Running ATLAS at scale on Amazon
Spring 2015 HEPiX, Oxford UK

Michael Ernst<mernst@bnl.gov>
Carlos Gamboa<cgamboa@bnl.gov>
John Hover<jhover@bnl.gov>
Hironori Ito<hito@bnl.gov>
Michael O'Connor<moc@bnl.gov>

Outline

Background/Context

Compute

- Capacity Management
- Image/VM authoring/configuration

Networking

Data, Storage, Workflow

Scaling Issues/Tests

References/Links

ATLAS Standard Processing Model

Data (mostly) pre-placed around the world out-of-band by Dynamic Data Management (DDM) system: Rucio, File Transfer Service (FTS)

PanDA (Production and Distributed Analysis)

- Pilots submitted to Grid “Compute Elements” (CEs) at sites, mostly driven by data location from Rucio.
- “Late-binding” pilot-based system. Pilots acquire slots, then request “real” jobs
- Data pulled in by job; either whole file(s), or remote open()
- Results staged out to grid “Storage Elements” (SEs)

Output data re-used in place and/or migrated to Tier 1s with tape storage for permanent archival.

Previous/ongoing ATLAS cloud efforts/activities

Some smaller-scale commercial cloud R&D (Amazon, Google, Helix Nebula) performed at BNL and CERN.

Currently running at medium-large scale (1000 jobs) on ~10 academic clouds using University of Victoria Cloud Scheduler.

Heavy investment/conversion to Openstack at CERN

Openstack cluster at Brookhaven National Lab (720 cores)

Many other facility-based ventures and experiments...

Current RACF AWS Project

Amazon grant (~$US200K) to demonstrate general-purpose feasibility ***at full ATLAS scales***.

- Not R&D any more. Enable ATLAS to run on EC2 economically, for all job types, without scaling limitations.
- 12+ RACF, ATLAS, BNL, and ESNet people involved, across multiple groups and institution.
- New Scientific Computing group at AWS (5+ person team dedicated to BNL, with additional as needed). Also engaging with CMS and CERN.
- Weekly “Tag-Up” phone conference.

This has required addressing:

- **Compute:** Capacity and scaling
- **Networking:** Bandwidth and congestion management
- **Data and Storage:** Data model, storage and transfer
- **Workflow:** Application innovations/changes to leverage context

Compute (1)

The struggle in this area is to get ***capacity***

3 US AWS Regions: us-east-1, us-west-1, and us-west-2

- Required setting up provisioning infrastructure and data storage to handle each region as separate resource (e.g. ami IDs).

Use Spot market for all execute hosts.

Spot bidding policy:

- ***What is the slot actually worth to us? Bid that.***
- Simple policy--avoids accidental self-harm, overages.
- Heuristic: Always bid $(.25 * on-demand price for that type)/hr.

Leverage all usable instance types meeting ATLAS minimum profile (2GB RAM, 20GB disk/core)

Compute (2)

In order to economically use Spot pricing, ATLAS needed to complete implementation of the PanDA Event Service (ES).

This service permits a pilot job to perform units of work smaller than a full ATLAS job, e.g. about 10 minutes.

- Intermediate results are stored in an object store.
- These are later merged to create final output (identical to what would have resulted from a full-length job).
- Intermediate objects can be discarded.

The Event Service was already planned, but accelerated for this project.

Cloud Provisioning

This deployed hardware expected to scale to 100k jobs.

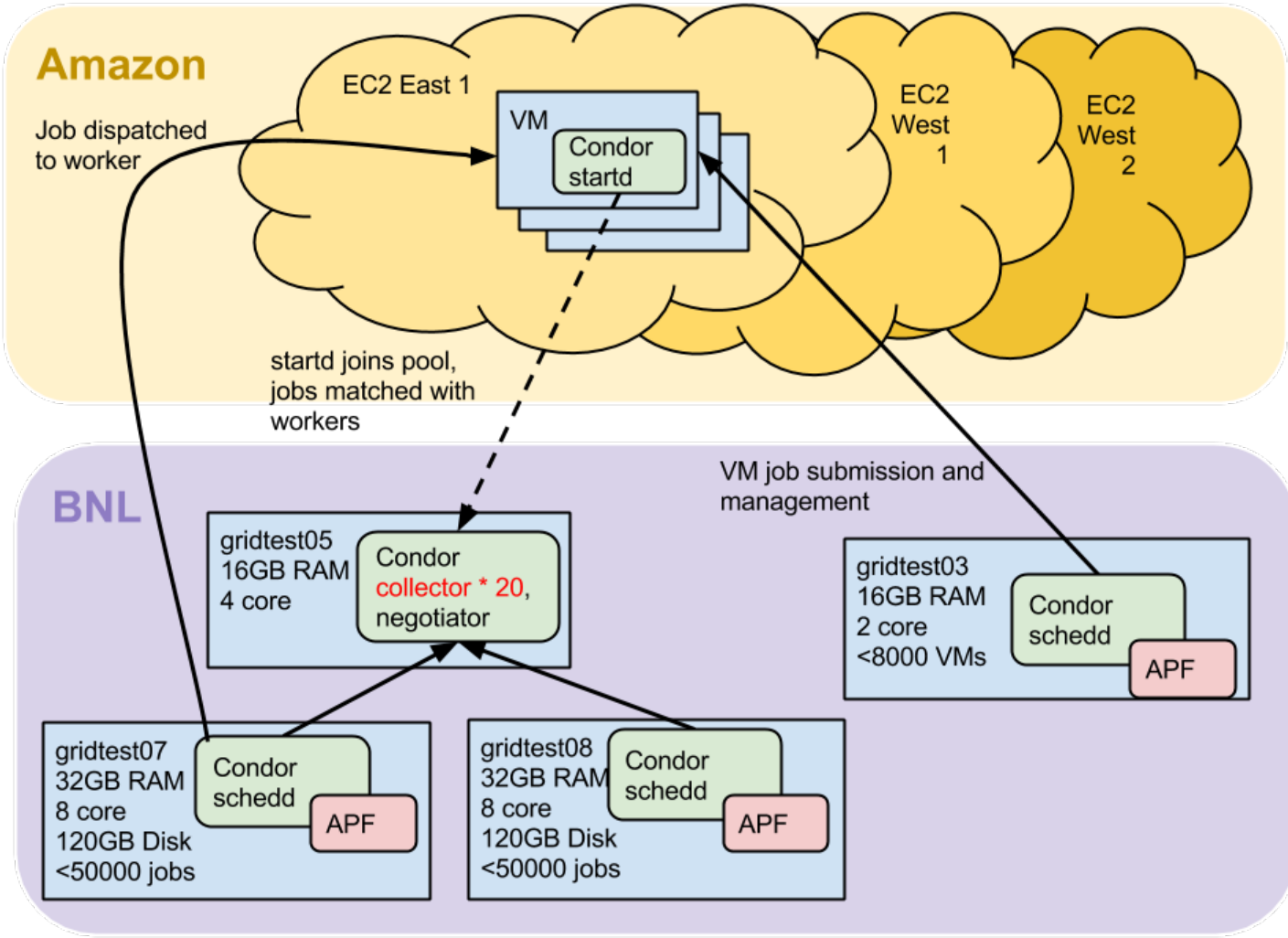


Image Authoring and Runtime Configuration

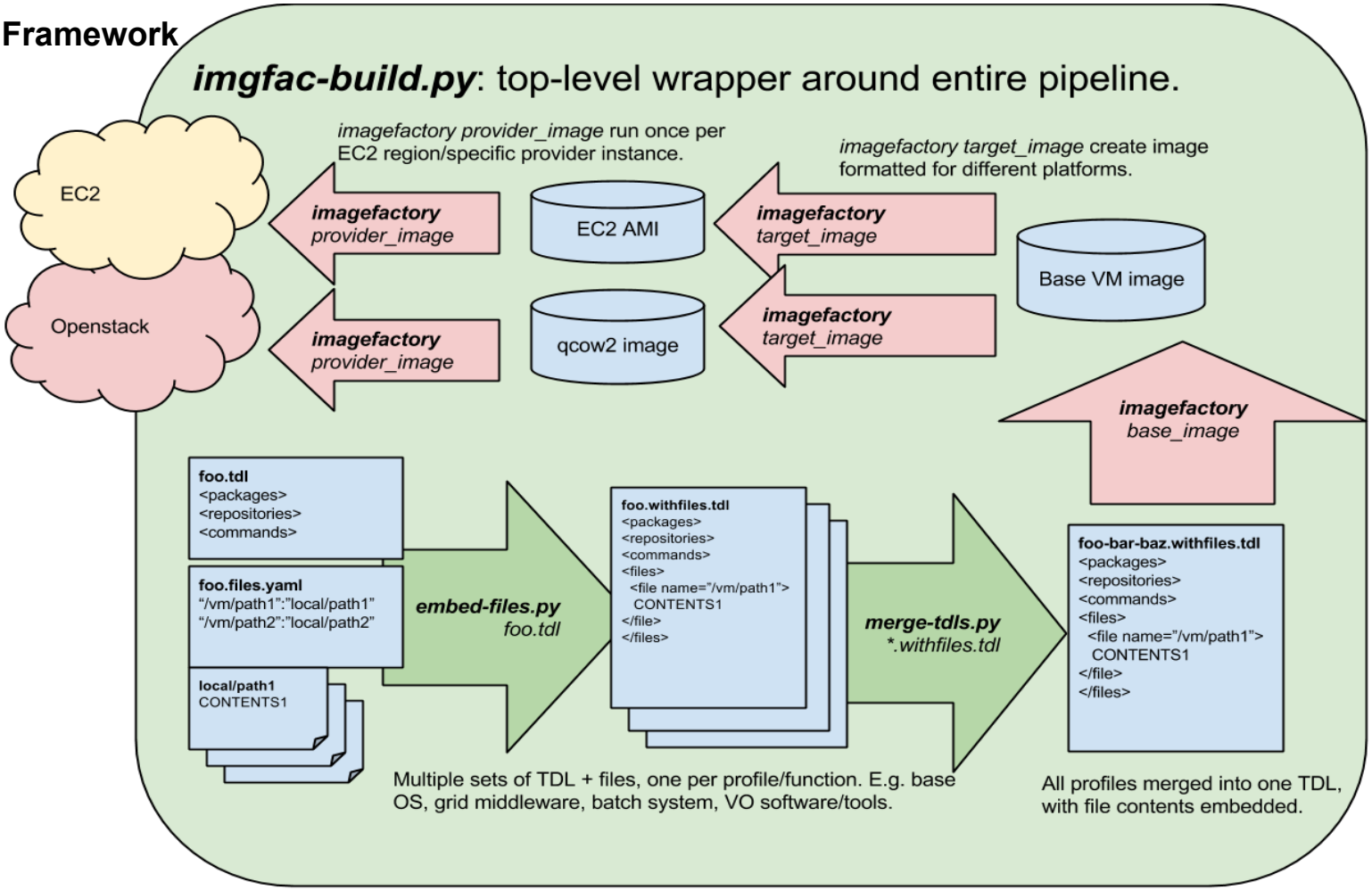
Design goals:

- Useful for ATLAS, but usable by other Open Science Grid (OSG) VOs.
- Eliminate runtime RPM installation. Fatal with O(1000) startups.
- Images deterministically reproduceable. No snapshotting.
- Provide the ability for other users to Do It Themselves (make toolset public).
- Flexibility between build-time and run-time customization. Both options OK.
- Open source only. Only use functions/services for which open source equivalents exist (EC2, S3). But not, e.g., Cloudwatch.
- Off-the-shelf, non-cloud (Puppet, Hiera, Condor, Yum) wherever possible. Off-the-shelf cloud (cloud-init, Imagefactory/Oz) only where needed.
- Keep custom parts small, simple, and/or optional.

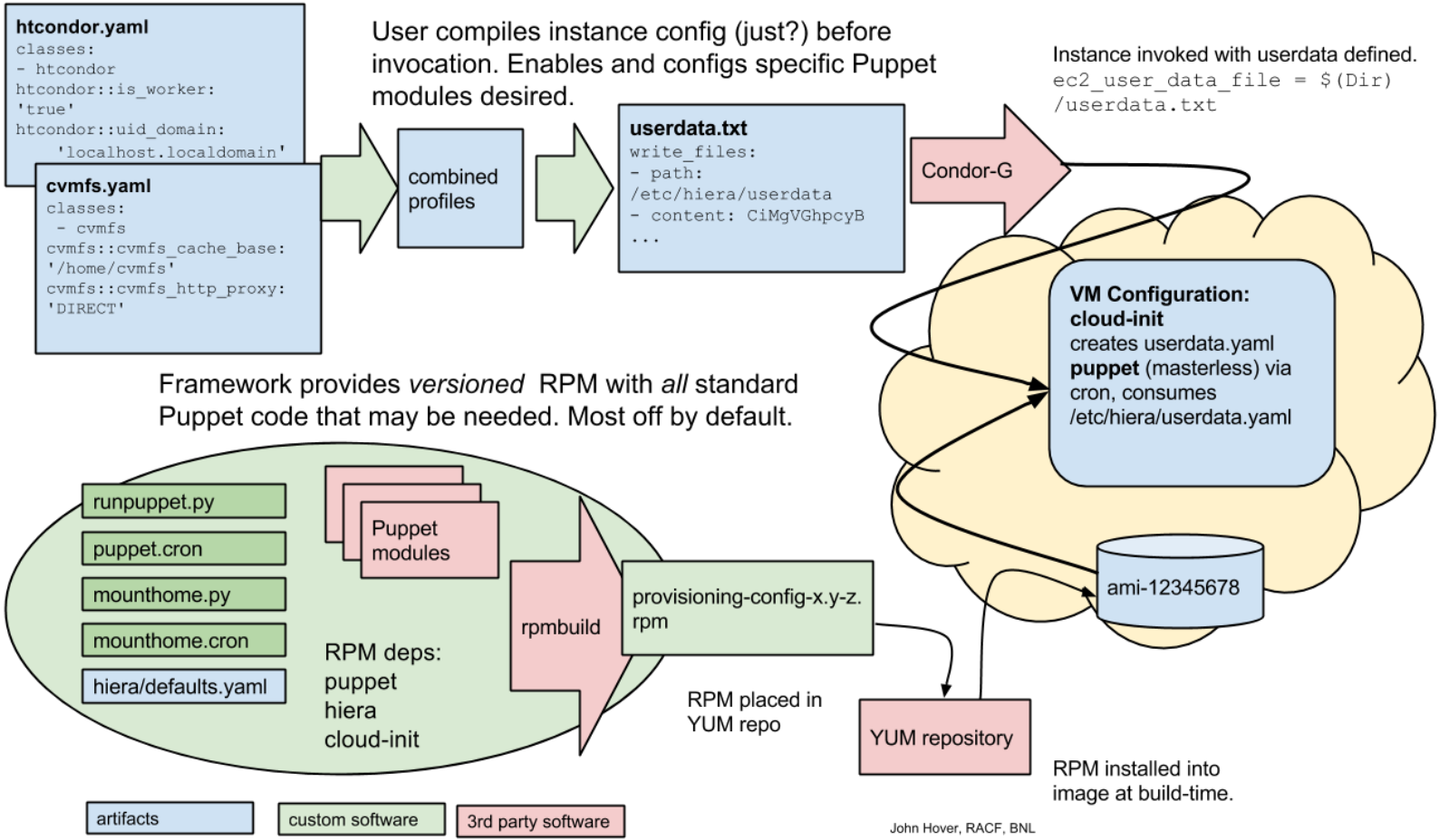
10,000 ft summary:

- Imagefactory 1.1.7 generates VMs from merged hierarchical templates.
- Masterless puppet consumes single Hiera file (injected via cloud-init write_file) at boot.

Build Framework



Runtime Configuration Framework



John Hover, RACF, BNL

Issues, Observations, Next Steps

- Amazon rolling out HVM instance types, but Imagefactory only supports PV types. Alex Zaytsev wrote a converter, and we are working with Imagefactory devs for native HVM support.
- **mounthome.py** runs at boot, finds all ephemeral drives, groups them, formats them, and mounts them. Will be converted to Puppet module.
- BTW: Running in parallel on BNL Openstack with same system.
- Orchestrating the creation of heterogeneous collections of nodes is still hard. Heat/CloudFormation doesn't provide a way to tell nodes information about each other. This is necessary for legacy apps. MCollective?
- Squid deployment a simple example of need. Looking into Shoal.

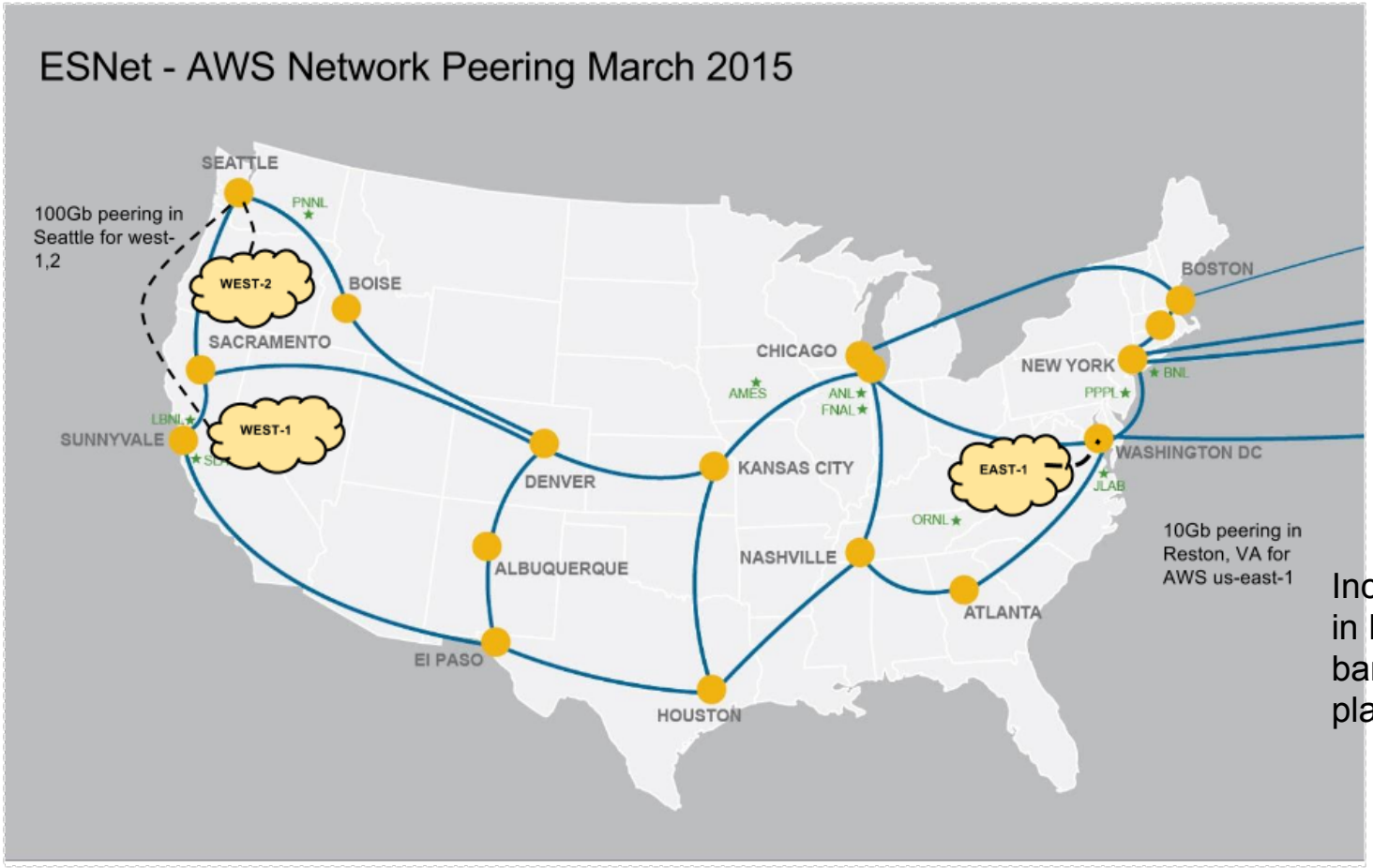
Networking (1)

Peering with ESNet (soon Internet2 and Geant) to allow data flows between experiment dedicated storage and Amazon

- Peering in Reston VA (10Gb), and Seattle WA (100Gb).
- **DirectConnect** to BNL via ESNet (10Gb)

DirectConnect is the router/advertising and flow config. Enables:

- QoS/congestion control.
- Virtual Private Clouds for custom topology (if needed).
- Public internet IPs with host in EC2.



European GEANT peering on the way.

Increases in East bandwidth planned.

Networking (2)

This arrangement was a prerequisite for Amazon's agreement to waive ***data egress charges*** (as long as they make up less than 15% of the total bill.)

Peering prevents our traffic from travelling over Amazon's commercial internet service providers.

DirectConnect allows ESNet to prevent our traffic from interfering with other BNL campus internet users.

So far, it appears it will be easy to keep data transfer under 15% (we did ~10% during Nov 2014 test, with worst-case data handling). If we keep output in EC2 and continue processing, transfer percentages will go down.

Data and workflow (1)

Make clouds **first class** ATLAS data storage component.

ATLAS standard model of endpoint-based “storage elements” (SEs)

- SEs integrated into data transfer and bookkeeping systems.

Previous testing used static BNL storage (already in the model) for input/output.

- But, creates high site-cloud network bandwidth usage, limiting scale.

New approach puts standard SEs in EC2 (one per region), with back-end mapping to S3 via s3fs.

- ATLAS systems being adapted to use S3 natively soon, i.e., S3 will ***be*** the Storage Element.

ATLAS Issues and Solutions

Traditional ATLAS workflows have 3+ hour (up to 5 day) jobs. No provision for termination (no checkpointing).

- We accelerated development of ***event service***, where pilots stage out intermediate results after each event.

Difficulty switching to S3-based data. No S3 native storage endpoint concept in ATLAS model.

- S3 support in beta for File Transfer Service (FTS). Already used to push data to S3.
- S3 support planned for ATLAS data management/catalogue system (Rucio)

Data and workflow (2)

Event service needs storage for intermediate results. Initial phase uses BNL Ceph object storage, with *merge jobs* running at BNL

Goal is to use S3 as intermediate storage, with merge jobs running in EC2.

But the standard approach to storage in ATLAS has been:

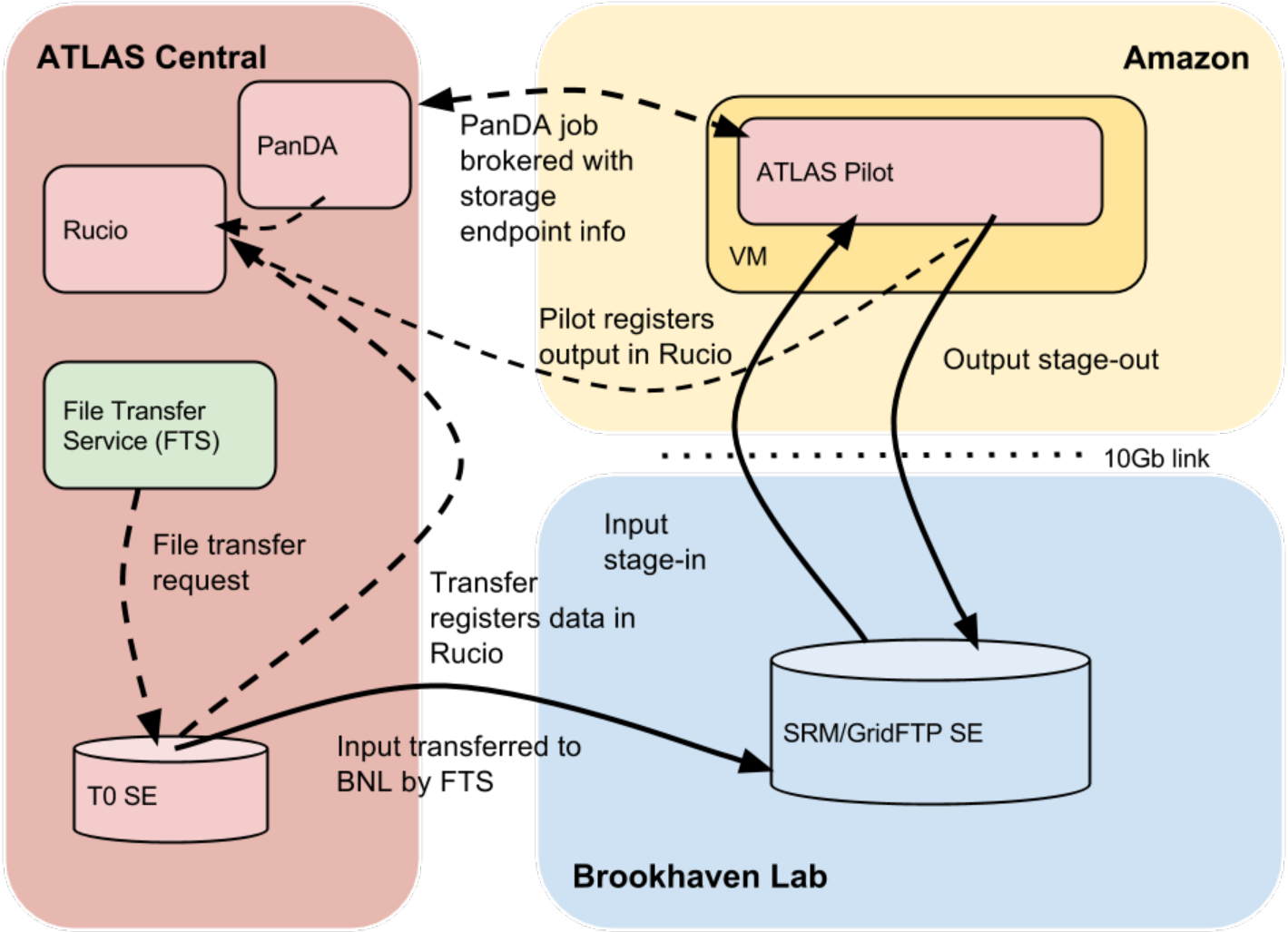
"delete only when you have to", since pulling from tape is slow.

With paid storage, like S3, we have to adapt to a:

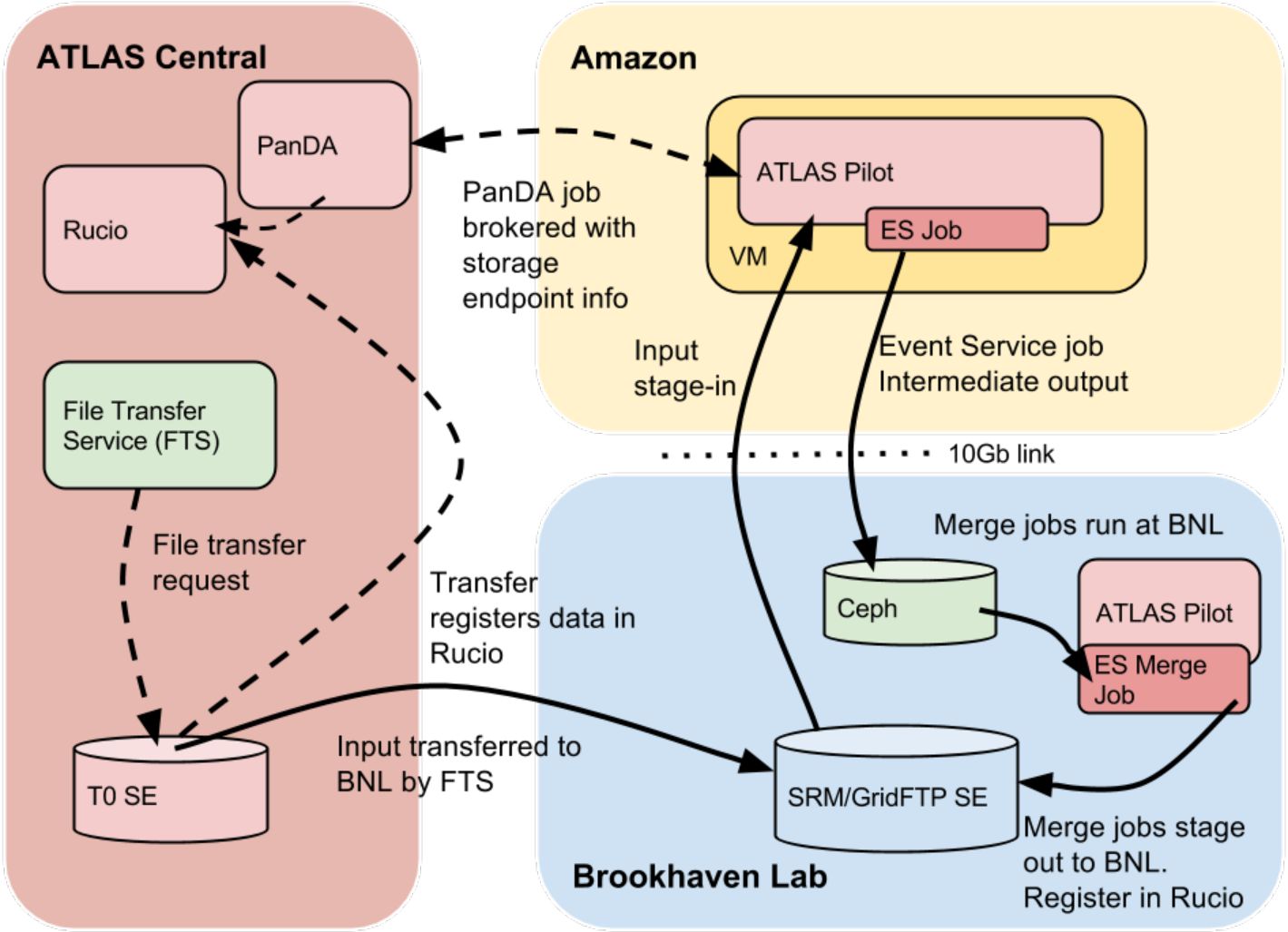
"actively delete anything you don't need"

and since cost is (size * time), our approach needs to be

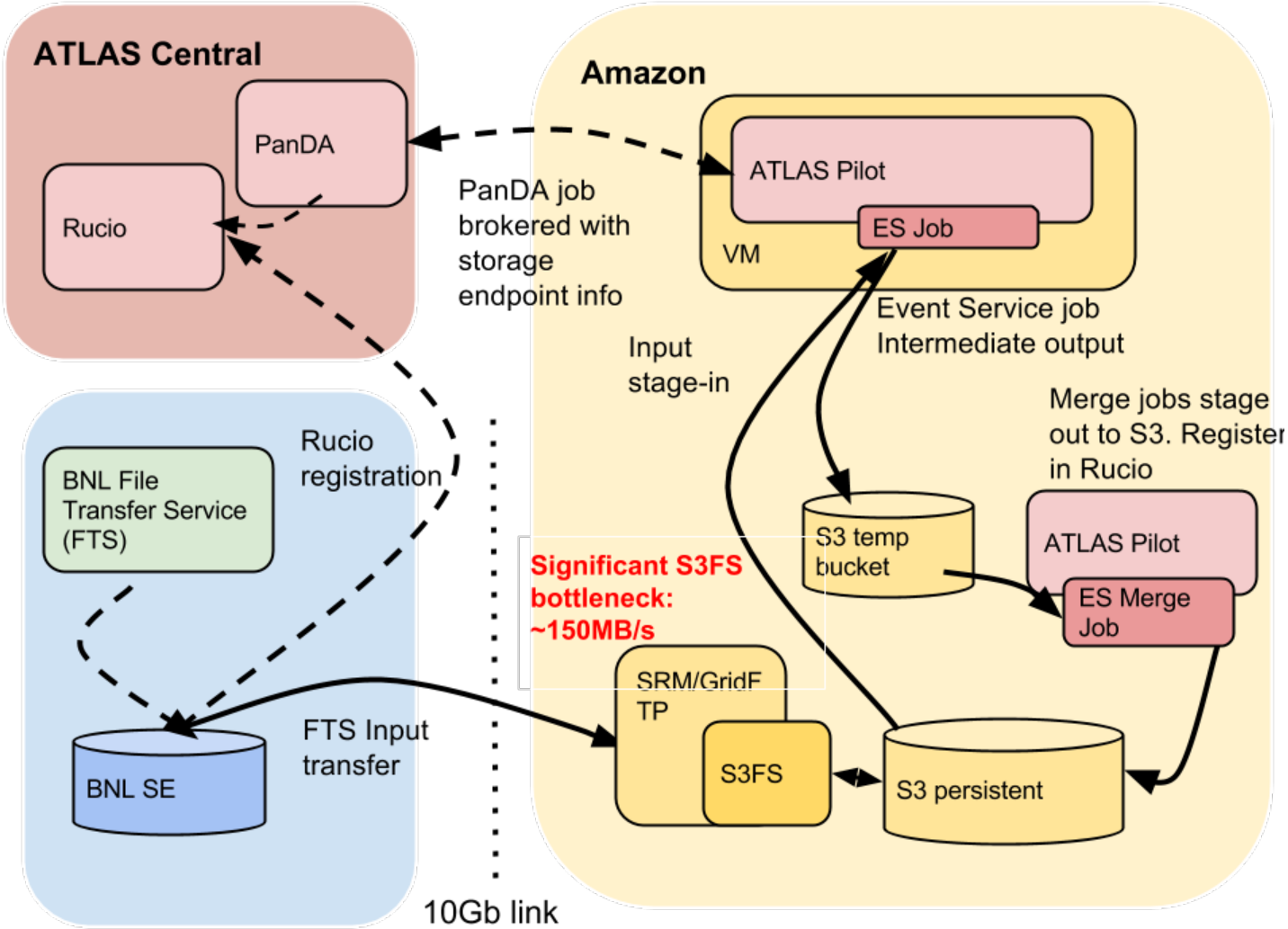
"actively delete **as soon as** you don't need it"



Every job transfers to/from BNL.



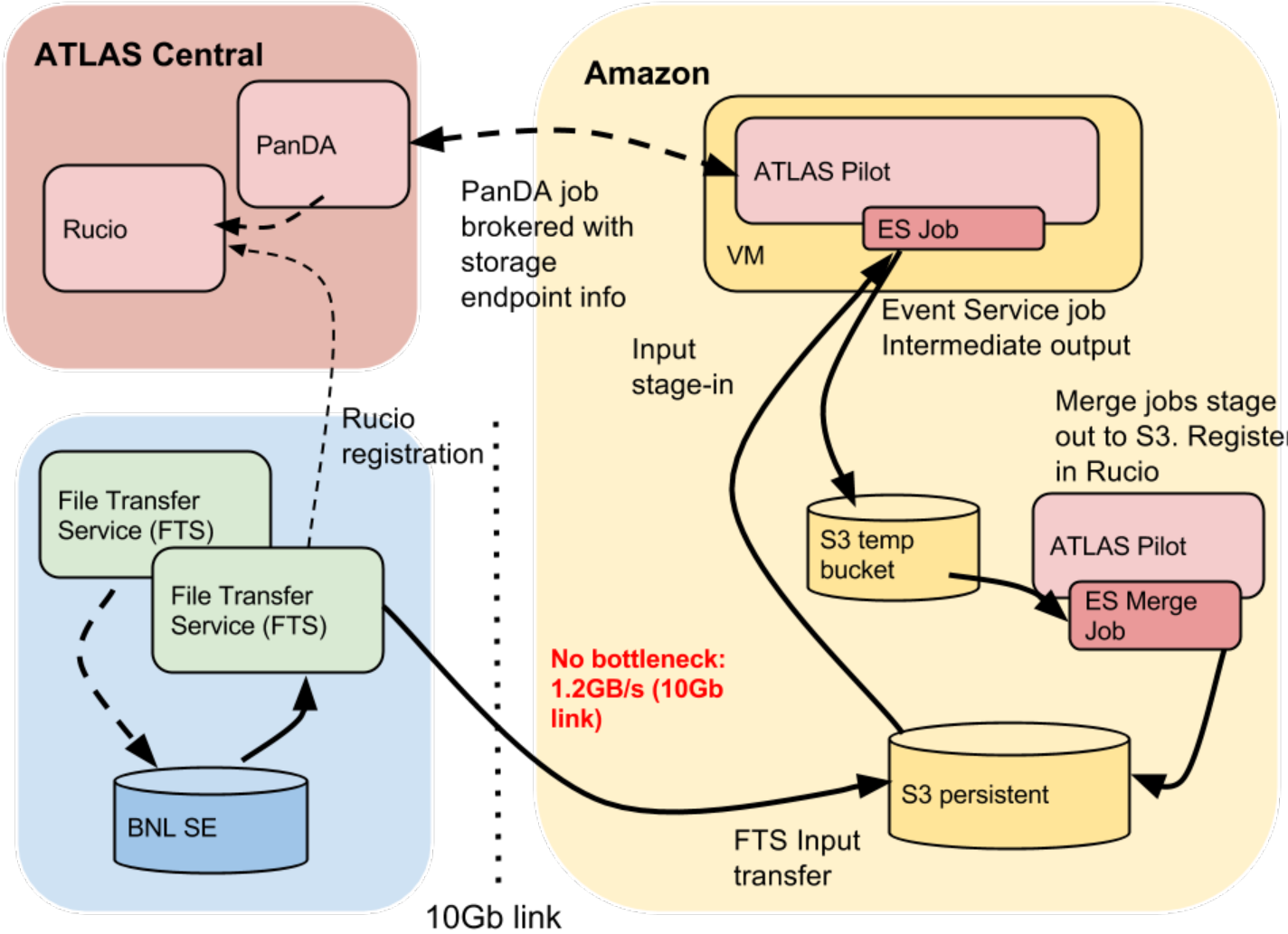
No distinction between EC2 regions. All share BNL data.



Separate setup for each EC2 region. Each now a Tier 2.

Addl. test results in extra slides. Not long-term solution.

Long-term goal: Run entirely within EC2, with link limitations only affecting SE to SE transfers. “Site” stage-in/out via S3.



Jobs only using S3, no scaling bottlenecks.

Data and workflow (2)

Checksums

- Allow customers to specify S3 object checksum algorithm?
- Currently ATLAS calculates its own rather than trusting the S3 MD5 checksum.

Application software

- Currently ATLAS jobs get their application software from CVMFS (Cern VM Filesystem), a read-only, global, HTTP-based software repository.
- The CVMFS client has beta support for S3 back-end. Would mean placing ATLAS apps in S3 (~3TB). Several $100/mo?

Scaling Test Runs (1)

Early 2013:

Amazon scaling tests focussed on Condor-level tweaks:

- Tune OS limits (1M open files, 65K max processes).
- Split daemons on several servers.
- Multiple collector processes, to avoid internal bottleneck.
- Enable shared port, CCB for networking efficiency.
- Enable session auth.

Scaling Test Runs (2)

November 2014: (First test under current project.)

- ~20,000 cores (job slots) on ~2500 nodes.
  - Duplicated the scale and functionality of the BNL ATLAS Tier 1 facility.
- 3 regions. 3 instance types (PV only). 7 days.
- Data staged in and out from BNL (no S3).
  - BNL_CLOUD, BNL_CLOUD_MCORE, and ANALY_BNL_CLOUD PanDA queues.
- Real workload: Mostly simulation but a bit of user analysis too. Lower job failure rate.
  - Saturated 10Gb link between BNL and AWS.
  - Moderate failure from spot terminations, ½ intrinsic performance of BNL workers. But run was economical.
- Spent $25,000 compute, $2500 data transfer. 10%

Scaling Test Runs (3)

Next run (April?):

- ~50,000 cores on ~6000 nodes (more 32-core nodes)
- 3 regions. 12 instance types (including HVM-only types).
- ~7+ days. Event service jobs.
- Regions-specific Panda queues: e.g. BNL_EC2E1, BNL_EC2W1

Focus on efficiency in addition to functionality, compile detailed cost/performance stats. Check Condor benchmarking.

Double-check economics of data transfer waiver.

Just waiting on:

- Pilot S3 or (SRM->S3 mapping) support for stage in/out.
- Confirmation of peering and DirectConnect for west-1,2 regions.

Scaling Test Runs (4)

Final(?) test

- ~100,000 cores (10,000 nodes?)
- 3 regions, ~12 instance types.
- Event Service only, with intermediate storage in S3
- Data stage-in and stage-out to S3
- Fully leverage peering/DirectConnect

This would temporarily duplicate the entire US ATLAS processing capacity. (Tier 1, 7 Tier 2s at 8 universities).

- Will require S3 support for Event Service *and* EC2-based merge jobs. Nice to have Event Service support for user analysis jobs.

References

This talk:

https://docs.google.com/presentation/d/1ZW3YBzbQkP50m_IpsS0gB9NiL4Hl8CnGYigm0p1gst8/edit?usp=sharing

Provisioning Toolkit Links:

https://docs.google.com/presentation/d/1s5x2KNDIrzlZRsKhhqvJCnN23zyUgSz6YYb2ZYLpW2A/edit?usp=sharing

http://svn.usatlas.bnl.gov/svn/griddev/provisioning-config/

http://svn.usatlas.bnl.gov/svn/griddev/provisioning-toolkit/

http://svn.usatlas.bnl.gov/svn/griddev/provisioning-templates/

Other Links:

https://twiki.cern.ch/twiki/bin/view/PanDA/EventServer

Questions/Discussion

Extra Slides/References

“Impedance mismatch” between commercial and scientific computing

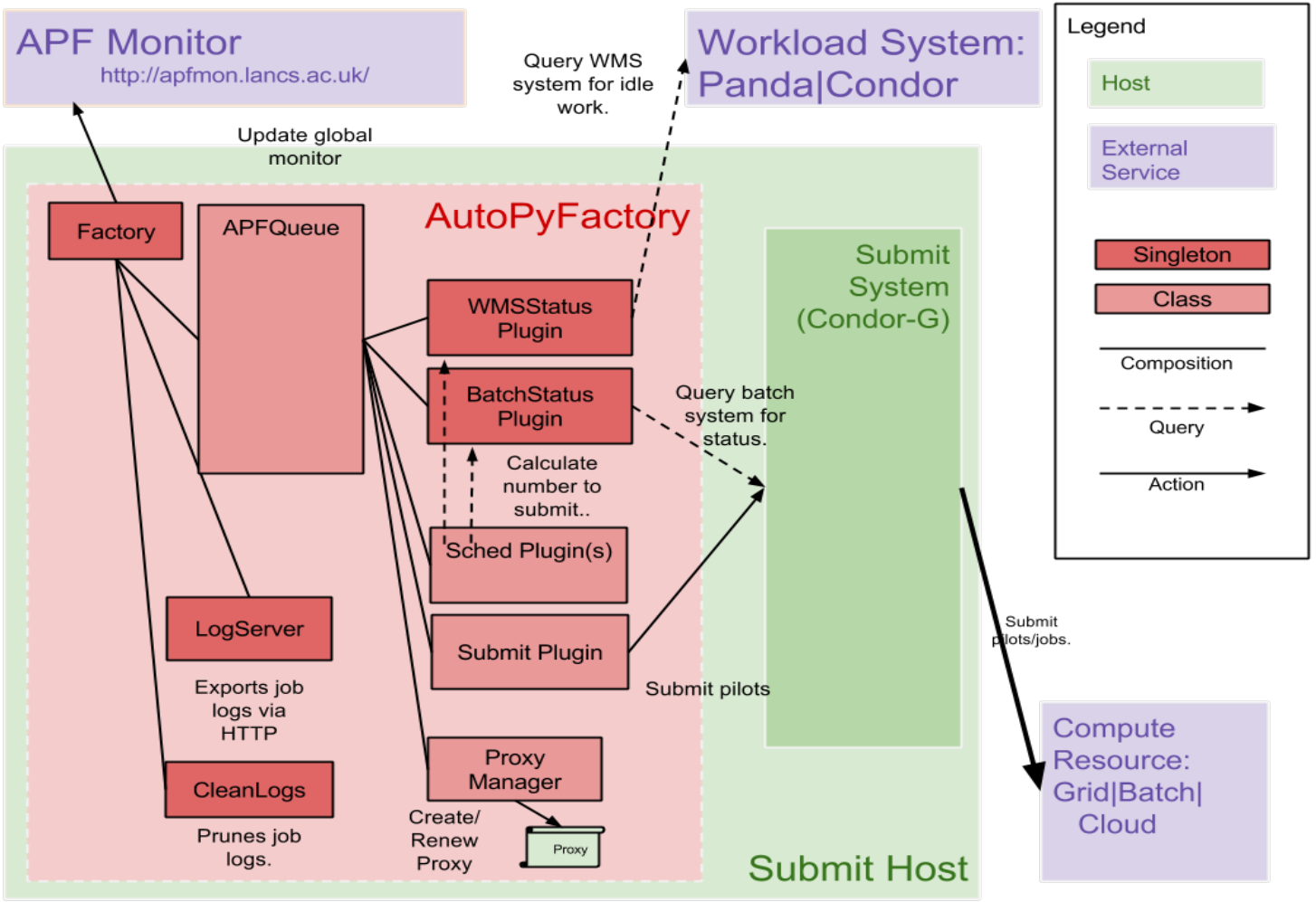
Amazon w/ commercial customers	**Scientific computing**
Customers often write applications from scratch to run natively on the cloud	Legacy software (written for grid, local usage) being migrated to cloud (adapted where necessary).
S3/HTTP distributed data; easy to adapt to.	Heavy commitment to GridFTP/SRM “endpoint-oriented” storage.
Customers usually happy to natively leverage all Amazon-provided services.	All services, processes, frameworks must work across Amazon and Openstack, since they will run on other public and private clouds.
For-profit company managers tell developers “make it happen.”	If scientists perceive usage as “too hard” or “too confusing” they will reject it, regardless of economic rationale.
Commercial customers usually able to adopt paid solutions for various problems.	Open source only.
Commercial customers may use limited instance types, with limited numbers. Often focussed on node performance.	Use nearly all instance types, in large numbers. Oriented toward core-based capacity, but not performance per se.
Peering/DirectConnect customer is usually the Amazon customer (except in the case of resellers).	Peering/DirectConnect customer different from Amazon customer. (e.g. ESNet vs. ATLAS (and other BNL Amazon users).
All storage costs money. Pro-active deletion required.	All storage dedicated and free. Deletion only needed for space.

“Impedance” (2)

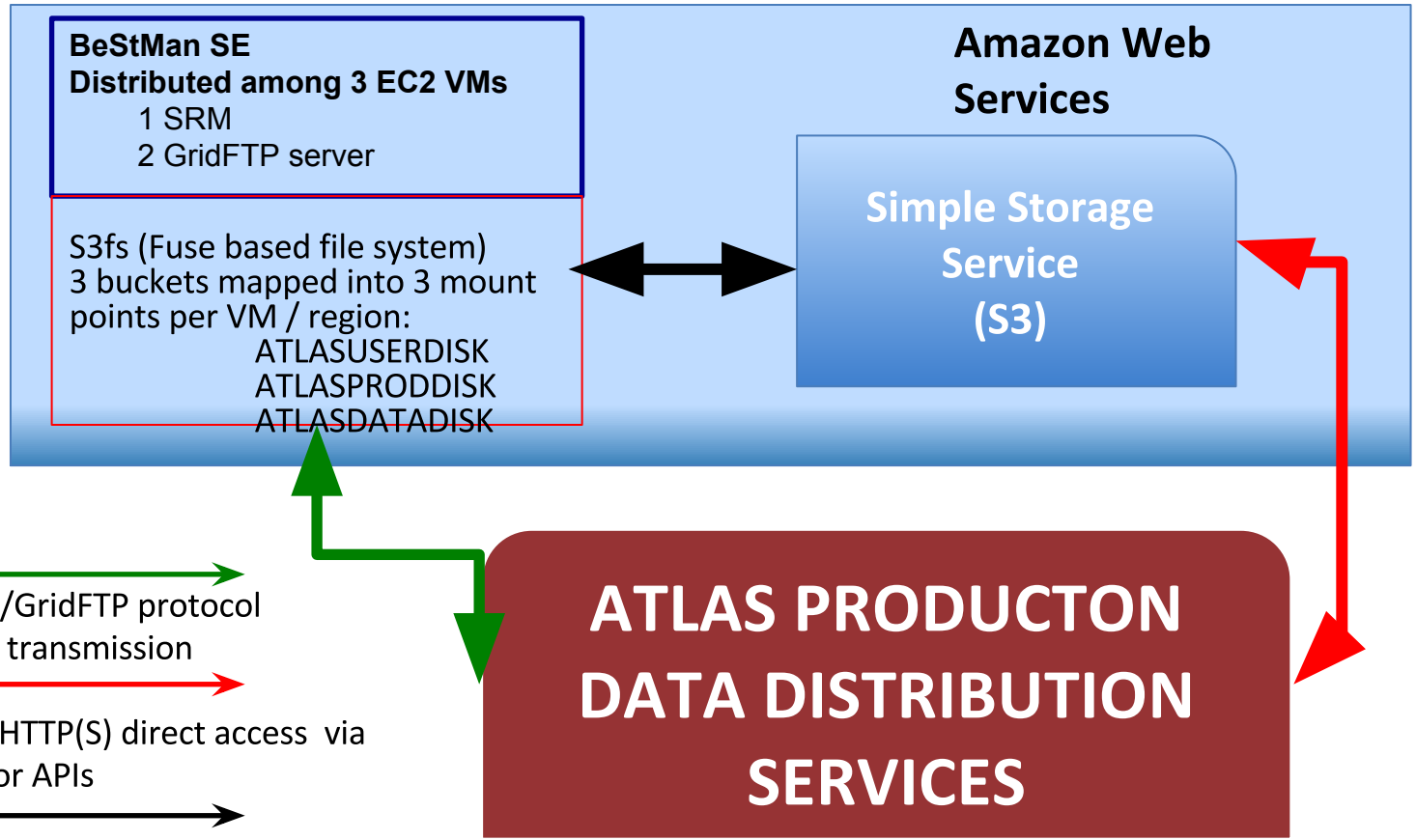
None of these mismatches were a “show stopper”. They just needed to be recognized and dealt with carefully.

Many commercial customers have constraints similar to ATLAS, so none of these are really new.

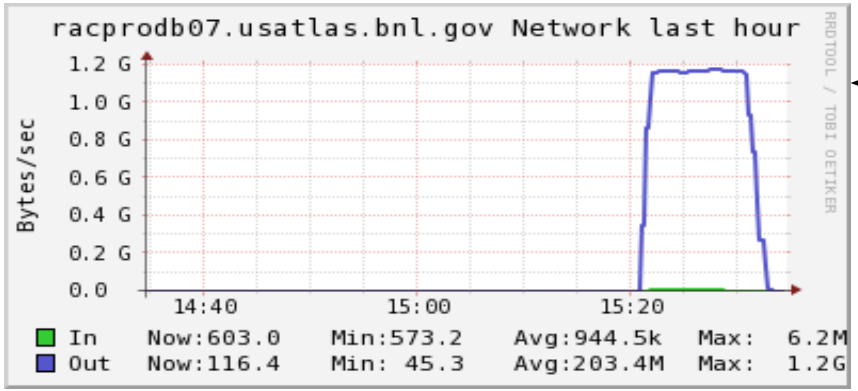
AutoPyFactory Design



Example us-east-1 region



Throughput tests destination Amazon S3 bucket (us-



Test 1 via Python/Boto API

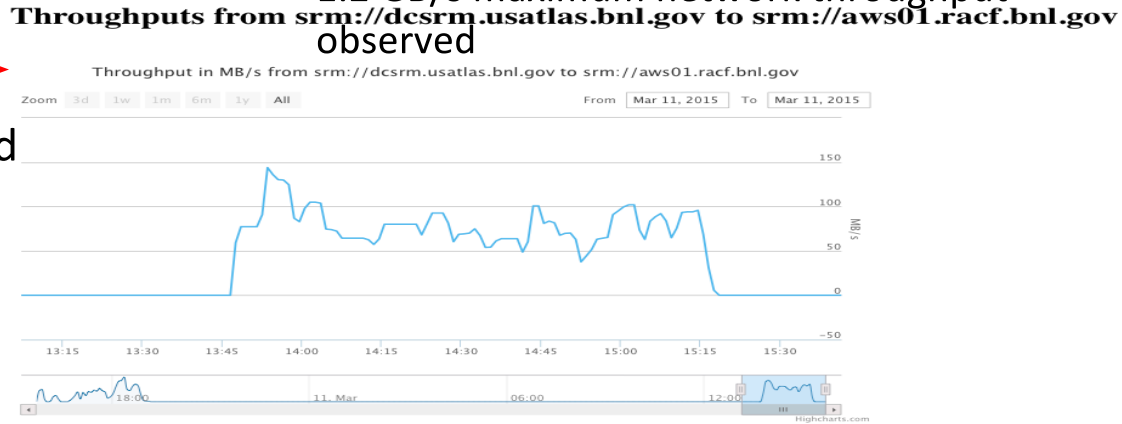
To saturate the network connectivity among a client at BNL to a S3 AWS bucket

202 files of 3.6 GB per file copied. 128 parallel transfers/job allowed.

1.2 GB/s maximum network throughput observed

Test 2 via SE BeStMan / S3fs

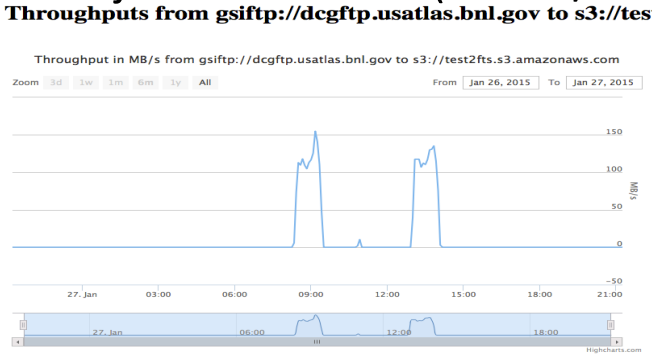
118 files of 3.6 GB/file transferred from BNL dCache using FTS

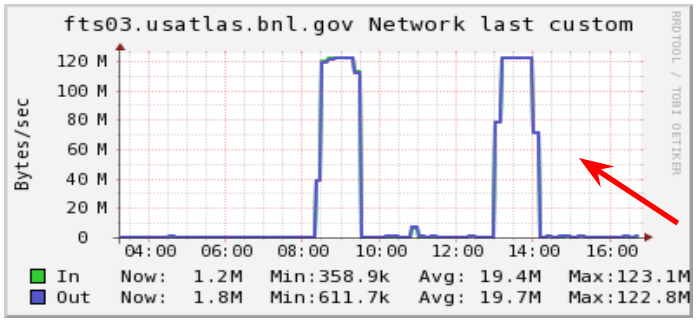


A 10Gb/s direct connect network link established among BNL and AWS S3 (us-east-1) used in the tests

Test 3. Via direct access to S3 from BNL dCache via FTS

Two jobs of 120 files (3.6 GB/file) transferred from BNL dCache using FTS

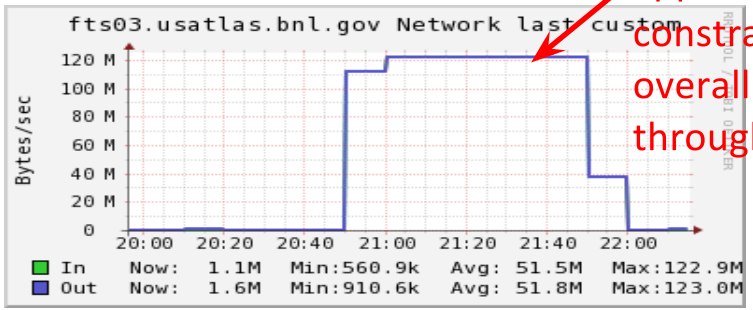




FTS host submitter current 1Gb/s NIC appeared to constrain the overall transfer throughput

Test 4. Via direct access between two S3 AWS buckets using FTS

120 files of 3.6 GB/file transferred



Tests

Test 1.
A proxy server populated with 202 files (3.6GB) size to run a custom job to access AWS S3 via python/Boto:

- 47GB memory
- 16 processors 3GHz
- 2x10Gb/s LACP Network connectivity
- Data stored in s DS3500 backend storage
  17 SAS disks /15kRPM configured in a
  RAID 0 layout

Test 2.
Destination BeStMan/S3fs SE
SE installation distributed among 3 homogenous EC2 VM (hi1.4xlarge):

- -57GB memory
- -16 processors 2.40Ghz
- -10Gb/s network connectivity
- -2 SDD Volumes dedicated for S3fs cache.

Source SE

- BNL dCache Massive Storage system

Tests (2)

Test 3,4
Destination AWS bucket us-east-1 region
Source Test 3
BNL dCache Massive Storage system
Source Test 4
AWS bucket us-east-1 region

FTS submit host
-47 GB memory
-24 processors, 2.7GHz
-1Gb/s network