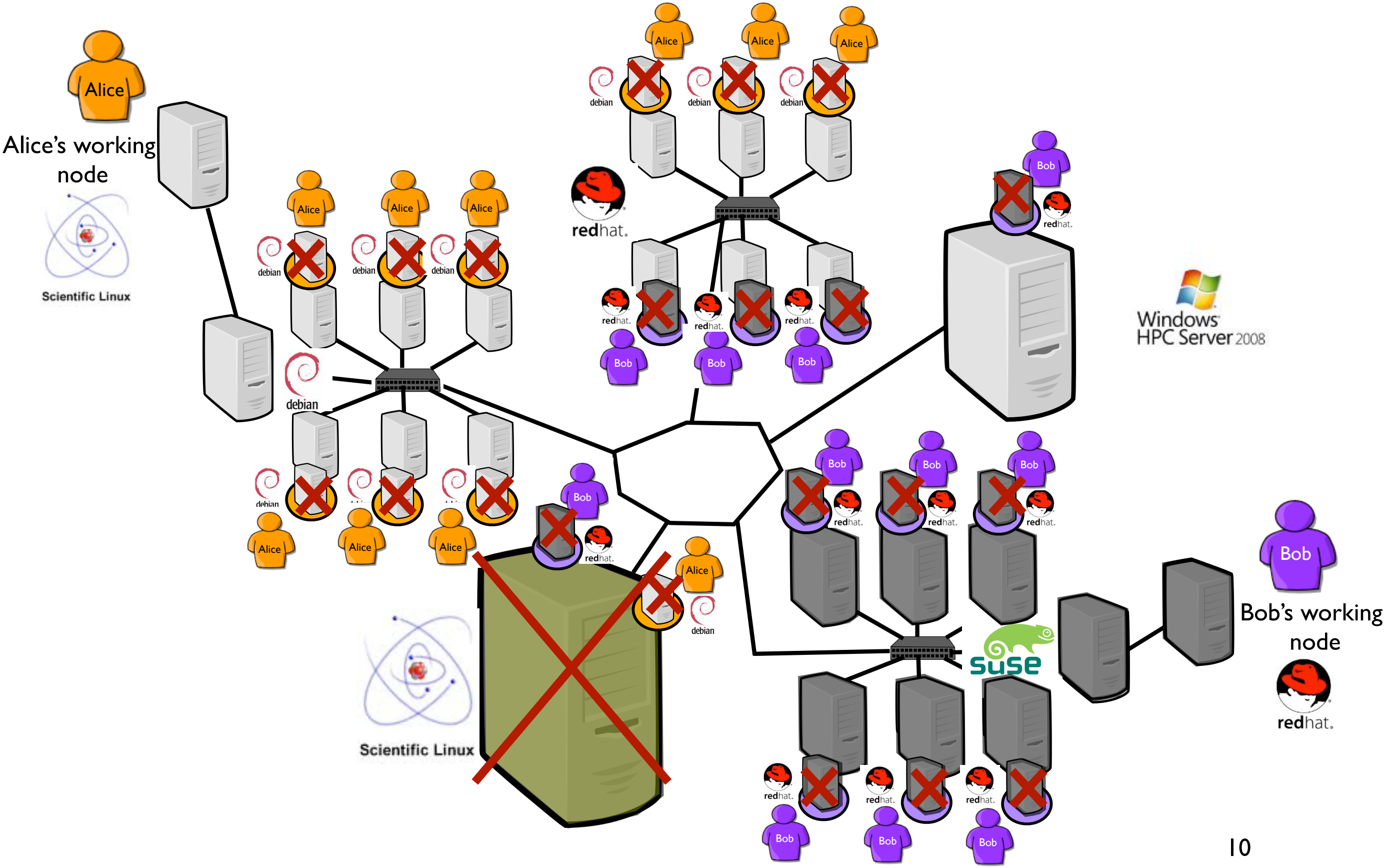
Back To The Alice/Bob Example



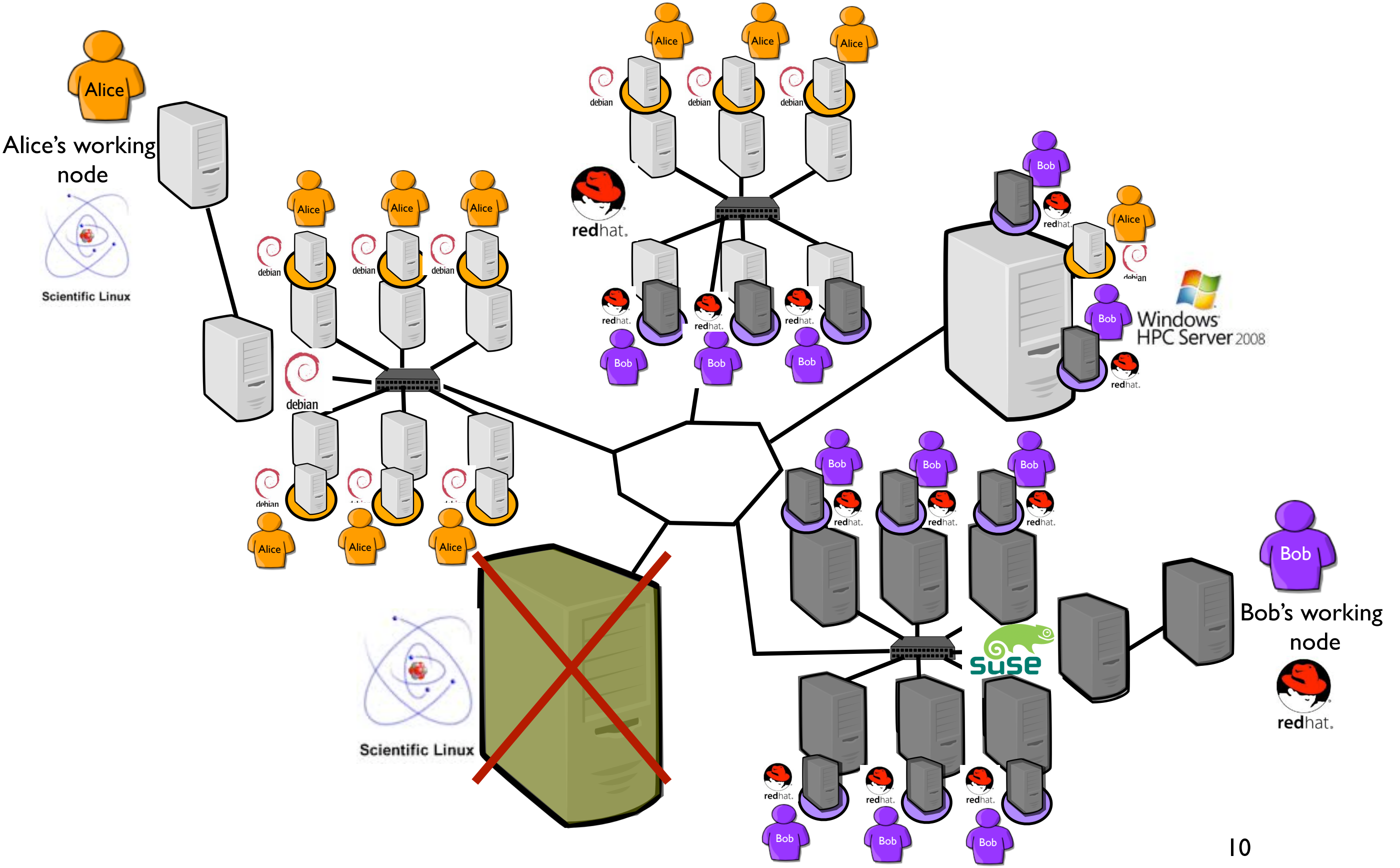
Back To The Alice/Bob Example



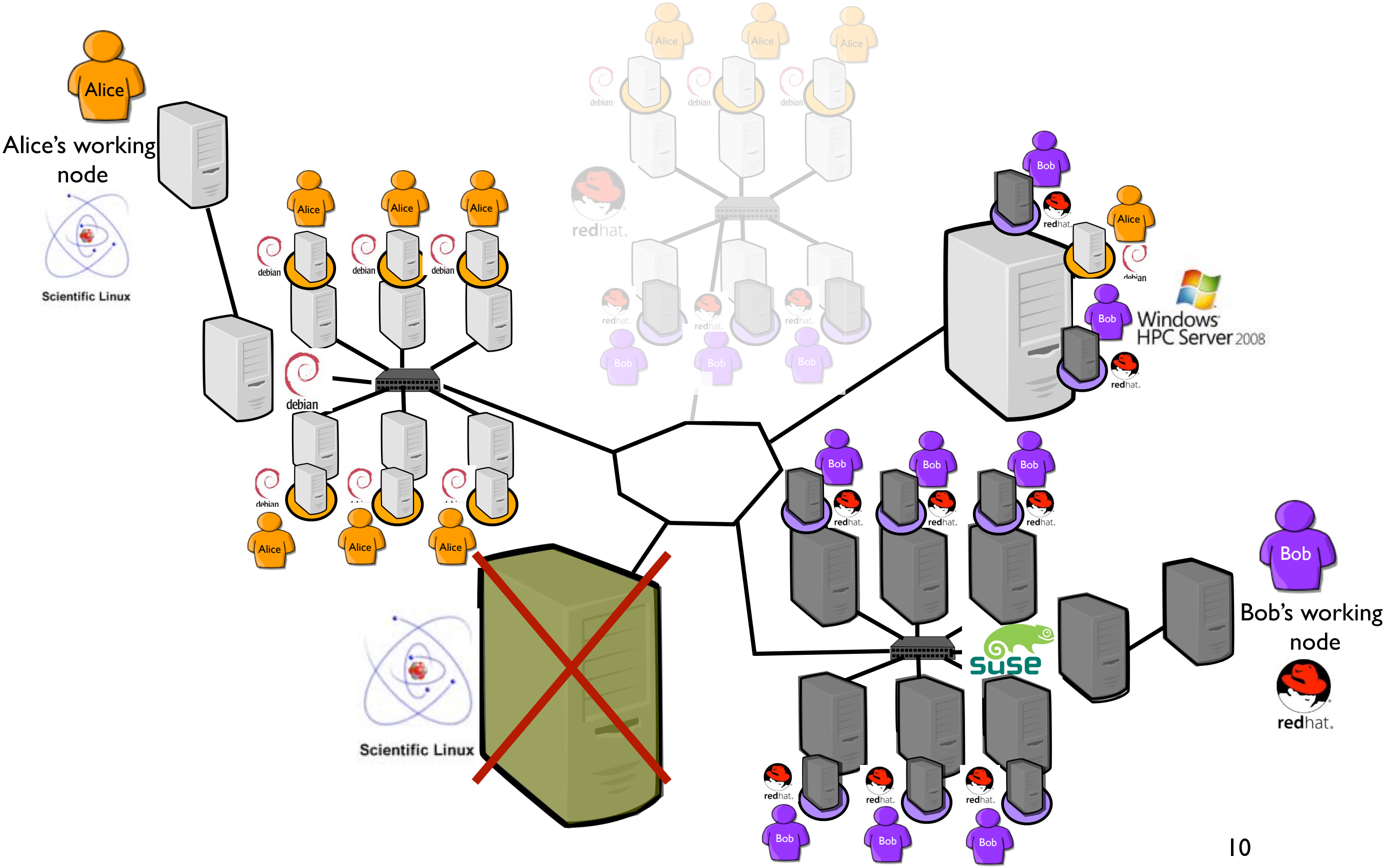
Back To The Alice/Bob Example



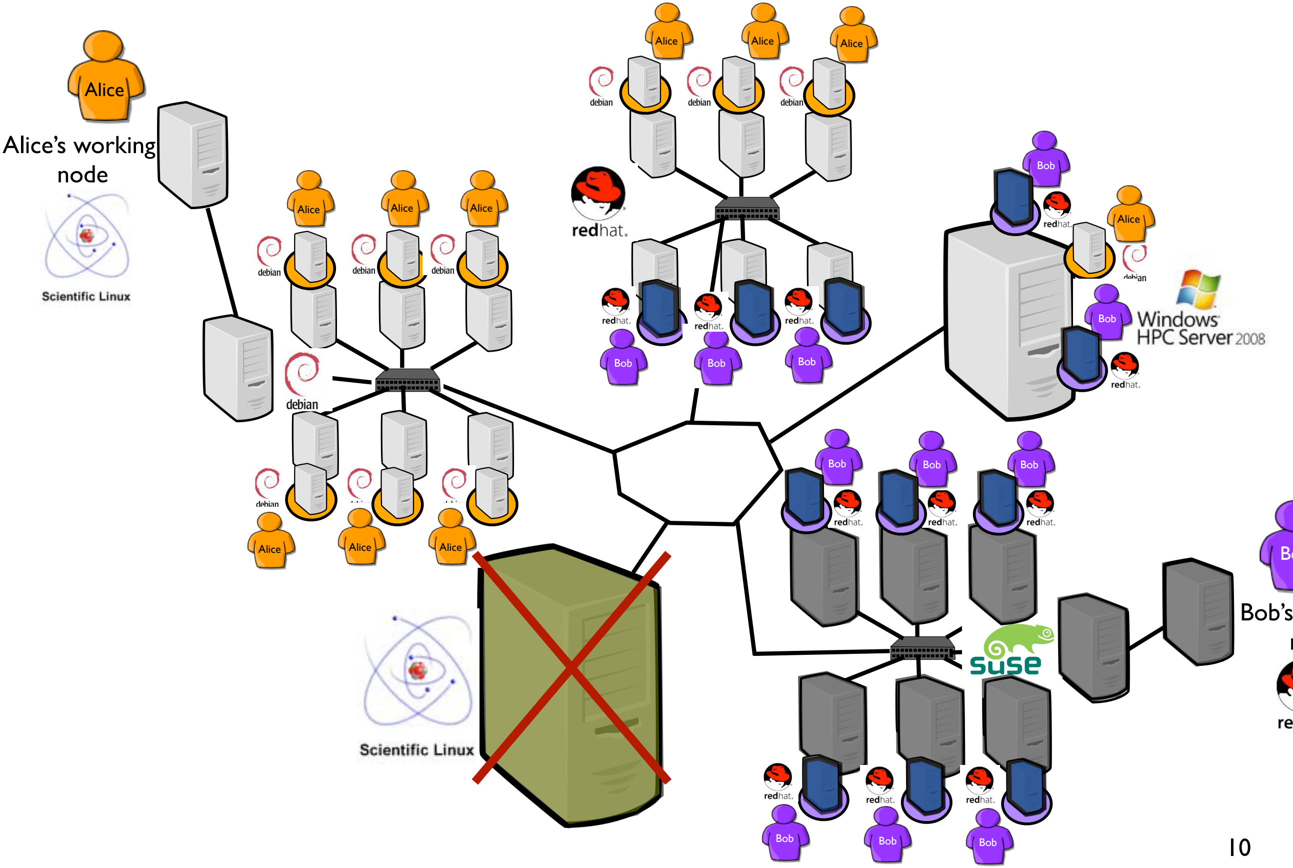
Back To The Alice/Bob Example



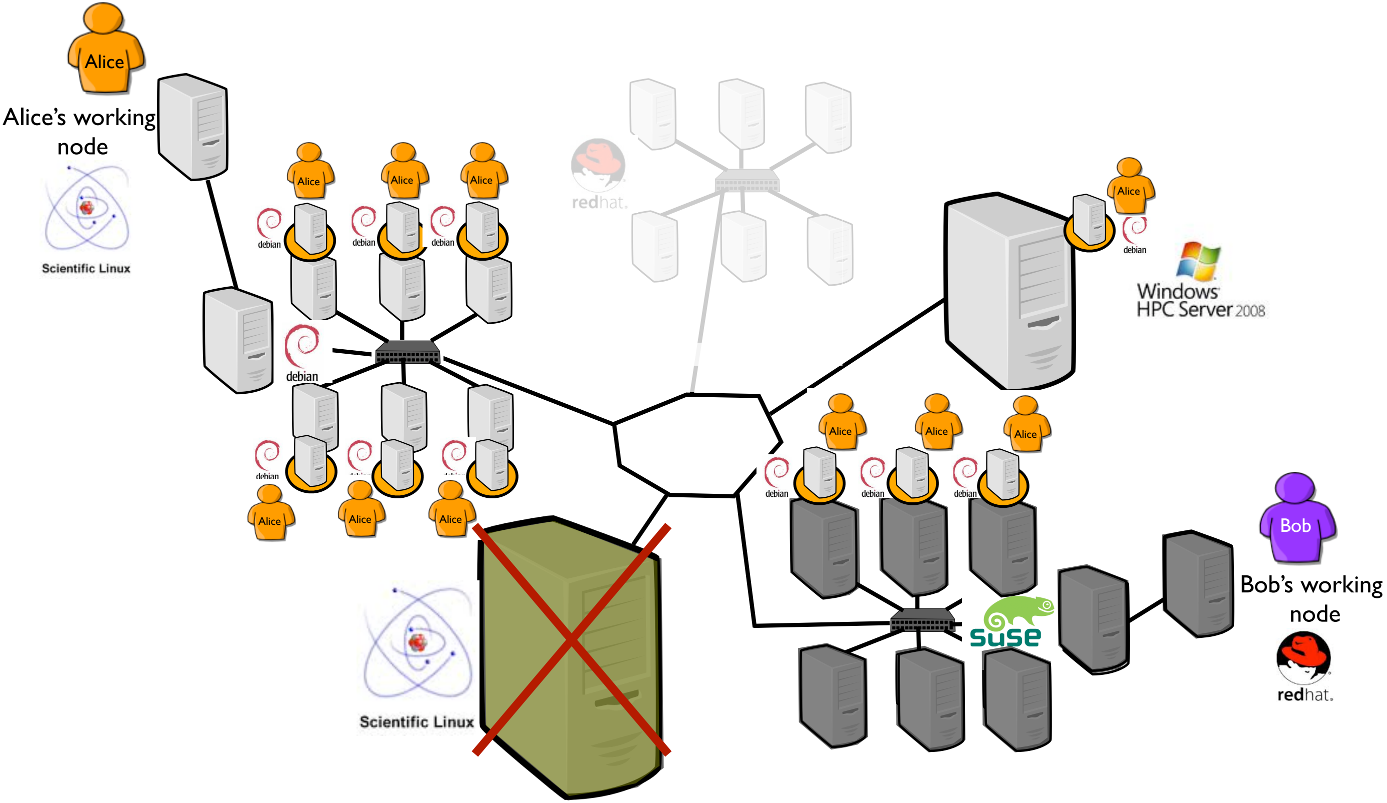
Back To The Alice/Bob Example



Back To The Alice/Bob Example



Back To The Alice/Bob Example



xxxx Computing

- xxxx as Utility

“We will probably see the spread of *computer utilities,* which, like present electric and telephone utilities, will service individual homes and offices across the country”

xxxx Computing

- xxxx as Utility

“We will probably see the spread of *computer utilities,* which, like present electric and telephone utilities, will service individual homes and offices across the country”

Len Kleinrock, 1969

credits: I. Fos Three Point Checklist

1961, Prof. John McCarthy

Focus on dynamical scheduling concerns

What can be done thanks to VM capabilities

Context

Job scheduling strategies for clusters/grids:
static allocation of resources / ”user-intrusive”

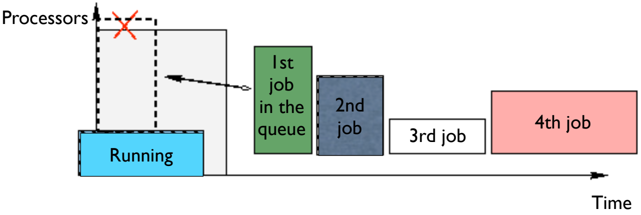
Based on user estimates (time/resources)
For a bounded amount of time
(e.g. 4 nodes for 2 hours)

Resources are reassigned at the end
of the slot without considering real
needs of applications
*(in the worst case, running applications can
be simply withdrawn from resources, i.e. G5K
best effort mode)*

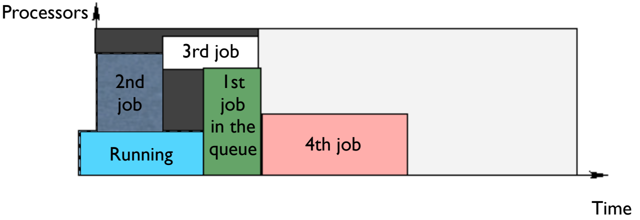
⇒ Coarse-grain exploitation
of the architecture

Context

- Batch scheduler policies: closed to FCFS



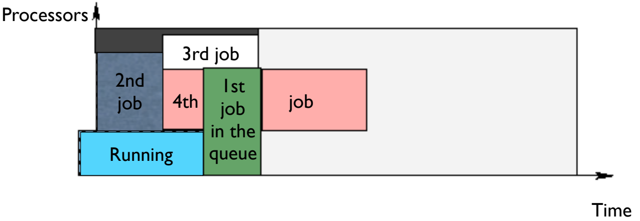
Jobs arrive in the queue and have to be scheduled.



FCFS + Easy backfilling

Jobs 2 and 3 have been backfilled.

Some resources are unused (dark areas)



Easy backfilling with preemption

The 4th job can be started without impacting the first one.

A small piece of resources is still unused.

⇒ consolidation and preemption to finely exploit distributed resources

Consolidation and Preemption

- Few schedulers include preemption mechanisms based on checkpointing solutions:
  - 😬 Strongly middleware/OS dependent
  - 😬 Still not consider application resource changes
- SSI approaches include both consolidation and preemption of processes:
  - 😬 Strongly middleware/OS dependent
  - 😬 SSI developments are tedious (most of them have been given up)
- Exploit all VM capabilities (start/stop - suspend/resume - migrate)

Consolidation and Preemption

- The Entropy proposal

  F. Hermenier, Ph.D. in CS (University of Nantes / 2009)
  Use of Live migration capability to finely exploit cluster resources [Hermenier et al. 09]

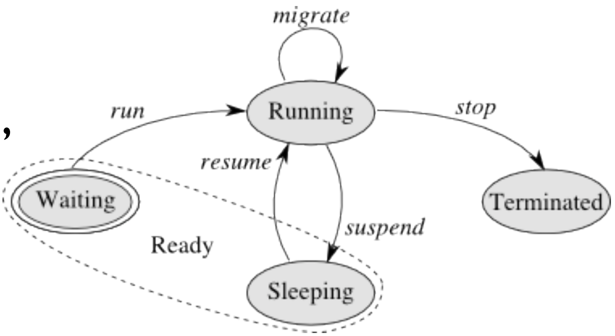
  Generalization: the Cluster-Wide Context Switch concept [Hermenier et al. 10]

- Use case - energy concerns in Datacenters

Cluster-Wide Context Switch

- General idea: manipulate vJobs instead of jobs (by encapsulating each submitted job in one or several VMs)

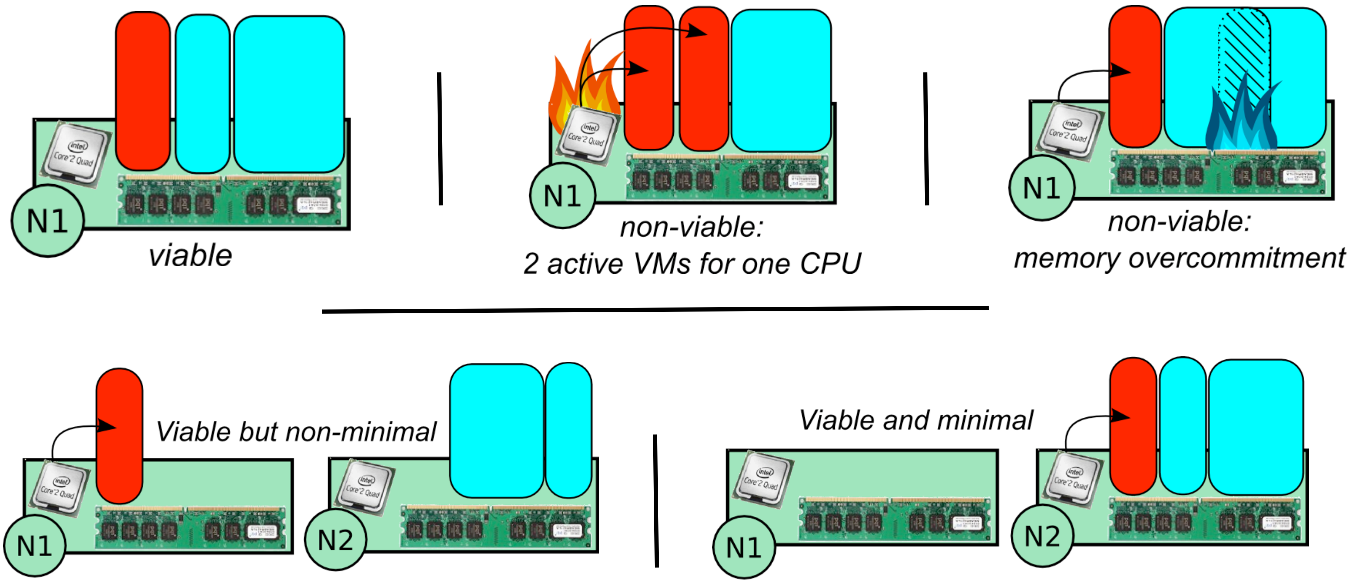
- In a similar way of usual processes, each vjob is in a particular state:



- A cluster-wide context switch (a set of VM context switches) enables to efficiently rebalance the cluster according to the: scheduler objectives / available resources / waiting vjobs queue

The Entropy Proposal

- To finely exploit resources (efficiency and energy constraints)
- Find the "right" mapping between VM needs and resources provided by PM



N1

viable

N1

non-viable:
2 active VMs for one CPU

N1

non-viable:
memory overcommitment

N1

Viable but non-minimal

N2

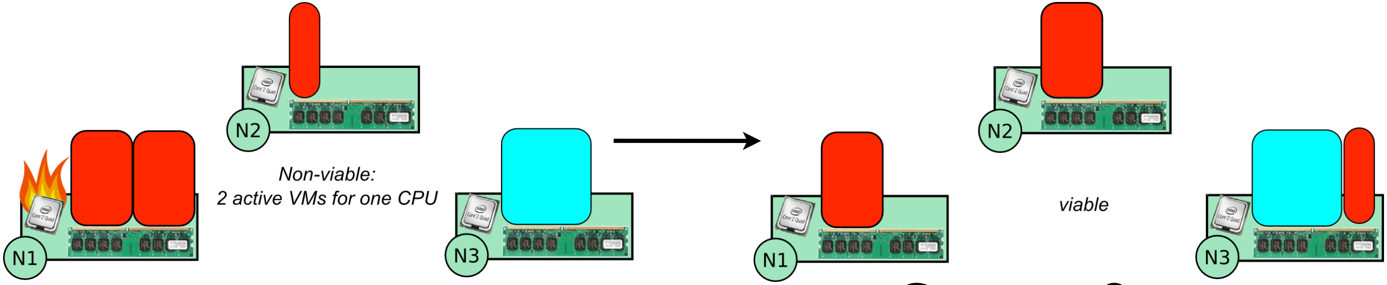
N1

Viable and minimal

N2

credits: F. Hermenier, Mines Nantes

The Entropy Proposal

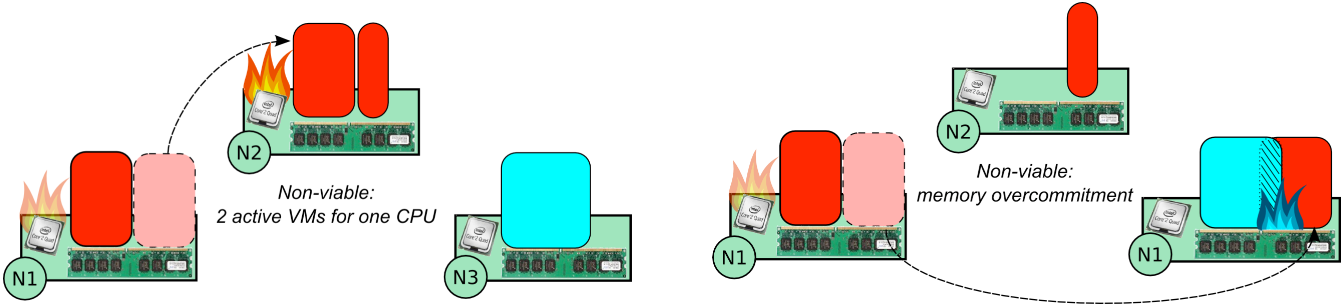


Non-viable:
2 active VMs for one CPU

viable

Current Status

Correct Status



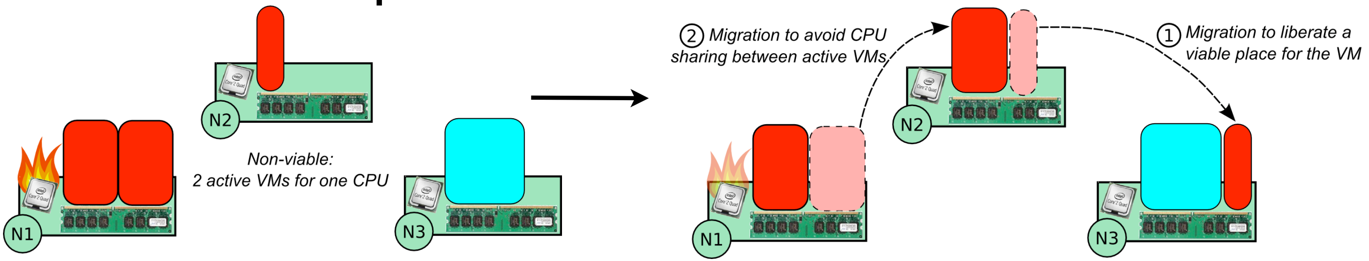
Non-viable:
2 active VMs for one CPU

Non-viable:
memory overcommitment

Non-viable manipulations

The Entropy Proposal

- Order VM Operations

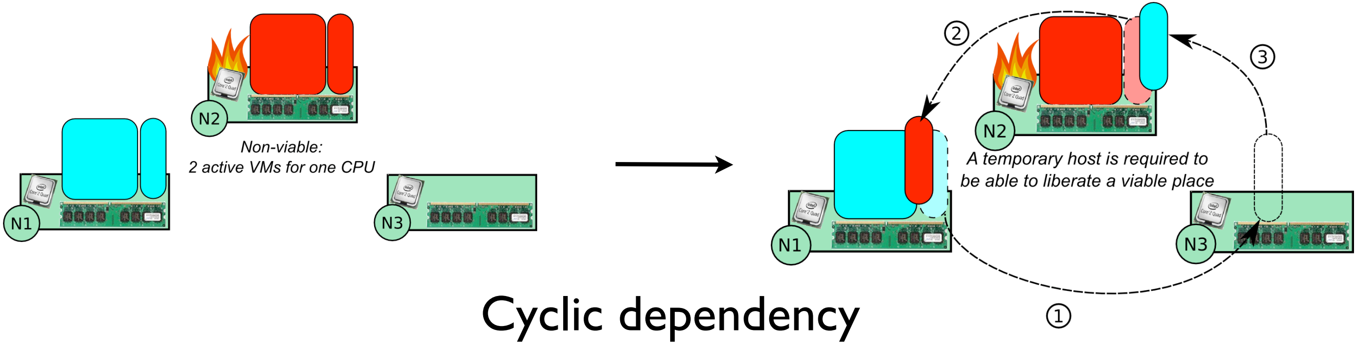


Non-viable:
2 active VMs for one CPU

② Migration to avoid CPU sharing between active VMs

① Migration to liberate a viable place for the VM

Sequential dependency



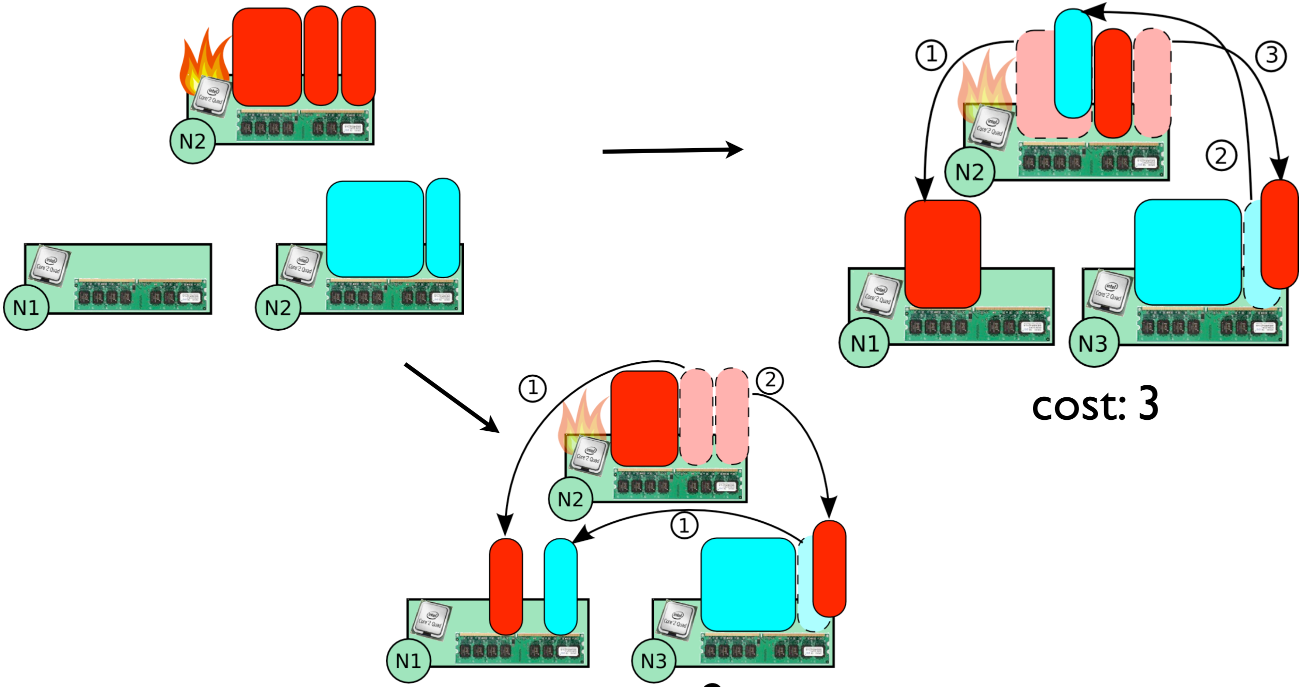
Non-viable:
2 active VMs for one CPU

A temporary host is required to be able to liberate a viable place

Cyclic dependency

The Entropy Proposal

- Optimizing the reconfiguration process



cost: 3

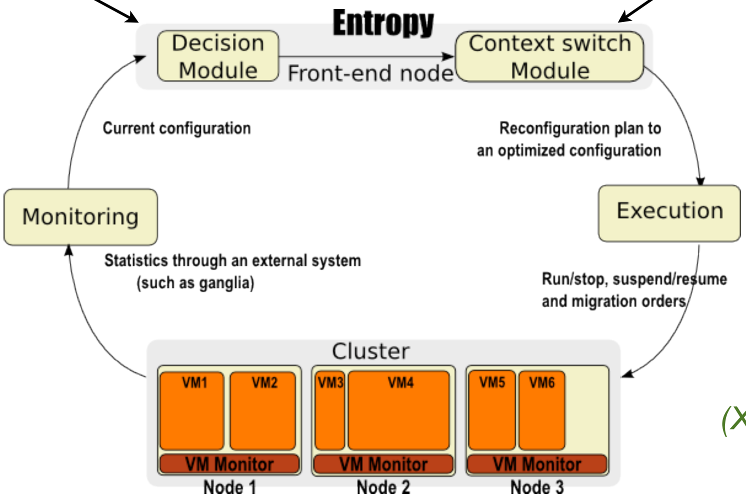
cost: 2

The Entropy Proposal

- The big picture: an autonomic model

Scheduling algorithm: select the jobs to run
(objectives/strategies defined by administrators)

Compute an efficient reconfiguration plan
to reach the expected configuration
(through the Choco constraint solver)



Actions are done
through *drivers*
(XEN XML-RPC API / ssh)

- http://entropy.gforge.inria.fr, irc.freenode.net #entropy,

The Entropy Proposal

- To sum up

😁 An autonomic framework to make the implementation of vjobs scheduling policies easier

Strength: composition of constraints
Developed since 2006 (ANR SelfXL / MyCloud, ANR Emergence, 10 persons)

😎 “Prix de la croissance verte numérique” in 2009

😬 Scalability of both computation and execution of the reconfiguration plan

😜 Work in progress

Performance/scalability/...