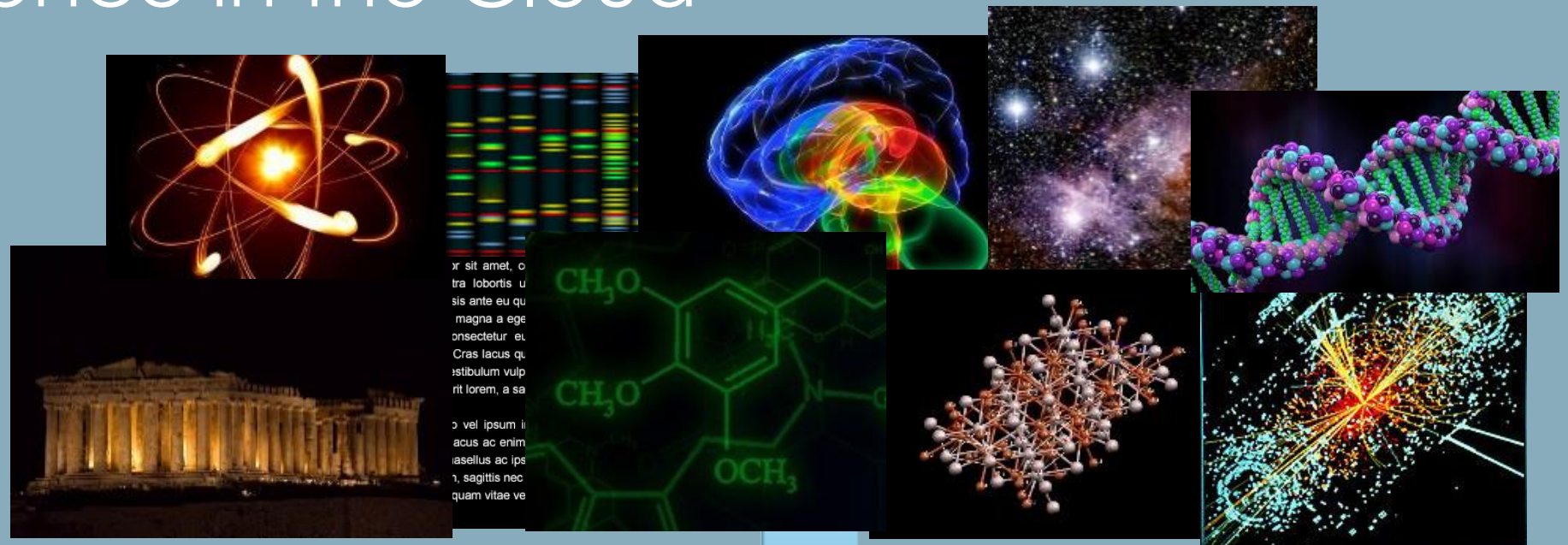
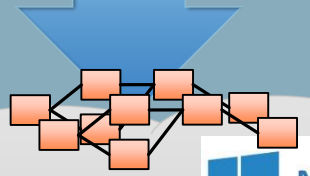
Autonomic Cloud Provisioning For Scientific Workflows

Ryan Chard

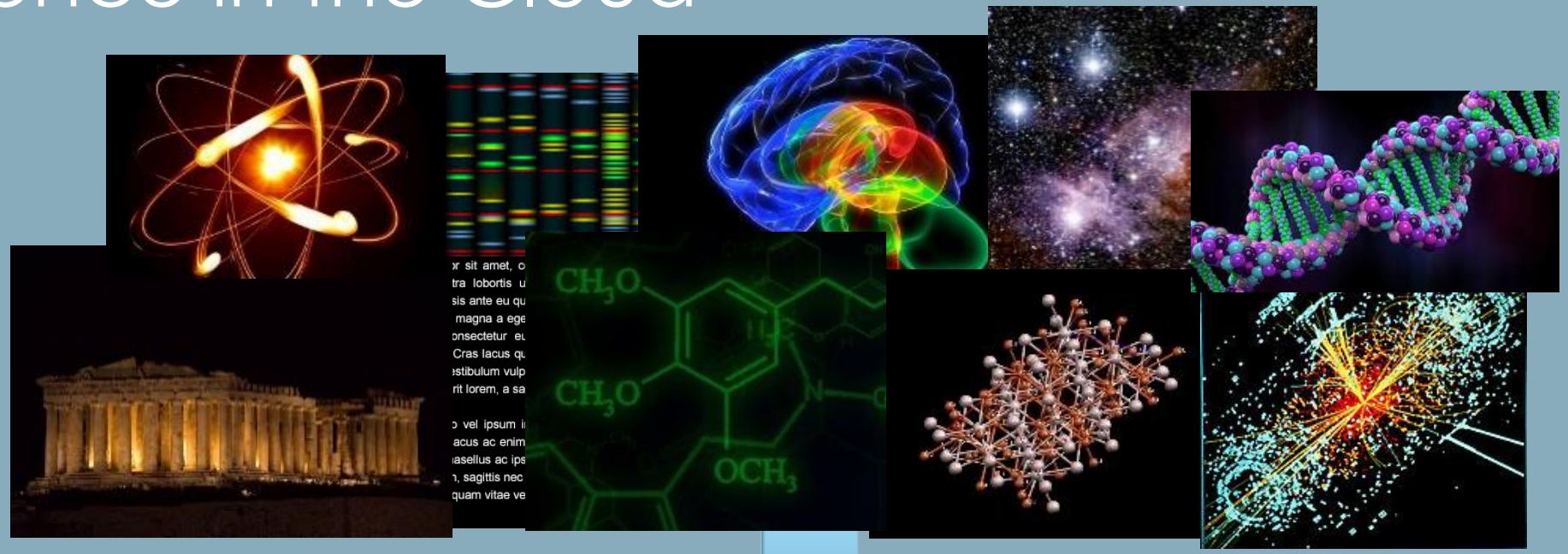
Victoria University of Wellington

Science in the Cloud





Science in the Cloud



Taverna

Galaxy

pegasus

Kepler
Your Science. Enabled.

amazon web services™

Microsoft Azure

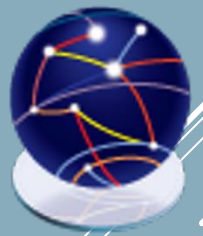
openstack™
CLOUD SOFTWARE

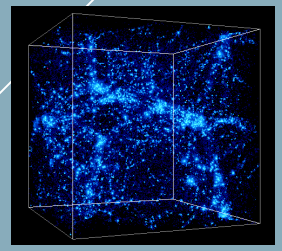
The Globus Galaxies Platform

A platform for creating cloud-hosted Science as a Service gateways:

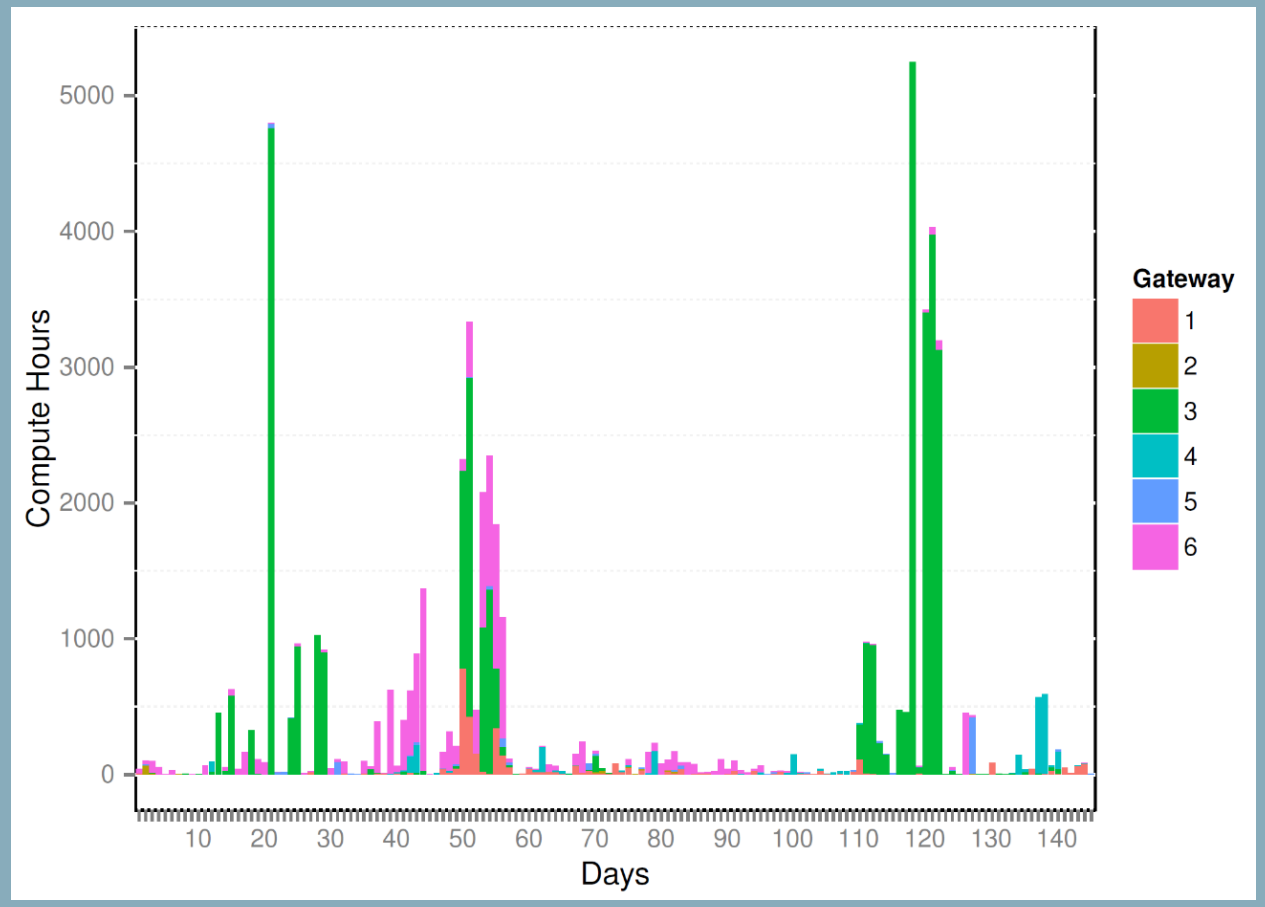
- **Galaxy** to create, manage, execute, share workflows
- **Globus** for data and identity management
- **HTCondor** for job scheduling and execution
- **AWS** for executing analyses
  - Spot instances
- **Elastic provisioner**

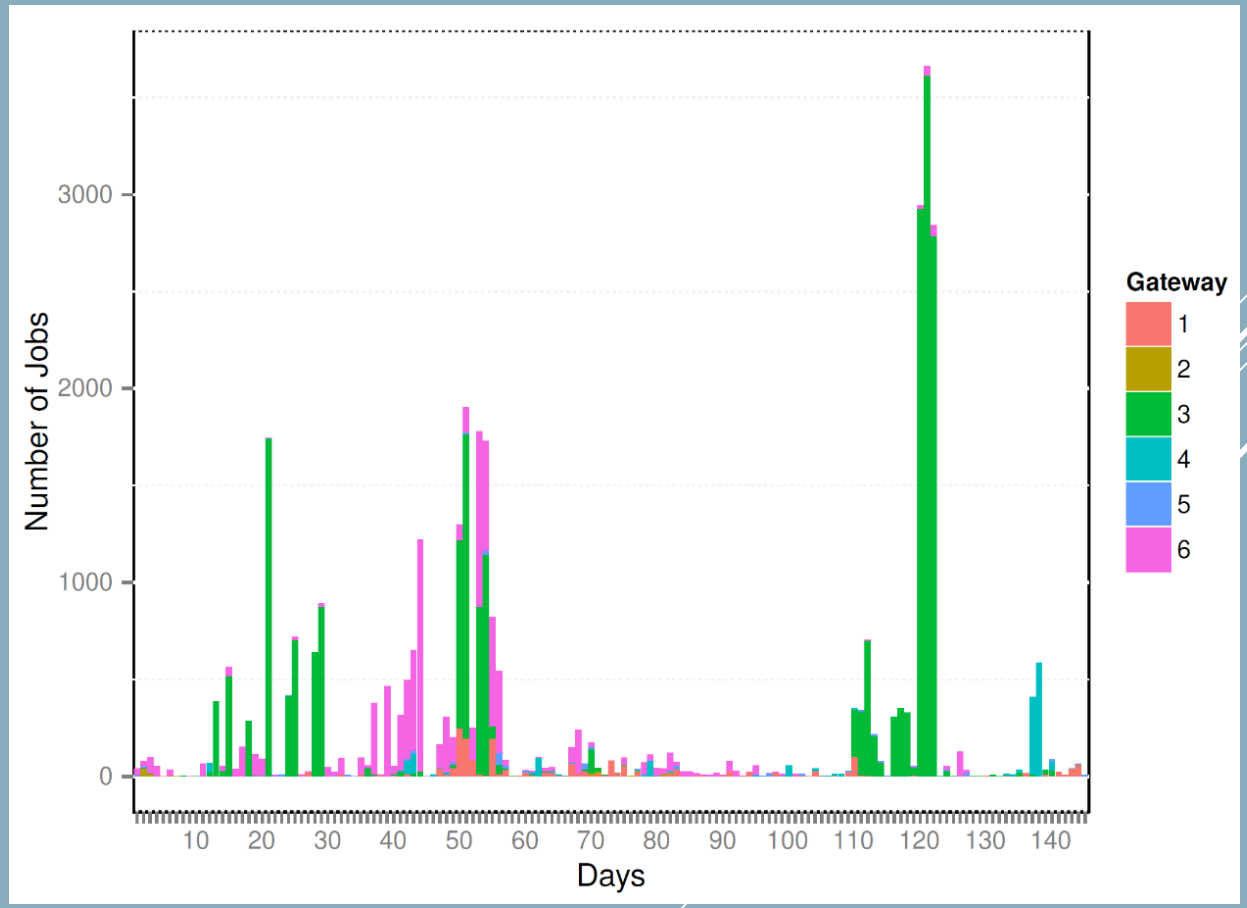
20 gateways, 300+ researchers





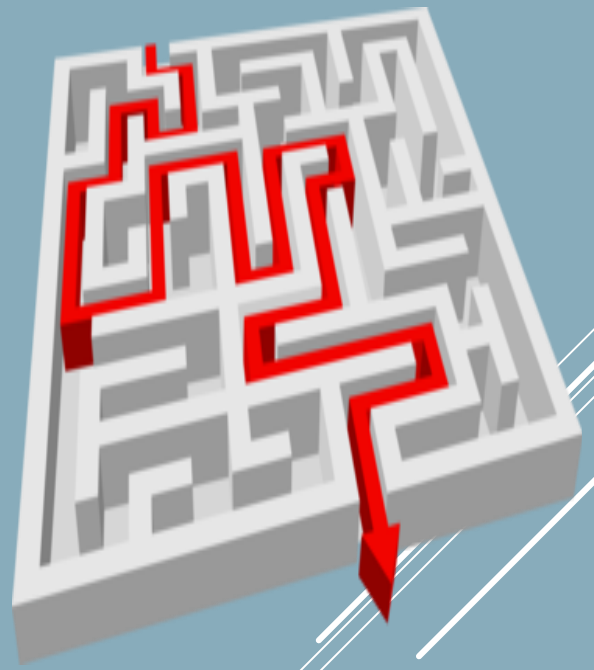
Globus Genomics Usage

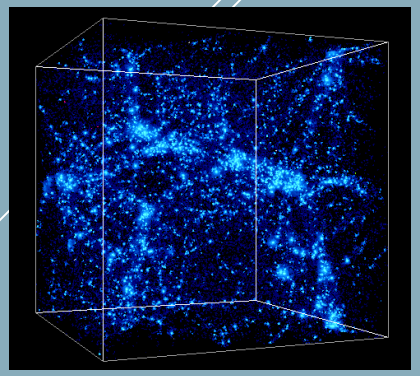




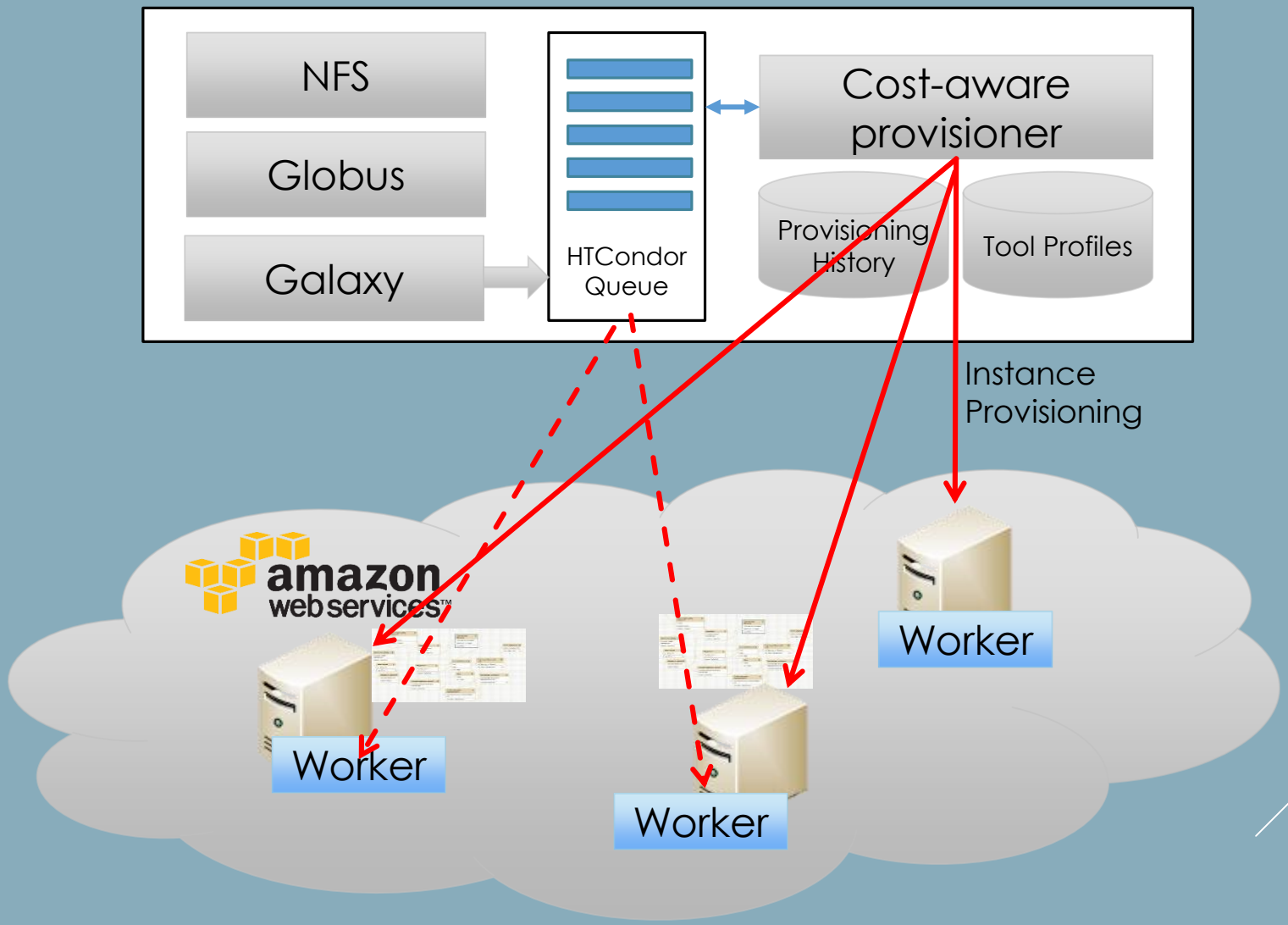
Cloud Provisioning Challenges:

- Naïve use can be expensive and limit performance
- 38 instance types, multiple regions and availability zones
  - Compute/memory/network/GPU optimised
  - Different pricing models (On-demand, spot, reserved)
- Differing tool requirements
- Technical challenges
  - Instance setup, tools, dependencies, termination
- Autoscaling/resource management

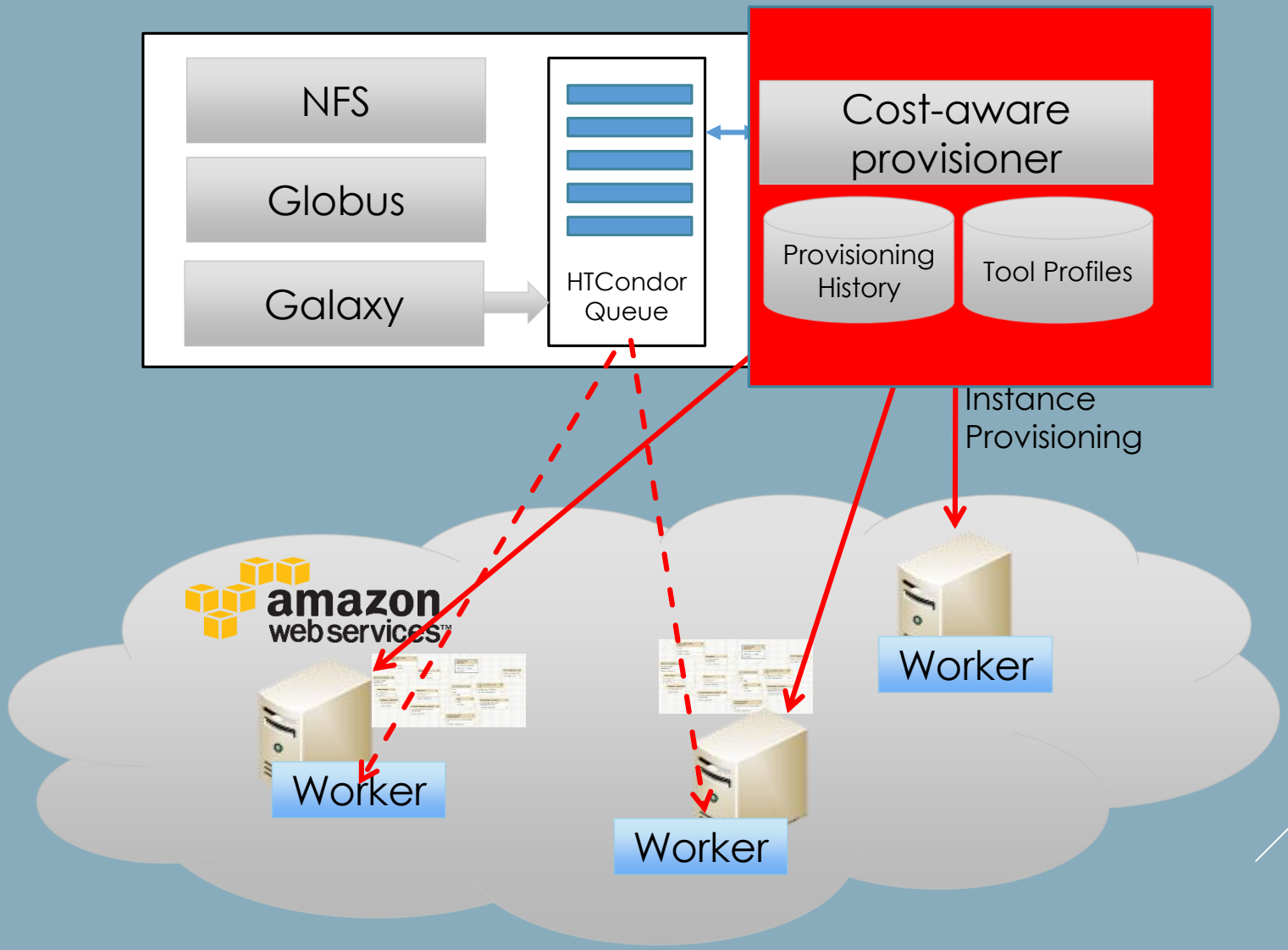




A Globus Galaxies Gateway

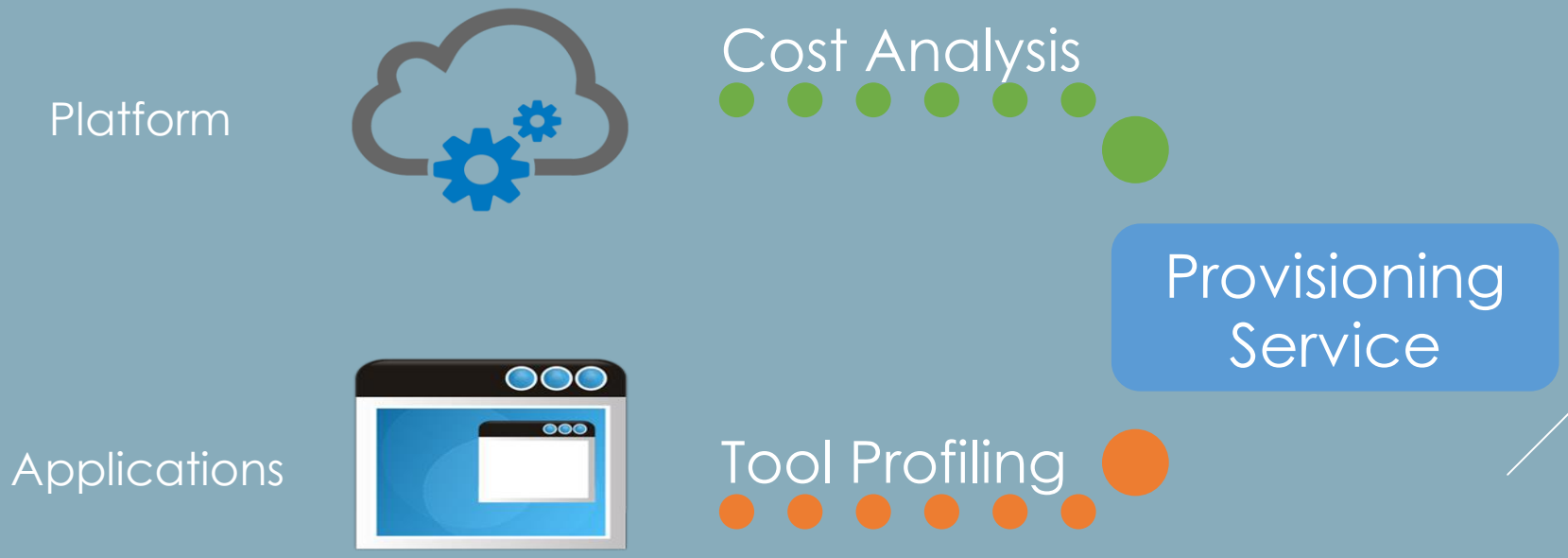


Provisioning System



Improving the Cloud Provisioner

- Simplify cloud usage
- Make it cheaper and faster
- Monitor arbitrary queues to acquire resources and fulfil workloads



Smarter Cloud Use

Cost

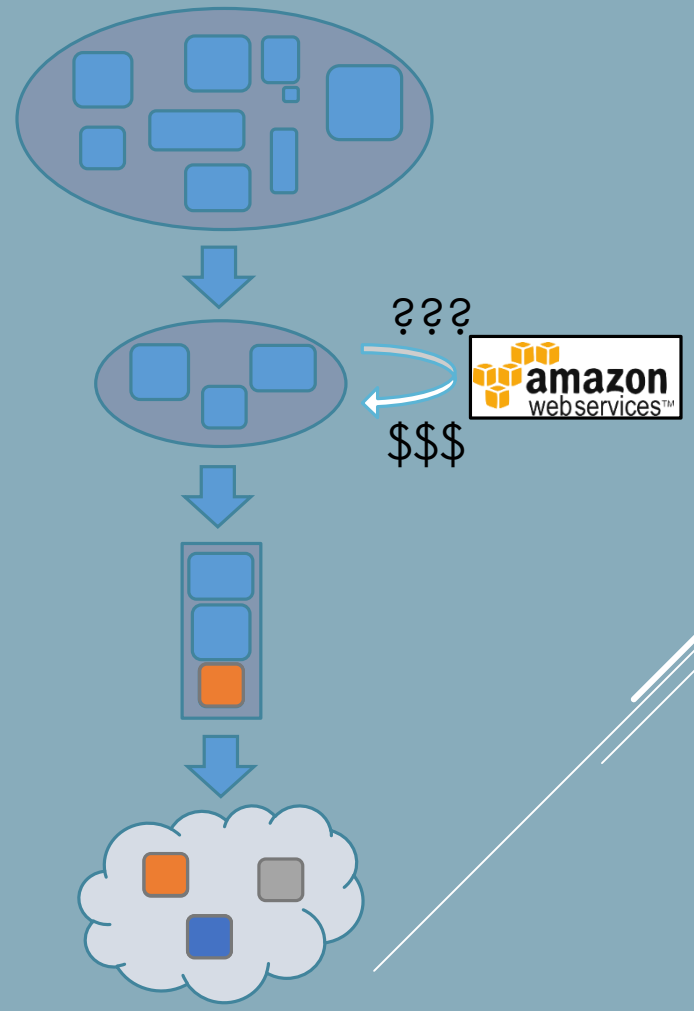
- Broaden search scopes
  - Use many instance types and all AZs
- Use profiles to match workloads to instances
  - Encode memory, CPU, and disk requirements in ClassAds etc.
- Evaluate spot market prices
  - All availability zones (AZs) and all suitable instance types
- Self termination

Throughput

- Over-provision requests
  - Request more resources than needed
- Repurpose excess spot requests
  - Migrate requests to idle jobs
- Revert to on-demand

Cost-aware Provisioning Approach

1. Filter instance types with profiles
2. Determine price for each instance type across all AZs
3. Rank potential requests
4. Make requests and monitor
5. Cancel or repurpose excess active requests once one is fulfilled



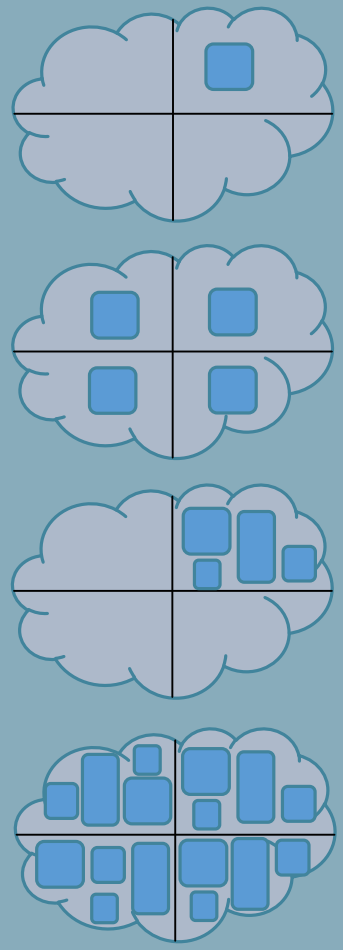
Evaluation

Six production GG gateways

- Galaxy/Condor logs + spot price history
- 145 day period

Evaluation by mapping jobs to AWS spot price history

Four provisioning scopes



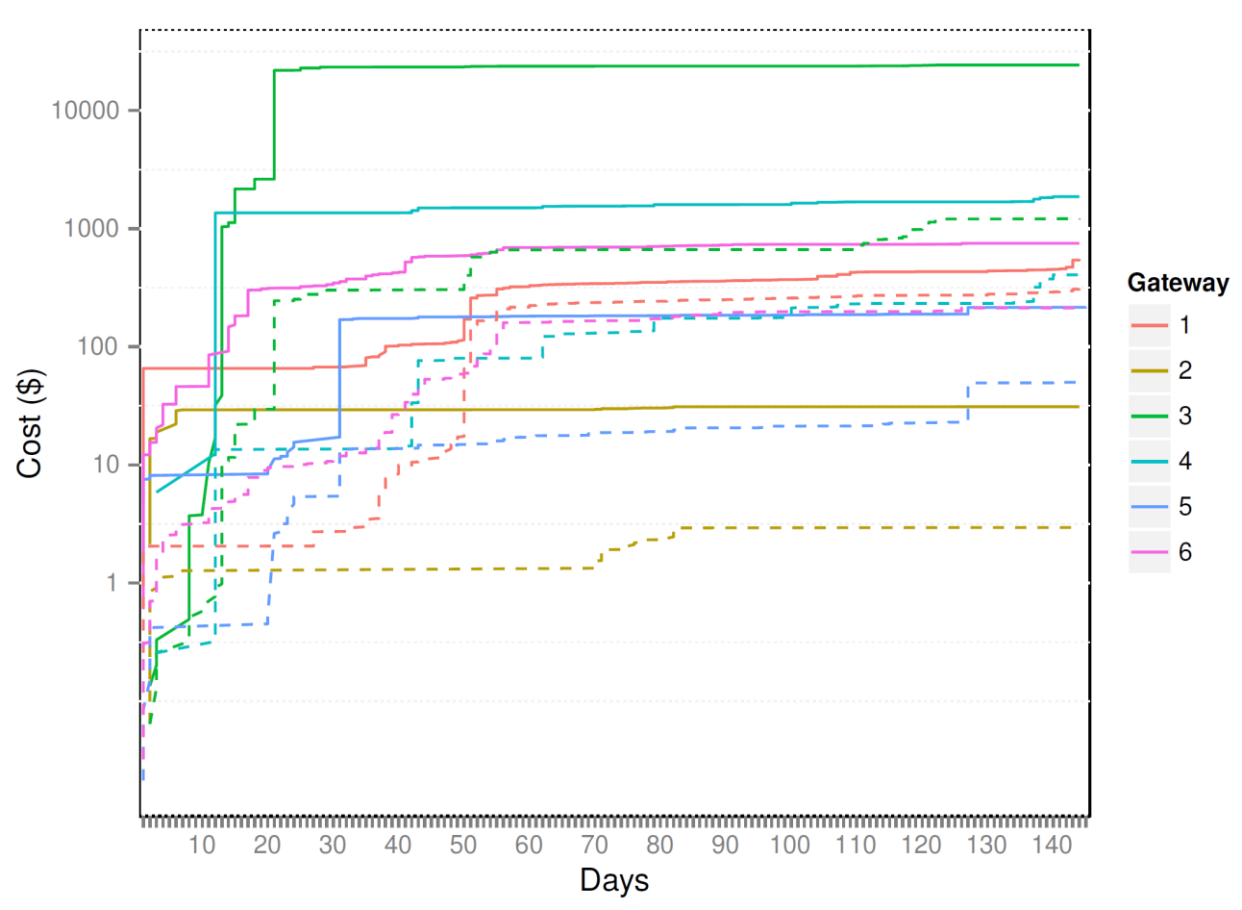
Single Instance, Single AZ (SI-SAZ)

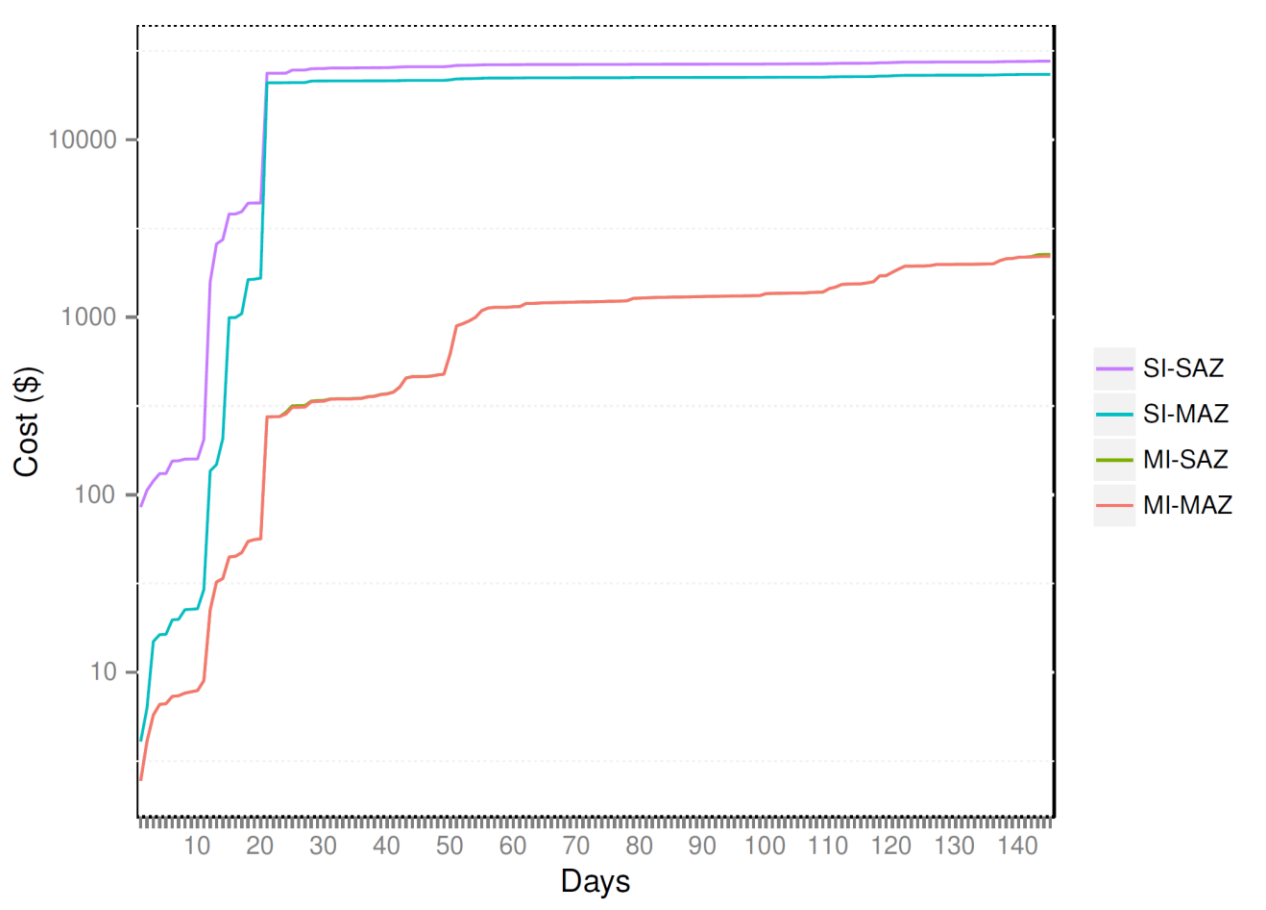
Single Instance, Multi AZ (SI-MAZ)

Multi Instance, Single AZ (MI-SAZ)

Multi Instance, Multi AZ (MI-MAZ)

Cost-aware Results





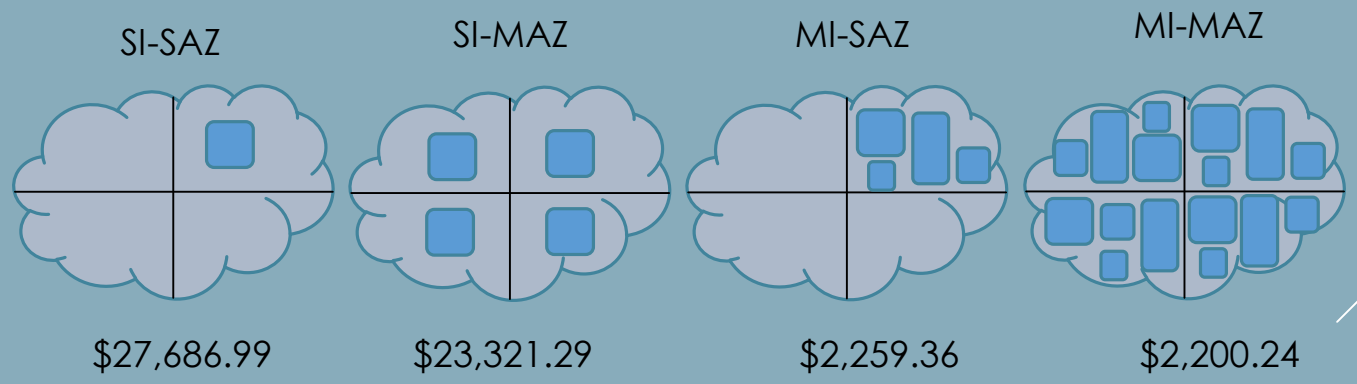
Cost/Day

Cost-aware Results

Increasing the search scope to include multiple instance types and availability zones results in cost savings between 43% and 95% per gateway

Overall reduction in cost of 92.1%

The difference between MI-SAZ and MI-MAZ less than 1%



Profiling Application Executions

What

- Matching tools to resources
- AWS is flexible
  - 38 instance types
- We need a way to determine resource usage
  - Identify suitable instance types
- Trial and error is tedious

Why

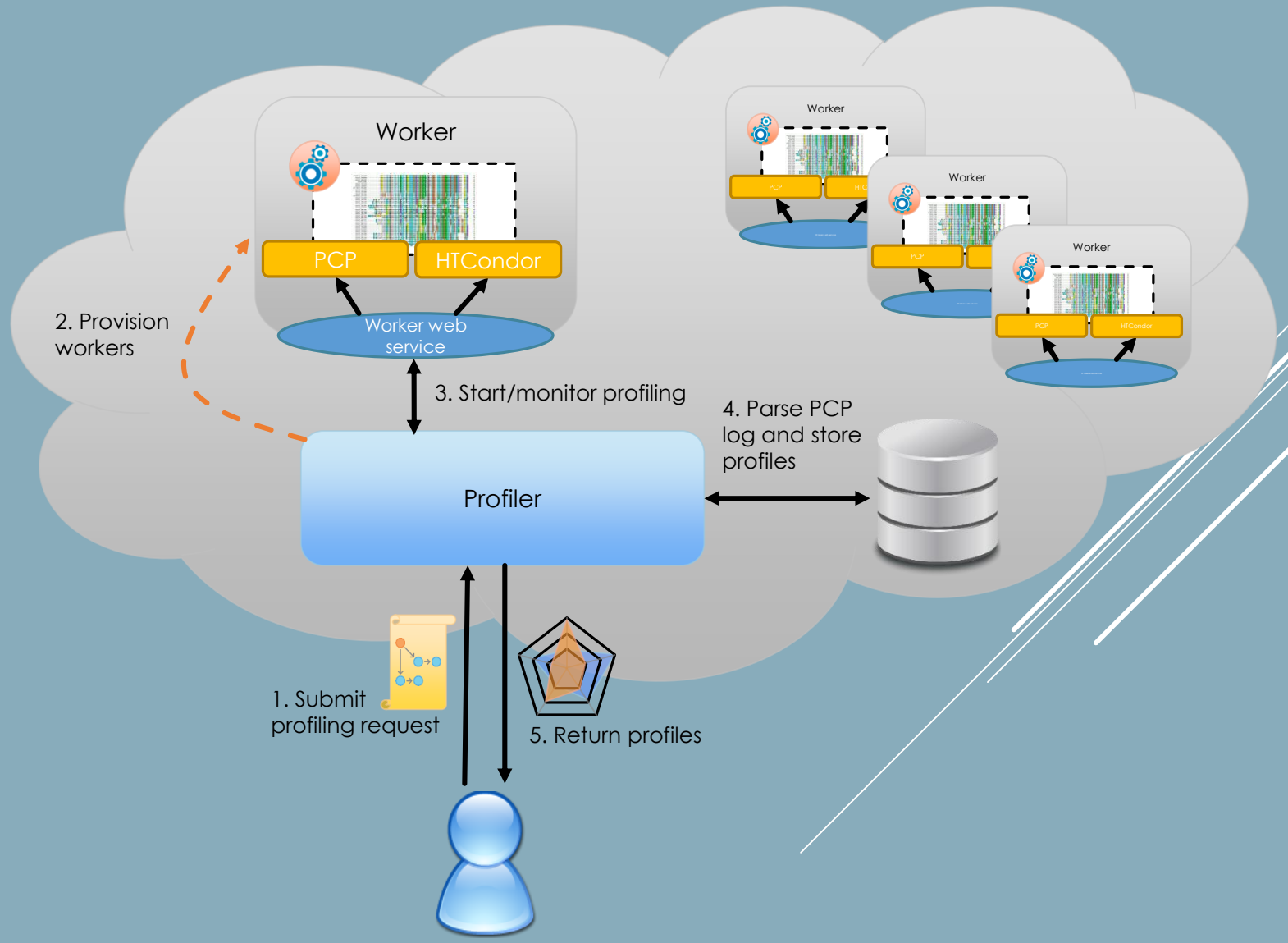
- Predict execution time and cost
- Enable co-allocation of jobs
- Minimise monetary cost of execution
- Maximise performance

How

- A profiling service!

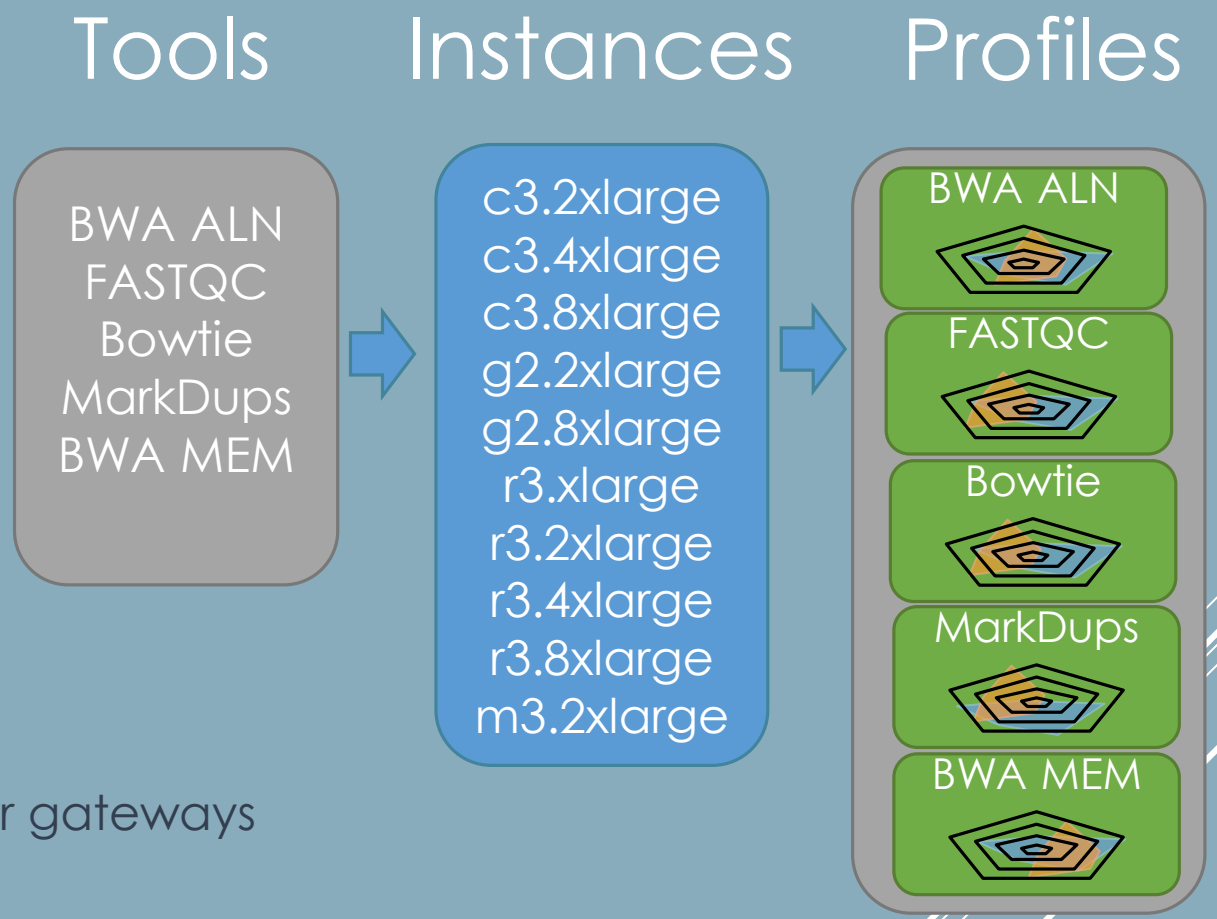
A Cloud Tool Profiling Service

1. Describe profile requests in JSON
2. Provision resources and apply a profiling Web Service
3. Use Performance Co-pilot (PCP) to capture usage
4. Capture and process PCP logs
5. Return profiles as JSON (or logs via s3)

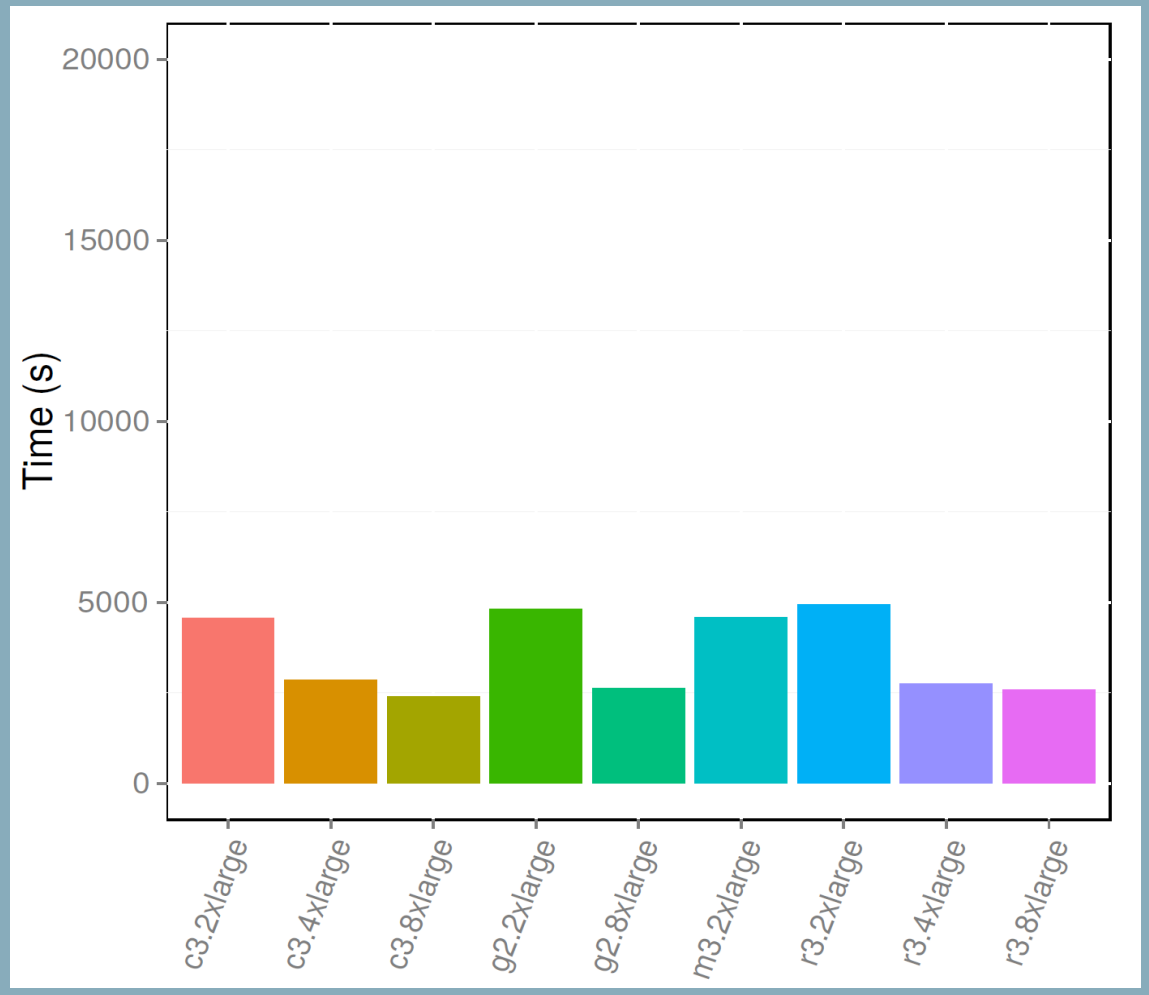


Evaluation

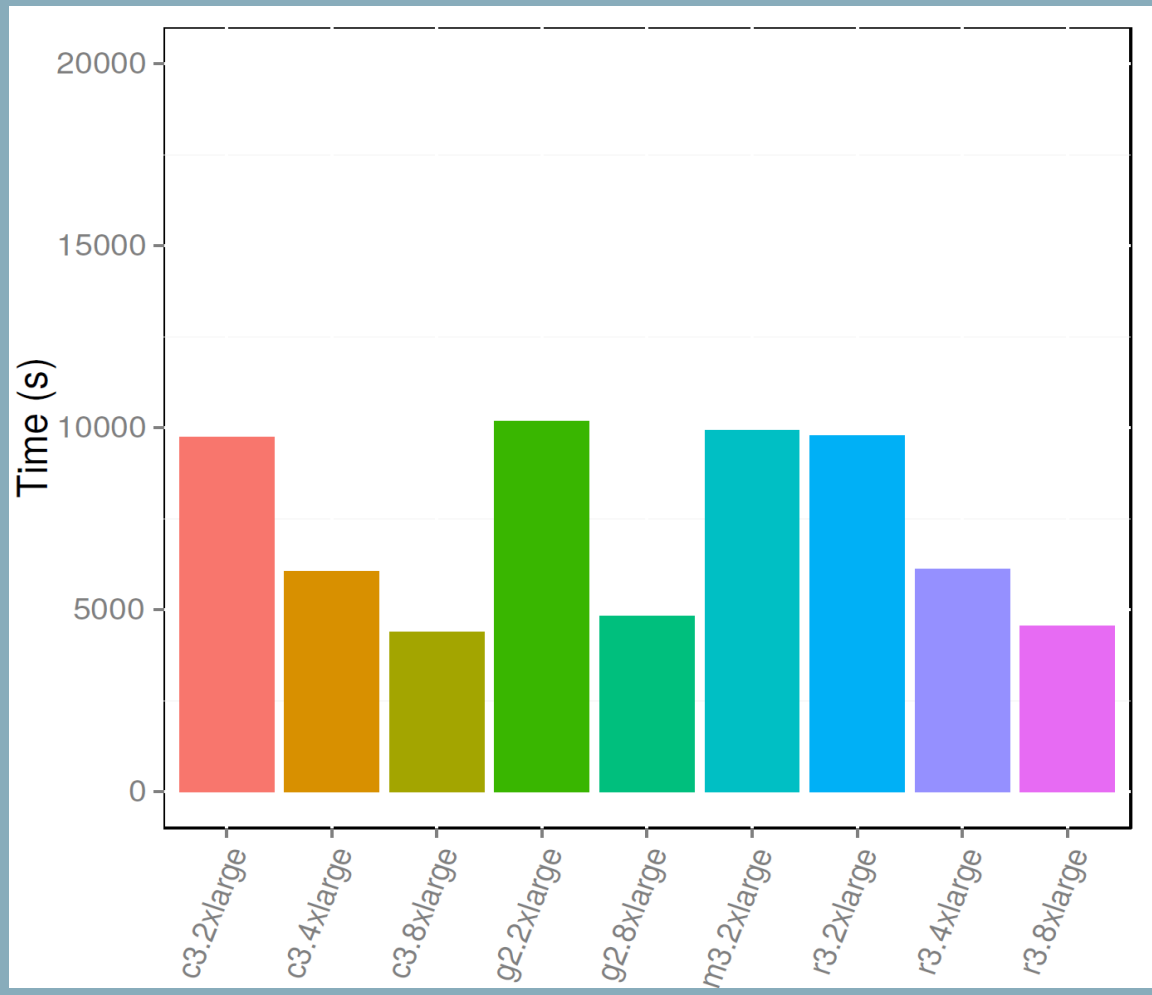
- 10 instance types
  - Instance storage (not EBS)
- Five Globus Genomics tools
  - Range from 20-90 minute exec time
  - Account for 17.7% of total compute across four gateways
- Capture:
  - Execution time, CPU utilisation, memory utilisation, disk, and network
- Assess impact on cost



Execution Time

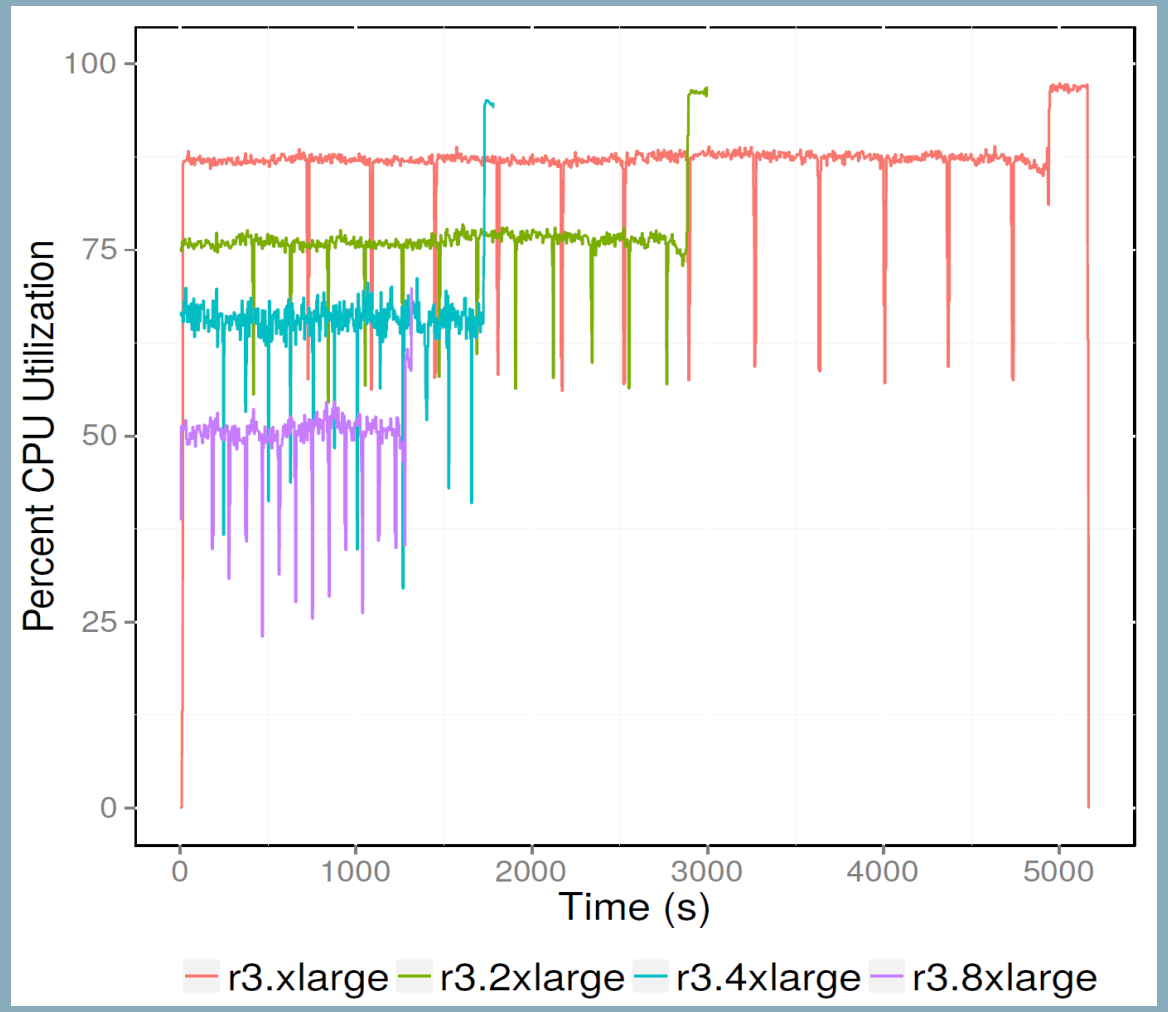


BWA MEM

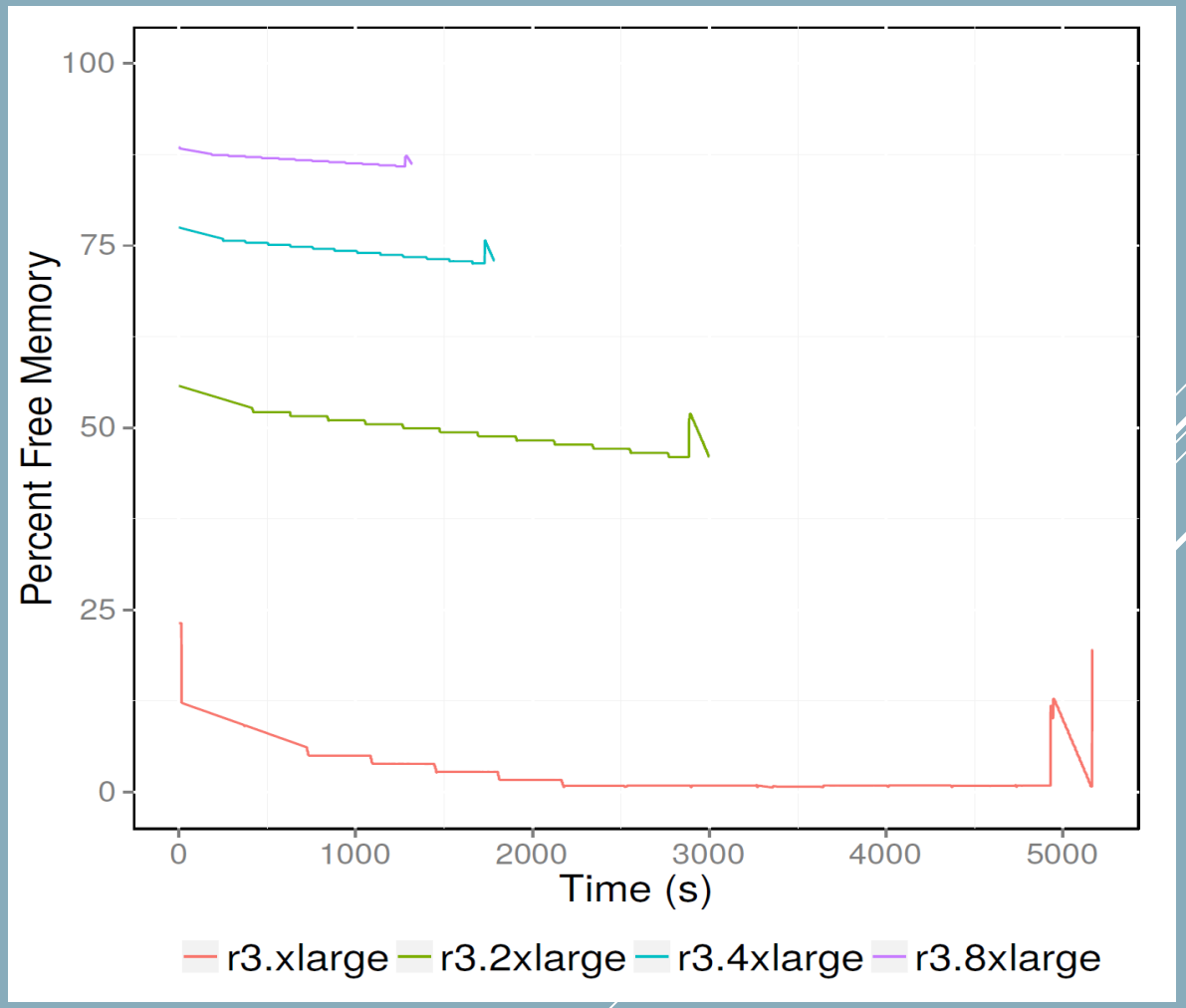


BWA ALN

Utilisation (Bowtie)



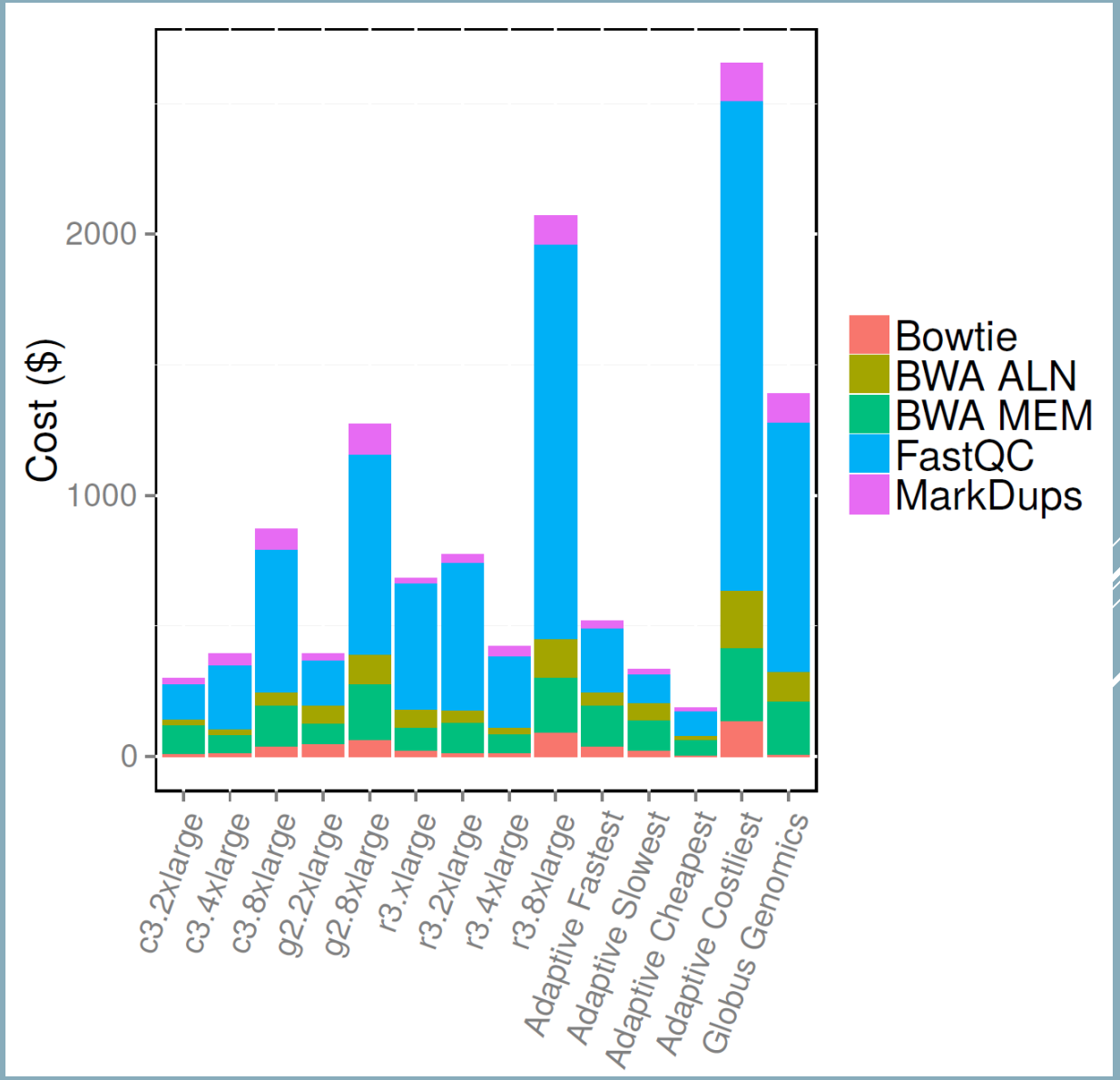
CPU



Memory

Spot Instance Price

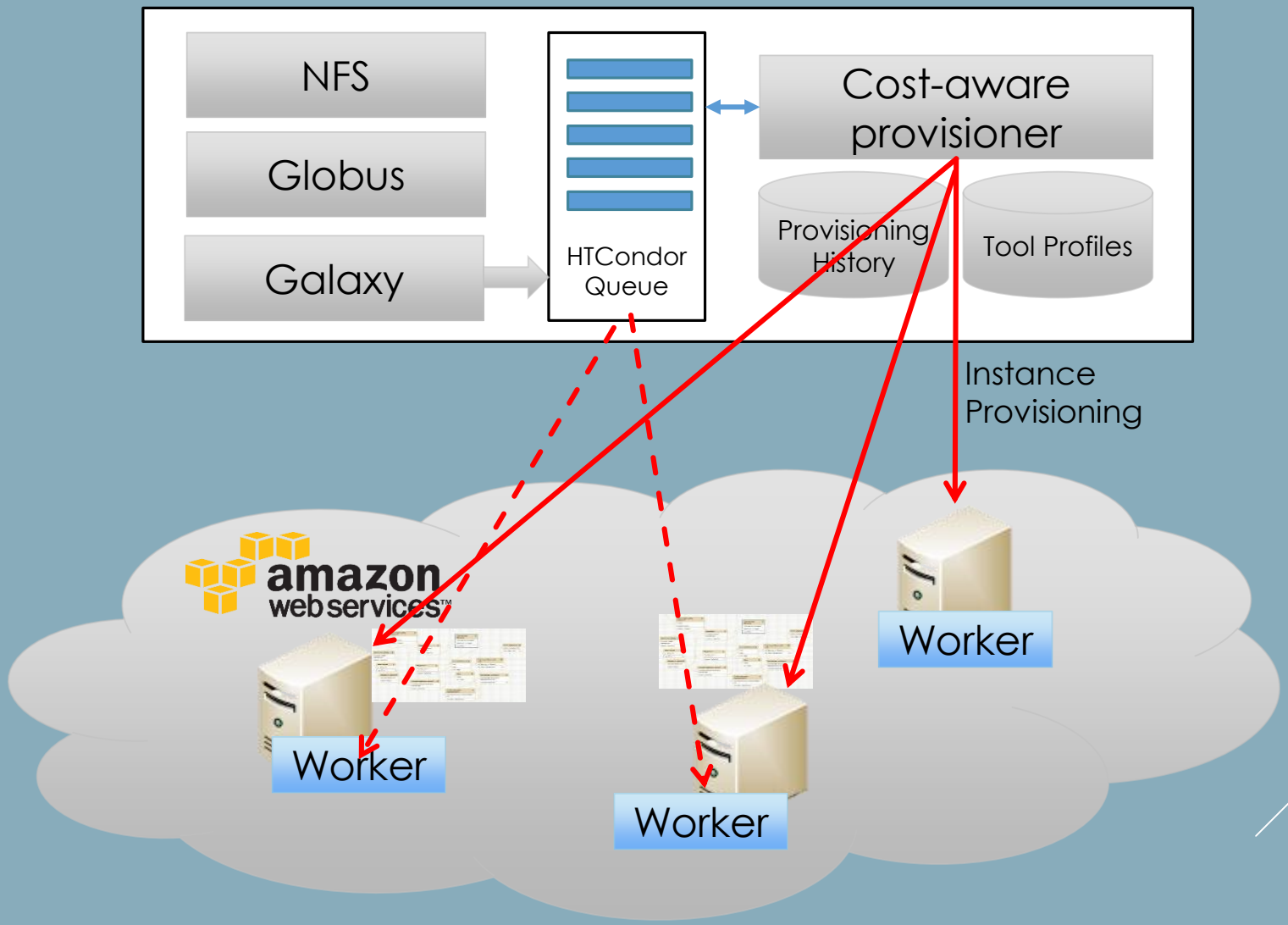
- A dataset of 2000 production jobs from GG gateways
- Use submission time and AWS spot price history to determine price per instance type
- Adaptive fastest, slowest, cheapest, costliest = quickest, slowest, cheapest, most expensive instance types
- Globus Genomics = naïve instance/zone selection



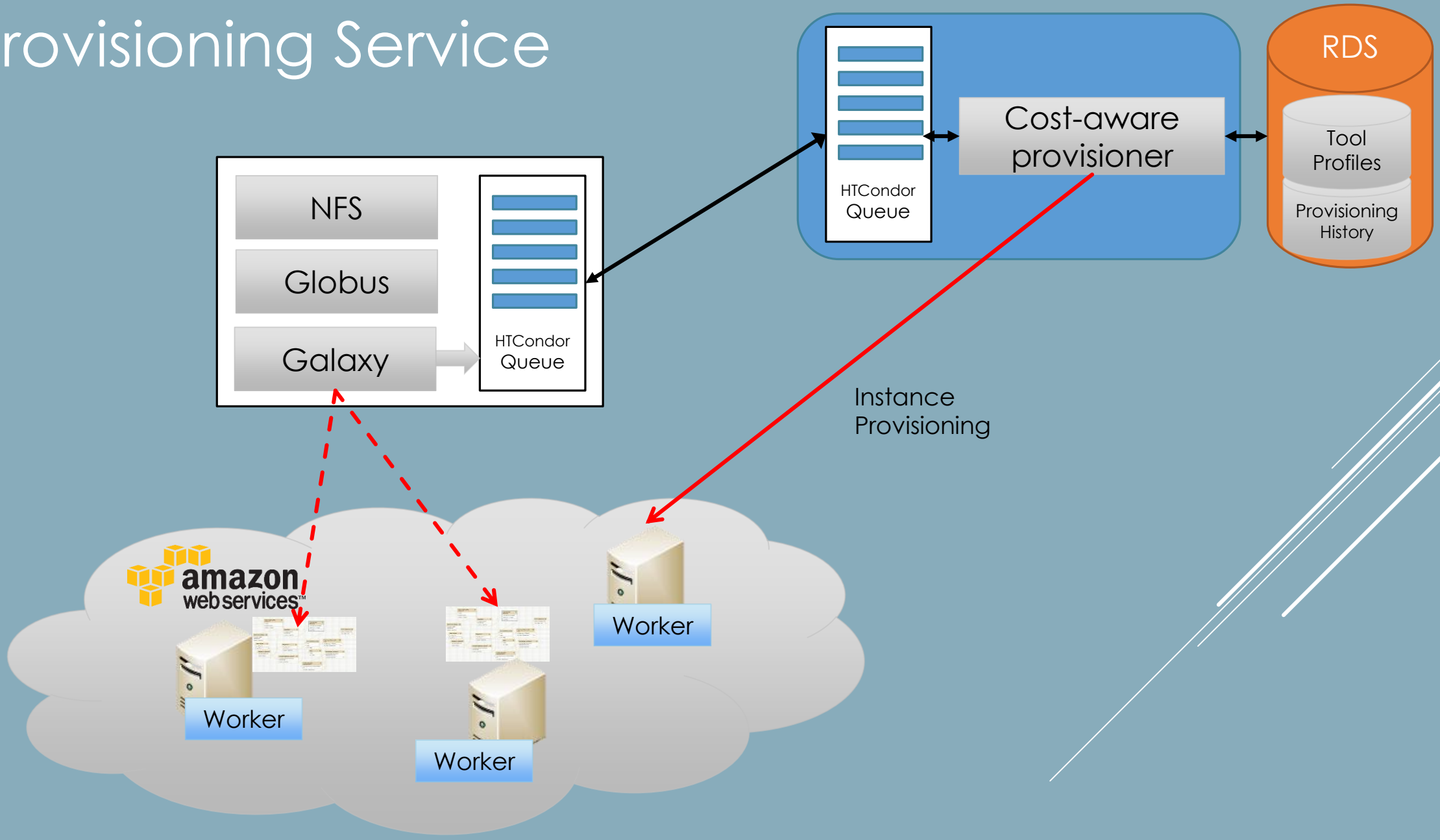
Provisioning as a Service

- A service to acquire and manage cloud resources
- Gateways subscribe to the service
- Combines profiles and cost-aware techniques
- Monitor a queue (HTCondor/Spark) and provision resources on demand

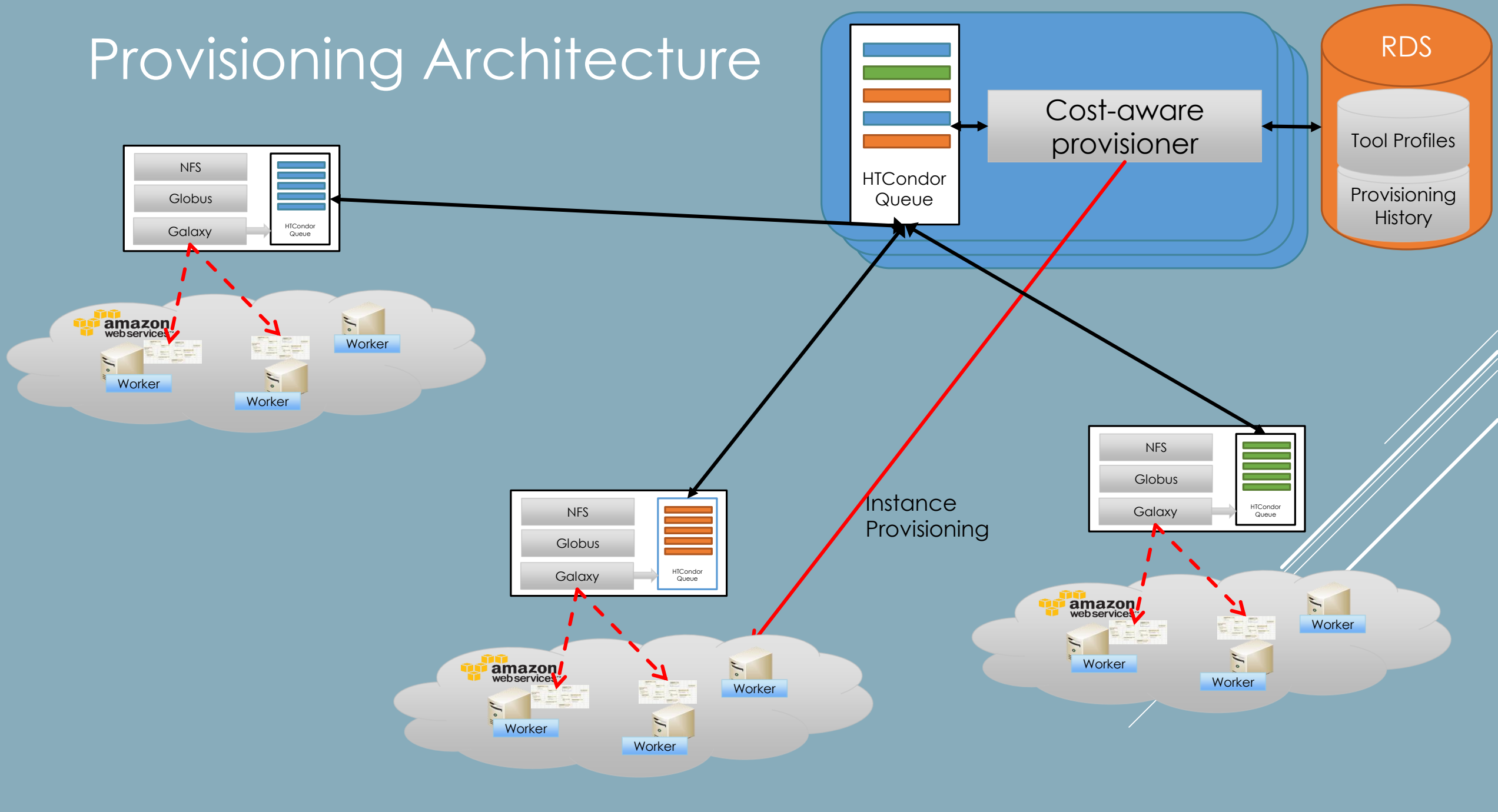
The Globus Galaxies Platform



A Provisioning Service

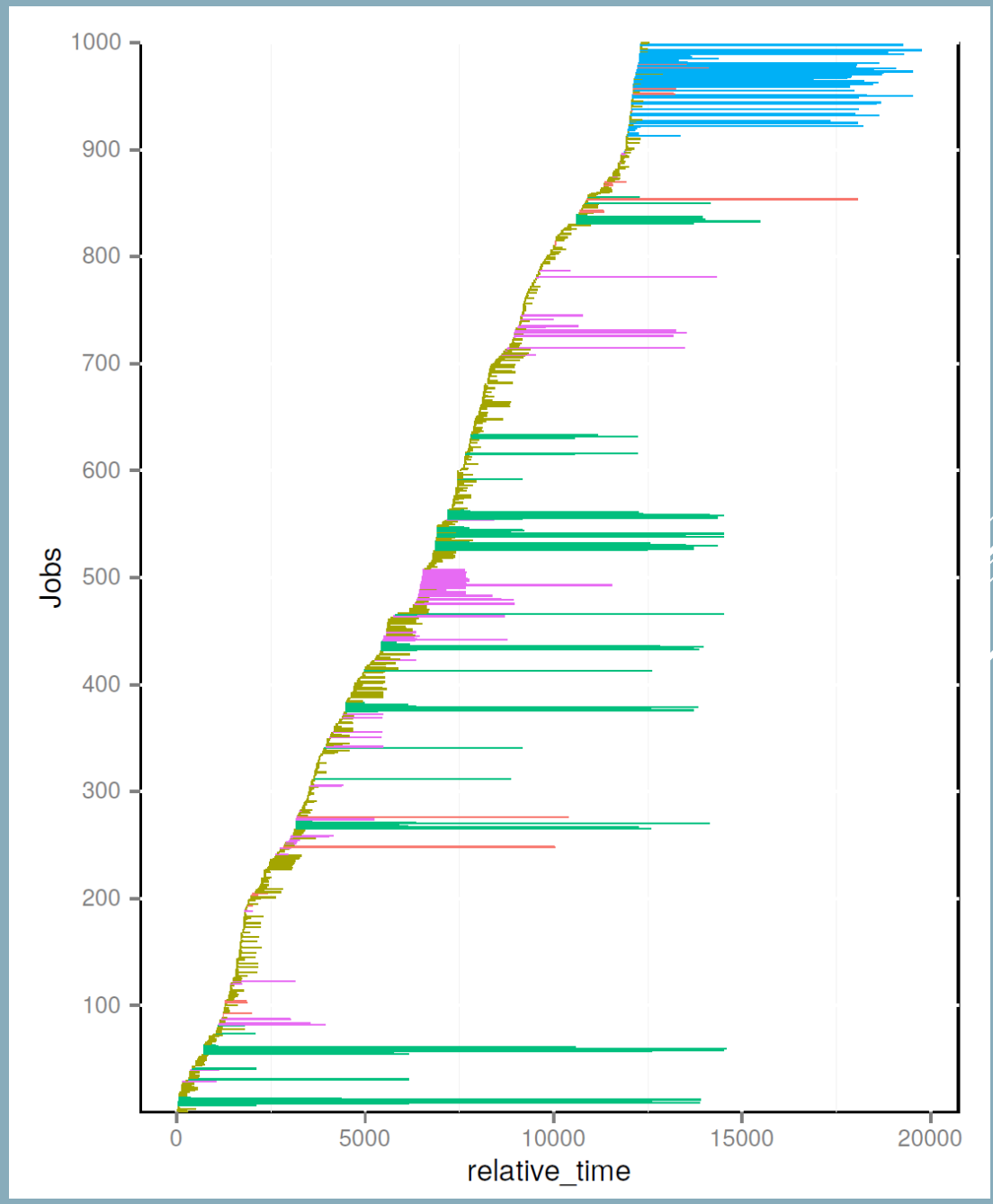


Provisioning Architecture



Evaluation

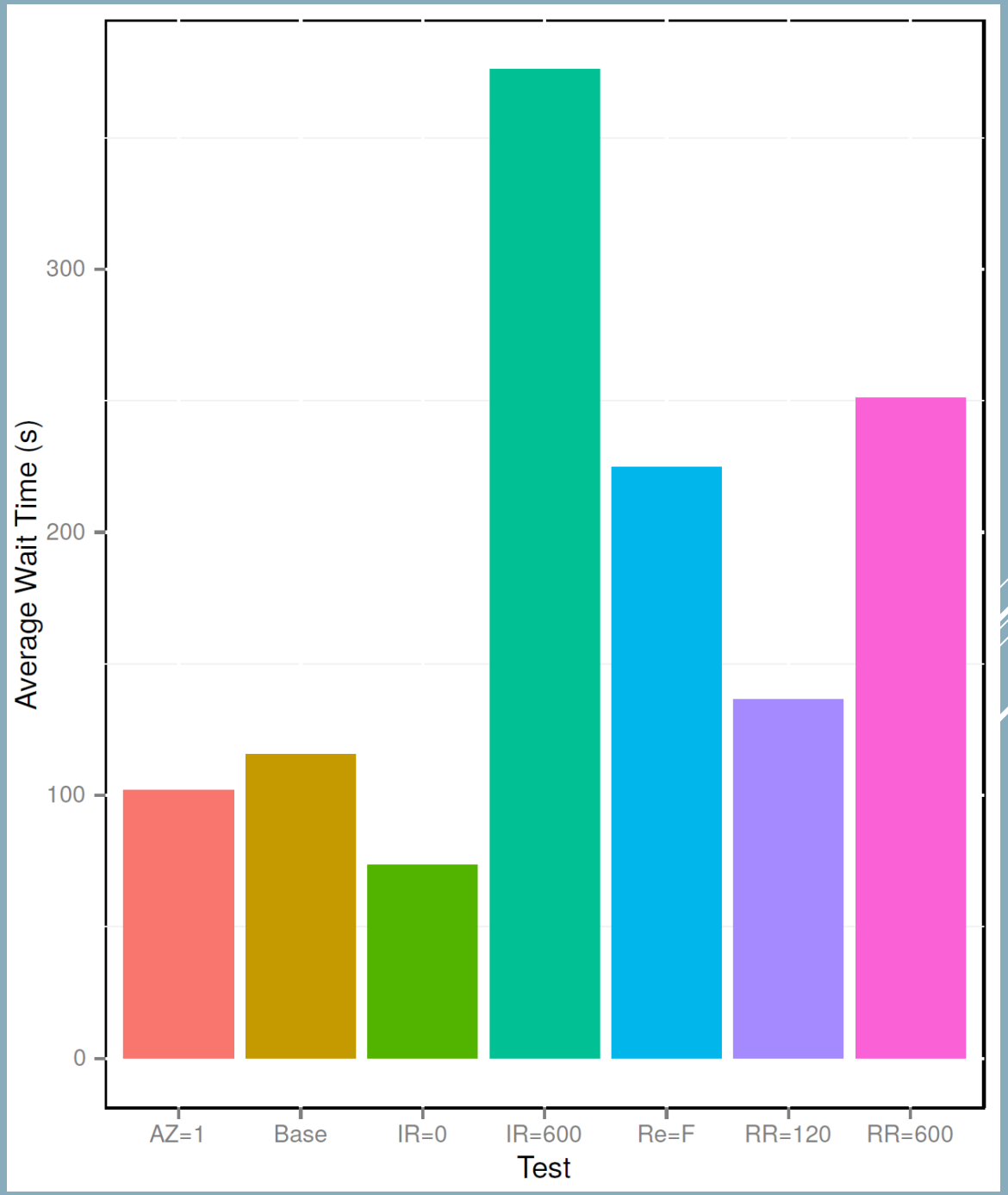
- Tune configurable parameters and see how it changes performance
  - Run rate, idle requirement, repurpose requests, resource restrictions
- A reproducible dataset of production GG workloads (busiest days from 5 gateways)
  - 1000 jobs (total ~8500 in 24 hours)
  - ~3.5 hours of jobs
  - ~8 hours execution duration
  - Transform exec time (sleep) by instance type
- Monitor number of requests, fulfilled instances, and cost to fulfil workload



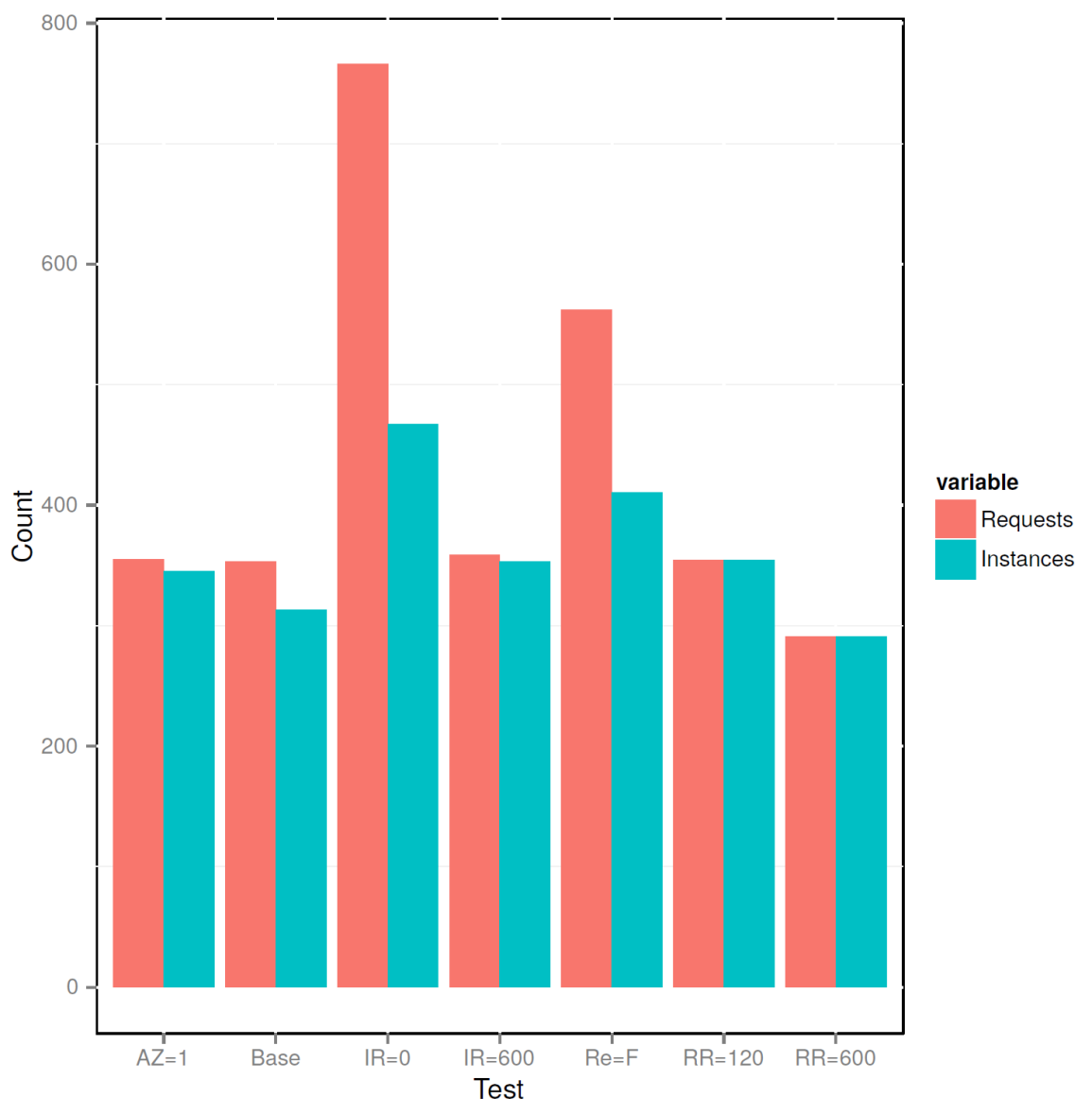
Average Wait Per Job

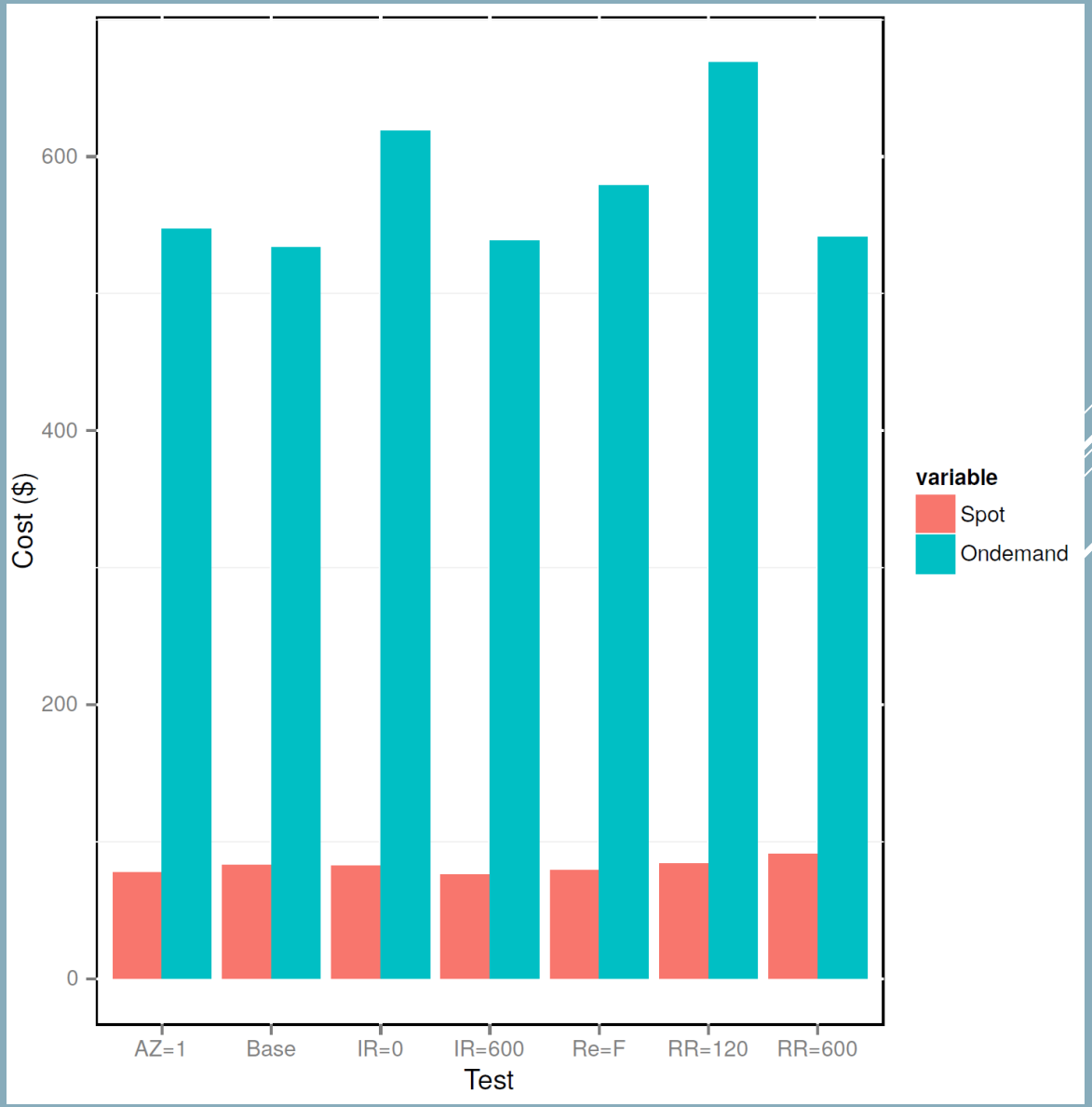
- Dashboard like configuration

- **AZ=1**: single availability zone
- **IR**: Idle requirement of jobs before resources are provisioned
- **Re**: Whether or not requests are repurposed once orphaned
- **RR**: Run rate, or interval between provisioning rounds
- **Base**: base case (Re=T, RR=5, IR=120,AZ=All)



Instances and Cost





Summary

- Using the cloud naïvely can increase costs and decrease performance
- There are simple techniques that can substantially reduce the cost of using the cloud
- Matching tools to resources is essential for effective cloud use
- The provisioning service and profiler are available on github

Thanks!

- Questions?