# Operating a distributed IaaS Cloud for BaBar

Ian Gable

Randall Sobie, Ashok Agarwal, Patrick Armstrong, Andre Charbonneau, Kyle Fransham, Roger Impey, Colin Leavett-Brown, Michael Paterson, Wayne Podaima, Matt Vliet

University of Victoria, Victoria, Canada

National Research Council of Canada, Ottawa

ATLAS ADC Cloud Workshop 2011-05-19

University of Victoria NRC·CNRC

# Outline

- Motivation
  - HEP Legacy Data Project
  - CANFAR: Observational Astronomy
- System Architecture
  - Job flow
  - Credentials
  - Data access
- Some Operational Experience
- Summary

# Motivation

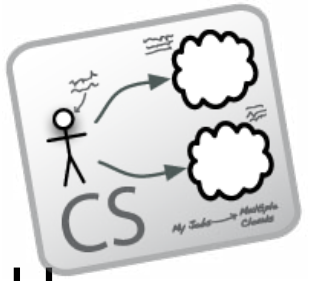
- Projects requiring modest resources we believe to be suitable to Infrastructure-as-a-Service (IaaS) Clouds:


  - The High Energy Physics Legacy Data project (2.5 FTEs). Target application BaBar MC and user analysis.

CANFAR
Canadian Advanced Network for Astronomical Research

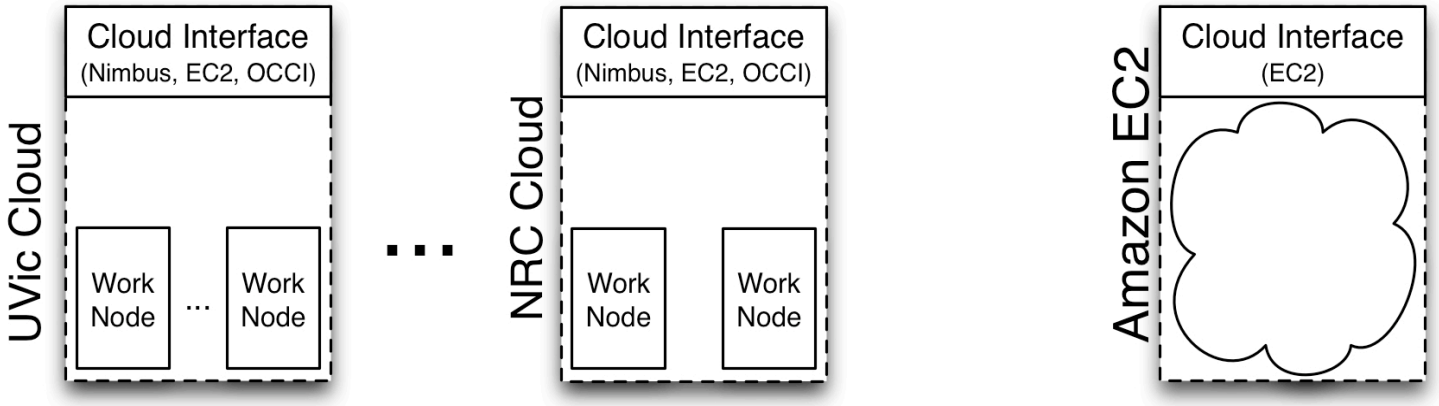
  - The Canadian Advanced Network for Astronomical Research (CANFAR). Supporting observational astronomy data analysis. Jobs are embarrassingly parallel like HEP( 1 image is like one event).
- We expect an increasing number of IaaS clouds to be available for research computing.

# Our Solution: Condor + Cloud Scheduler



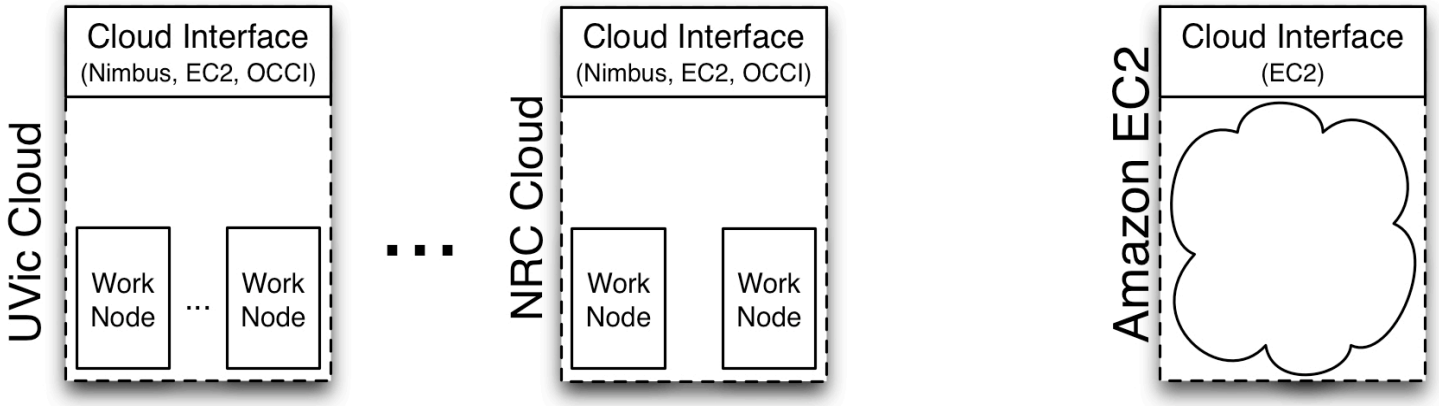

- Cloud Scheduler is a simple python package created by UVic and NRC to boot VMs on IaaS clouds based on Condor queues.
- Users create a VM with their experiment software installed
  - A basic VM is created by our group, and users add on their analysis or processing software to create their custom VM
- Users then create condor batch jobs as they would on a regular cluster, but they specify which VM should run their images.
- Aside from the VM creation step, this is very similar to the regular HTC workflow.

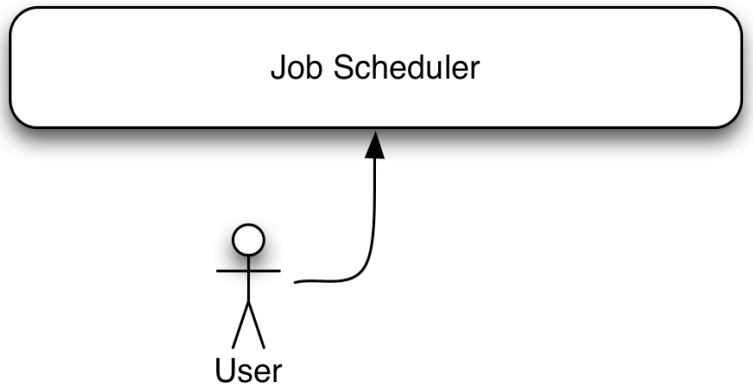
# Step 1




Research and Commercial clouds made available with some cloud-like interface.

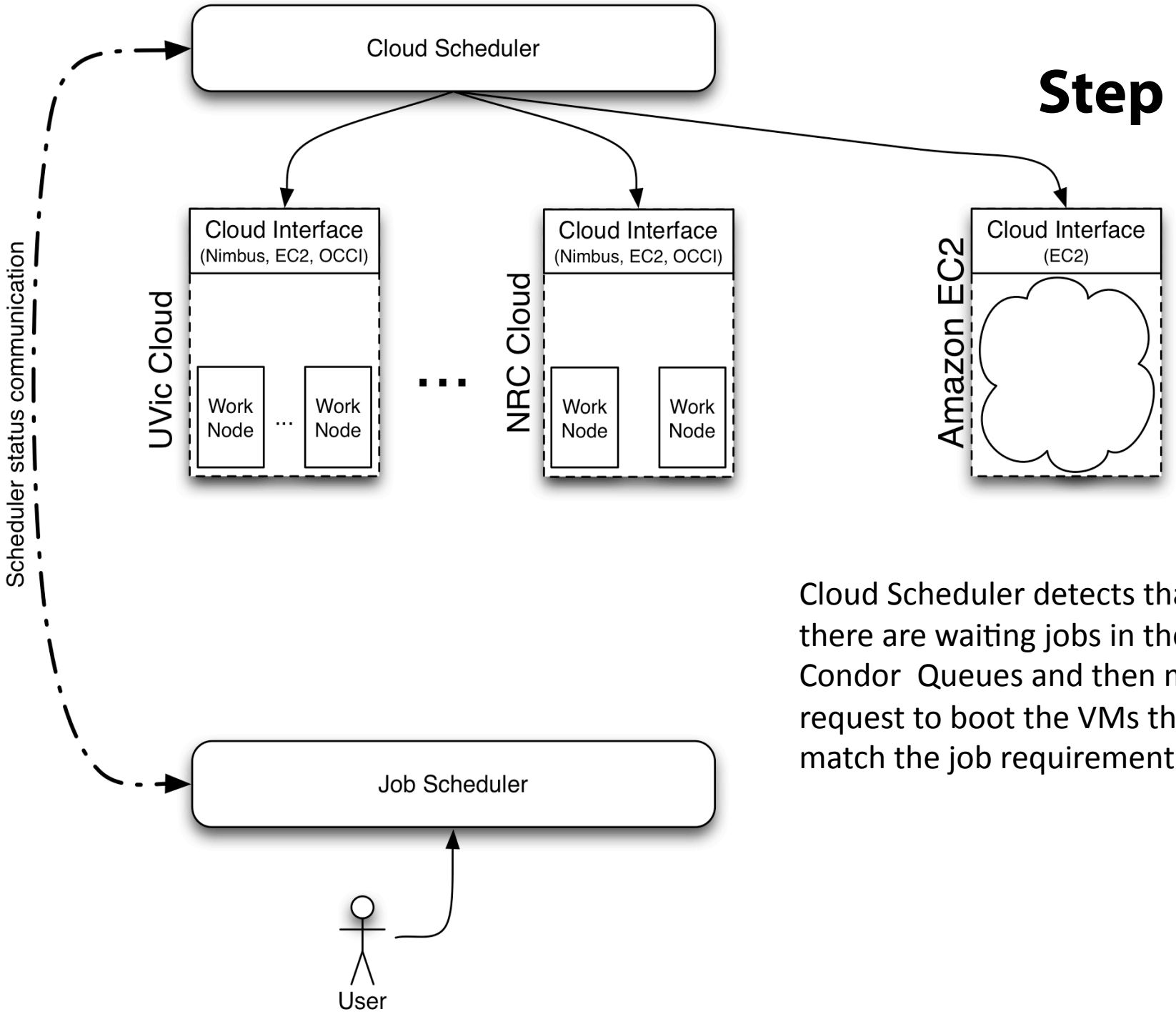
# Step 2



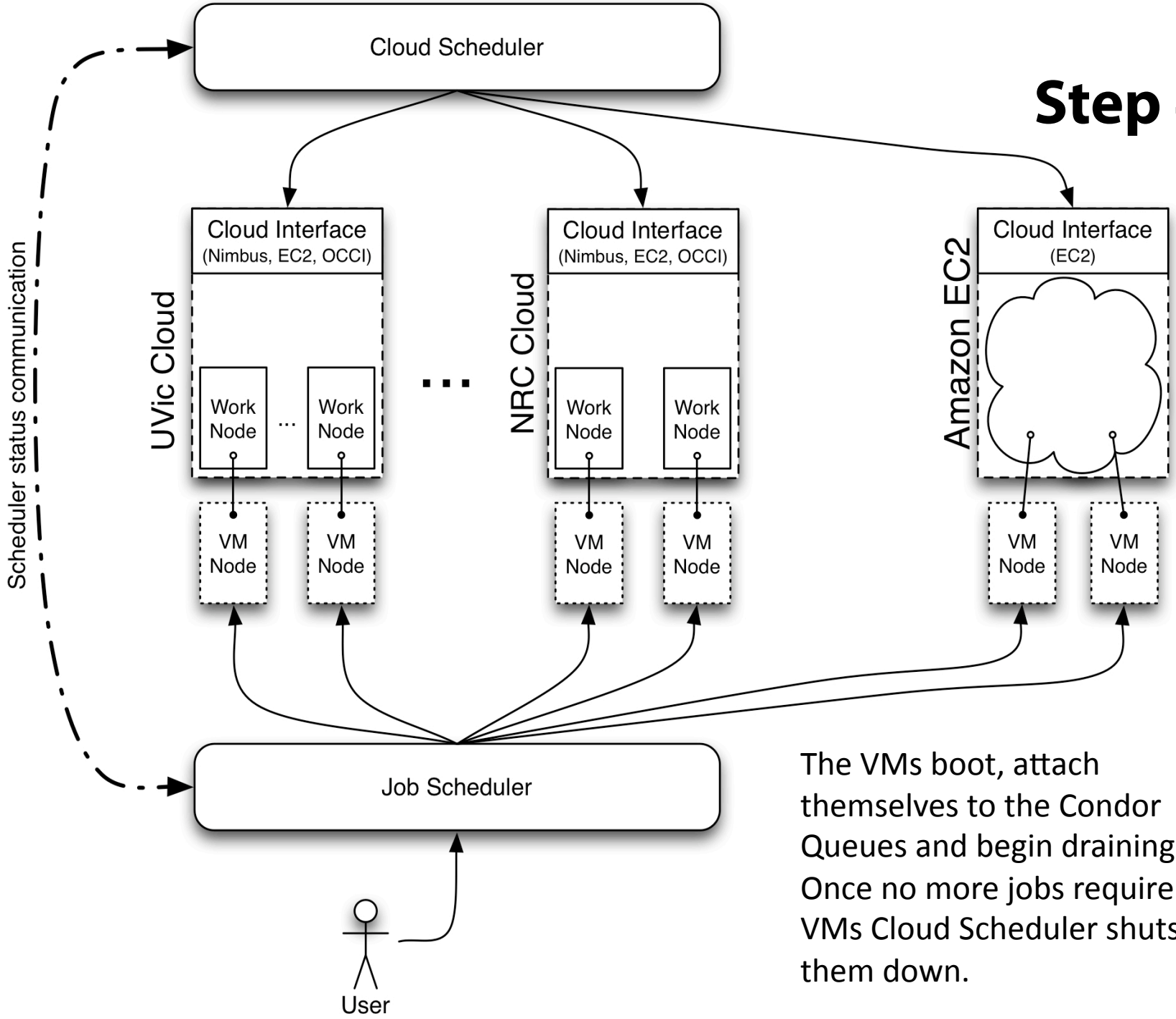

User submits to Condor Job scheduler that has no resources attached to it.

# Step 3




Cloud Scheduler detects that there are waiting jobs in the Condor Queues and then makes request to boot the VMs that match the job requirements.

# Step 4




The VMs boot, attach themselves to the Condor Queues and begin draining jobs. Once no more jobs require the VMs Cloud Scheduler shuts them down.

# Implementation Details

- Condor used as Job Scheduler
  - VMs contextualized with Condor Pool URL and service certificate.
  - Each VM has the Condor startd daemon installed, which advertises to the central manager at start.
  - GSI host authentication used when VMs joining pools
  - User credentials delegated to VMs after boot by job submission
  - Condor Connection broker to get around private IP clouds

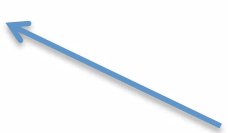
- Cloud Scheduler
  - User proxy certs used for authenticating with IaaS service where possible (Nimbus). Otherwise using secret API key (EC2 Style).
  - Can communicate with Condor using SOAP interface (slow at scale) or via condor_q
  - Primarily support Nimbus and Amazon EC2, with experimental support for OpenNebula and Eucalyptus
  - VMs reused until there are no more jobs requiring that VM type.
  - Images pulled via http or must preexist at the site.

# Condor Job Description File

```
Universe        =       vanilla
Log             =       SP-3429-Tau11-Run2-R24a3-3.l1341
Output          =       SP-3429-Tau11-Run2-R24a3-3.o1341
Error           =       SP-3429-Tau11-Run2-R24a3-3.e1341
Input           =       a52.tcl
should_transfer_files    =       YES
when_to_transfer_output  =       ON_EXIT


Requirements = VMType =?= “BaBarAnalysis-52”
+VMLoc           =       "http://vmrepo.heprc.uvic.ca/BabarAnaylysis-52"
+VMCPUArch       =       "x86_64"
+VMStorage       =       "1"
+VMCPUCores      =       "1"
+VMMem           =       "2555"
+VMAMI           =       "ami-64ea1a0d"
+VMInstanceType  =       "m1.small"
+VMJobPerCore    =       True
getenv           =       True

Queue
```

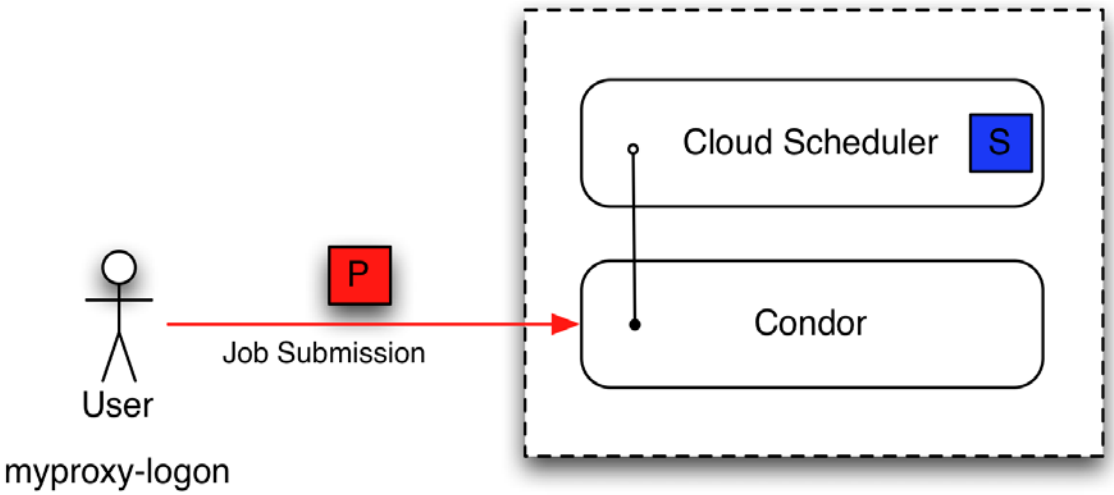


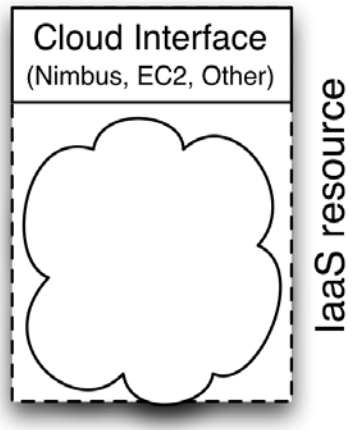
Custom condor attributes

# Credentials Movement

- User does a myproxy-logon
- User submits a condor job with proxy
- IaaS API request made to Nimbus with proxy
- Cluster service certificate is injected into the VM (could be replaced with user cert).
- VM boots and joins the condor pool using GSI authentication with the service certificate.
- Condor submits job to VM with delegated user proxy.

# Credentials Step 1

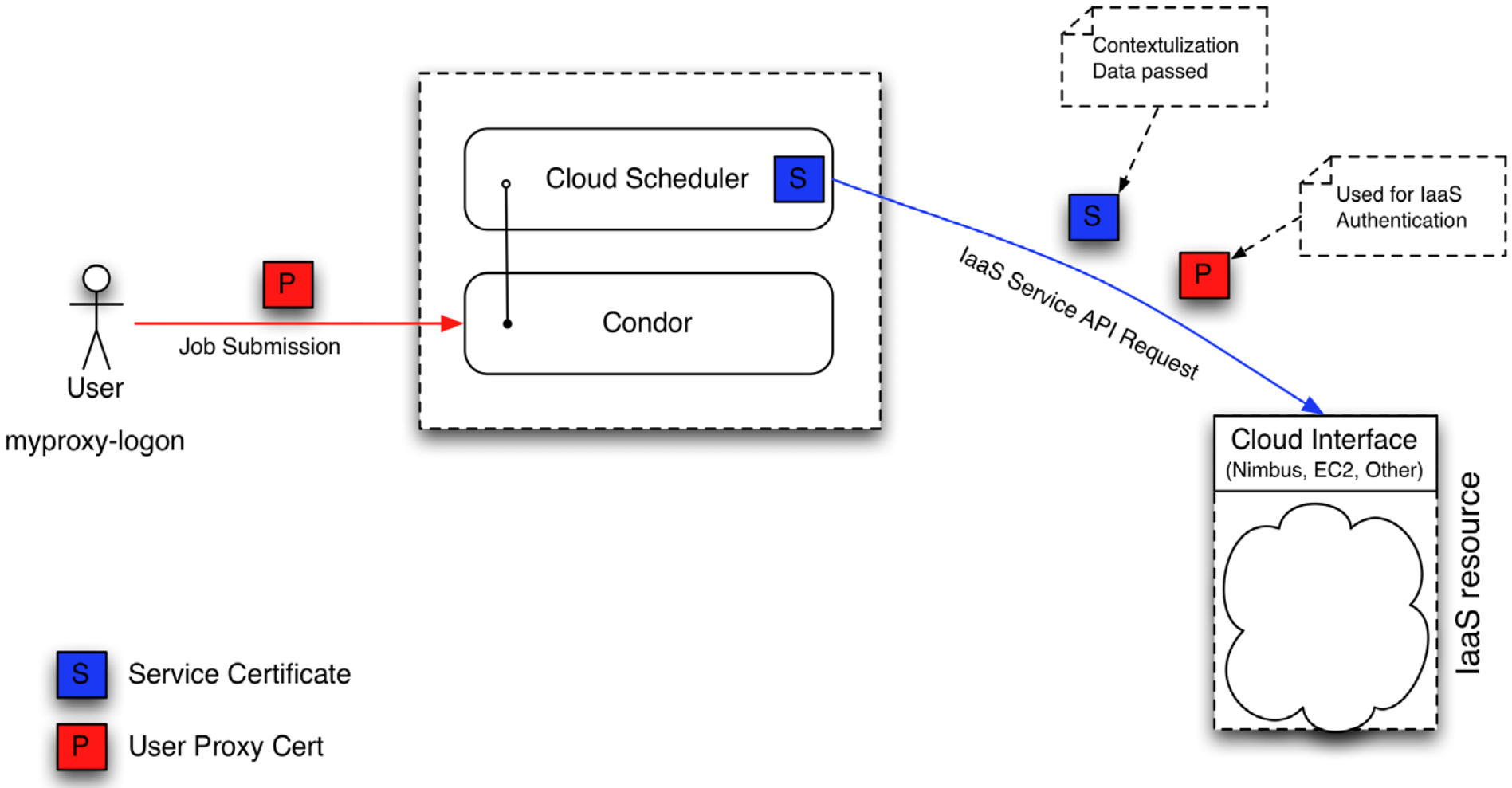


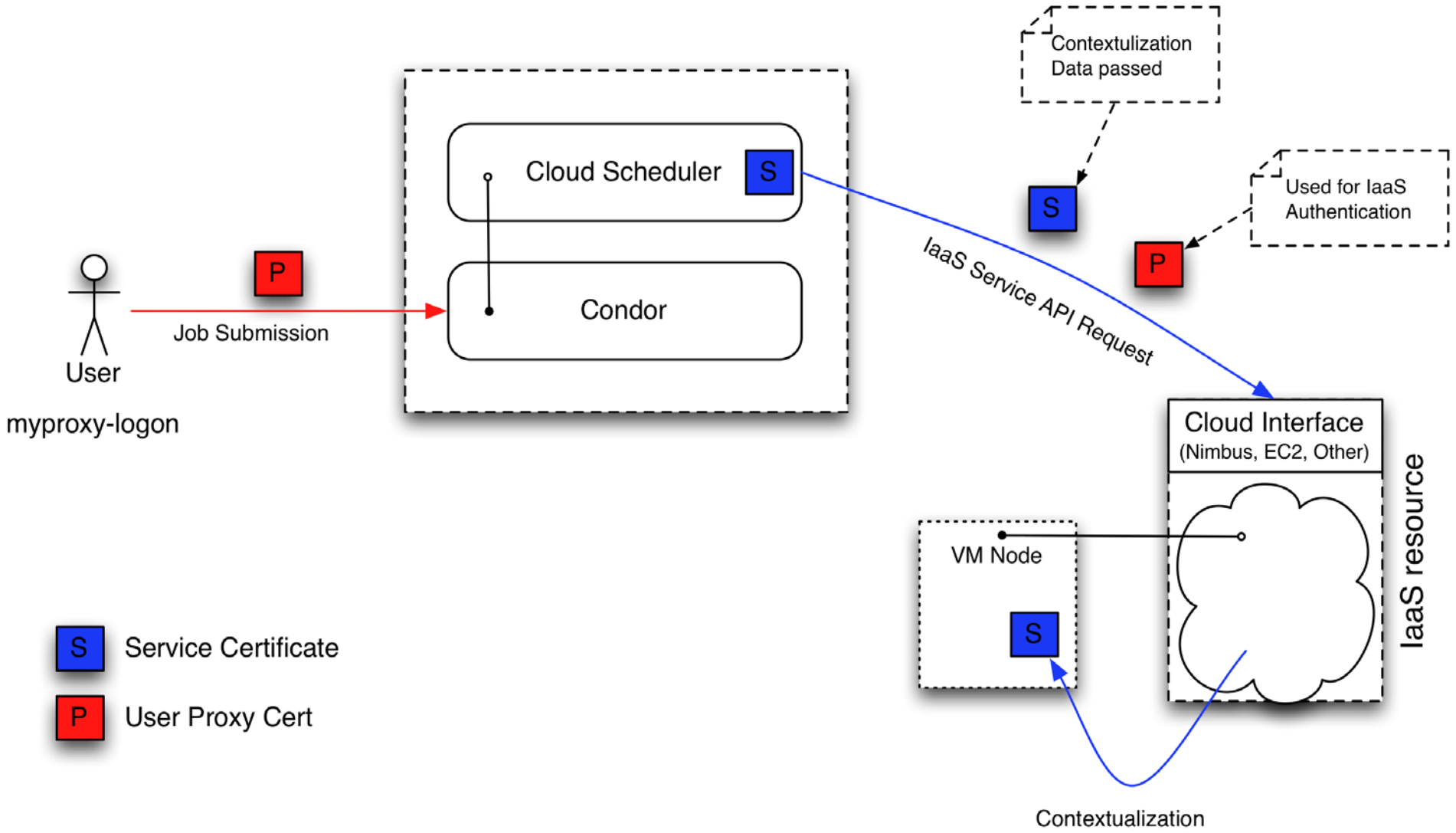

S Service Certificate

P User Proxy Cert

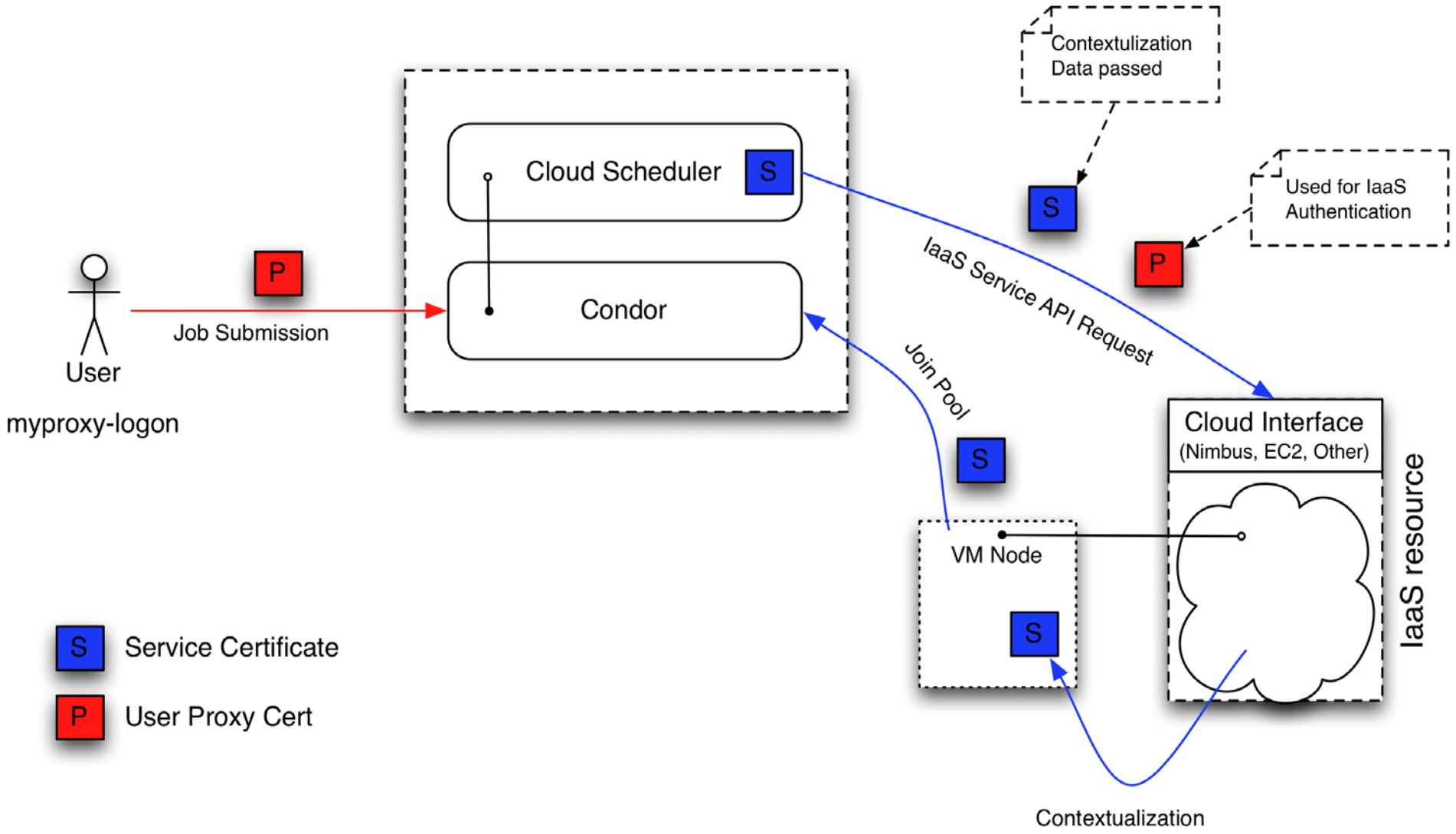
# Credentials Step 2



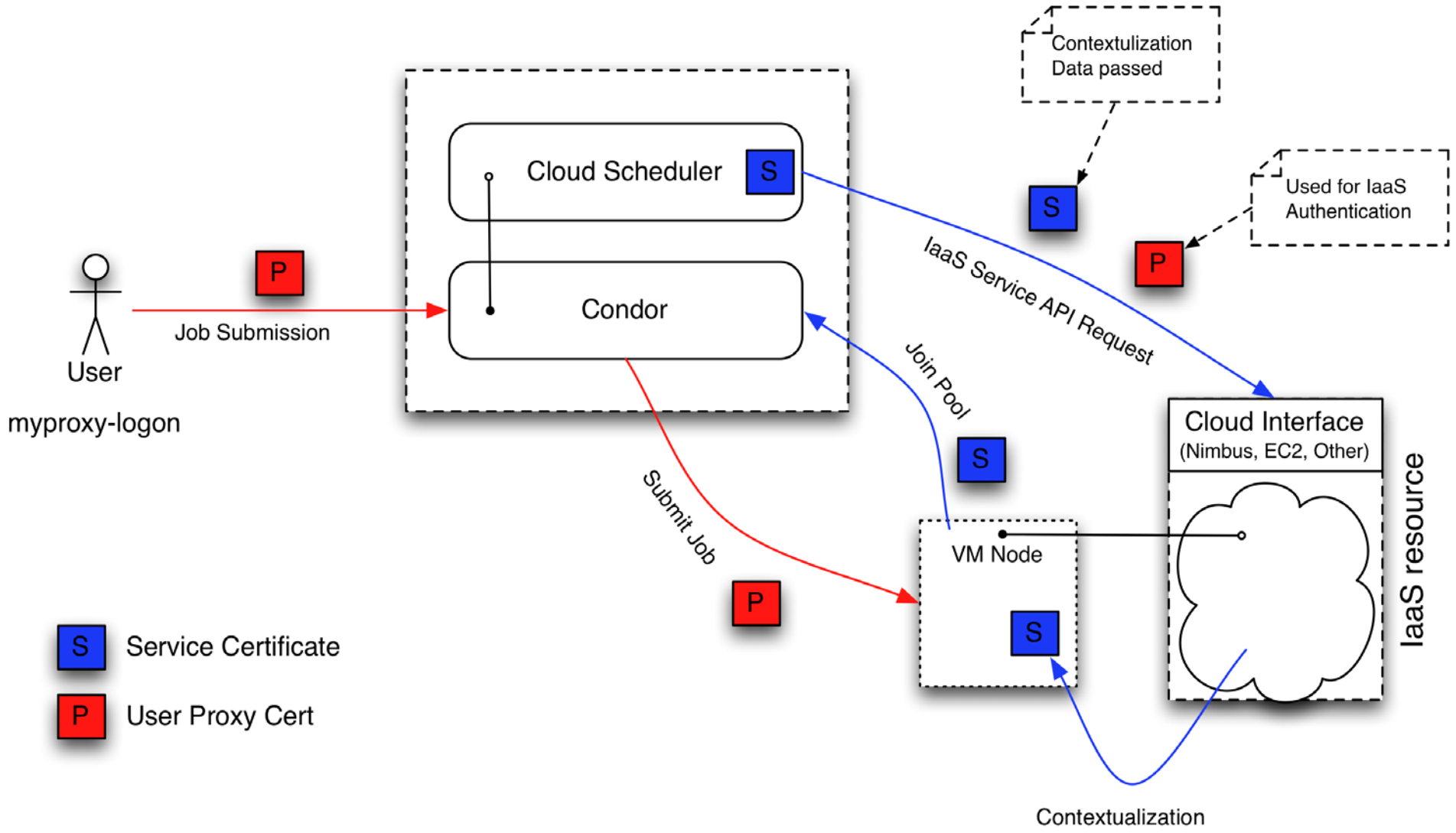
# Credentials Step 3



# Credentials Step 4



# Credentials Step 5



# Experimental BaBar Cloud Resources

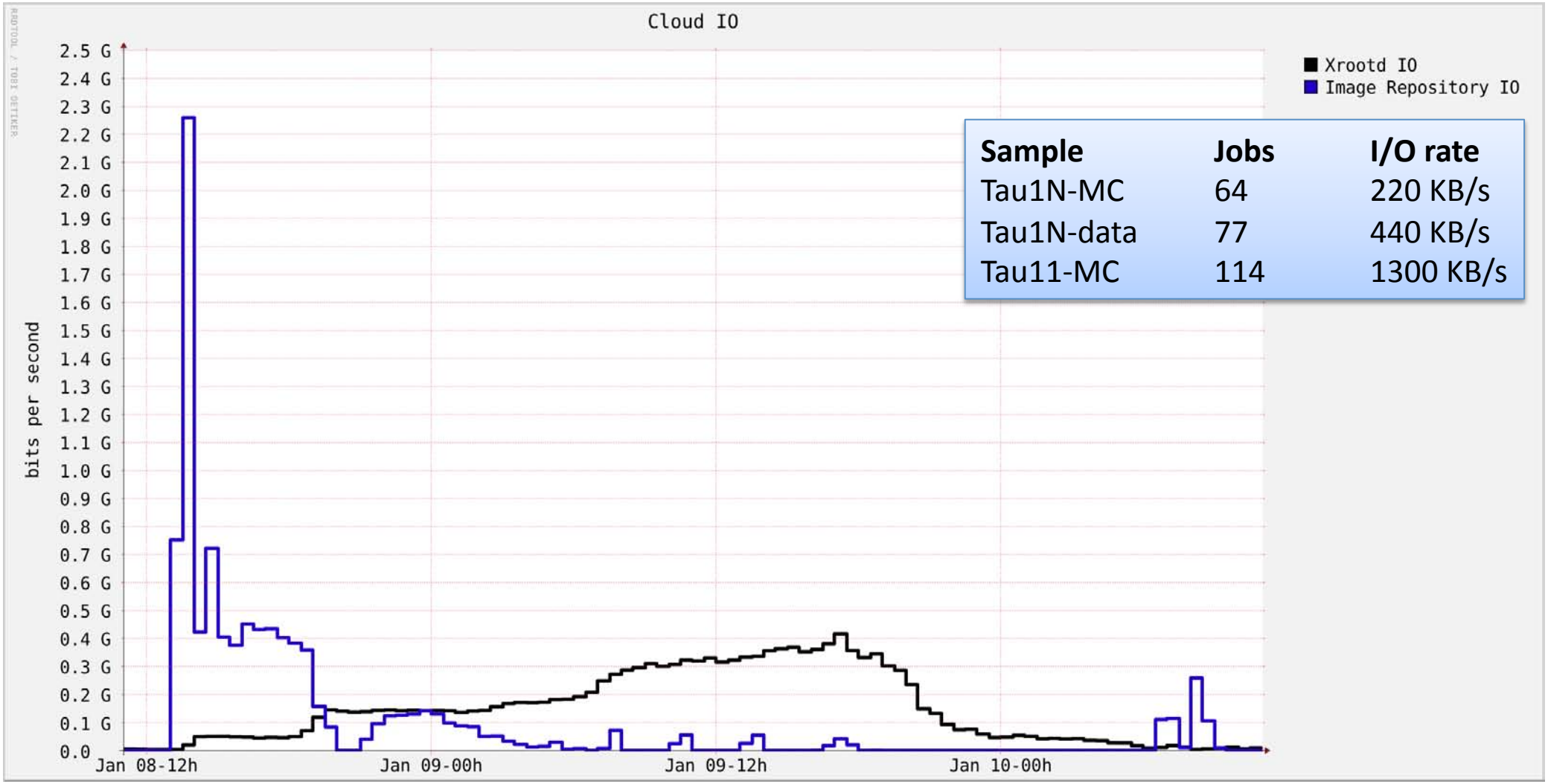
| Resource | Cores | Notes | Collaboration Certified |
|---|---|---|---|
| FutureGrid @Argonne Lab | 100 Cores Allocated | Resources allocation to support BaBar | ★ |
| Elephant Cluster @UVic | 88 Cores | Experimental cloud cluster hosts (xrootd for cloud) | ★ |
| NRC Cloud in Ottawa | 68 Cores | Hosts VM image repository (repoman) | ★ |
| Amazon EC2 | Proportional to $ | Grant funding from Amazon | ★ |
| Hermes Cluster @Uvic | Variable (280 max) | Occasional Backfill access | |

★ Certified for Monto Carlo Production by Babar Collaboration

# Data Access

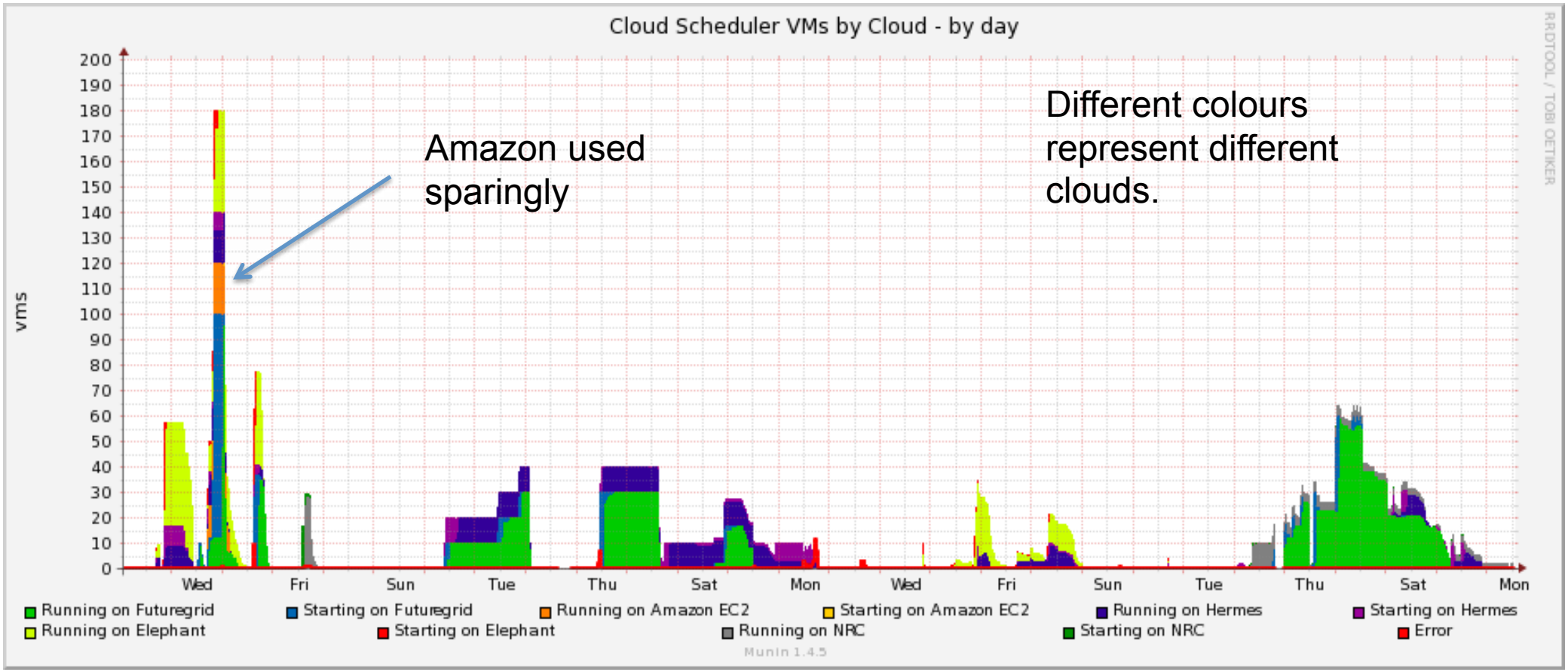
- Data access is the biggest challenge
- We are using XRootd for remote data access with mixed results.
  - Getting CPU efficiency from 90% to 30%.
  - Need a lot more work to understand how to optimize on a site by site-site basis (can we contextualize based on location?)
- Amazon EC2 remote data access to slow to be practical. Data sets stored in Amazon S3.
- Output moved back with GridFTP.

# Cloud I/O for BaBar User Analysis



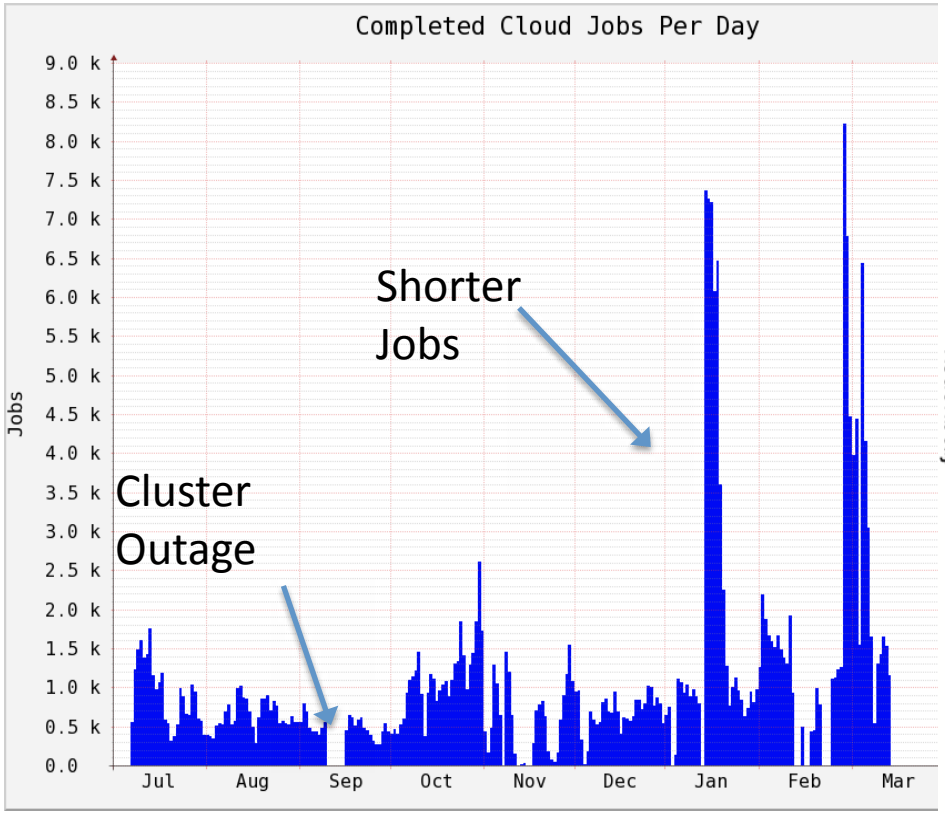

| Sample | Jobs | I/O rate |
|---|---|---|
| Tau1N-MC | 64 | 220 KB/s |
| Tau1N-data | 77 | 440 KB/s |
| Tau11-MC | 114 | 1300 KB/s |

# A Typical Week (Babar)

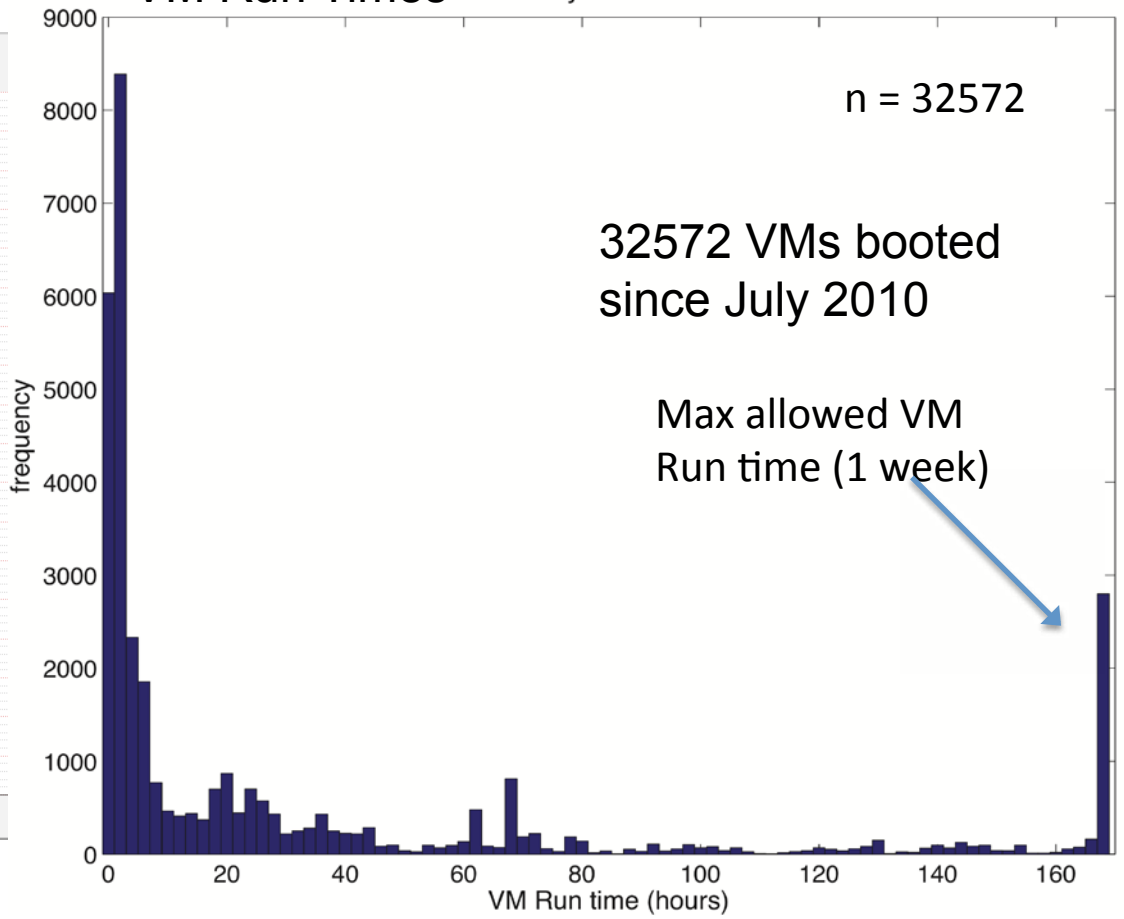


# CANFAR

Cloud Jobs




VM Run Times



# Some Lessons Learned

- Monitoring cloud resources from the site perspective is difficult
  - Resources are ephemeral hence hard to instrument and gather all the information you would normally expect.
- Debugging user VM problems is hard for users, and hard for support
  - What do you do when the VM network doesn't come up.
- No two EC2 API implementations are the same
  - Nimbus, OpenNebula, Eucalyptus all different
  - Each IaaS implementation has implemented a different subset of functionality and have their own quirks.
- Users insulated from cloud failures. If a cloud is failing to boot VMs then the job doesn't get pulled.

# Other Aspects

- CVMFS used for BaBar base software.
  - Reduces VM size by at least 10G
- Conditions database accessed remotely via XrootD
- HEPiX Virtualization working group
  - VM images exchange with CERN
  - Working on secure image exchange (images signed with grid certs)
- Contributing to Nimbus Development
- Working on BaBar Long Term Data Analysis System at SLAC , 'Cloud in a Box'

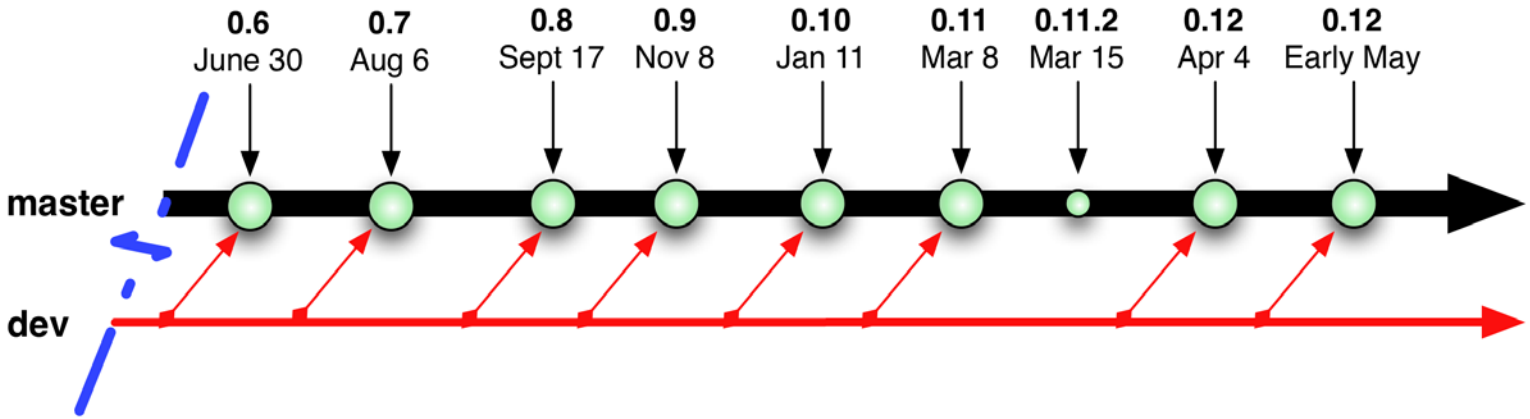
# Summary

- We have built a  distributed IaaS system for user analysis and MC simulation
  - Boots user customized VMs
- System in routine production now for CANFAR (Astronomy) and BaBar
- Working on I/O and scalability
- Lots of interesting challenges

# More Information

- Ian Gable <igable@uvic.ca>
- cloudscheduler.org
- Code on GitHub:
  - http://github.com/hep-gc/cloud-scheduler
  - Run as an open source project

Roadmap



# Acknowledgements


canarie

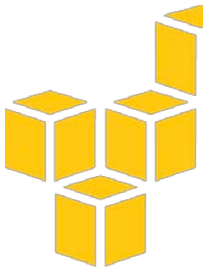
Canada's Advanced Research and Innovation Network
Le réseau évolué de recherche et d'innovation du Canada


University of Victoria

NRC·CNRC

CANFAR

Canadian Advanced Network for Astronomical Research



amazon web services™

# Beginning of Extra Slides

# About the code

```
Ian-Gables-MacBook-Pro:cloud-scheduler igable$ cat source_files |
xargs wc -l
       0 ./cloudscheduler/__init__.py
       1 ./cloudscheduler/__version__.py
     998 ./cloudscheduler/cloud_management.py
    1169 ./cloudscheduler/cluster_tools.py
     362 ./cloudscheduler/config.py
     277 ./cloudscheduler/info_server.py
    1086 ./cloudscheduler/job_management.py
       0 ./cloudscheduler/monitoring/__init__.py
      63 ./cloudscheduler/monitoring/cloud_logger.py
     208 ./cloudscheduler/monitoring/get_clouds.py
     176 ./cloudscheduler/utilities.py
      13 ./scripts/ec2contexthelper/setup.py
      28 ./setup.py
      99 cloud_resources.conf
    1046 cloud_scheduler
     324 cloud_scheduler.conf
     130 cloud_status
    5980 total
```

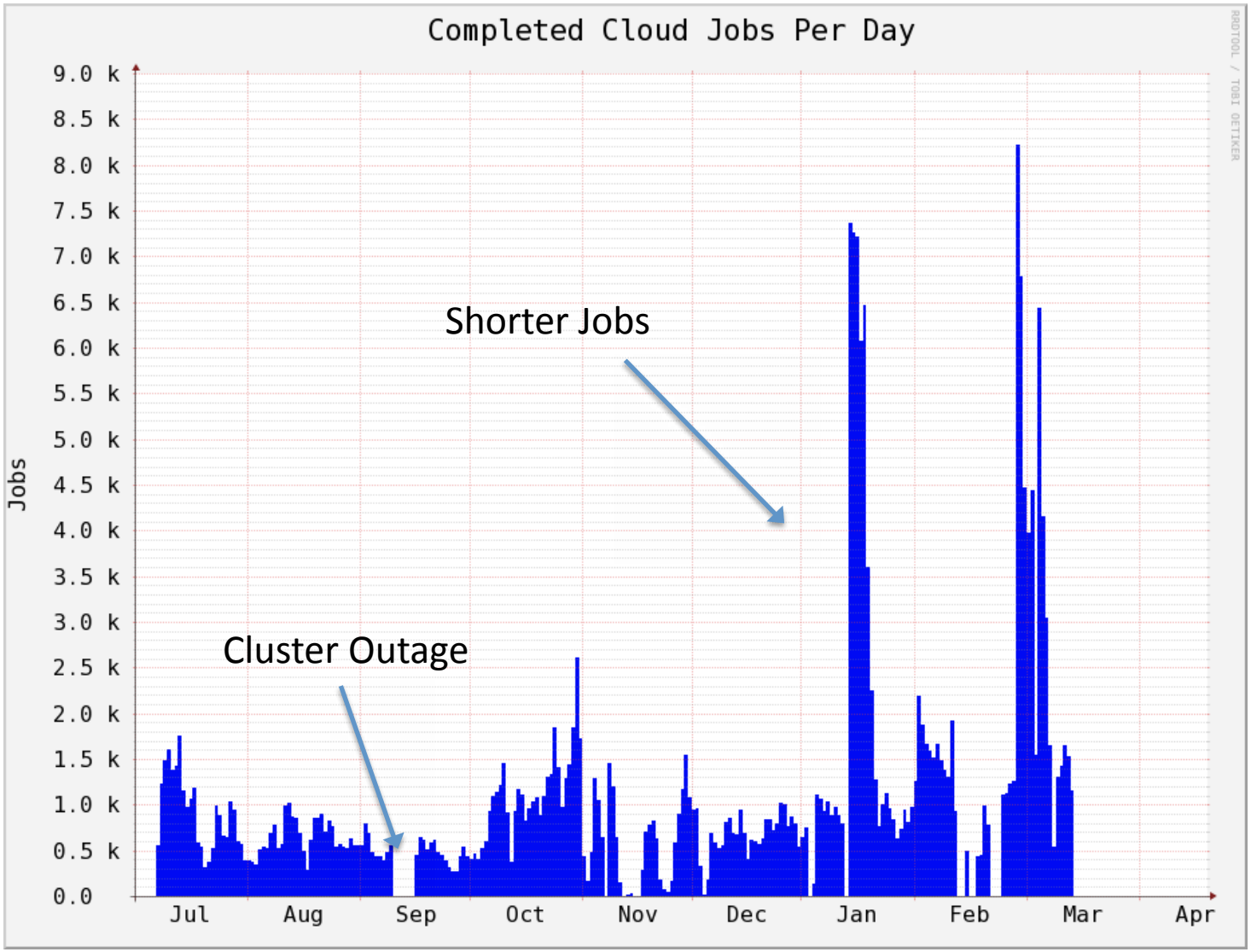
- Relatively small python package, lots of cloud interaction examples

http://github.com/hep-gc/cloud-scheduler

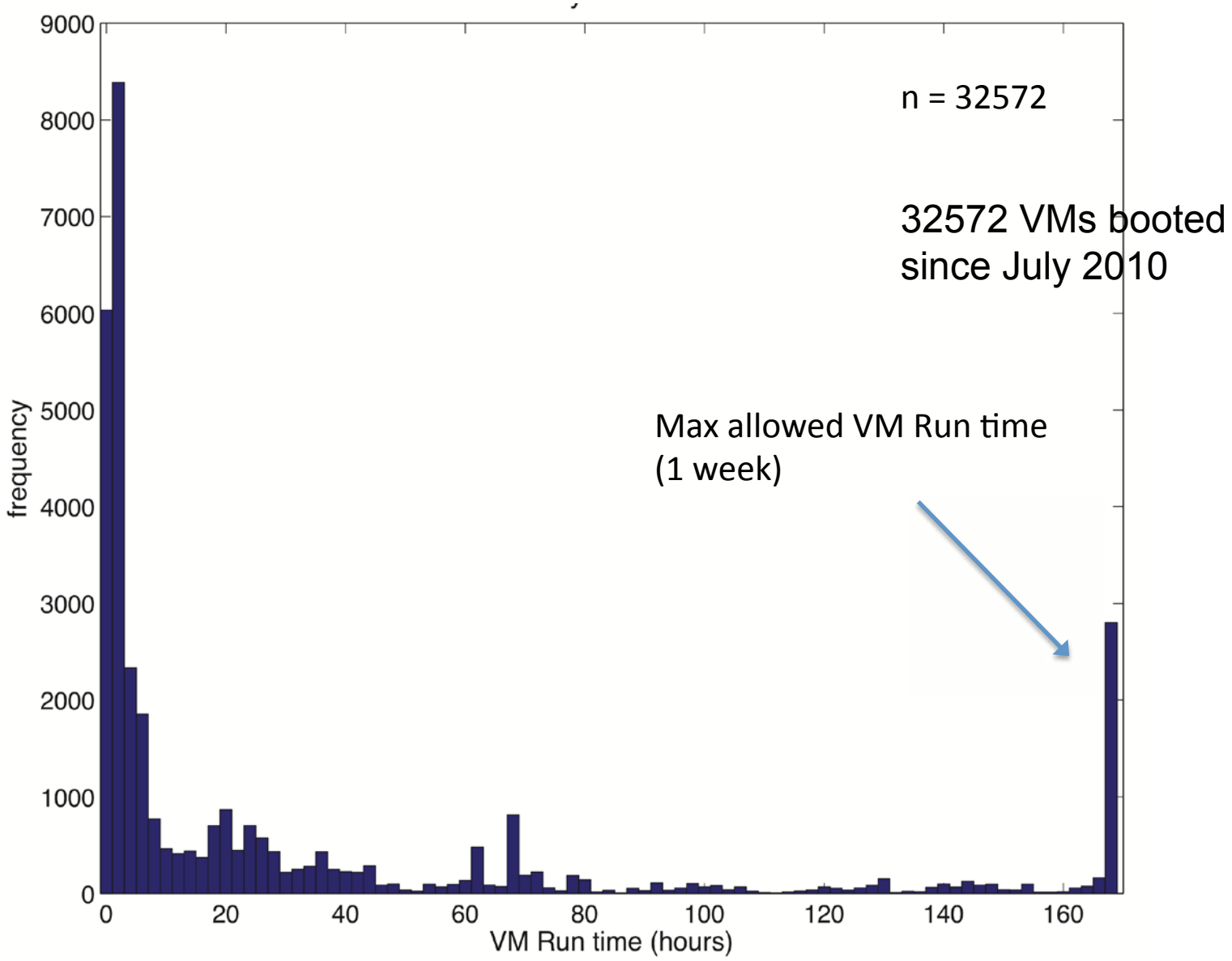
# HEP Legacy Data Project

- We have been funded in Canada to investigate a possible solution for analyzing BaBar data for the next 5-10 years.
- Collaborating with SLAC who are also pursuing this goal.
- We are exploiting VMs and IaaS clouds.
- Assume we are going to be able run BaBar code in a VM for the next 5-10 years.
- We hope that results will be applicable to other experiments.
- 2.5 FTEs for 2 years ends in October 2011.

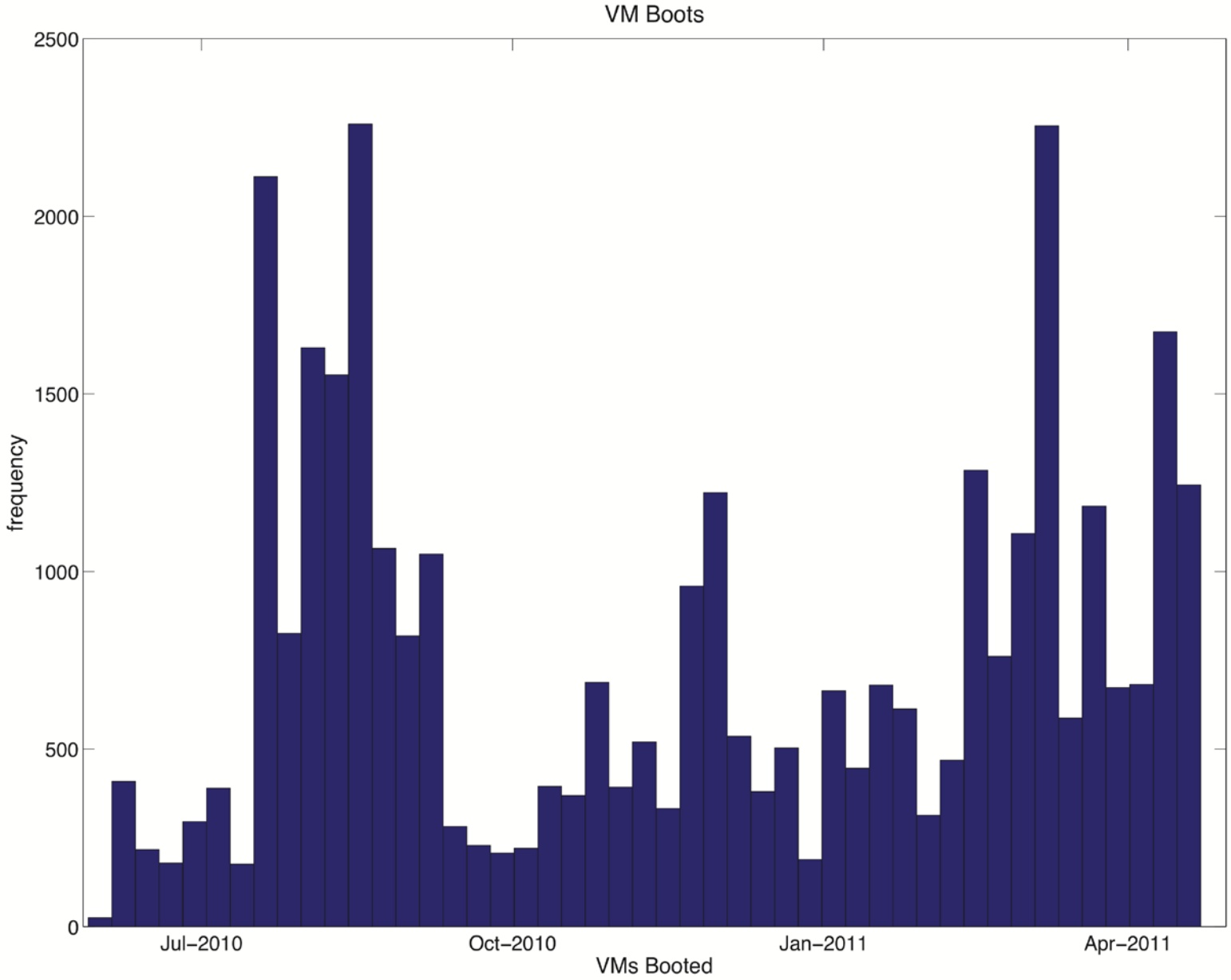
# Experience with CANFAR



# VM Run Times (CANFAR)



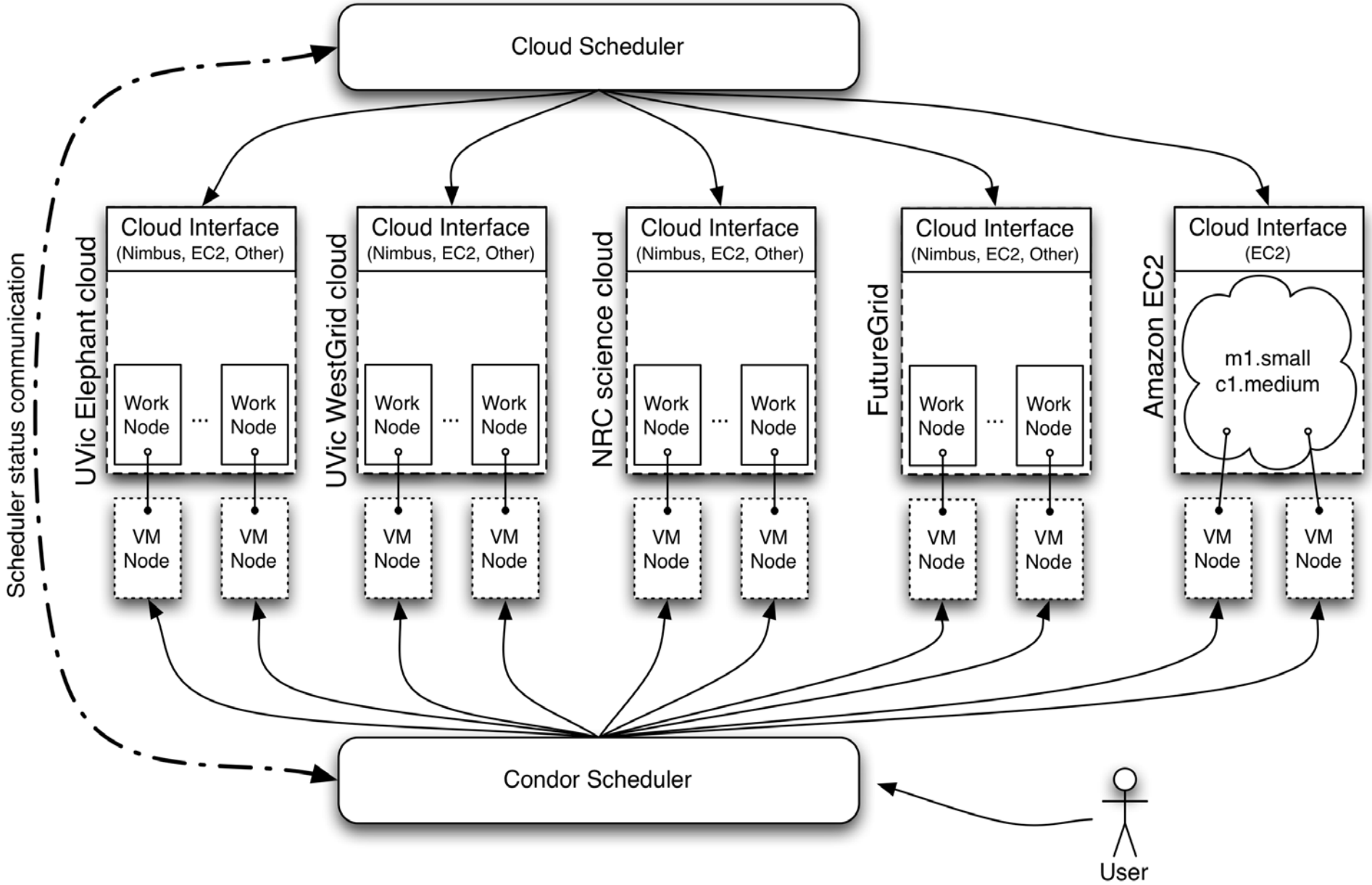
# VM Boots (CANFAR)



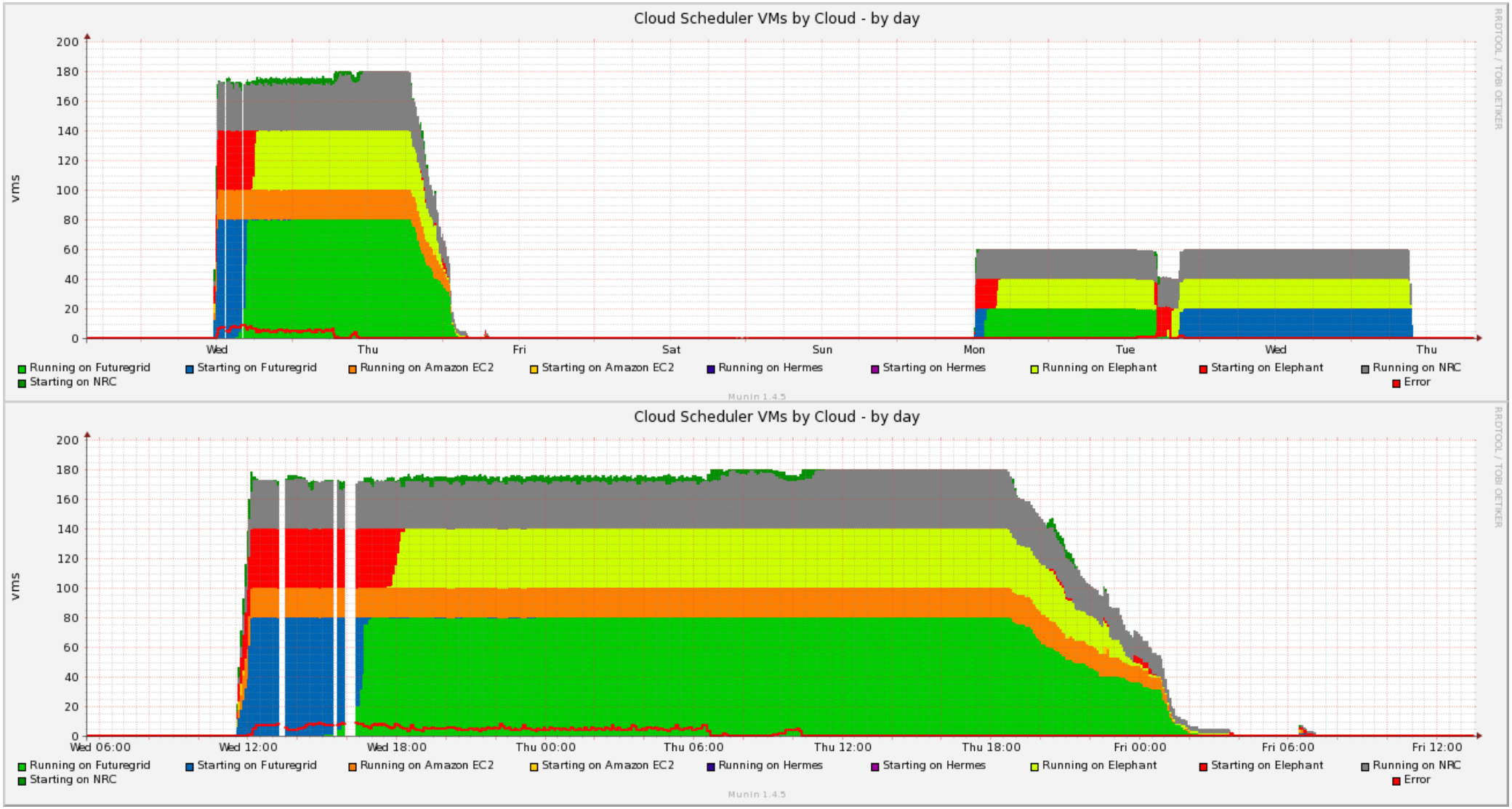
# Job Submission

- We use a Condor Rank expression to ensure that jobs only end up on the VMs they are intended to.
- Users use Condor attributes to specify the number of CPUs, memory, scratch space, that should be on their VMs (only works with IaaS that supports this).
- Users can also specify EC2 Amazon Machine Image (AMI) ID if Cloud Scheduler is configured for access to this type of cloud.
- Users prevented from running on VMs which were not booted for them (condor start requirement checked against DN).

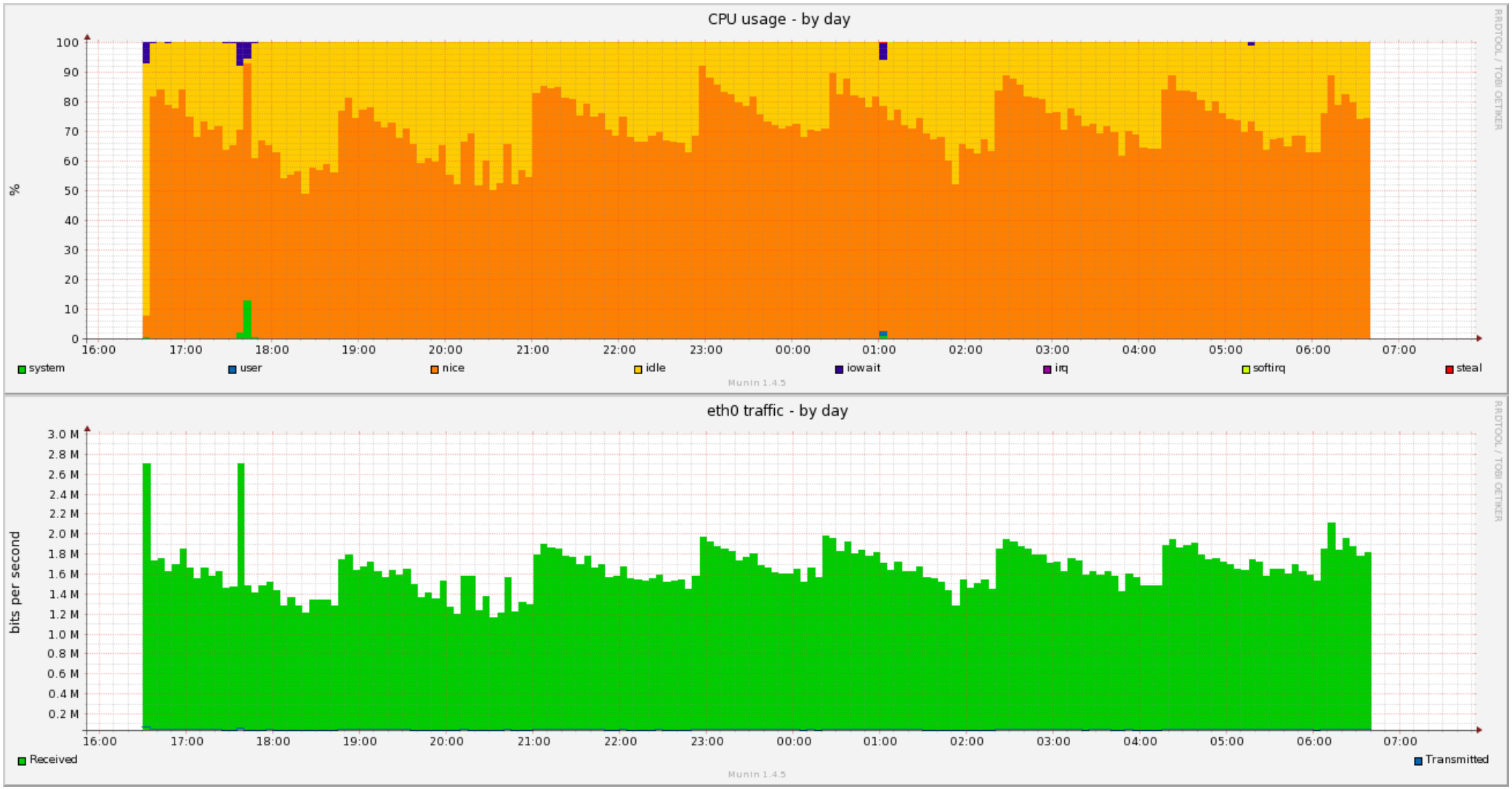
# BaBar Cloud Configuration



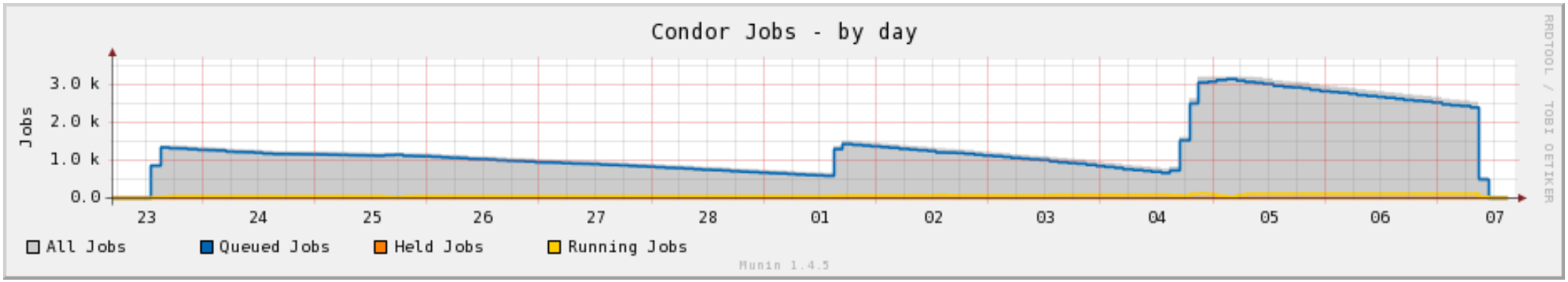
# Other Examples



# Inside a Cloud VM



# BaBar MC production




More resource added