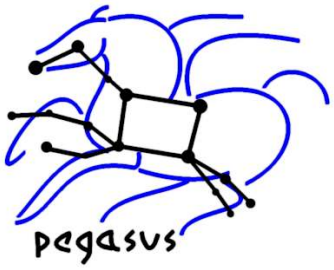
Scientific Workflows in the Cloud

Gideon Juve
USC Information Sciences Institute
gideon@isi.edu



Outline

- Scientific Workflows
  - What are scientific workflows?
- Workflows and Clouds
  - Why (not) use clouds for workflows?
  - How do you set up an environment to run workflows in the cloud?
- Evaluating Clouds for Workflows
  - What is the cost and performance of running workflow applications in the cloud?

Scientific Workflows

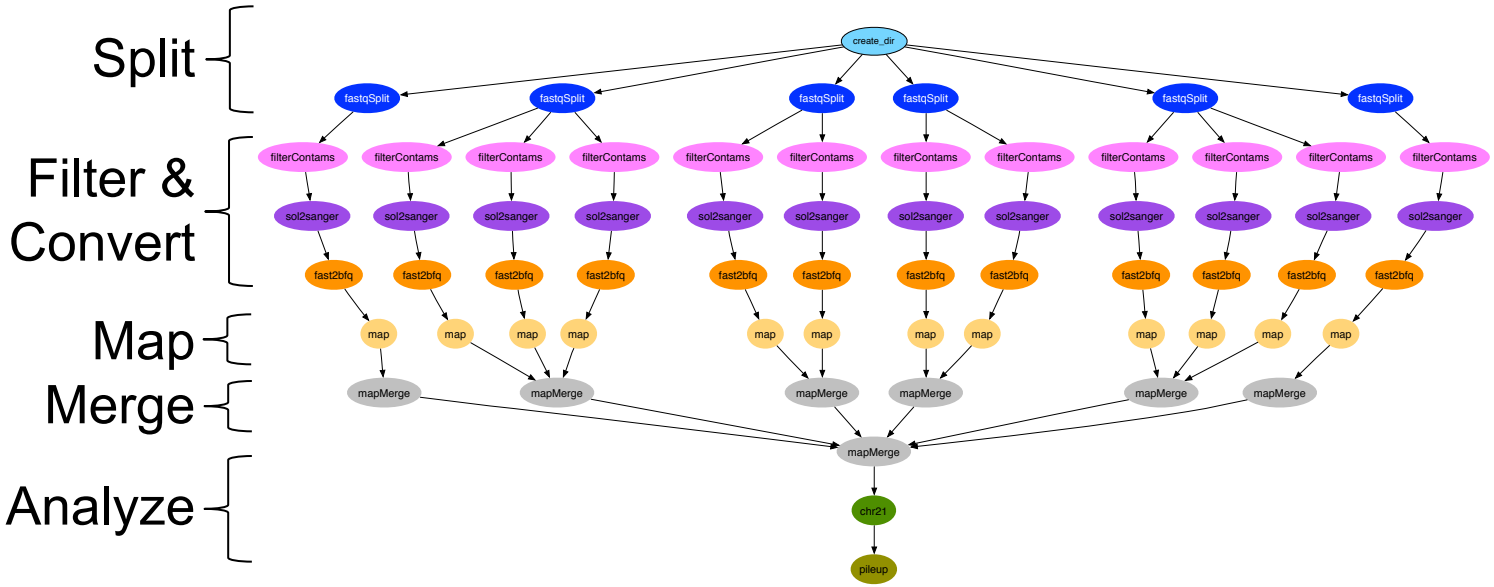
Science Applications

- Scientists often need to:
  - Integrate diverse components and data
  - Automate data processing steps
  - Repeat processing steps on new data
  - Reproduce previous results
  - Share their analysis steps with other researchers
  - Track the provenance of data products
  - Execute analyses in parallel on distributed resources
  - Reliably execute analyses on unreliable infrastructure

Scientific workflows provide solutions to these problems

Scientific Workflows

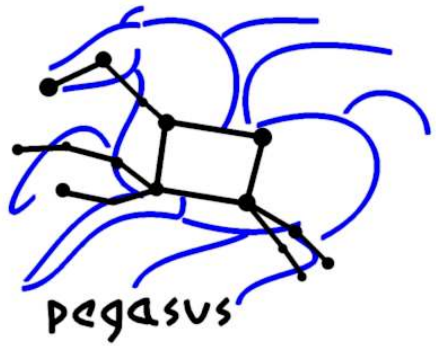
- Orchestrate complex, multi-stage scientific computations
- Expressed in high-level workflow languages
  - DAGs, scripting languages, data flow, actors, etc.
- Can be automatically parallelized on distributed resources



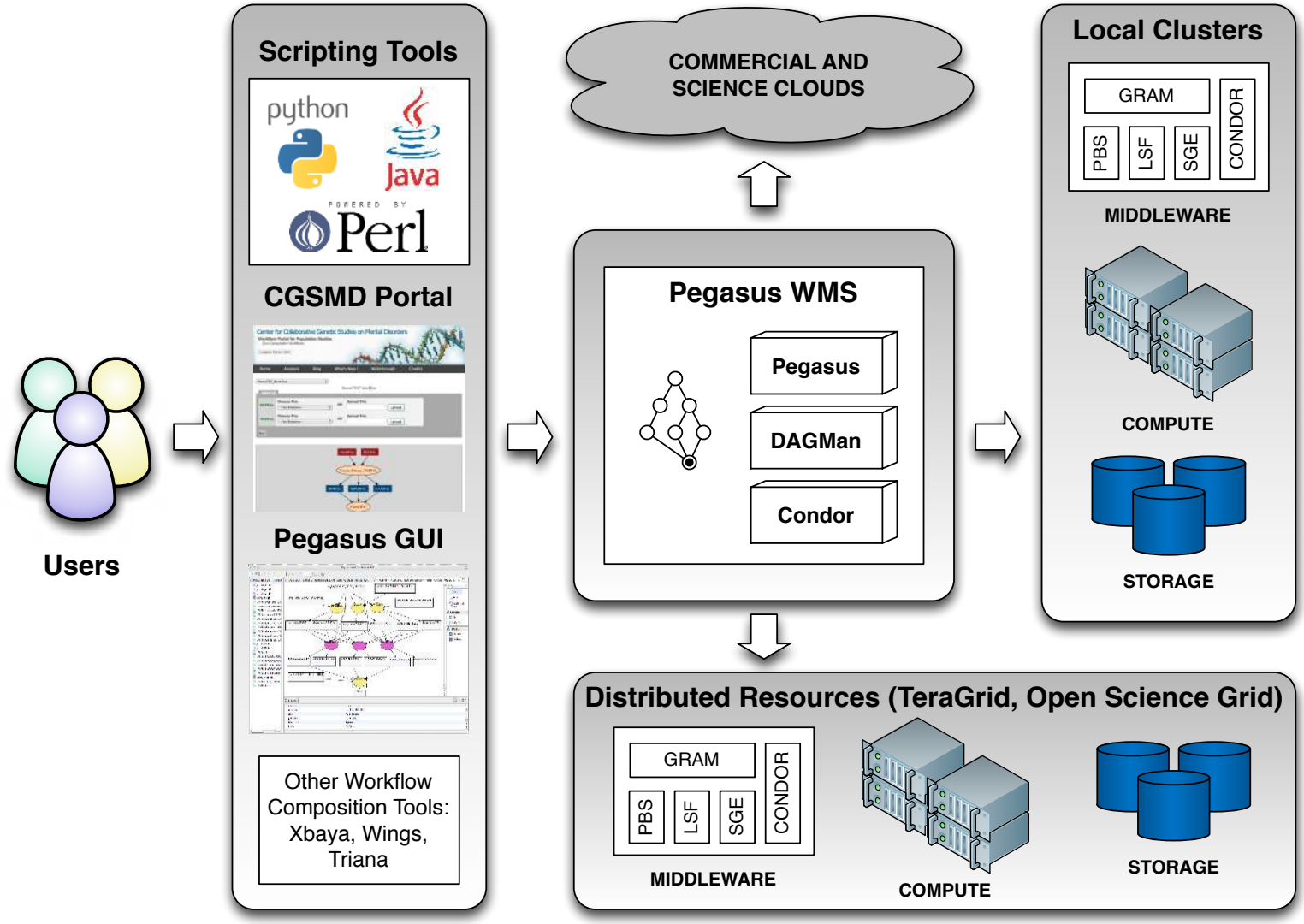
Epigenomics Workflow

Pegasus Workflow Management System

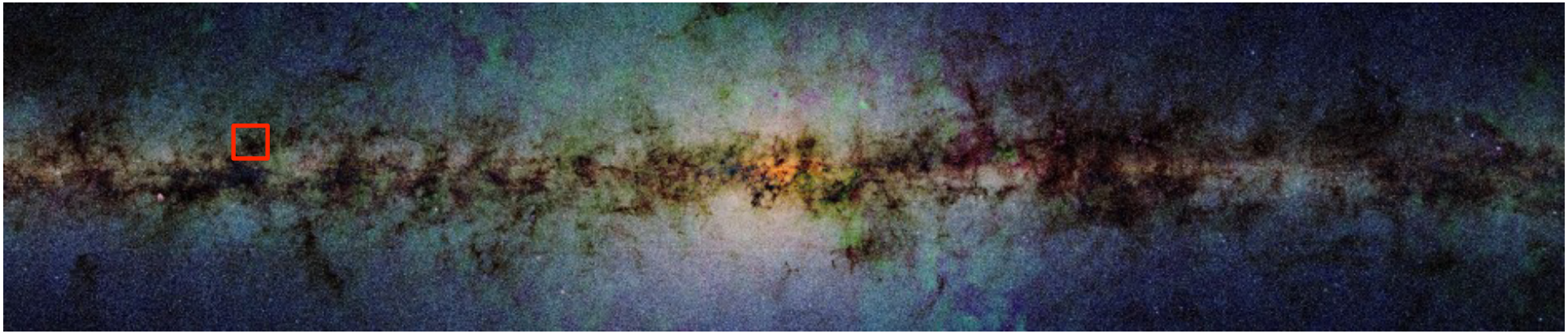
- Compiles abstract workflows to executable workflows
- Designed for scalability
  - Millions of tasks, thousands of resources, terabytes of data
- Enables portability
  - Local desktop, HPC clusters, grids, clouds
- Does not require application code changes
- Features
  - Replica selection, transfers, registration, cleanup
  - Task clustering for performance and scalability
  - Reliability and fault tolerance
  - Monitoring and troubleshooting
  - Provenance tracking
  - Workflow reduction



Pegasus WMS


Scripting Tools
python
Java
POWERED BY
Perl
CGSMD Portal
Pegasus GUI
Other Workflow Composition Tools: Xbaya, Wings, Triana
Users
COMMERCIAL AND SCIENCE CLOUDS
Pegasus WMS
Pegasus
DAGMan
Condor
Local Clusters
GRAM
PBS
LSF
SGE
CONDOR
MIDDLEWARE
COMPUTE
STORAGE
Distributed Resources (TeraGrid, Open Science Grid)
GRAM
PBS
LSF
SGE
CONDOR
MIDDLEWARE
COMPUTE
STORAGE

Large-Scale, Data-Intensive Workflows



John Good (Caltech)

- Montage Galactic Plane Workflow
  - 18 million input images (~2.5 TB)
  - 900 output images (2.5 GB each, 2.4 TB total)
  - 10.5 million tasks (34,000 CPU hours)

× 17

- Scientific workflow management systems are designed to automatically distribute data and computations for these large applications

Workflows and Clouds

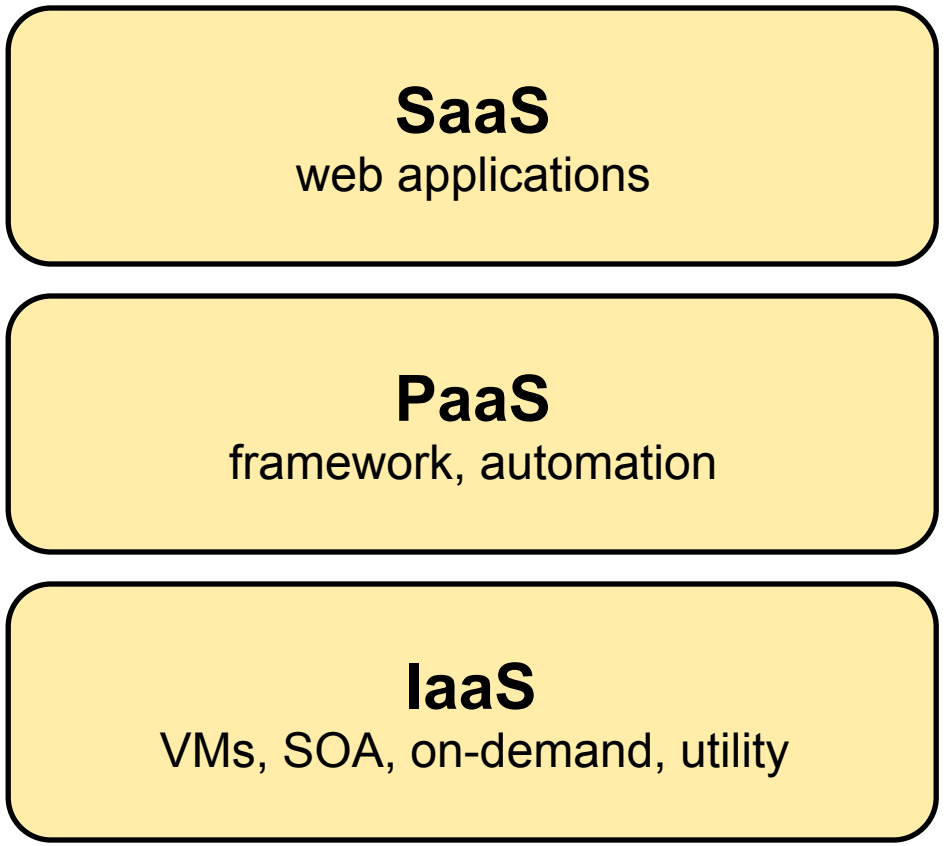
Benefits of Clouds for Workflows

- Custom Execution Environments
  - Workflows often contain diverse legacy code
  - VM images can be customized for the application
- Provisioning
  - On-demand, elastic
- Reproducibility, Sharing and Provenance
  - VM images capture the entire software environment
  - Can be re-launched to repeat experiment
  - Can be shared with other scientists (e.g. appliances)
  - Can be stored as part of provenance
- Economics (Commercial Clouds)
  - Pay instead of waiting
  - No premium to scale
  - Cost/performance tradeoffs

Drawbacks of Clouds for Workflows

- Administration
  - Administration is still required
  - The user is responsible for the environment
- Complexity
  - Deploying workflow applications
  - Provisioning
  - Cost/performance tradeoffs
- Performance
  - Virtualization overhead, non-HPC resources
- Other issues
  - Vendor lock-in
  - Security

Cloud Hierarchy



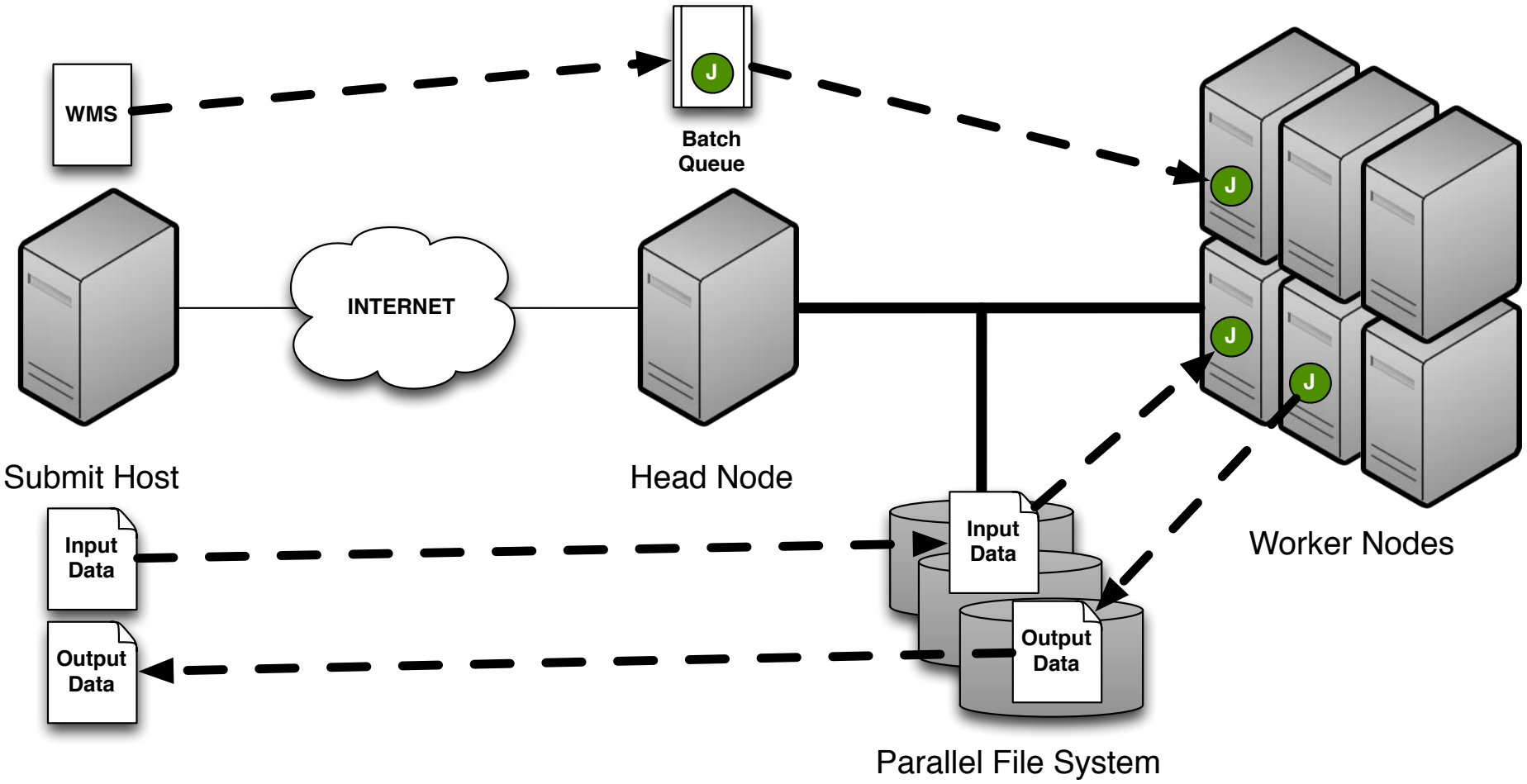
Galaxy

Science gateways

Amazon SWF

This talk will focus on deploying workflows on IaaS clouds

Typical Grid/HPC Cluster Setup



Virtual Clusters

- One approach to deploying workflows in the cloud is to replicate grid/cluster environments
- Grids and clusters are already configured for executing large-scale, parallel applications
- Infrastructure clouds only provide raw resources
- The challenge is to deploy **Virtual Clusters**
- Some software exists to support this:
  - Chef and Puppet
  - Nimbus Context Broker
  - cloudinit.d
  - StarCluster
  - Wrangler

Virtual Cluster Challenges

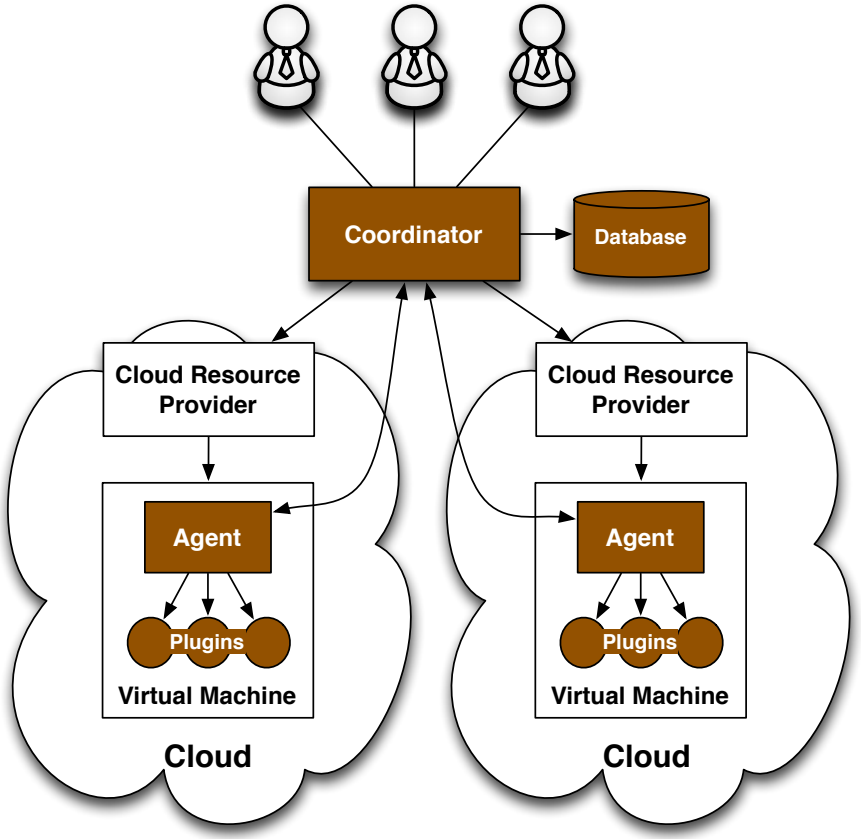
- Required environment is composed of multiple nodes with different roles
  - e.g. submit host, worker nodes, file system
- Infrastructure clouds are dynamic
  - Provision on-demand
  - Configure at runtime
- Deployment is not trivial
  - Manual setup is error-prone and not scalable
  - Scripts work to a point, but break down for complex deployments

Virtual Cluster Requirements

- Automatic deployment
  - Scriptable and repeatable
- Complex dependencies between VMs
  - No pre-defined architectures
- Dynamic provisioning / elasticity
  - Add and remove nodes at runtime
- Multiple cloud providers
  - Different provisioning interfaces
- Monitoring
  - Periodic failure checking and error reporting

Wrangler Deployment Service

- VC deployment service
  - Deployment description
    - Declarative XML syntax
    - DAG-based dependencies
  - User-defined plugins
  - Multiple Interfaces
    - Command-line, XML-RPC, Python API
  - Multiple Resource Providers
    - Current: EC2, Eucalyptus
    - Future: Nimbus, OpenStack



G. Juve, E. Deelman, “Automating Application Deployment in Infrastructure Clouds”, 3rd IEEE International Conference on Cloud Computing Technology and Science (CloudCom), 2011.

Deployment Request

<deployment>: Set of virtual machines that collectively implement an application.

<node>: A single VM.

<provider>: Specifies the cloud resource provider to use to provision the VM, and the VM parameters.

<plugin>: Specifies the plugin script, and parameters to use when configuring the VM.

<depends>: Enables user to specify ordering of nodes for provisioning and configuration.

<ref>: Enables nodes to be configured using attributes of other nodes.

```
<deployment>
    <node name="server">
        <provider name="amazon">
            <image>ami-912837</image>
            <instance-type>c1.xlarge</instance-type>
            ...
        </provider>
        <plugin script="nfs_server.sh">
            <param name="EXPORT">/mnt</param>
        </plugin>
    </node>
    <node name="client" count="3" group="clients">
        <provider name="amazon">
            <image>ami-901873</image>
            <instance-type>m1.small</instance-type>
            ...
        </provider>
        <plugin script="nfs_client.sh">
            <param name="SERVER">
                <ref node="server" attribute="local-ipv4">
            </param>
            <param name="PATH">/mnt</param>
            <param name="MOUNT">/nfs/data</param>
        </plugin>
        <depends node="server"/>
    </node>
</deployment>
```

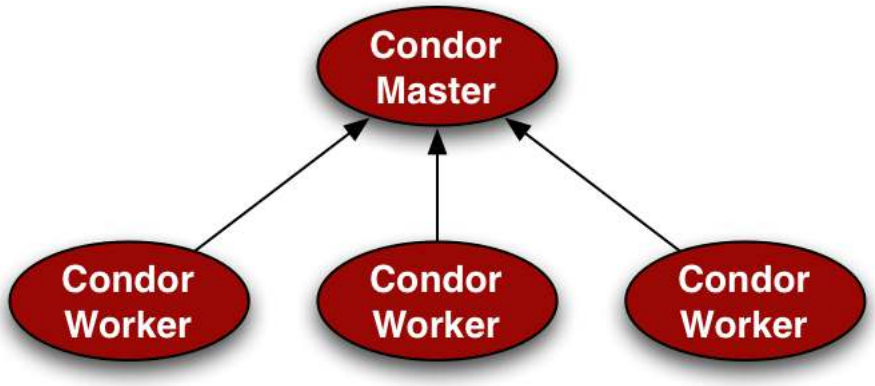
Create identical VMs

Depend on a single node or a group

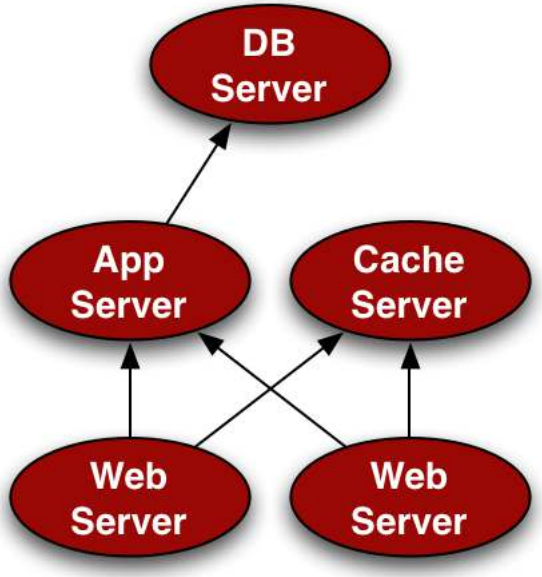
Client needs server's IP address

Dependencies

- Specifies dependencies between VMs
  - e.g. NFS client requires NFS server
- Model as a Directed Acyclic Graph (DAG)
  - Determines order of configuration
  - Easy to reason about adding / removing VMs



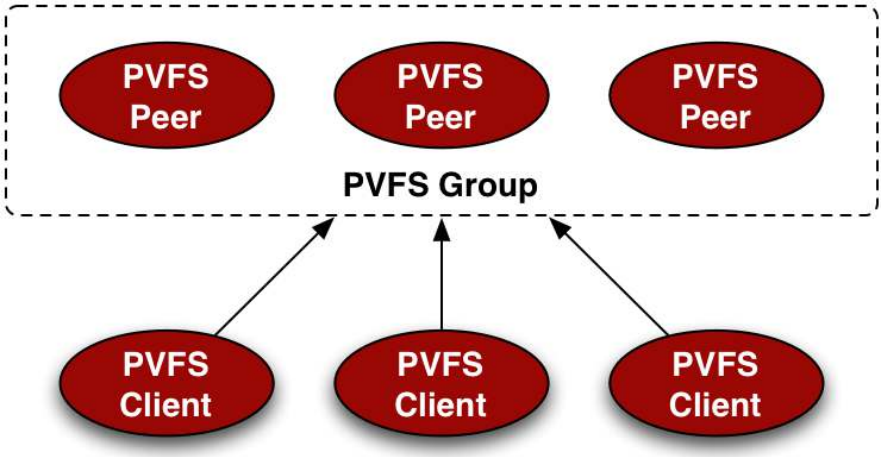
Condor Pool



Web Application

Groups

- A collection of mutually-dependent nodes
- For systems that require nodes to know about each other
  - P2P systems
  - parallel file systems
- Configuration happens after dependencies register, not after they are configured
  - Can get IP address, but not custom attributes



PVFS Parallel File System

Plugins

- Define the behavior of a VM
- Implemented as shell/perl/python scripts
- User-defined, automatically transferred to VM

Plugin Commands

start: Generate configuration files, start necessary processes, advertise custom attributes.

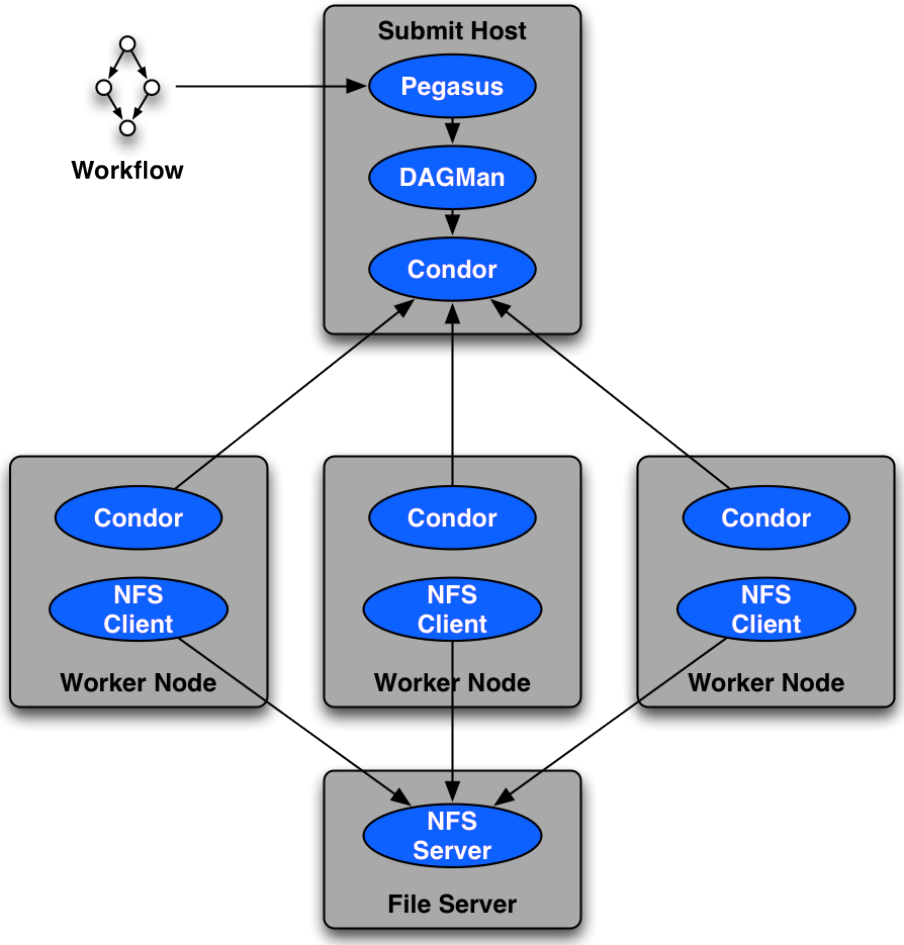
stop: Terminate running processes and clean up.

status: Check to make sure that the VM is in a valid state.

```
#!/bin/bash -e
PIDFILE=/var/run/condor/master.pid
SBIN=/usr/local/condor/sbin
CONF=/etc/condor/condor_config.local
if [ “$1” == “start” ]; then
    if [ "$CONDOR_HOST" == "" ]; then
        echo "CONDOR_HOST not specified”
        exit 1
    fi
    cat > $CONF <<END
CONDOR_HOST = $CONDOR_HOST
END
    $SBIN/condor_master –pidfile $PIDFILE
elif [ “$1” == “stop” ]; then
    kill –QUIT $(cat $PIDFILE)
elif [ “$1” == “status” ]; then
    kill -0 $(cat $PIDFILE)
fi
```

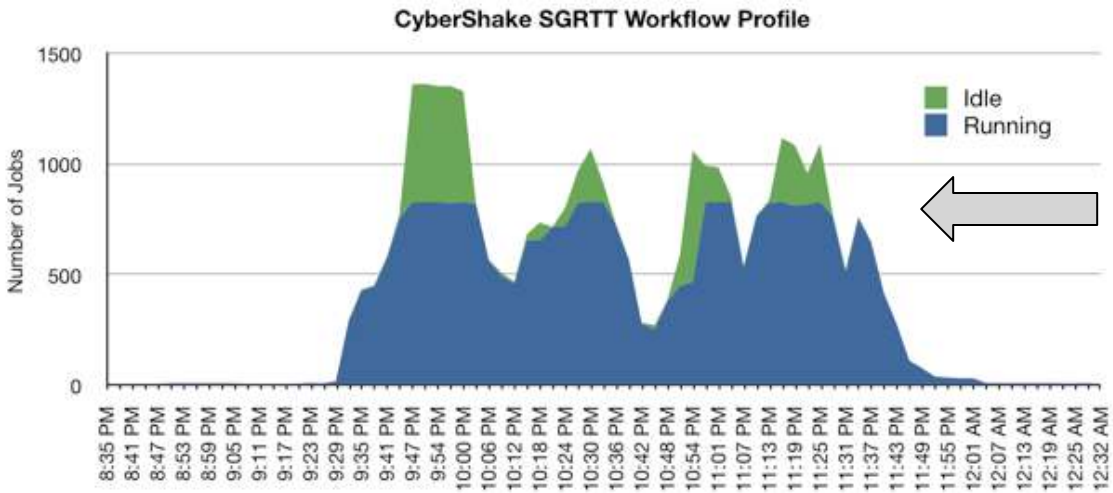
condor_master.sh

Example Deployment



Dynamic Provisioning

- Resource requirements of workflows change over time
- Would like to adapt dynamically
  - Solution is application-specific
  - Auto Scaling for web servers
- Important research topic



800 cores provisioned

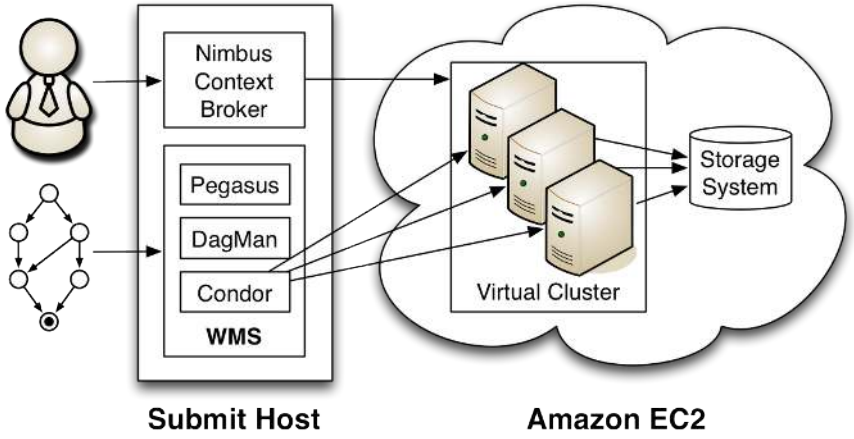
Resource utilization was less than 50%

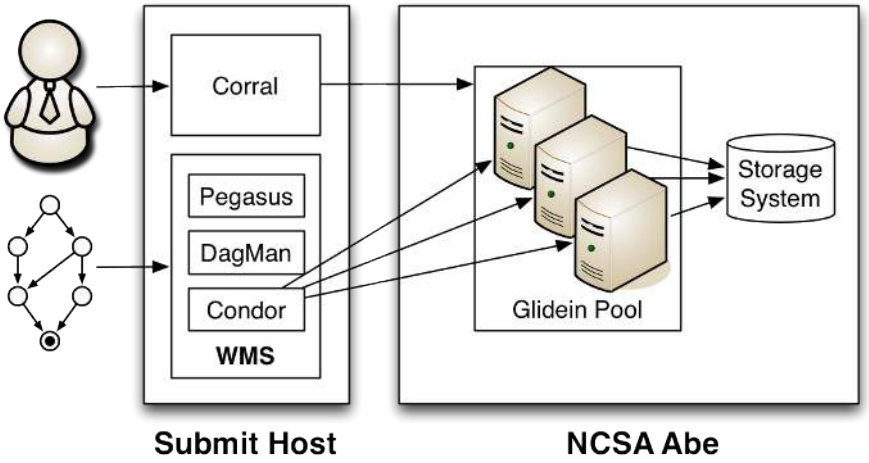
Evaluating Clouds for Workflows

Evaluating Clouds for Workflows

- Resource evaluation
  - Characterize performance, compare with grid
  - Resource cost, transfer cost, storage cost
- Storage system evaluation
  - Compare cost and performance of different distributed storage systems for sharing data in workflows
- Case study: mosaic service (long-term workload)
- Case study: periodograms (short-term workload)

Execution Environments



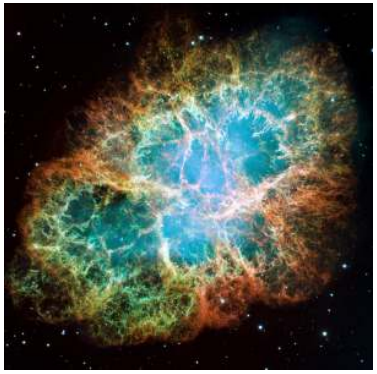


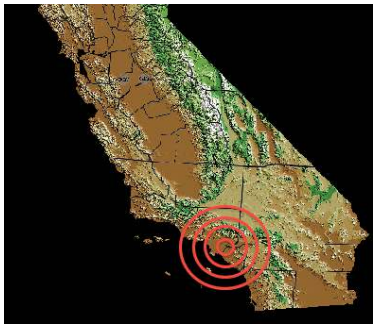
Cloud (Amazon) **Grid (TeraGrid)**

Cloud and grid environments are functionally identical

Applications Evaluated

- Montage (astronomy)
  - **I/O: High (95% of time waiting on I/O)**
  - Memory: Low
  - CPU: Low
- Epigenome (bioinformatics)
  - I/O: Low
  - Memory: Medium
  - **CPU: High (99% of time in CPU)**
- Broadband (earthquake science)
  - I/O: Medium
  - **Memory: High (75% of time tasks use > 1GB)**
  - CPU: Medium





Resource Performance Evaluation

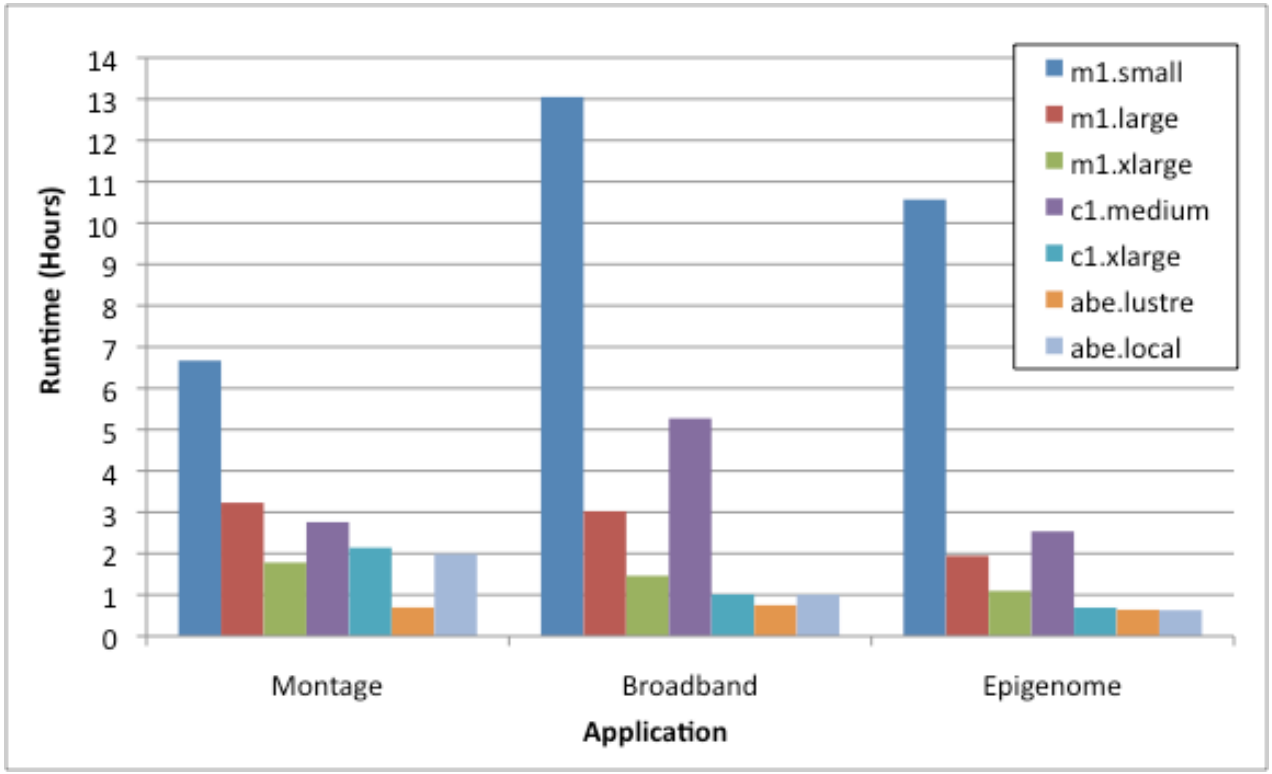
- Run workflows on single instances of different resource types (using local disk)
- Goals:
  - Compare performance/cost of different resource types
  - Compare performance of grid and cloud
  - Characterize virtualization overhead

Type	Arch.	CPU	Cores	Memory	Network	Storage	Price
m1.small	32-bit	2.0-2.6 GHz Opteron	1/2	1.7 GB	1-Gbps Ethernet	Local disk	$0.085/hr
m1.large	64-bit	2.0-2.6 GHz Opteron	2	7.5 GB	1-Gbps Ethernet	Local disk	$0.12/hr
m1.xlarge	64-bit	2.0-2.6 GHz Opteron	4	15 GB	1-Gbps Ethernet	Local disk	$0.68/hr
c1.medium	32-bit	2.33-2.66 GHz Xeon	2	1.7 GB	1-Gbps Ethernet	Local disk	$0.17/hr
c1.xlarge	64-bit	2.33-2.66 GHz Xeon	8	7.5 GB	1-Gbps Ethernet	Local disk	$0.68/hr
abe.local	64-bit	2.33 GHz Xeon	8	8 GB	10-Gbps InfiniBand	Local disk	N/A
abe.lustre	64-bit	2.33 GHz Xeon	8	8 GB	10-Gbps InfiniBand	Lustre	N/A

Resource Types Used

G. Juve, E. Deelman, K. Vahi, G. Mehta, G. B. Berriman, B. Berman, P. Maechling, Scientific Workflow Applications on Amazon EC2, e-Science, 2009.

Cloud Resource Performance

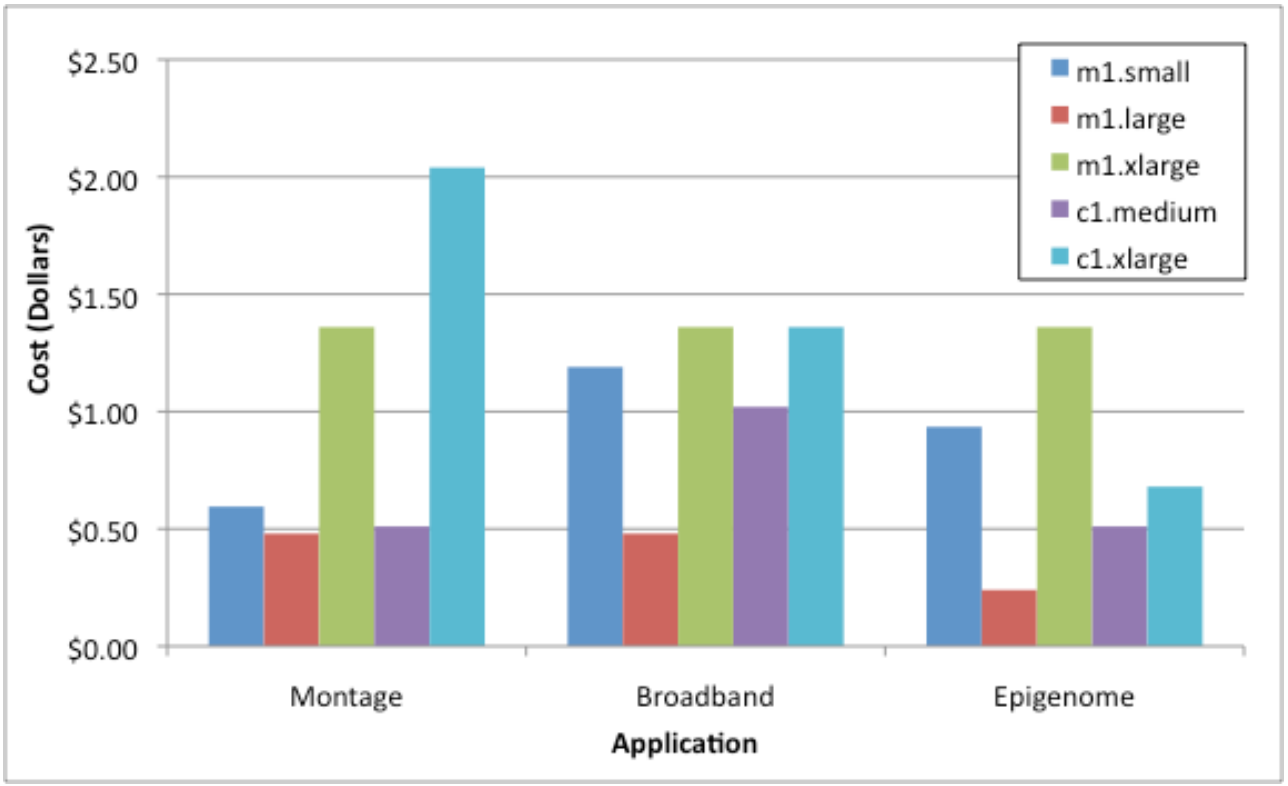


- Large difference in performance
- The best choice will depend on the application and the cost/ performance requirements of the user
- Virtualization overhead is less than 10%
- Parallel file system is biggest advantage for Abe

Cost Analysis

- Resource Cost
  - Cost for VM instances
  - Billed by the hour
- Transfer Cost
  - Cost to copy data to/from cloud over network
  - Billed by the GB
- Storage Cost
  - Cost to store VM images, application data
  - Billed by the GB-month, # of accesses

Resource Cost



- The per-workflow cost is low
- m1.small is not the cheapest
- m1.large is most cost-effective
- Resources with best performance are not cheapest
- Per-hour billing affects price/ performance tradeoff

Transfer Cost

Application	Input	Output	Logs
Montage	4291 MB	7970 MB	40 MB
Broadband	4109 MB	159 MB	5.5 MB
Epigenome	1843 MB	299 MB	3.3 MB

Transfer Sizes

Application	Input	Output	Logs	Total
Montage	$0.42	$1.32	< $0.01	$1.75
Broadband	$0.40	$0.03	< $0.01	$0.43
Epigenome	$0.18	$0.05	< $0.01	$0.23

Transfer Costs

- Cost of transferring data to/from cloud
  - Input: $0.10/GB (first 10 TB, sometimes discounted)
  - Output: $0.17/GB (first 10 TB)
- **Transfer costs can be high**
  - For Montage, transferring data costs more than computing it
- Costs can be reduced by storing input data in the cloud and using it for multiple workflows

Storage Cost

- Storage Charge
  - Price for storing data
  - Per GB-month
- Access Charge
  - Price for accessing data
  - Per operation
- **Short-term storage is relatively inexpensive**

- S3
  - Storage: $0.15 / GB-month
  - Access: PUT: $0.01 / 1,000
  - GET: $0.01 / 10,000
- EBS
  - Storage: $0.10 / GB-month
  - Access: $0.10 / million IOs

Application	Volume Size	Monthly Cost
Montage	5GB	$0.66
Broadband	5GB	$0.60
Epigenome	2GB	$0.26

Storage of Input data in EBS

Image	Size	Monthly Cost
32-bit	773 MB	$0.11
64-bit	729 MB	$0.11

Storage of VM images in S3

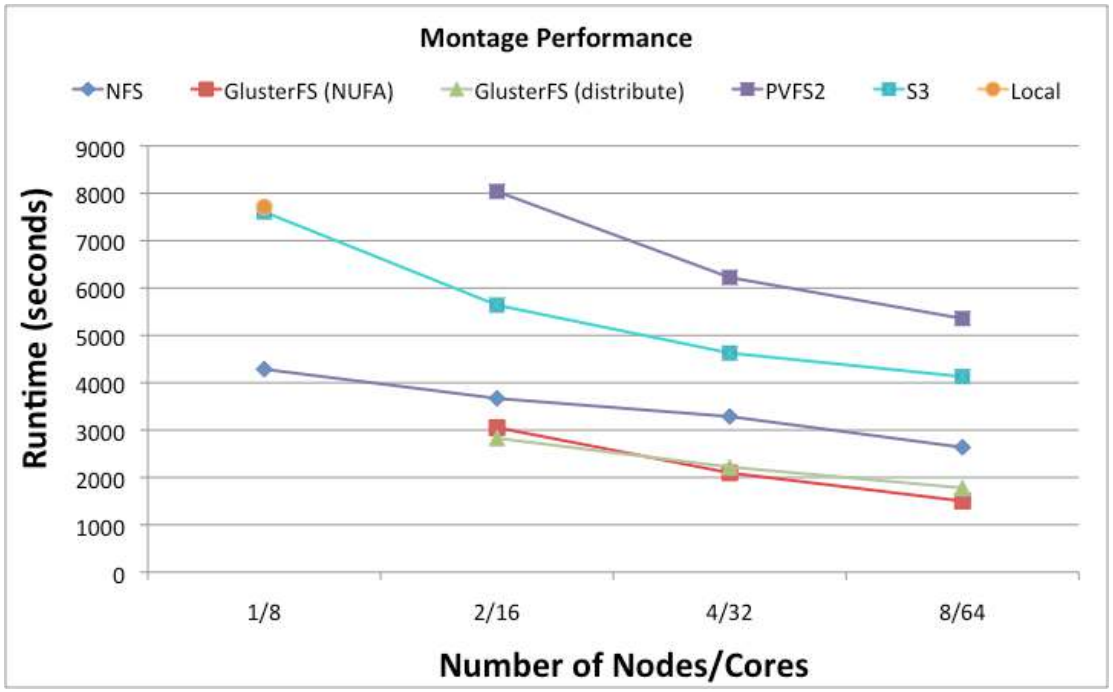
Storage System Evaluation

- Investigate options for storing intermediate data for workflows on a virtual cluster
  - Input and output data are transferred or stored
- Goals
  - Determine how to deploy storage systems
  - Compare traditional file systems with cloud storage systems
  - Compare performance/cost of storage systems
- Challenges
  - No custom kernels = no Lustre, Ceph (possible now)
  - XtreemFS did not perform well enough to finish experiments
  - Used c1.xlarge resources, no HPC instances at the time
  - Storage systems tuned for streaming workloads
  - Infinite number of combinations and configurations

Storage Systems

- Local Disk
  - RAID0 across ephemeral devices with XFS
  - RAID helps with the EC2 “first-write penalty”
- NFS: Network file system
  - 1 dedicated node (m1.xlarge)
- PVFS: Parallel, striped cluster file system
  - Workers host PVFS and run tasks
  - Using patched 2.6.3 because 2.8 series was not stable
- GlusterFS: Distributed file system
  - Workers host GlusterFS and run tasks
  - NUFA, and Distribute modes
- Amazon S3: Object-based storage system
  - Non-POSIX interface required changes to Pegasus
  - Data is cached on workers

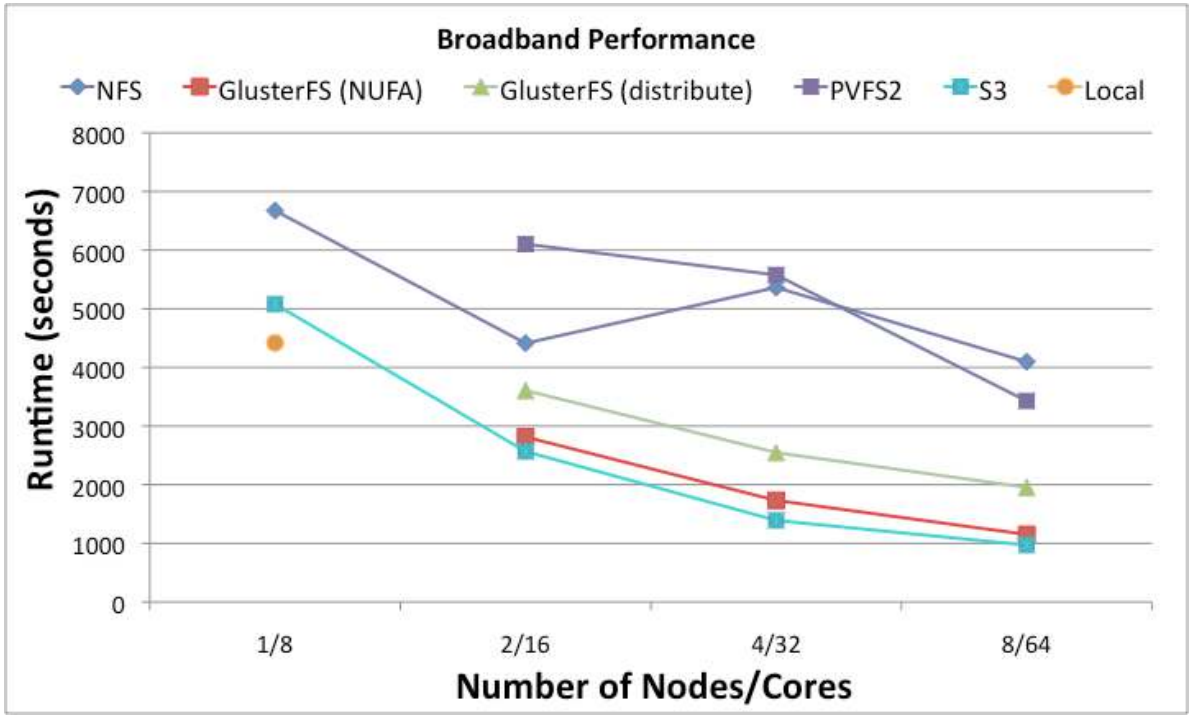
Storage System Performance (1)



Montage – I/O-bound

- Large number (30K) of small files (a few MB)
- GlusterFS does very well
- S3 performs poorly because of latency
- PVFS does not handle small files well

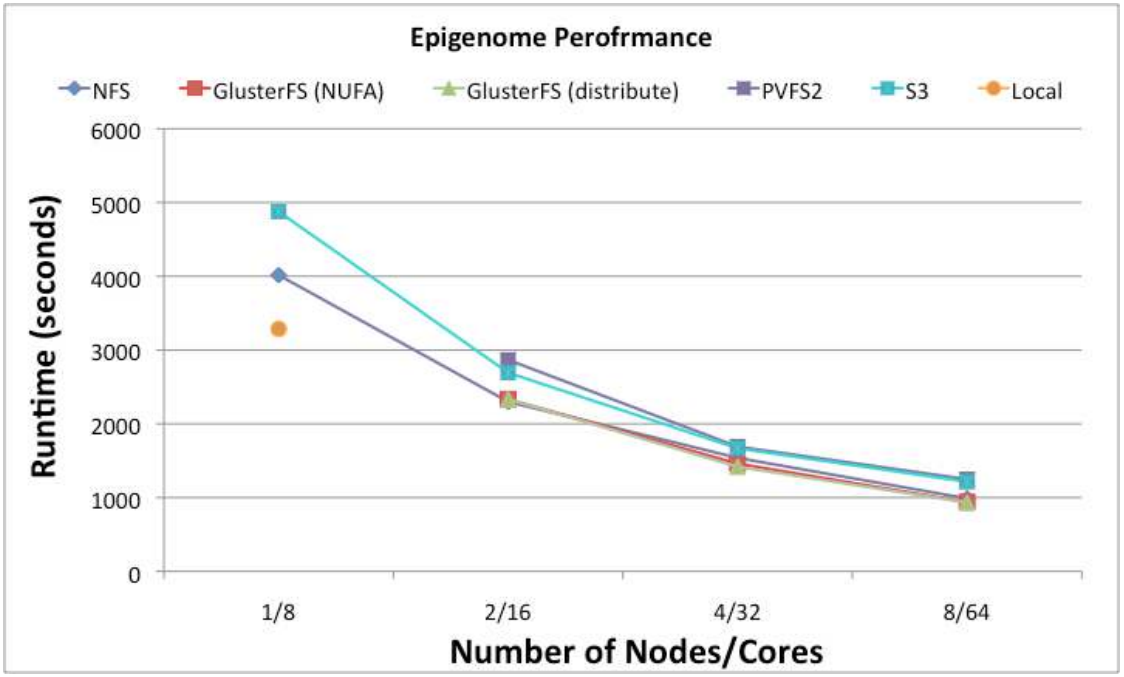
Storage System Performance (2)



Broadband – Memory intensive

- BB reuses many files – better S3 cache performance
- PVFS performance due to the large number of small files
- NFS performance is a mystery

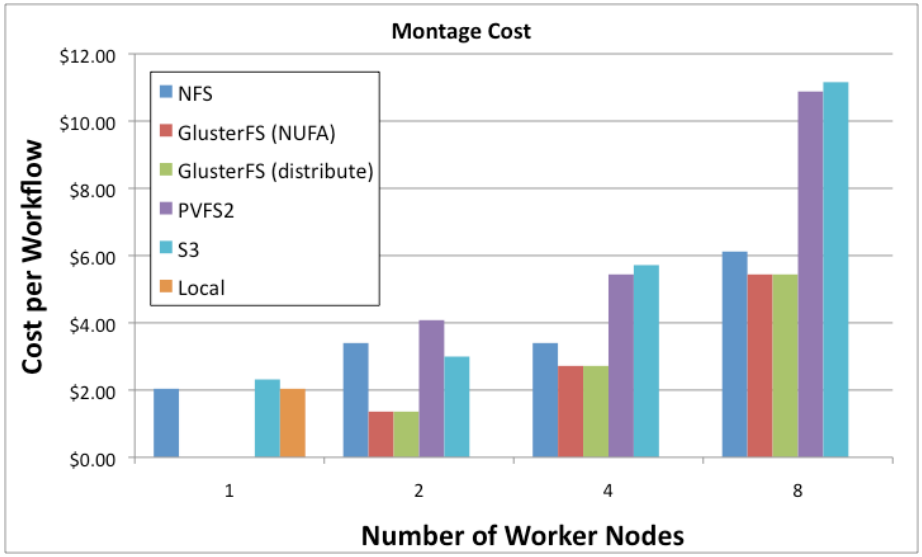
Storage System Performance (3)

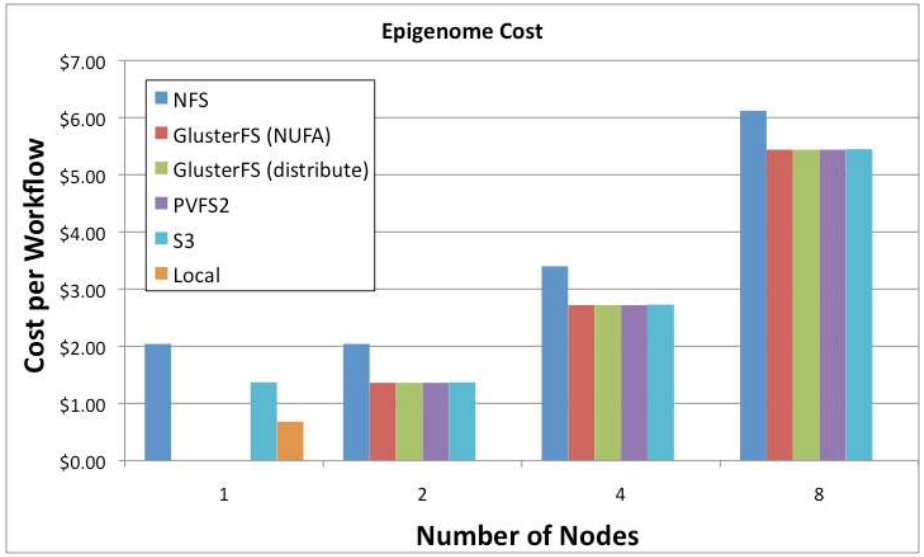


Epigenome – CPU-bound

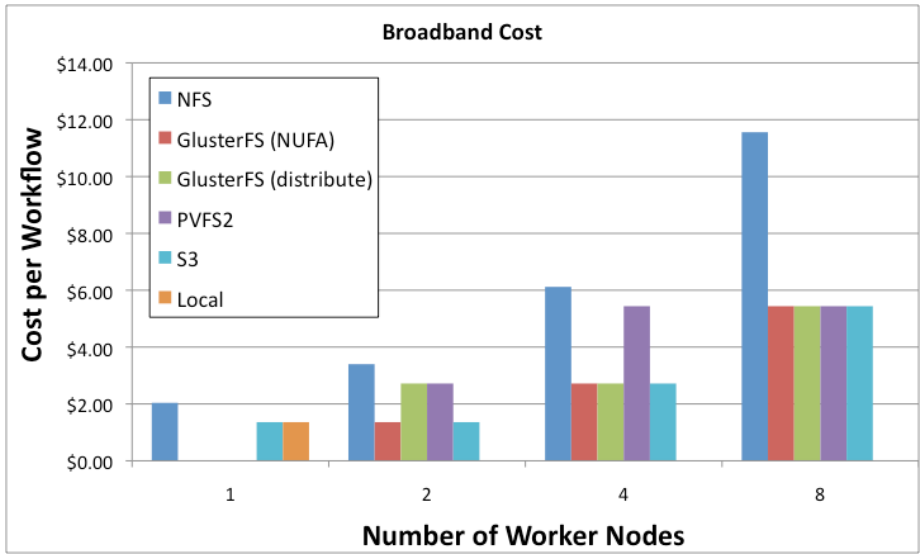
- Storage system did not matter much for CPU-bound application

Resource Cost (by Storage System)





- S3, NFS are at a disadvantage because of extra charges
- **Performance benefit of using multiple nodes does not offset increased cost**



Case Study: Mosaic Service

- Montage image mosaic service
  - Currently hosted by NASA IPAC
  - Computes mosaics on-demand using local cluster
- Current service workload:
  - 10 TB 2MASS dataset
  - 1,000 x 4 degree mosaics / month
- **Which is more cost-effective, cloud or local?**
  - 3 year operating cost comparison
  - Assume management costs are the same for both

Rho Oph Dark Cloud, Image Courtesy of 2MASS, IPAC, Caltech

G. B. Berriman, Gideon Juve, Ewa Deelman, Moira Regelson, Peter Plavchan, The Application of Cloud Computing to Astronomy: A Study of Cost and Performance, e-Science, 2010.

Cost Comparison

Item	Cost ($)
12 TB RAID 5 disk farm and enclosure (3 yr support)	12,000
Dell 2650 Xeon quad-core processor, 1 TB staging area	5,000
Power, cooling and administration	6,000
Total 3-year Cost	**23,000**
Cost per mosaic	**0.64**

Local Option

Item	Price ($)
Input Transfer (10 TB)	1,024.00
Output Transfer (24 TB)	3,691.41
Storage (10 TB)	36,864.00
Input I/O (10 TB)	24.58
Output I/O (27 TB)	67.50
Compute (c1.medium)	4,467.60
Total	**46,139.08**
Cost per mosaic	**1.28**

Amazon EBS Option

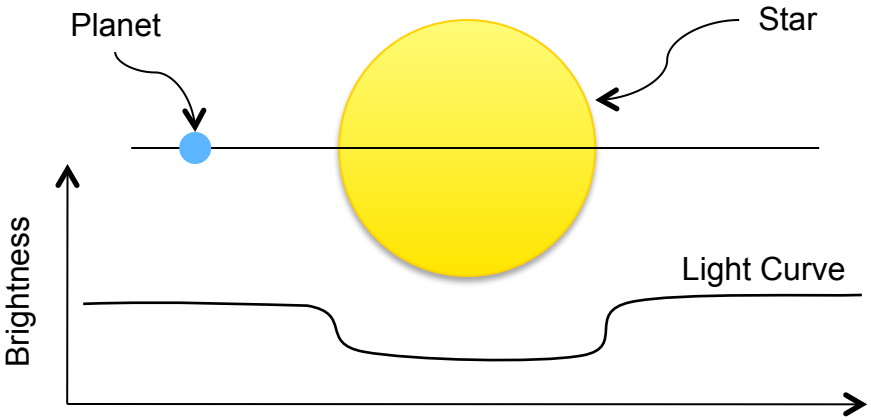
Item	Price ($)
Input Transfer (10 TB)	1,024.00
Output Transfer (24 TB)	3,691.41
PUT Ops (5.24 M)	52.43
GET Ops (14.4 M)	14.40
Compute (c1.medium)	4,467.60
Normal Storage (10 TB)	55,296.00
Total w/ Normal Storage	**64,545.84**
Cost per mosaic (Normal)	**1.79**
Reduced Redundancy Storage (10TB)	36,864.00
Total w/ RR Storage	**46,113.84**
Cost per mosaic (RR)	**1.28**

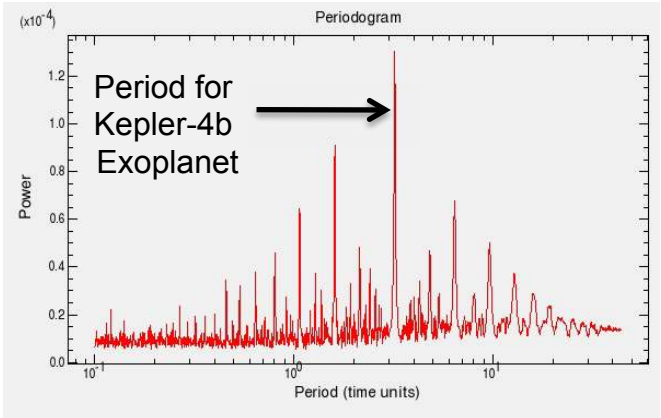
Amazon S3 Options

Amazon cost is ~2X local

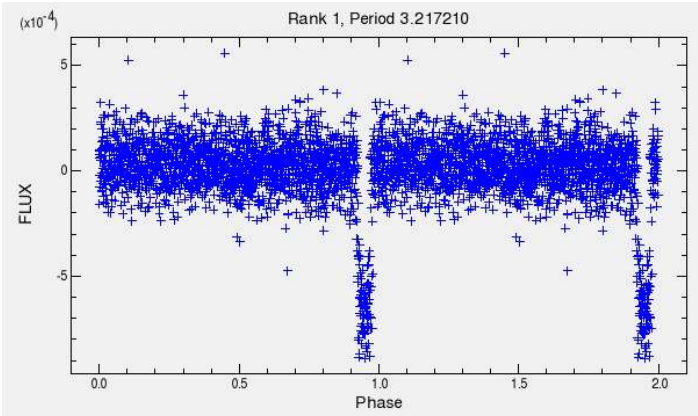
Case Study: Periodograms

- What is a periodogram?
  - Calculates the significance of different frequencies in time-series data to identify periodic signals.
  - Useful tool in the search for **exoplanets**
- NStED Periodogram tool
  - Fast, parallel implementation of periodograms algorithms in portable C





BLS periodogram for Kepler -4b, the smallest transiting exoplanet discovered by Kepler to date.



Phased Light Curve for Kepler-4b showing transiting exoplanet signal.

G. B. Berriman, Gideon Juve, Ewa Deelman, Moira Regelson, Peter Plavchan, The Application of Cloud Computing to Astronomy: A Study of Cost and Performance, e-Science, 2010.

Kepler Periodogram Atlas

- Compute periodogram atlas for public Kepler dataset
  - ~200K light curves X 3 algorithms X 3 parameter sets
  - Each parameter set was a different "Run", 3 runs total
  - **EC2 worked great for small workflows, grid was easier for large workflows**

		Run 1 (EC2)	Run 2 (EC2)	Run 3 (TeraGrid)
Runtimes	**Tasks**	631992	631992	631992
	Mean Task Runtime	7.44 sec	6.34 sec	285 sec
	Jobs	25401	25401	25401
	Mean Job Runtime	3.08 min	2.62 min	118 min
	Total CPU Time	1304 hr	1113 hr	50019 hr
	Total Wall Time	16.5 hr	26.8 hr	448 hr
Inputs	**Input Files**	210664	210664	210664
	Mean Input Size	0.084 MB	0.084 MB	0.084 MB
	Total Input Size	17.3 GB	17.3 GB	17.3 GB
Outputs	**Output Files**	1263984	1263984	1263984
	Mean Output Size	0.171 MB	0.124 MB	5.019 MB
	Total Output Size	105.3 GB	76.52 GB	3097.87 GB
Cost	**Compute Cost**	$179.52	$94.61	$4,874.24
	Output Cost	$15.80	$11.48	$464.68
	Total Cost	**$195.32**	**$106.08**	**$5,338.92**

Compute is ~10X Transfer

Amazon: 16 x c1.xlarge instances = 128 cores
Ranger: 8-16 x 16 core nodes = 128-256 cores

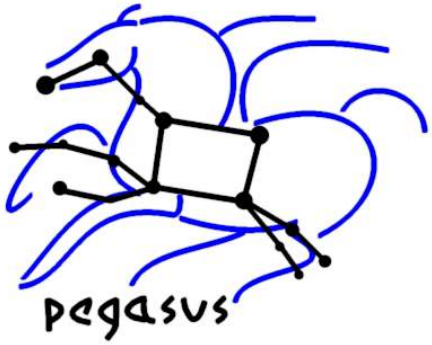
Actual cost **Estimated cost**

Conclusions

- Workflows help scientists orchestrate complex, multi-step simulations and analyses
- Clouds have many benefits and drawbacks for workflows
- Deploying workflows in the cloud is difficult, but there are tools that can help
- The performance of workflows in the cloud is manageable
- There are many cost/performance tradeoffs to consider
- Clouds are not cheaper or easier than alternatives
- More work needs to be done on tools for deployment, PaaS workflows, and dynamic provisioning

Pegasus Tutorial

- Goes through the steps of creating, planning, and running a simple workflow
- Virtual machine-based
  - VirtualBox
  - Amazon EC2
  - FutureGrid



http://pegasus.isi.edu/tutorial

http://pegasus.isi.edu/futuregrid/tutorials

Acknowledgements

- Pegasus Team
  - Ewa Deelman
  - Karan Vahi
  - Gaurang Mehta
  - Mats Rynge
  - Rajiv Mayani
  - Jens Vöckler
  - Fabio Silva
  - WeiWei Chen
- SCEC
  - Scott Callaghan
  - Phil Maechling
- NExScI/IPAC
  - Bruce Berriman
  - Moira Regelson
  - Peter Plavchan
- USC Epigenome Center
  - Ben Berman

Funded by NSF grants OCI-0722019 and CCF-0725332

Pegasus website: http://pegasus.isi.edu