Universitat
Autònoma
de Barcelona

Scheduling in Virtual Infrastructure

Departamento de Arquitectura de
Computadores y Sistemas Operativos
(CAOS/DACSO)
Escuela de Ingeniería
Universidad Autónoma de Barcelona

Chalapathi Valupula
Dirigida por :Elisa Heymann.

Index

- Objective
- Introduction
- Pre-Staging Model Architecture
- Pre-Staging Model Implementation
- Results
- Concluding Remarks & Open lines

Objectives

- Scheduling Virtual Infrastructure for efficiently reusing Virtual Machines
    - ➢ Makes use of virtualization technology advantages
    - ➢ Successive multiple job executions
    - ➢ Reducing the overhead induced by Virtualization
- Designing Pre-Staging Architecture
    - ➢ Integrates seamlessly with existing infrastructure
    - ➢ Is backwards compatible with traditional interfaces
    - ➢ Deals with Virtual Machine life cycle
- Implementation and experimentation
    - ➢ Prove Pre-Staging Architecture viability
    - ➢ Detect bottlenecks and other limitations
    - ➢ Measure Pre-Staging model performance

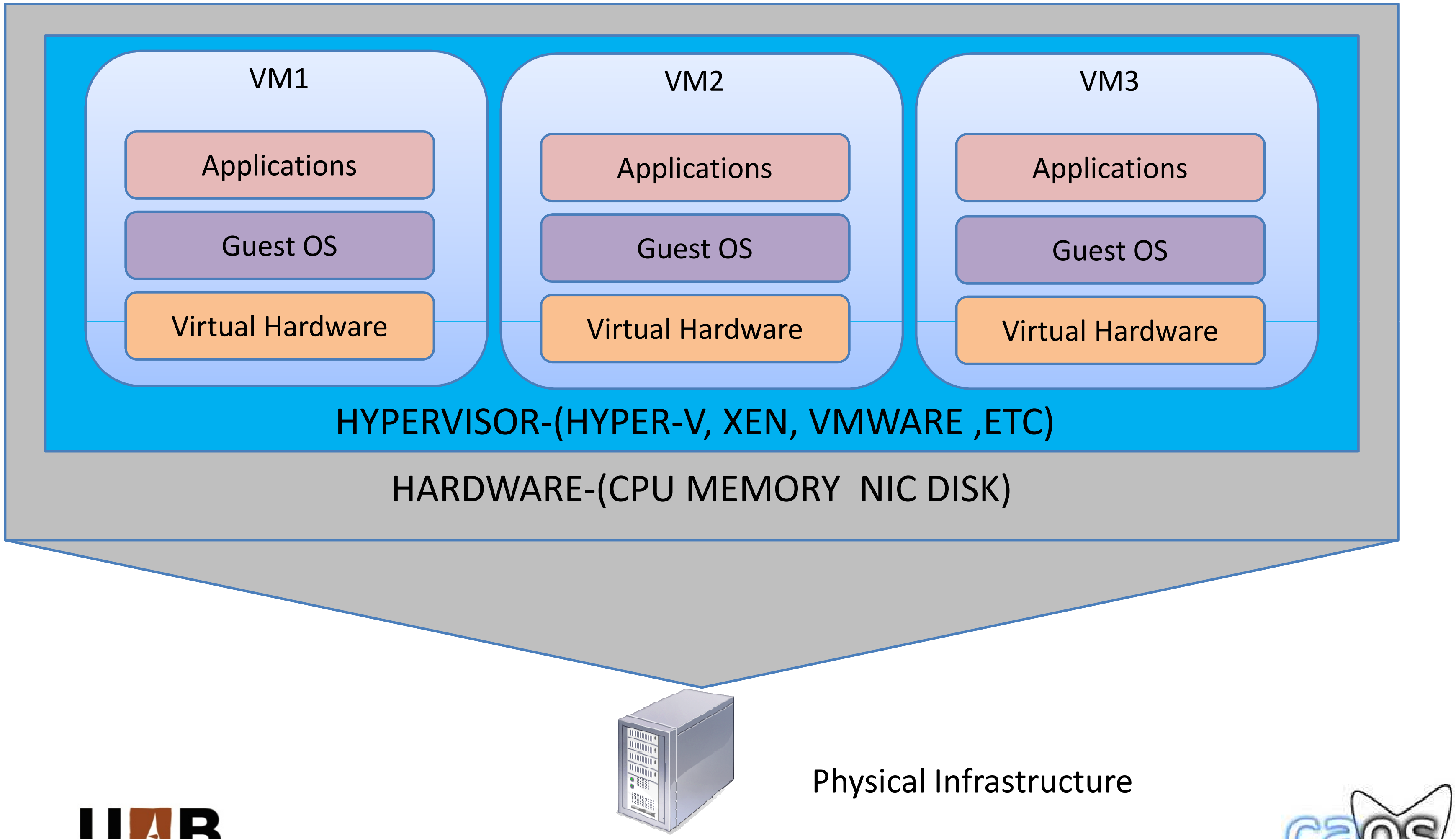
Introduction

- As technology evolves, computational resources increase
- We can use Distributed architectures such as Grids and Clouds to solve large scientific problems
- Distributed architectures are inherently complex:
  - ➢ Resources scattered all around the globe and are heterogeneous
  - ➢ Distributed administration: no centralized control
  - ➢ Efficient resource management is essential in those architectures in order to achieve high performance

Virtualization Concept

- Virtualization is a framework or methodology of dividing the resources of a computer into multiple execution environments, by applying one or more concepts or technologies such as hardware and software partitioning, time-sharing, partial or complete machine simulation, emulation, quality of service, and many others.
- Virtualized resources enable a more efficient resource management
- Each instance of such execution is called a Virtual Machine(VM)

Virtualization Concept



VM1

Applications

Guest OS

Virtual Hardware

VM2

Applications

Guest OS

Virtual Hardware

VM3

Applications

Guest OS

Virtual Hardware

HYPERVISOR-(HYPER-V, XEN, VMWARE ,ETC)

HARDWARE-(CPU MEMORY NIC DISK)

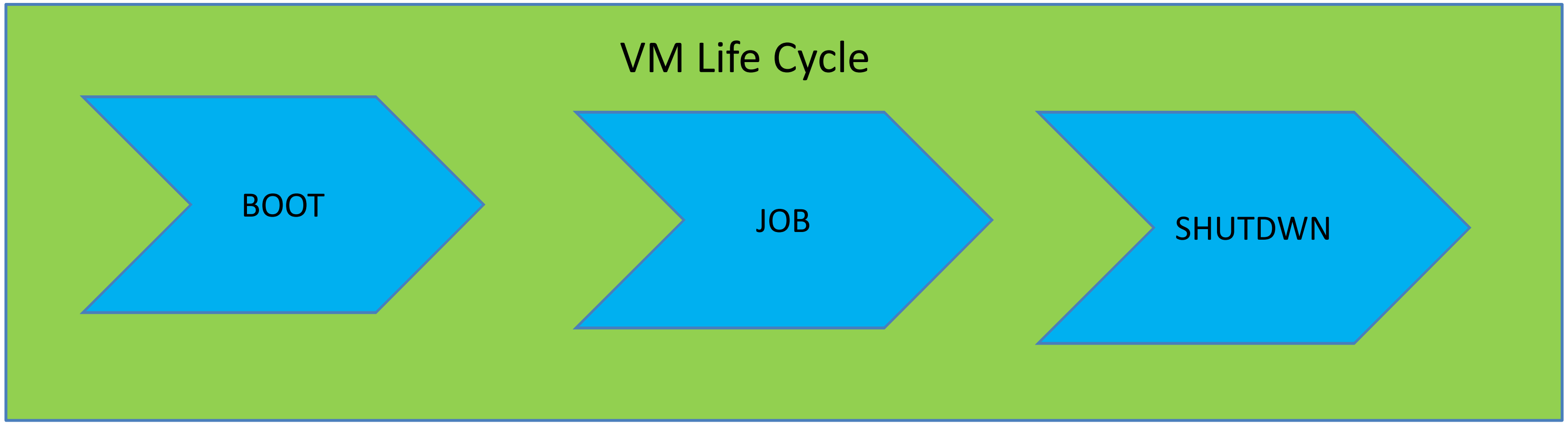
Physical Infrastructure

Virtualization Concept

- Few advantages of Virtual Machines
  - ➢ Hardened security
  - ➢ Platform isolation
  - ➢ Easy reconfiguration
  - ➢ Better Reliability, Availability and Serviceability

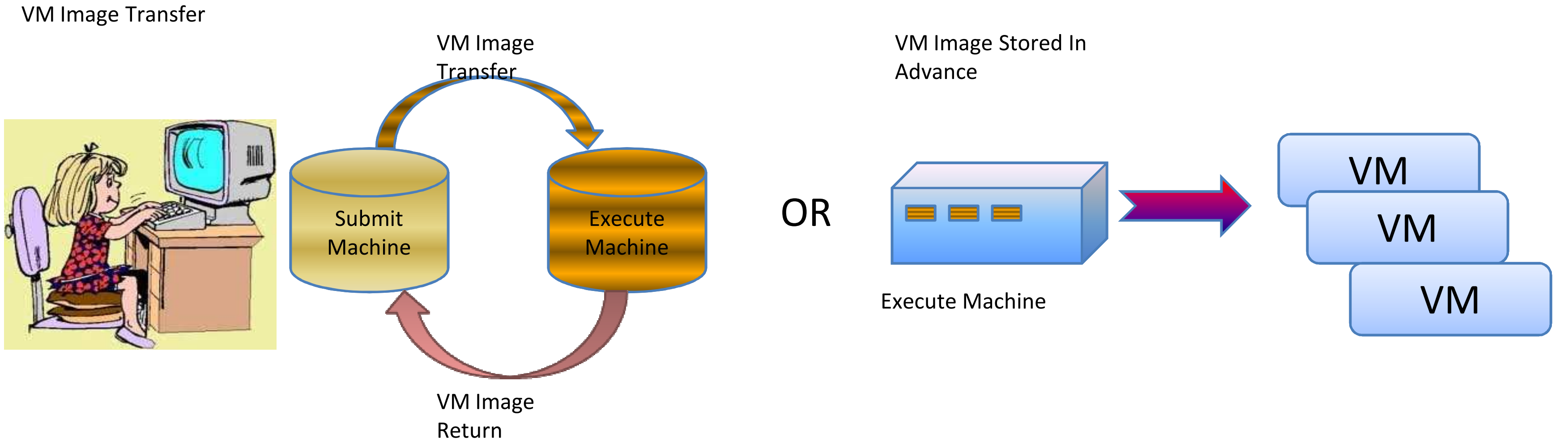
Virtual Machine Life Cycle

- Virtual Machine Life Cycle
  - Boot Up of the VM
  - Running Job on VM
  - Completion Job and Shutdown of the VM



Condor Virtual Machine Universe

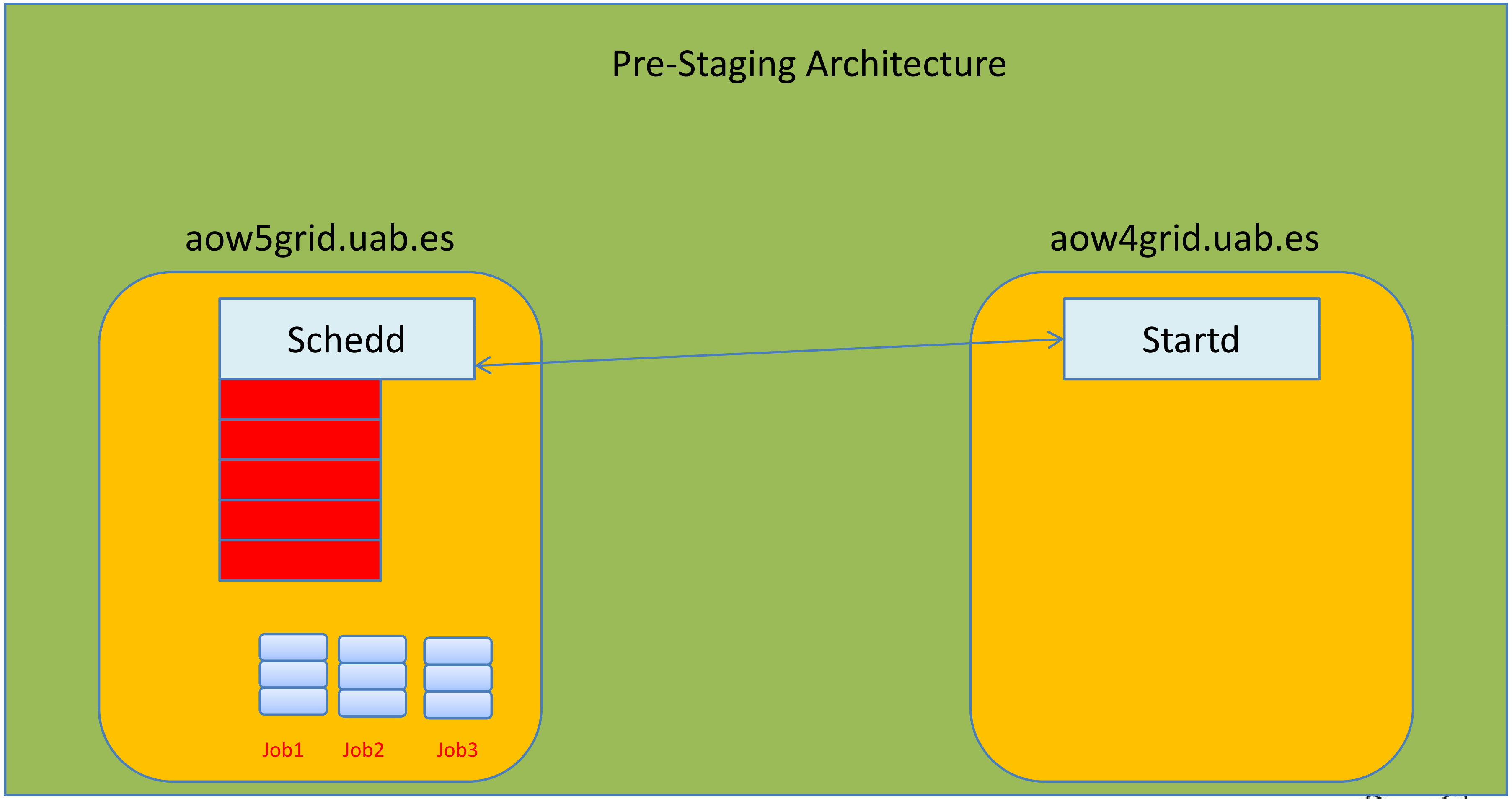
- Virtual Machine Job
  - Starting Boot Up of the VM
  - Running VM On
  - Completion Shutdown of the VM
  - Result Modified VM image (Optional)



Pre-Staging Architecture

- We design Pre-Staging Architecture based on the Condor distributed system
- We choose a to establish a Condor pool by using two systems aow5grid.uab.es and aow4grid.uab.es
- Machine aow5grid.uab.es is used as Condor manager which matches the jobs to resources and manages the jobs ,as well as a submit machine from where we submit our jobs.
- Machine aow4grid.uab.es is used as a Condor execute machine where our jobs are executed.
- Both these machine are pre-configured to support Virtual Machines.

Pre-Staging Architecture



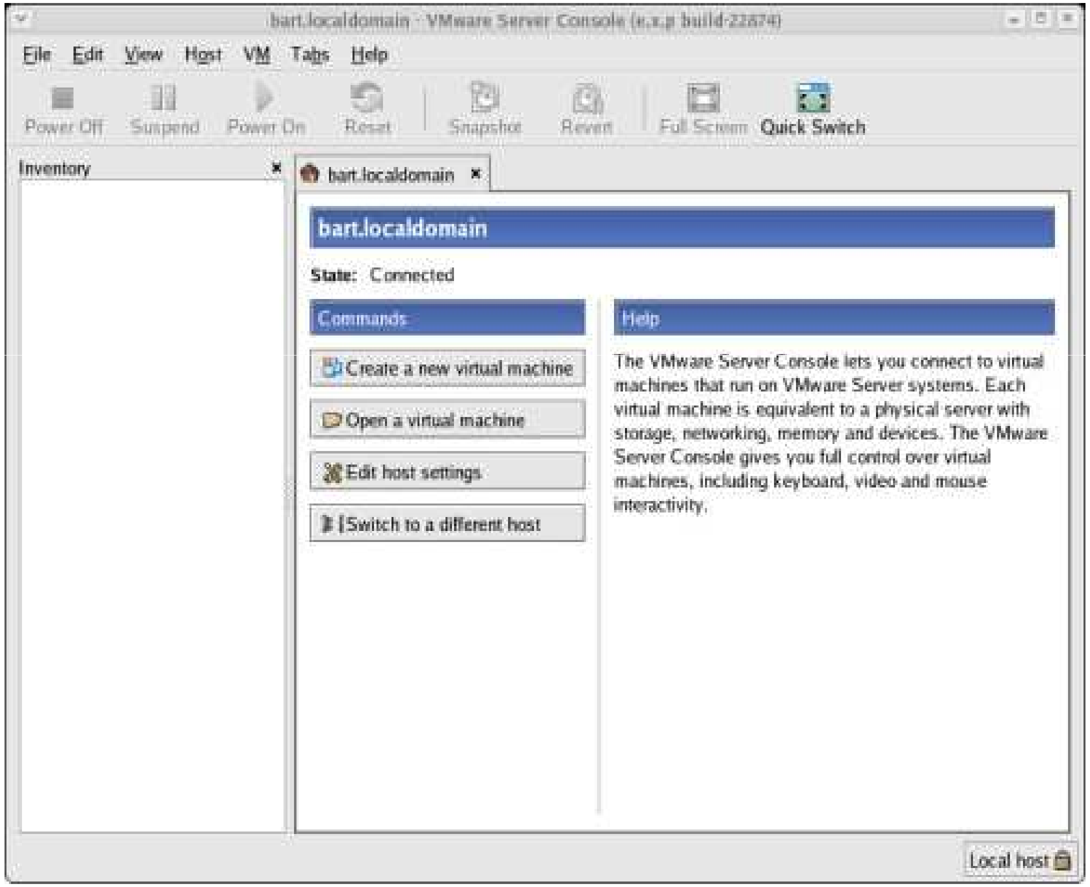
Pre-Staging Architecture

- Now that we have designed the physical Infrastructure
- We design a Virtual Machine Image by using VMWare server 1.0.1
- Our Virtual Machine Image for this project is called as aow12grid.uab.es
- The Virtual Machine Image is configured with the same version of Condor demons that are running on the physical machines aow5grid.uab.es and aow4grid.uab.es to avoid any compatibility issues
- We also insert custom ClassAd attributes in a machine ad via the config file on the Virtual Machine Image

  Machine = “aow12grid.uab.es"
  STARTD_ATTRS = $(STARTD_ATTRS) Machine

Virtual Machine Image

- We create VMWare server Virtual Machine Image
- Using VMWare console
- Fedora VMI
  - ➢ Aow12grid.uab.es
  - ➢ OS Fedora 9.
  - ➢ Memory: 157.9 MiB.
  - ➢ Processor: Intel(R) Pentium(R) 4 CPU 1.8GHz
  - ➢ Condor version-7.4.4
- Ubuntu VMI
  - ➢ Aow12grid.uab.es
  - ➢ Ubuntu10.10.
  - ➢ Memory: 264 MiB.
  - ➢ Processor: Intel(R) Pentium(R) 4 CPU 1.8GHz
  - ➢ Condor version-7.4.4



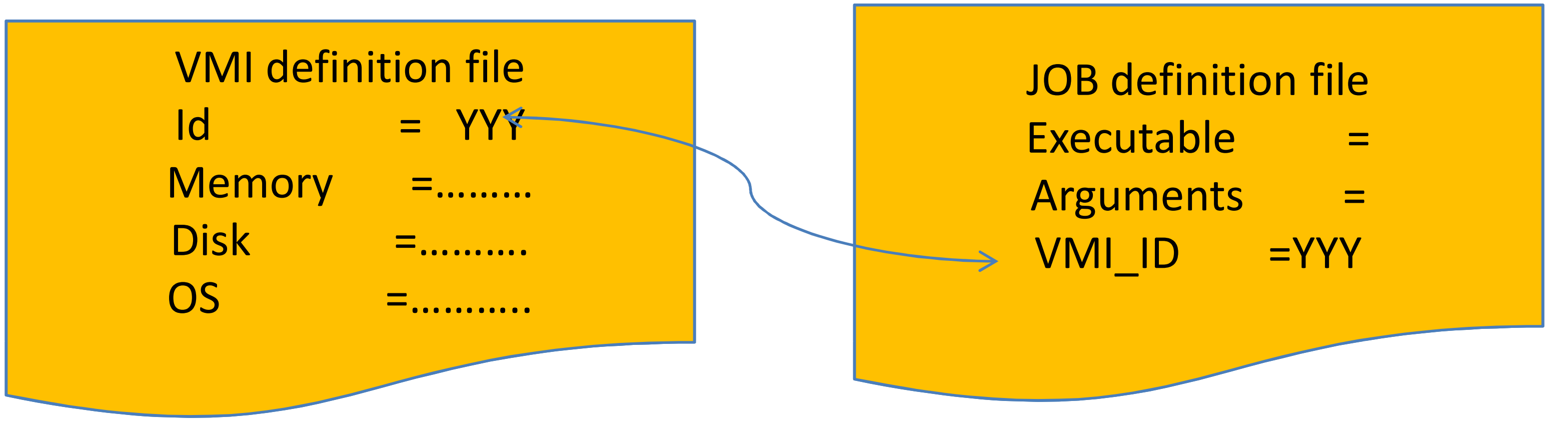
State: Connected

The VMware Server Console lets you connect to virtual machines that run on VMware Server systems. Each virtual machine is equivalent to a physical server with storage, networking, memory and devices. The VMware Server Console gives you full control over virtual machines, including keyboard, video and mouse interactivity.

Virtual Machine Image

- Few characteristics of Virtual Machine Image
  - **Image Compatibility** The Virtual Machine image must be in a format usable by the hypervisor software in use at the execute machine.
  - **Architecture Compatibility** The operating system running in the Virtual Machine must be compatible with the system architecture exposed by the hypervisor.
  - **Dynamic Reconfigurability** The guest system inside the VM must be able to have certain properties, such as its MAC address, IP address, hostname, and Condor job scheduler set at boot time.

Typical use case



VMI definition file
Id = YYY
Memory =………
Disk =……….
OS =………..

JOB definition file
Executable =
Arguments =
VMI_ID =YYY

Every job and VM has a description file

- VM description file has
  - HW platform, OS, memory characteristics, ...
  - **A unique identifier**
- Job description file has
  - Executable, arguments, input files, ...
  - **A parameter links the job with the VM that should host**
- Users submit their jobs along with the description file
  - Architecture transparently manages job execution and return the results

Pre-Staging Model Implementation

- Now that we have the Pre-Staging architectural design
- We now implement our design form the submit machine (Aow5grid.uab.es)
- We select Pre-configured Virtual Machine Image (here Fedora)
- We create a Condor VM Universe submit file Submit1
    - ➢ [condor@aow5grid~]$ Condor_submit submit1

```
Universe                              = vm
Executable                            = without any job
Log                                   = simple.vm.log.txt
vm_type                               = vmware
vm_memory                             = 164
vmware_dir                            = /home/condor/condor-job/Fedora
vmware_should_transfer_files  = true
Queue
```

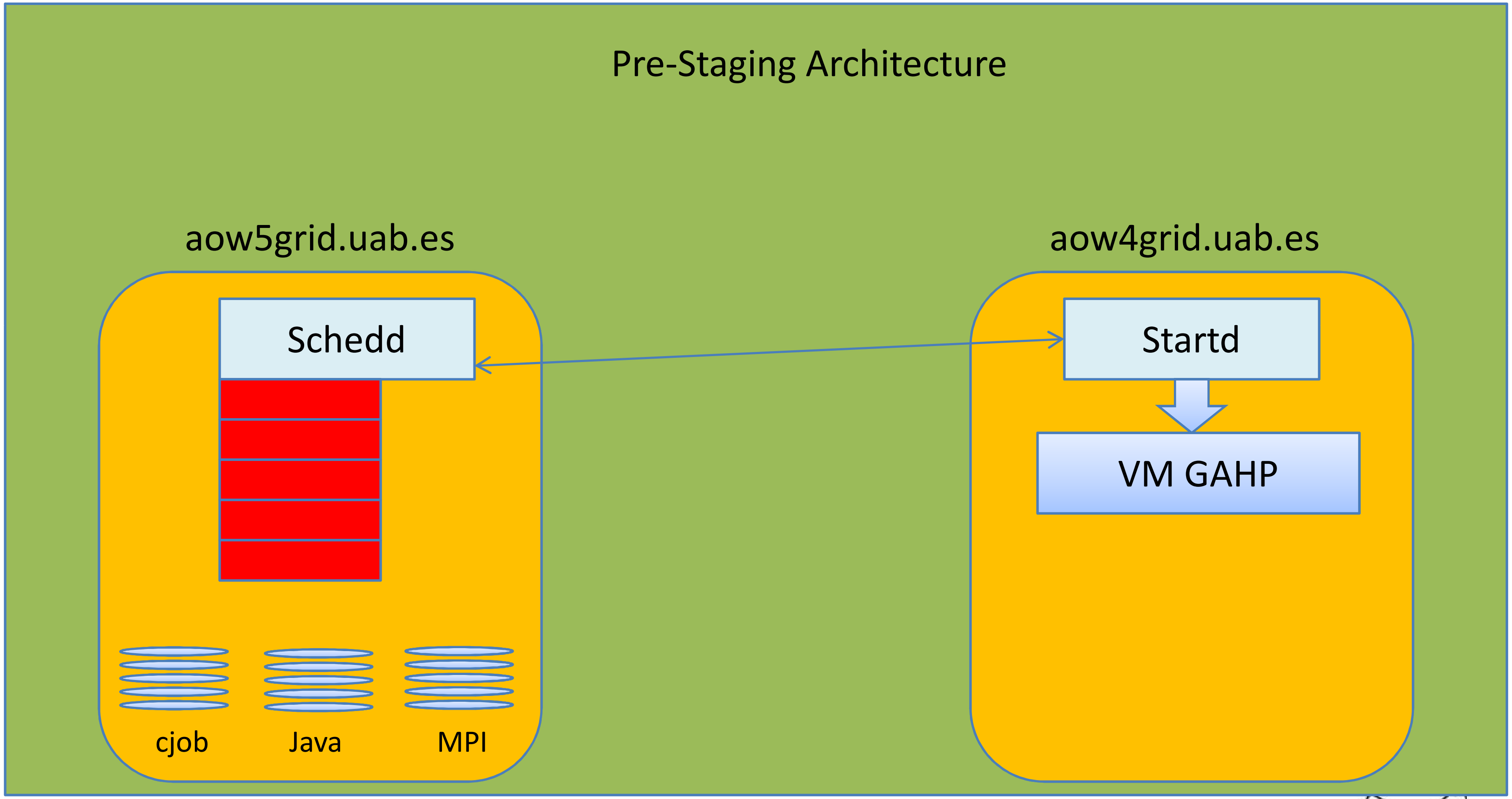
Pre-Staging Model Implementation

- We submit our Virtual Machine Image to Condor VM Universe using *Condor_submit submit1*
- As our Condor Pool consist of two Machines the condor manager (aow5grid.uab.es) runs the job on execute machine (aow4grid.uab.es)
- At this stage we check the status of our Condor pool using the command *Condor_status* which displays the status of the pool and no of machines.

```
[condor@aow5grid~]$ condor_status
Name              OpSys    Arch    State      Activity   LoadAv   Mem    ActvtyTime
Aow5grid.uab.es   LINUX    INTEL  Unclaimed  Idle        0.020    502    0+00:10:06
Aow4grid.uab.es   LINUX    INTEL   Claimed    Busy       0.940    512    0+00:00:40
```

UAB
Universitat Autònoma de Barcelona

Pre-Staging Model Implementation



Pre-Staging Architecture

aow5grid.uab.es

aow4grid.uab.es

Schedd

Startd

VM GAHP

cjob

Java

MPI

Universitat Autònoma de Barcelona

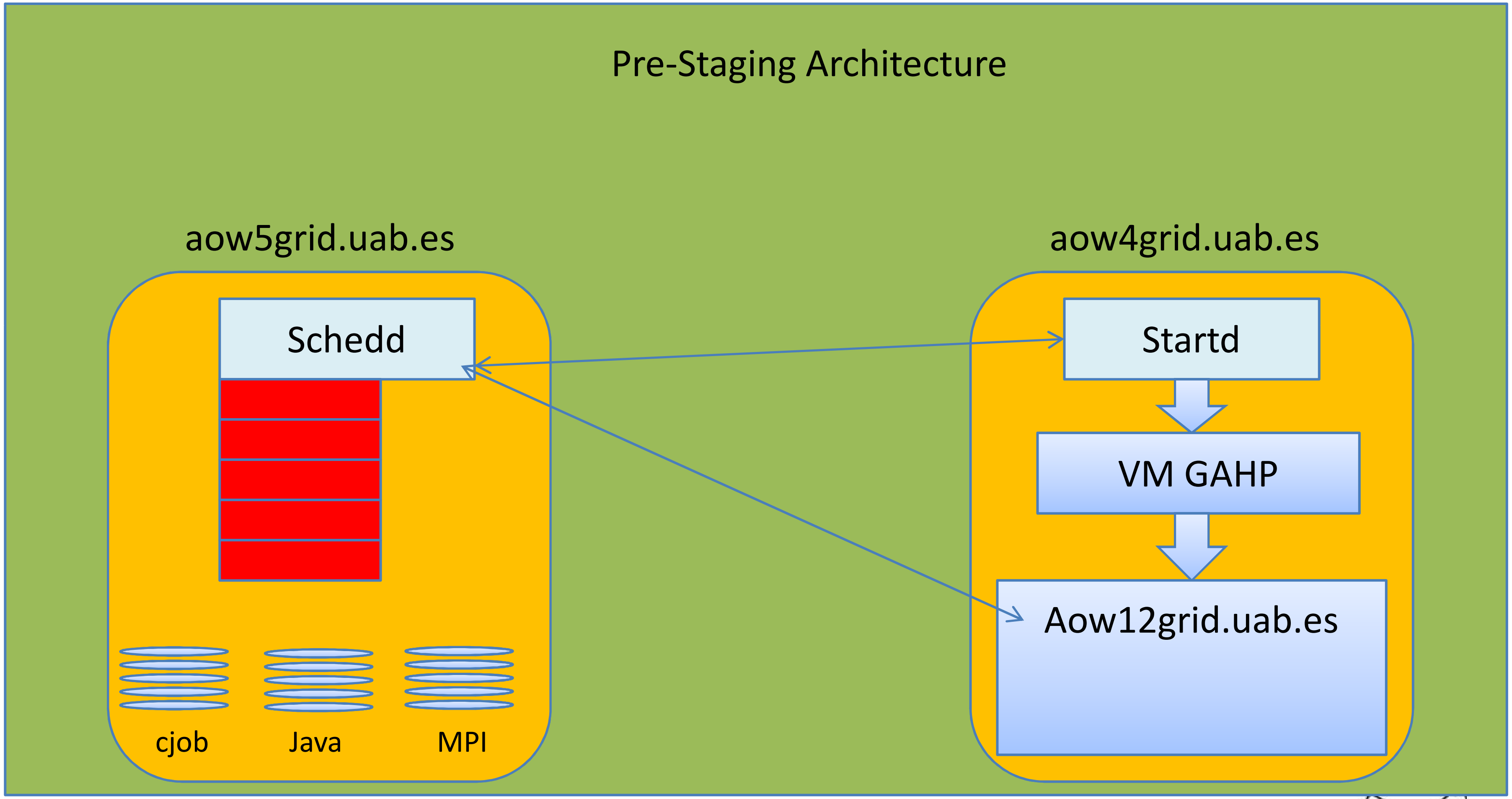
Pre-Staging Model Implementation

- At this stage we can clearly see that there are only two machines in our pool
- Because the Virtual Machine which we have submited takes some time to boot and join the pool
- After 13 minutes we again check the status of condor pool by the command *condor_status*

```
[condor@aow5grid~]$ condor_status
Name              OpSys    Arch    State      Activity    LoadAv   Mem      ActvtyTime
Aow5grid.uab.es   LINUX    INTEL  Unclaimed  Idle         0.020    502      0+00:25:31
Aow4grid.uab.es   LINUX    INTEL   Claimed    Busy         0.970    512      0+00:14:10
Aow12grid.uab.es LINUX     INTEL  Unclaimed  Idle          0.035   164      0+00:00:05
```

Universitat Autònoma de Barcelona

Pre-Staging Model Implementation



Pre-Staging Architecture

Pre-Staging Model Implementation

- We can clearly see that the Virtual Machine is in the pool now
- The Virtual Machine aow12grid.uab.es is now read to receive jobs and execute them
- Condor in Virtual Machine can gather information from host machine
  - E.g. load average, keyboard idle time
- We submit our jobs to the Virtual Machine by inserting custom ClassAd attributes into a job via the submit file like
  - Machine = “aow12grid.uab.es"

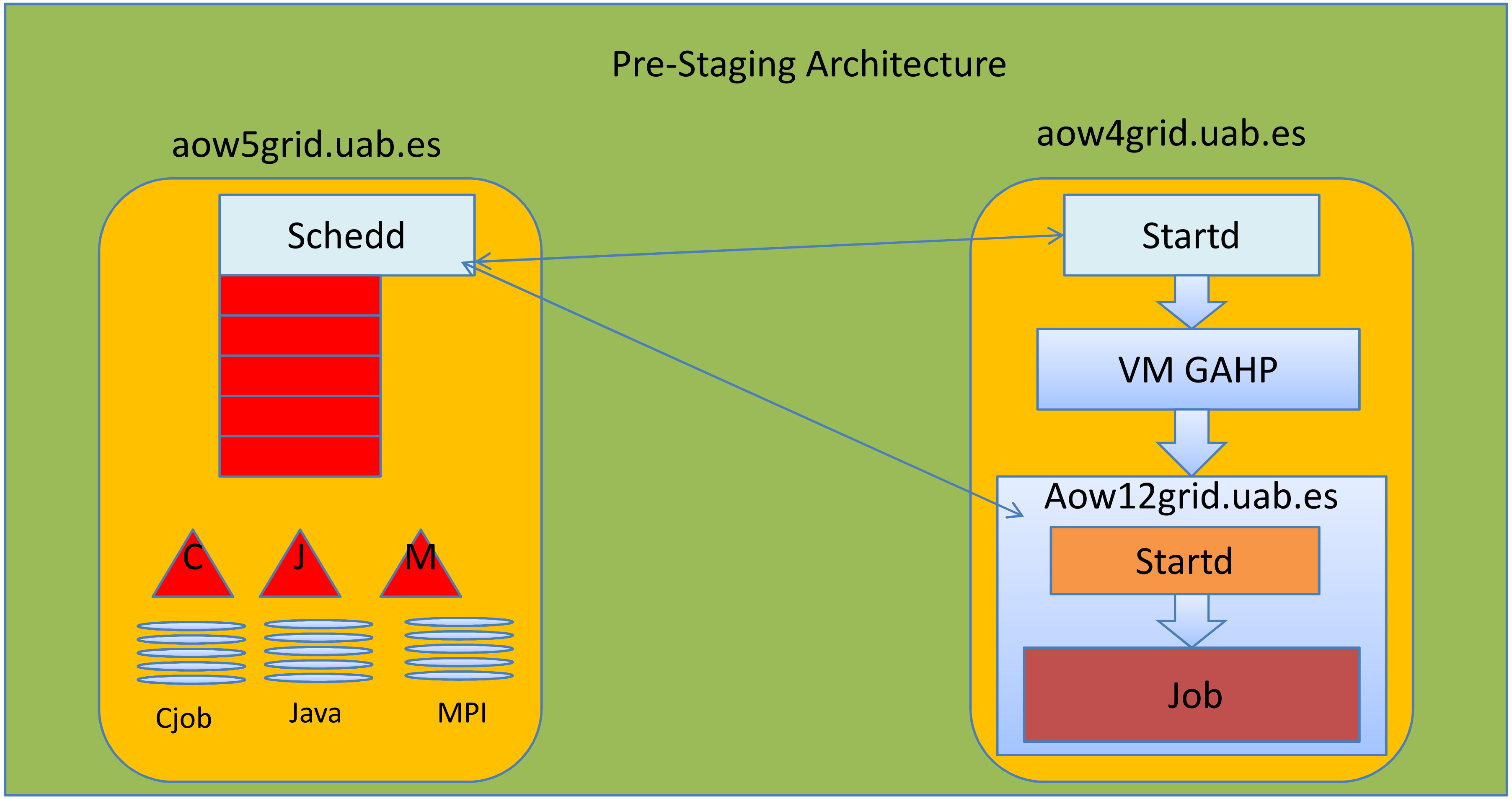
Pre-Staging Model Implementation

- Now that the Virtual Machine aow12grid is waiting in the pool
- We submit jobs to the Virtual Machine
  - ➢ [condor@aow5grid~]$ Condor_submit submit1

```
Universe                    = vanilla
Executable                  = C-application
Log                         = C-application.log
Output                      = C-application.out
Error                       = C-application . error
Requirements                = (Machine == "aow12grid.uab.es")
should_transfer_files       = YES
when_to_transfer_output     = ON_EXIT
Queue
```

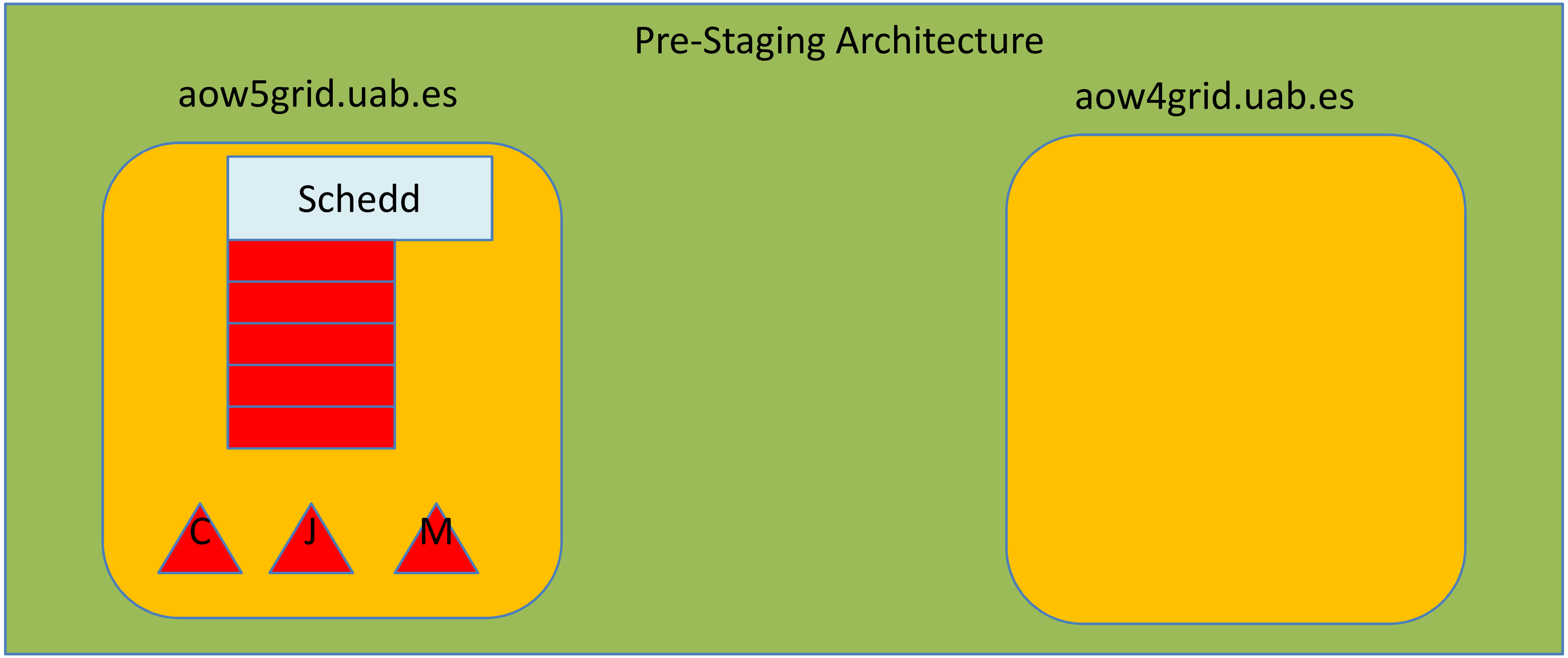
Universitat Autònoma de Barcelona

Pre-Staging Model Implementation



Pre-Staging Model Implementation

- After executing the jobs we want, we kill the Virtual Machine by just killing the Condor VM Universe job by using the command *Condor_rm*



Pre-Staging Model Implementation

- Once again we check the status of the pool to verify whether the Virtual machine is on the pool or not using *condor_status* command

```
[condor@aow5grid~]$ condor_status
Name             OpSys    Arch    State      Activity    LoadAv   Mem      ActvtyTime
Aow5grid.uab.es  LINUX    INTEL  Unclaimed  Idle        0.020    502      0+01:10:06
Aow4grid.uab.es  LINUX    INTEL   Claimed    Busy       0.940    512      0+00:59:27
```

- We can clearly see that the Virtual Machine is removed from the pool
- Thus we can execute multiple jobs using same Virtual Machine

Results

- Experimentation set 1

➢ **Submit Machine**

Aow5grid.uab.es

OS Fedora 9.

Memory 502.5 MiB

Processor: Intel(R) Pentium(R) 4 CPU 1.8GHz

Condor version-7.4.4

Vmware server 1.0.1

➢ **Execute Machine**

Aow5grid.uab.es

OS Fedora 9.

Memory 1.5 GiB.

Processor: Intel(R) Pentium(R) 4 CPU 1.8 GHz

Condor version-7.4.4

Vmware server 1.0.1

➢ **Virtual Machine Image**

Aow12grid.uab.es

OS Fedora 9.

Memory: 157.9 MiB.

Processor: Intel(R) Pentium(R) 4 CPU 1.8GHz

Condor version-7.4.4

➢ **Jobs**

C-application

Java application

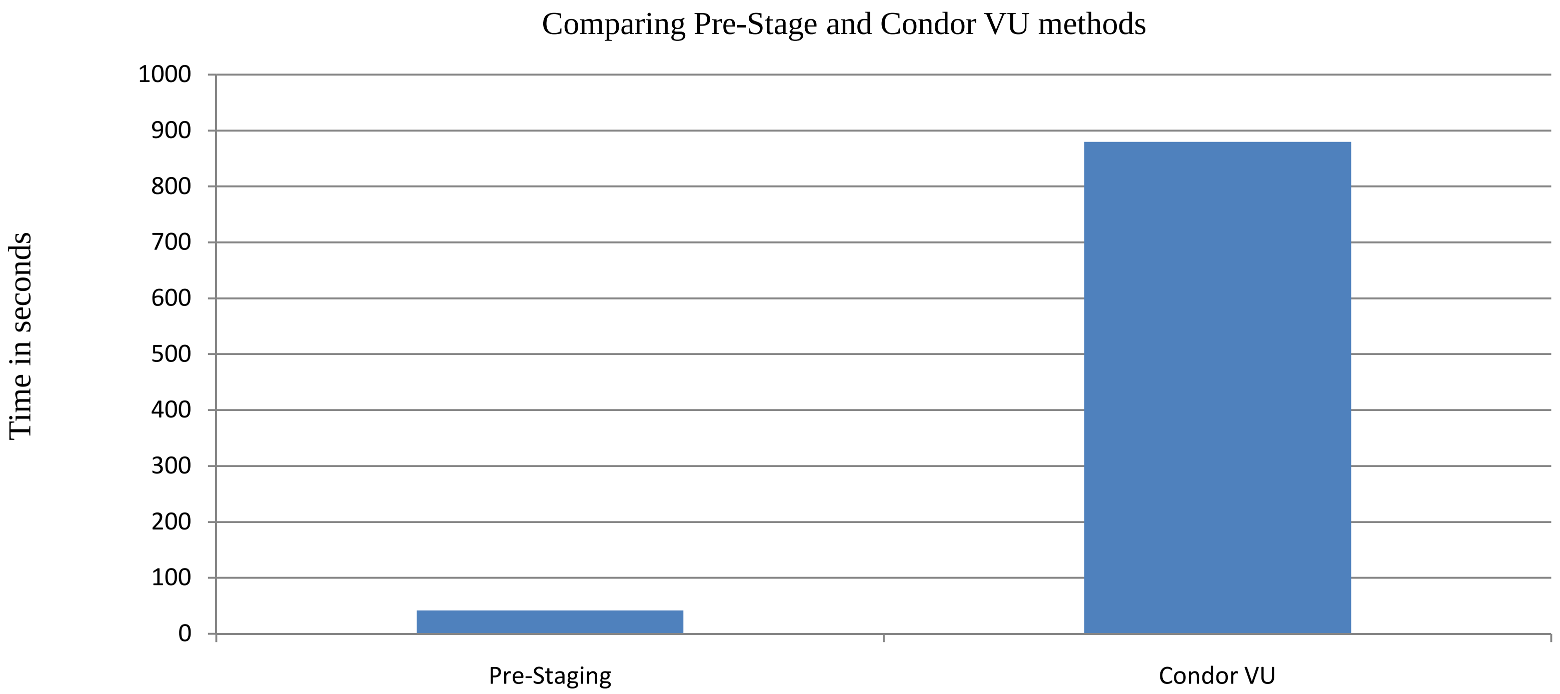
MPI application

Results

- By using the experimentation set 1 we execute all the three types of jobs using the Condor VM Universe for 15 times and take note of all the executions and make an average of each job
- We also execute all the three types of jobs by Pre-Staging Model for 15 time and make an average of execution times of each job

Type of Job	Pre-Staging Model	Condor VM Model
C-Language Application	42s	880s
Java Application	57s	972s
MPI Application	112s	1117s

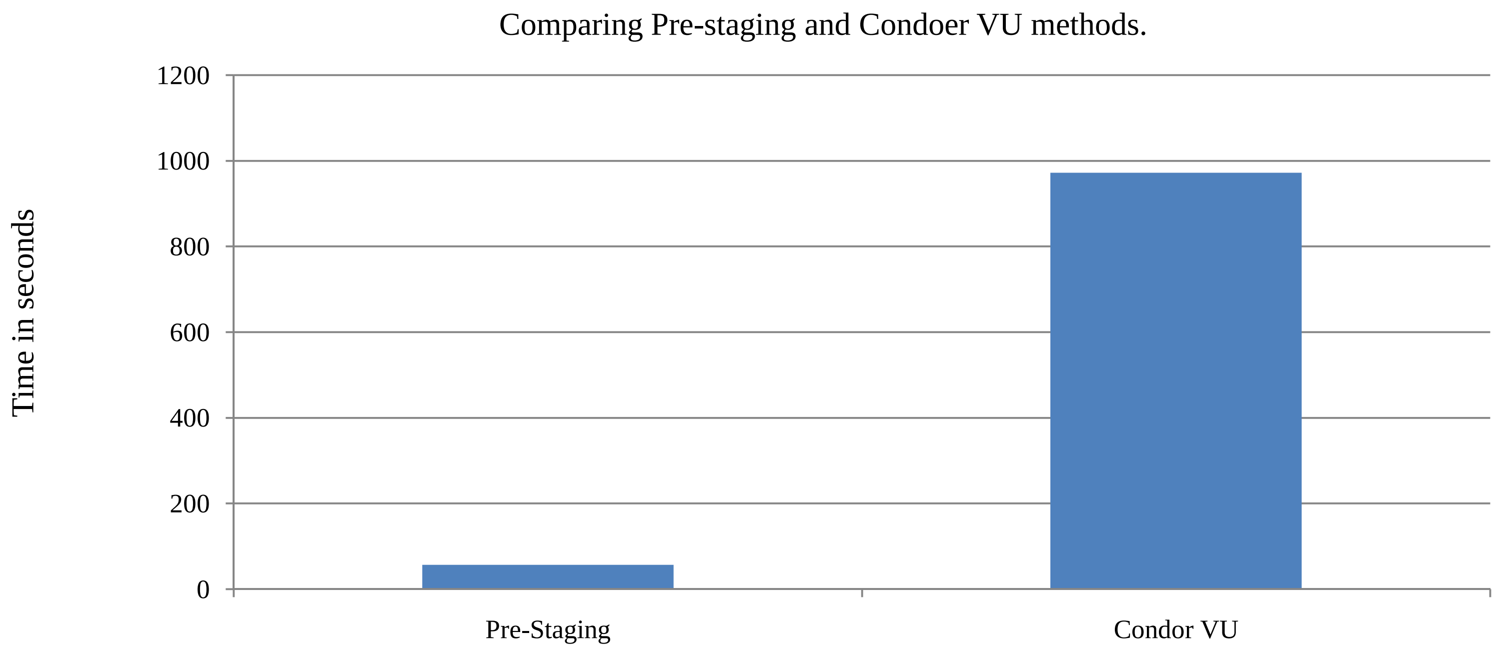
Results



- For C-Language Application we observe that by using the Pre-Staging Model the application gains nearly 21 times in execution time.

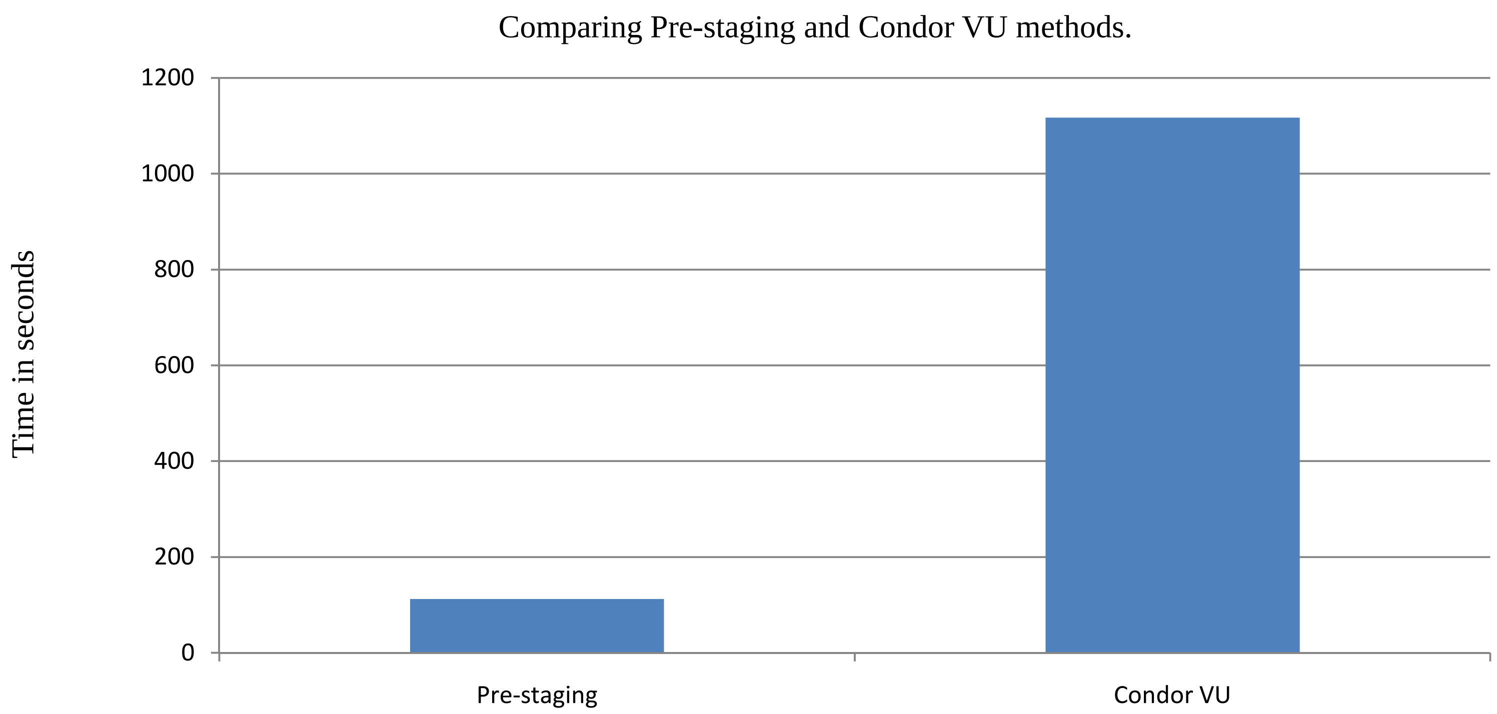
Universitat Autònoma de Barcelona

Results



- The Java application gains 17 times in comparison with Condor Virtual Machine Universe by the Pre-Staging Model.

Results



- The MPI application gains nearly 10 times in comparison to the Condor Virtual Machine Universe

Results

- Experimentation set 2

➢ **Submit Machine**

Aow5grid.uab.es

OS Fedora 9.

Memory 502.5 MiB

Processor: Intel(R) Pentium(R) 4 CPU 1.8GHz

Condor version-7.4.4

Vmware server 1.0.1

➢ **Execute Machine**

Aow5grid.uab.es

OS Fedora 9.

Memory 1.5 GiB.

Processor: Intel(R) Pentium(R) 4 CPU 1.8 GHz

Condor version-7.4.4

Vmware server 1.0.1

➢ **Virtual Machine Image**

Aow12grid.uab.es

OS Ubuntu10.10.

Memory: 264 MiB.

Processor: Intel(R) Pentium(R) 4 CPU 1.8GHz

Condor version-7.4.4

➢ **Jobs**

C-application

Java application

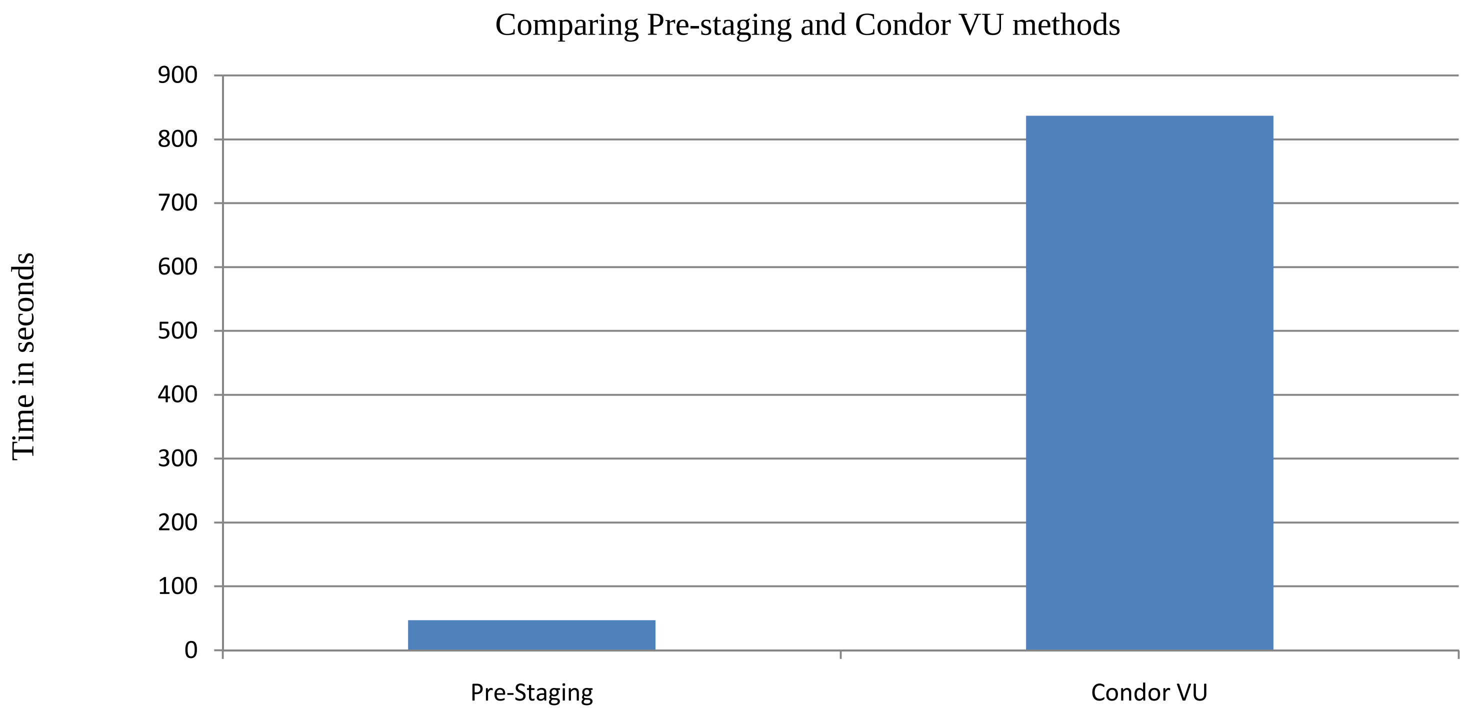
MPI application

Results

- By using the experimentation set 2 we execute all the three types of jobs using the Condor VM Universe for 15 times and take note of all the executions and make an average of each job
- We also execute all the three types of jobs by Pre-Staging Model for 15 time and make an average of execution times of each job

Type of Job	Pre-Staging Model	Condor VM Model
C-Language Application	47s	837s
Java Application	68s	937s
MPI Application	132s	1289s

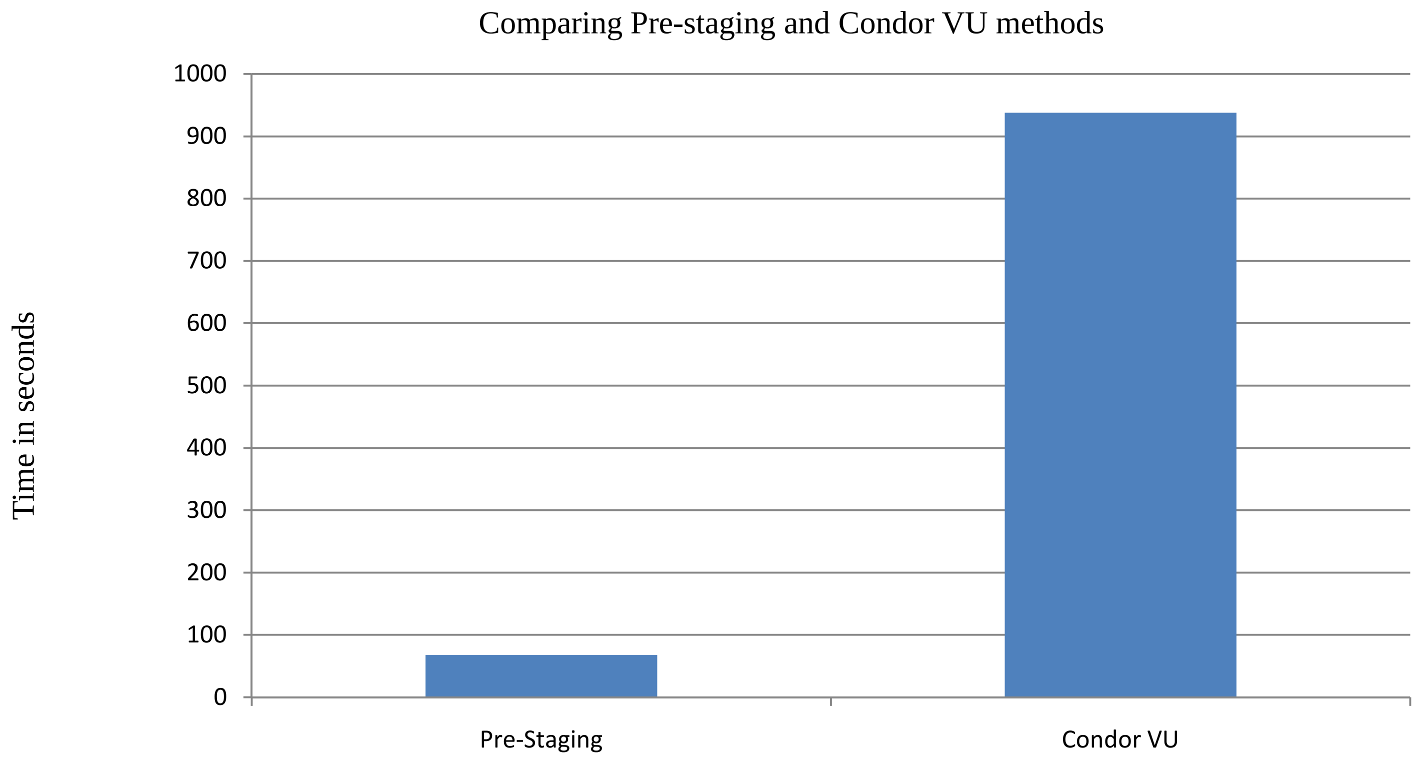
Results



- For C-Language Application we observe that by using the Pre-Staging Model the application gains nearly 17 times in execution time.

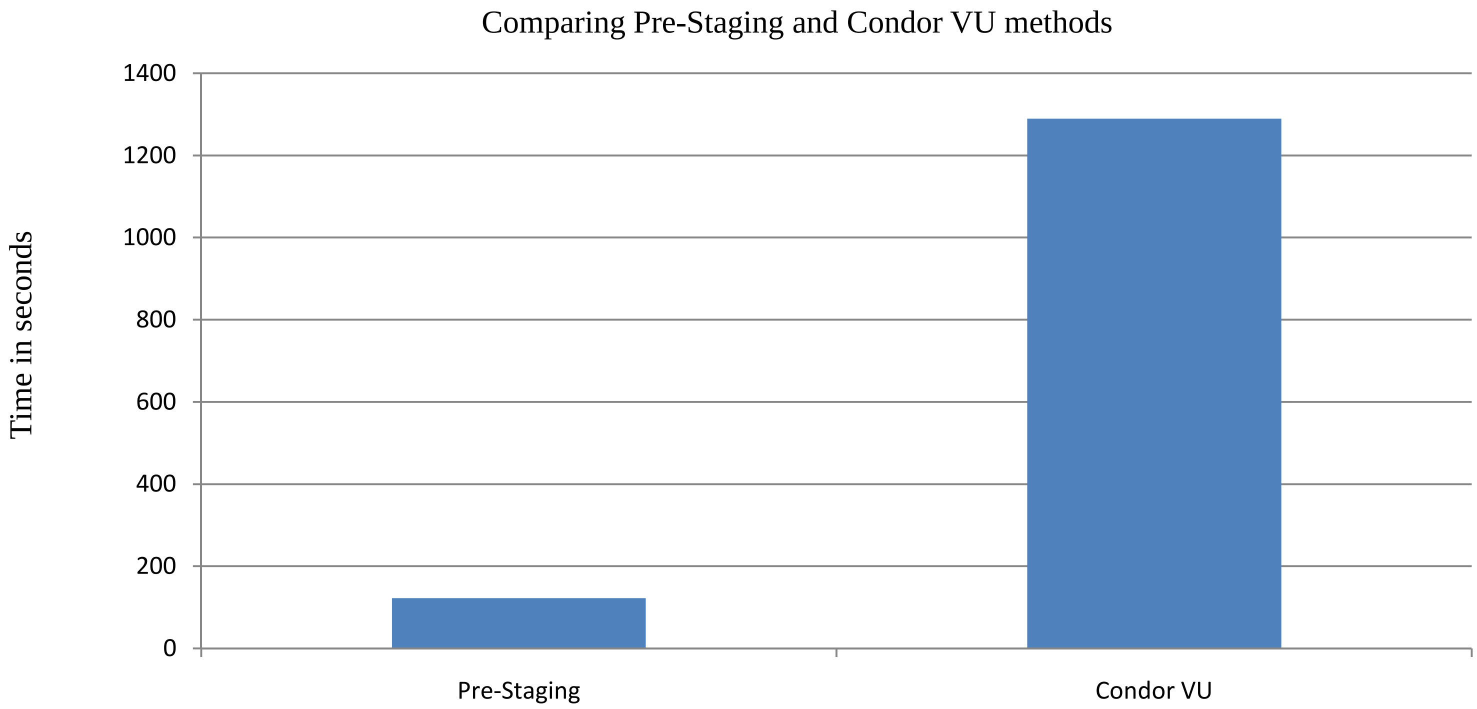
Universitat Autònoma de Barcelona

Results



- The Java application gains 14 times in comparison with Condor Virtual Machine Universe by the Pre-Staging Model.

Results



- The MPI application gains nearly 10 times in comparison to the Condor Virtual Machine Universe

Concluding Remarks

- Pre-Staging Model provides performance improvement for C and Java applications
- Performance improvement is decreased a bit for MPI application
- Pre-Staging Model facilitates reusability of Virtual Machines
- Condor Virtual Machine Universe is tedious and complicated
- Pre-Staging Model is simple and easy to use

Open Lines

- Replicating the Pre-Staging Model on a larger Scale like Virtual Organization Clusters
- Implementing the Pre-Staging Model without using Independent IP addresses , when implemented on a larger scale each Virtual Machine Image requires a independent IP address
- Reducing the complexity of custom ClassAds
- Testing the model for other virtualization software like Xen , KVM, Virtual Box etc
- Testing the Model viability for other High-Throughput scientific Applications like DNA sequencing where large memory sizes are required

Thanks for your attention

Any questions?