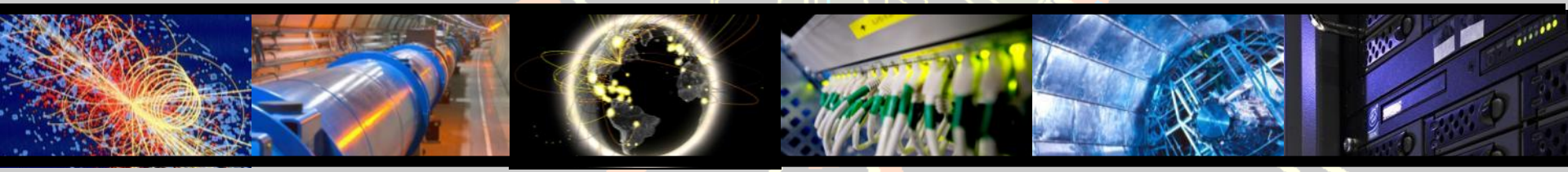
Plans for “Clouds” in the U.S. ATLAS Facilities

Michael Ernst
Brookhaven National Laboratory



The Experts and Actors

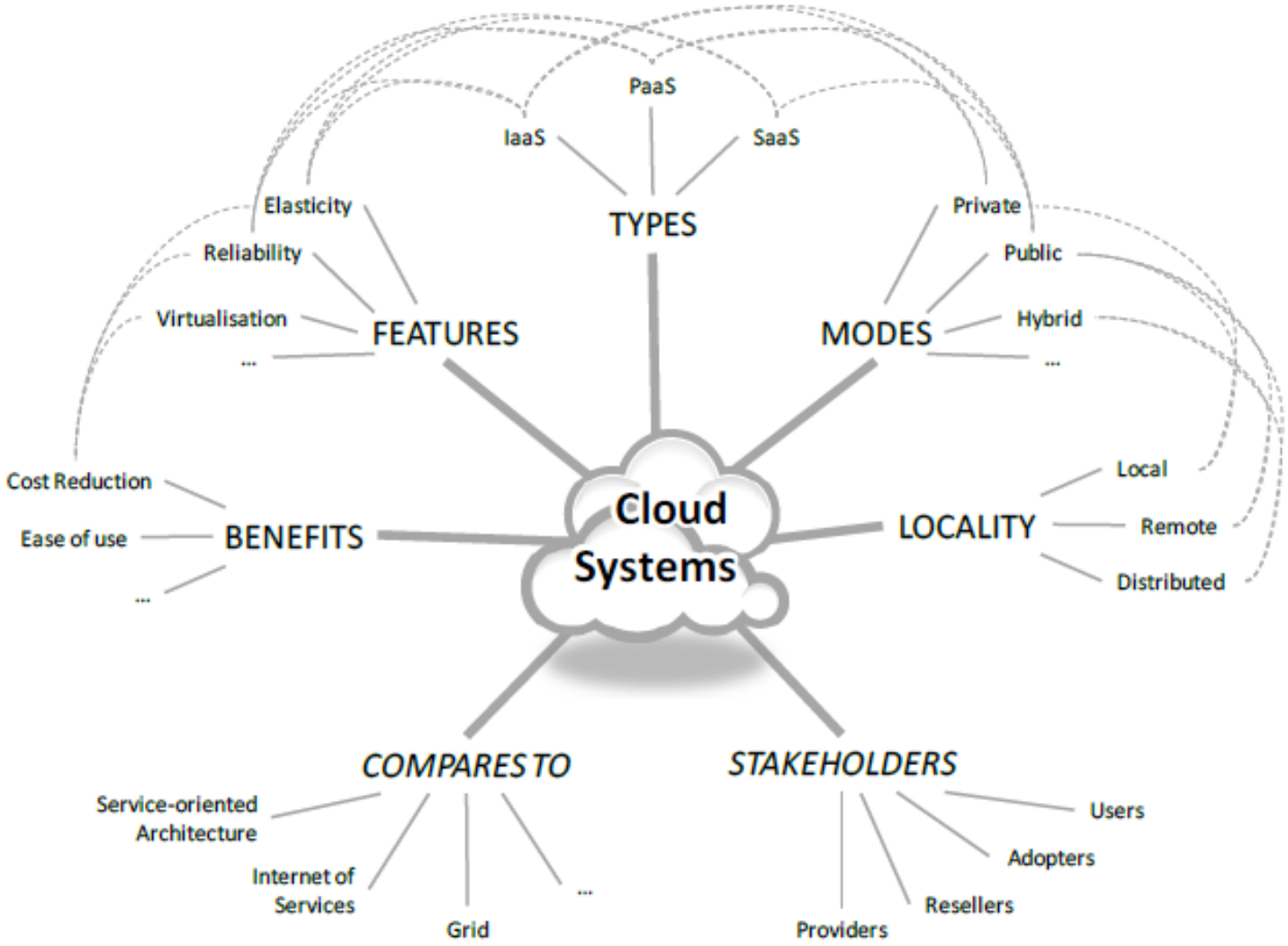
- This work was/is primarily carried out by
  - John Hover (BNL)
  - Jose Caballero (BNL)
  - Xin Zhao (BNL)
- People joining
  - Hironori Ito (BNL) w/ focus on Cloud Storage
  - Alexandr Zaytsev (BNL)
  - Lincoln Bryant (UoChicago)
- There are also cloud activities for analysis
  - Doug will talk about that after me

Outline

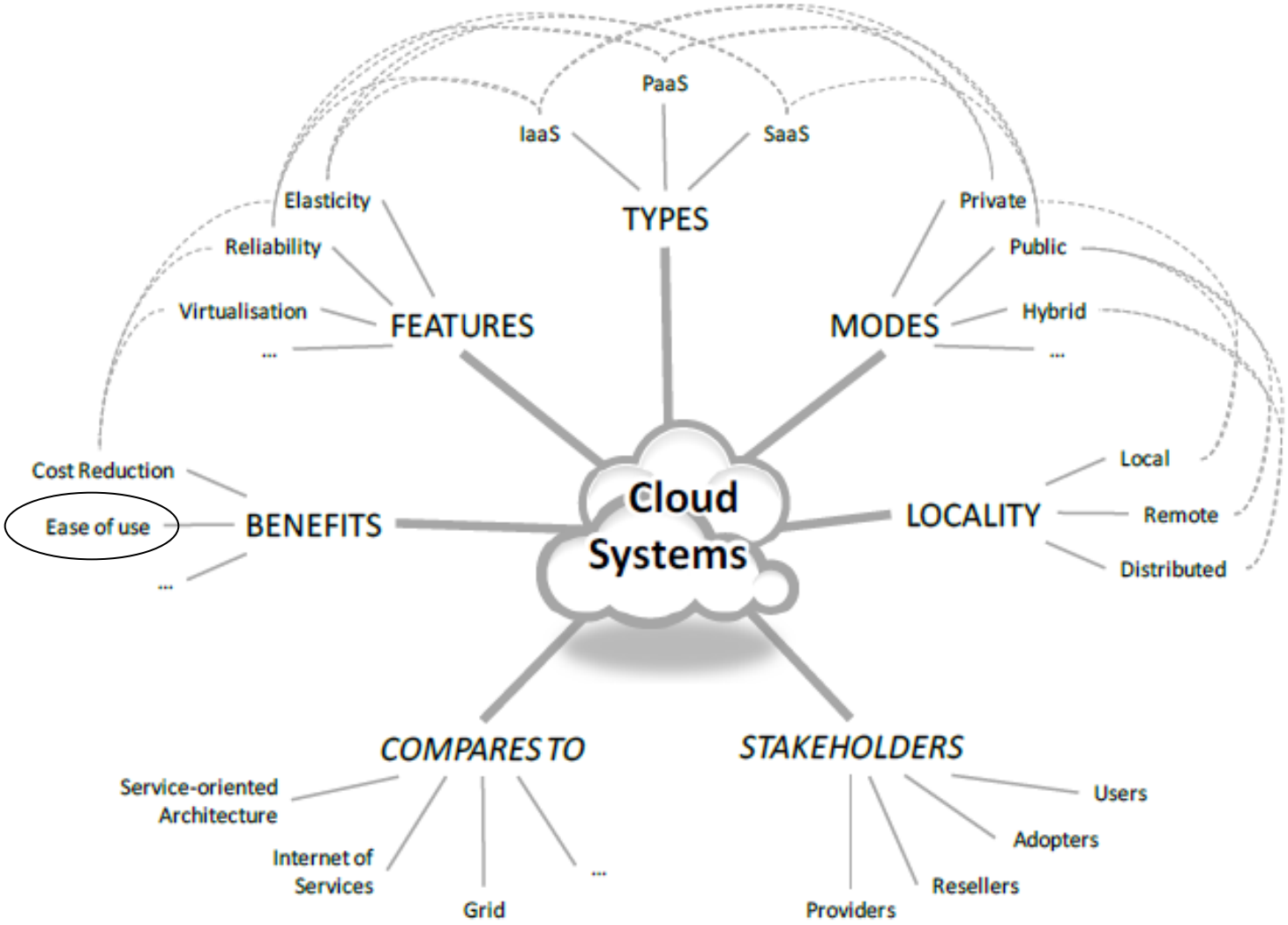
- Basic Cloud Properties
- Rationale/Benefits Recap
- Dependencies/Limitations
- Current BNL Status
  - Openstack
  - VMs with Boxgrinder
  - Panda queue
  - Local batch virtualization support
- Next Steps and Plans

mernst@bnl.gov

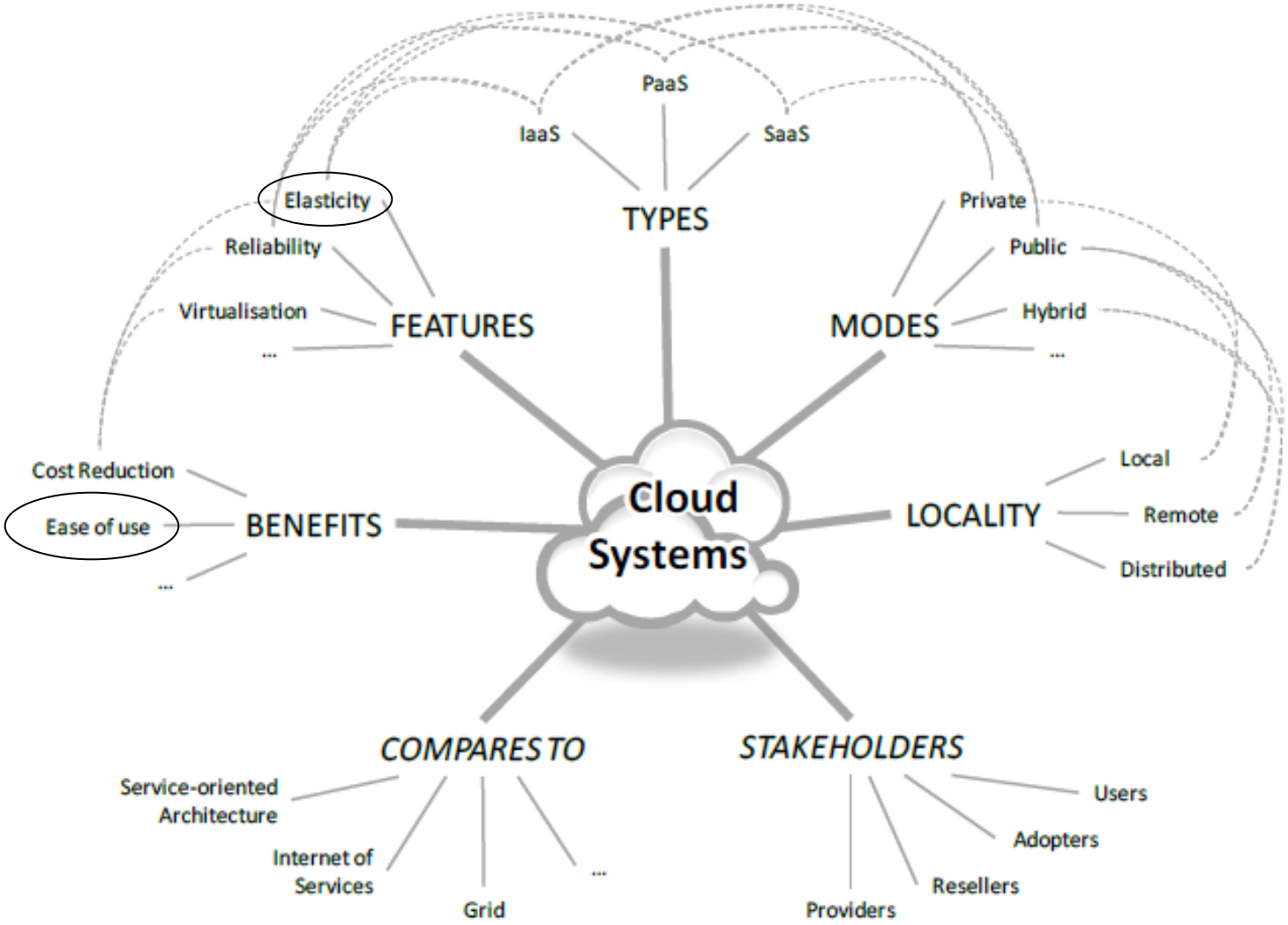
Aspects forming a Cloud System



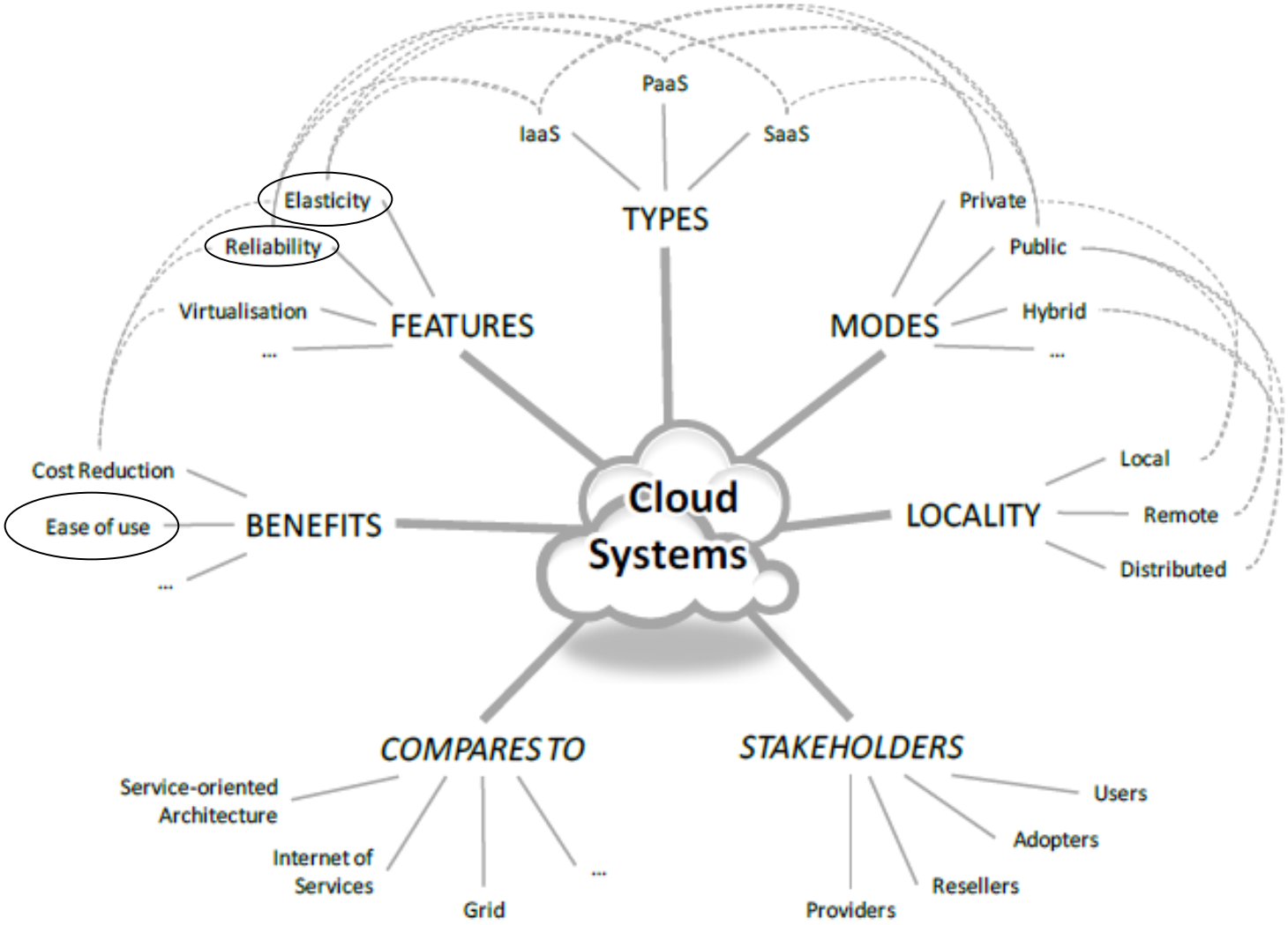
Aspects forming a Cloud System



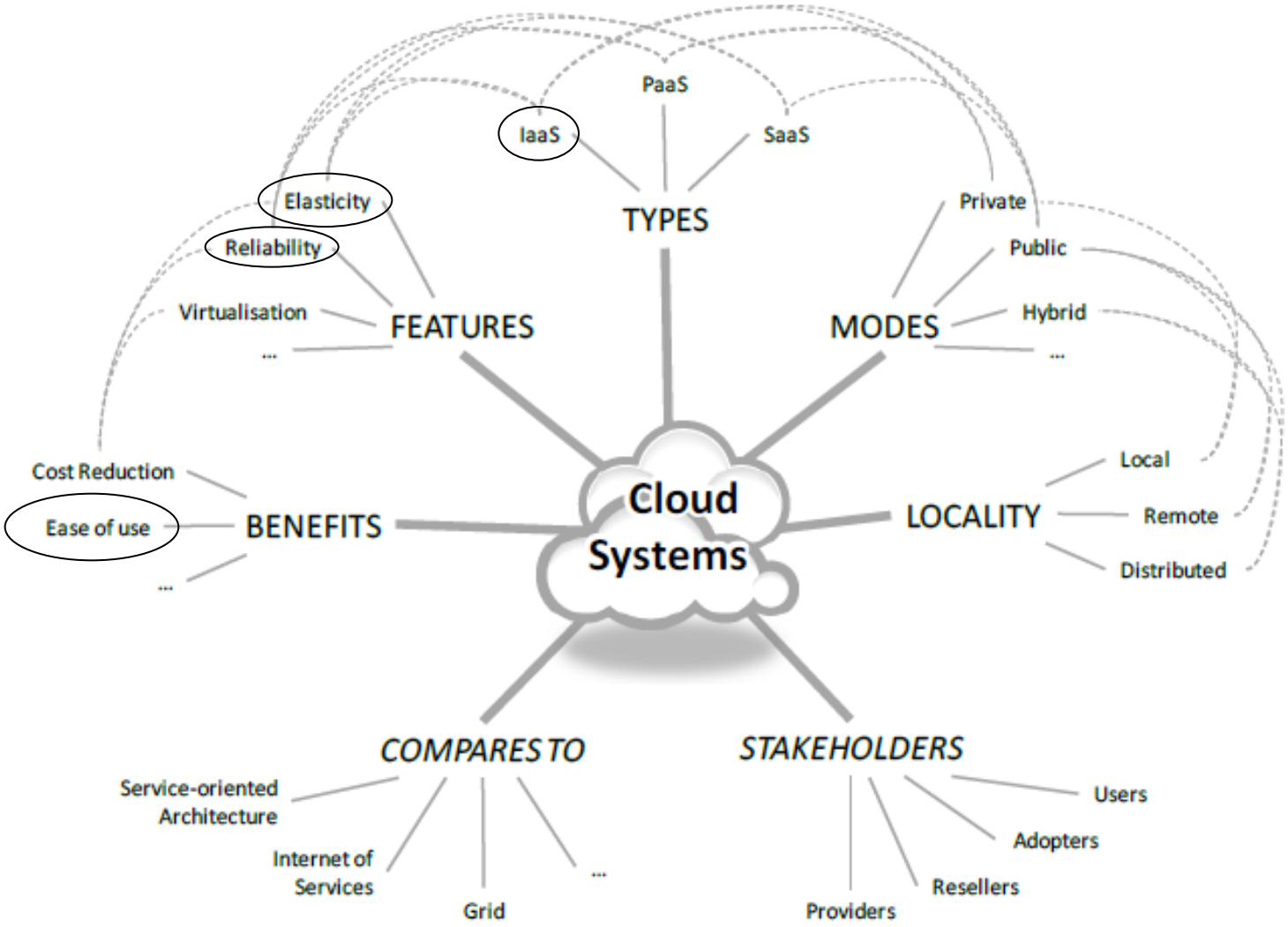
Aspects forming a Cloud System



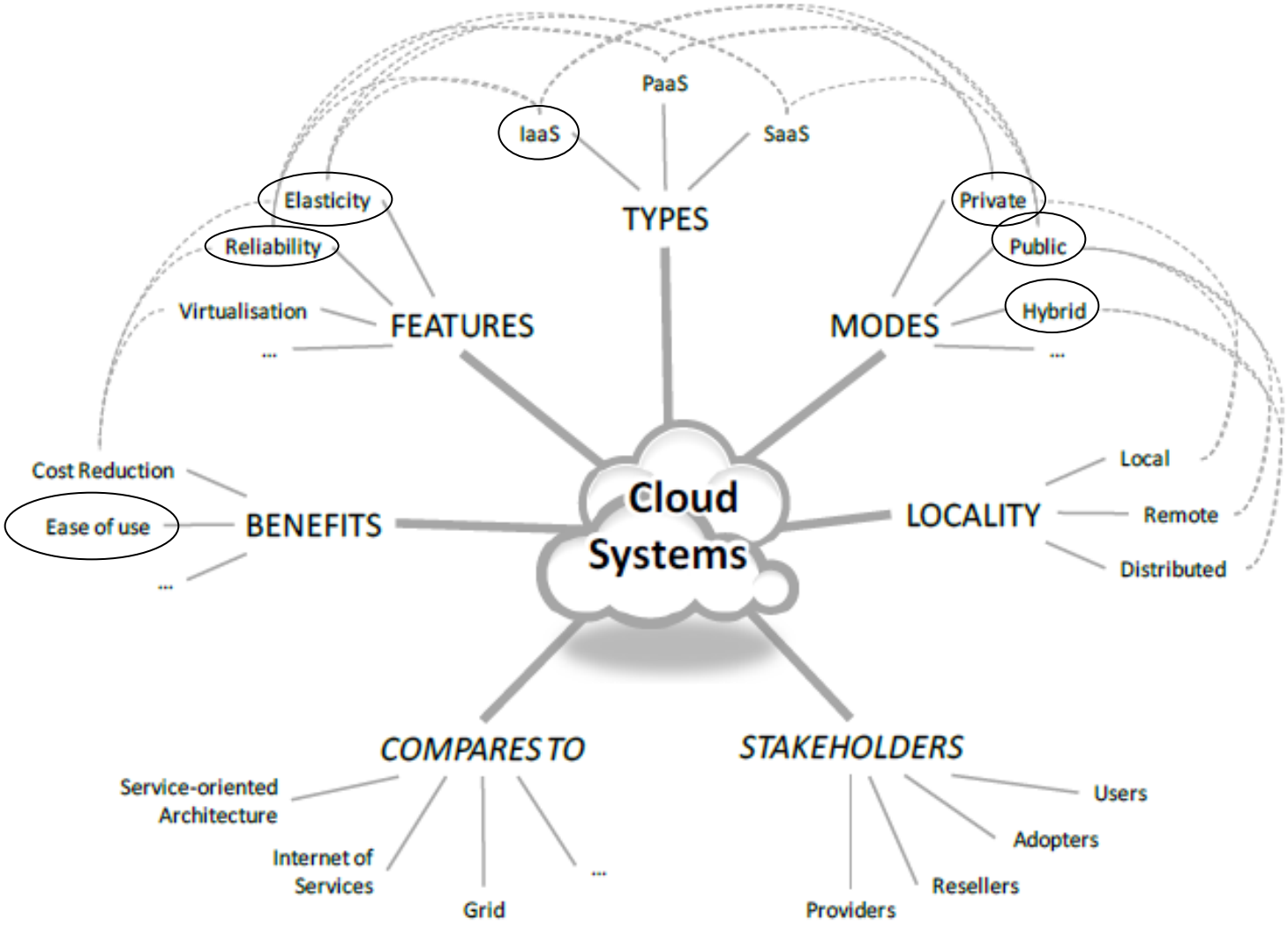
Aspects forming a Cloud System



Aspects forming a Cloud System



Aspects forming a Cloud System



Non-Functional Aspects

- Elasticity
  - Capability of infrastructure to adapt to changing, potentially non-functional requirements
    - Amount of data supported by app, # of concurrent users
  - There is horizontal and vertical scalability
    - Horizontal: # of instances to satisfy changing demands
    - Vertical: Size of instances
  - Cloud scalability involves rapid up- and downscaling

Non-Functional Aspects

- Availability and Reliability
  - Ensure constant operation without disruption
    - No data loss, no code reset during execution, etc
  - Typically achieved through redundant resource utilization
    - Many aspects move from hardware to software-based solutions (e.g. redundancy in the file system vs. RAID controllers, stateless front end servers vs. UPS, etc)
  - Provide redundancy for services and data so failures can be masked transparently
    - Fault tolerance also requires ability to introduce new redundancy, online and non-intrusively

Non-Functional Aspects

- Agility and Adaptability
  - Capabilities that strongly relate to Elasticity
    - On-time reaction to changes (# of requests and size)
    - Changes to environmental conditions (types of resources)
    - Requires resources to be autonomic and enable them to provide self-* capabilities

Dependencies/Limitations

- Using Cloud in larger US ATLAS facilities context requires
  - **X509 authentication mechanism. Current platform implementations all require username/passwords.**
  - **Accounting mechanism.**
  - **Automated, supported install and configuration.**
- Intrusive: Fundamental change
  - **Does represent a new lowest-level resource management layer.**
  - **But, once adopted all current management can still be used.**
- Networking and Security
  - **Public IPs require some DNS delegation, may also require additional addresses. (No public IPs at BNL due to lack of delegation).**
  - **Some sites may have security issues with the Cloud model**.

mernst@bnl.gov

Dependencies/Limitations

- ATLAS infrastructure not designed to be fully dynamic:
  - E.g. fully programmatic PanDA site creation and destruction
  - DDM assumes persistent endpoints
  - Others? Any element that isn't made to be created, managed, and cleanly deleted programmatically.

mernst@bnl.gov

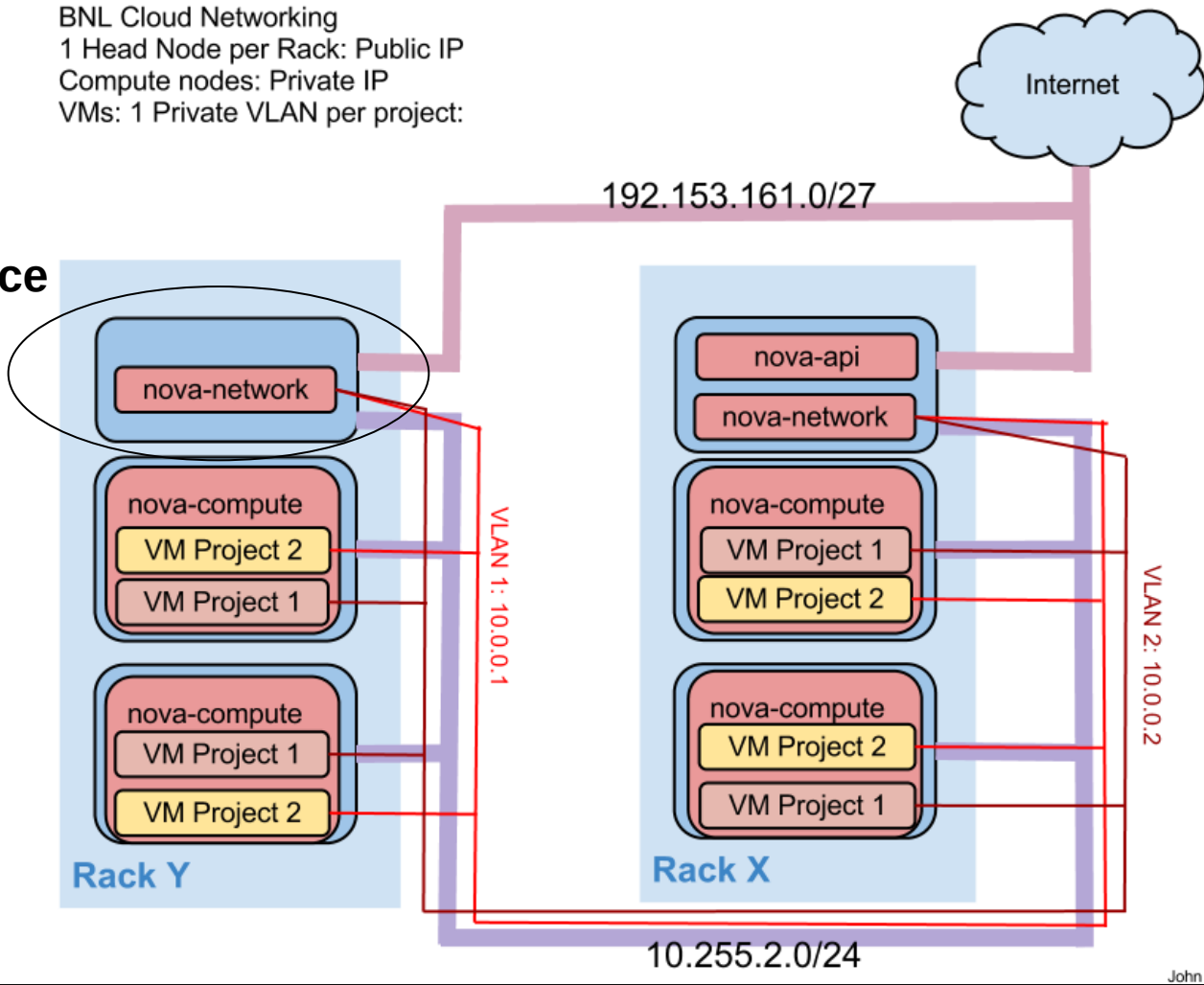
BNL OpenStack Cloud

- OpenStack 4.0 (Essex)
  - 1 Controller, 100 execute hosts (~300 2GB VMs), fairly recent hardware (3 years), KVM virtualization w/ hardware support.
  - Per-rack network partitioning (10Gb throughput shared)
  - Provides EC2 (nova), S3 (swift), and an image service (glance).
  - Essex adds keystone identity/auth service, Dashboard.
  - Programmatically deployed, with configurations publically available.
  - Fully automated compute-node installation/setup (Puppet)
  - Enables 'tenants'; partitions VMs into separate authentication groups, such that users cannot terminate (or see) each other's VMs.

mernst@bnl.gov

BNL OpenStack Layout

Potential I/O Bottleneck
- **multiple gateways**
- **public IP on second interface**



mernst@bnl.gov

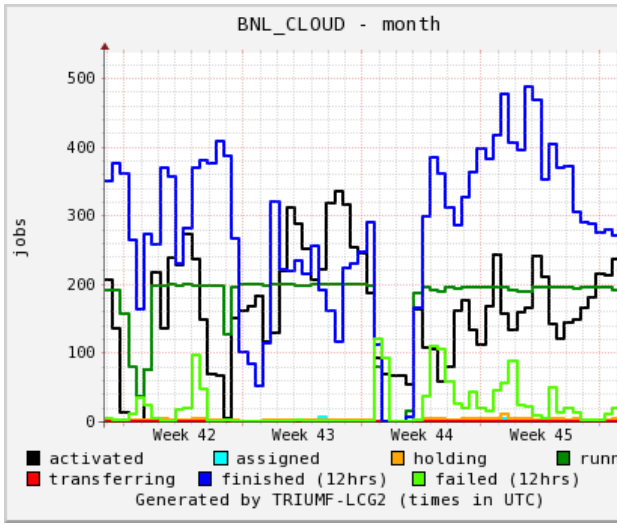
BNL VM Authoring

- Programmatic ATLAS Worker Node VM creation using Boxgrinder (BG)
  - http://boxgrinder.org/
  - http://svn.usatlas.bnl.gov/svn/griddev/boxgrinder/
- Notable features:
  - Modular appliance inheritance. The WN-atlas definition inherits the WN-osg profile, which in turn inherits from base.
  - Connects back to static Condor schedd for jobs.
  - BG creates images dynamically for kvm/libvirt, EC2, virtualbox, vmware via 'platform plugins'.
  - BG can upload built images automatically to Openstack (v3), EC2, libvirt, or local directory via 'delivery plugins'.
  - OSG now actively taking interest in BG, i.e. for testbeds

mernst@bnl.gov

PanDA

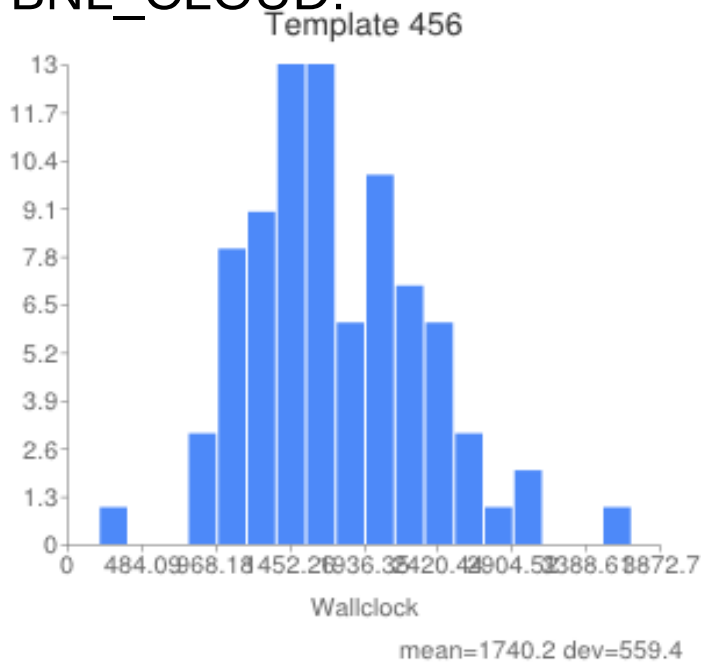
- BNL_CLOUD
  - Standard production Panda site.
  - Configured to use wide-area stagein/out (SRM, LFC), so same cluster can be extended transparently to Amazon or other public academic clouds.
  - Steadily running ~200 prod jobs on auto-built VMs for months. No facility related job failures
  - HC tests, auto-exclude enabled -- no problems so far.
  - Performance actually better than main BNL prod site. (next slide).
  - Also ran hybrid Openstack/EC2 cluster for a week. No problems.



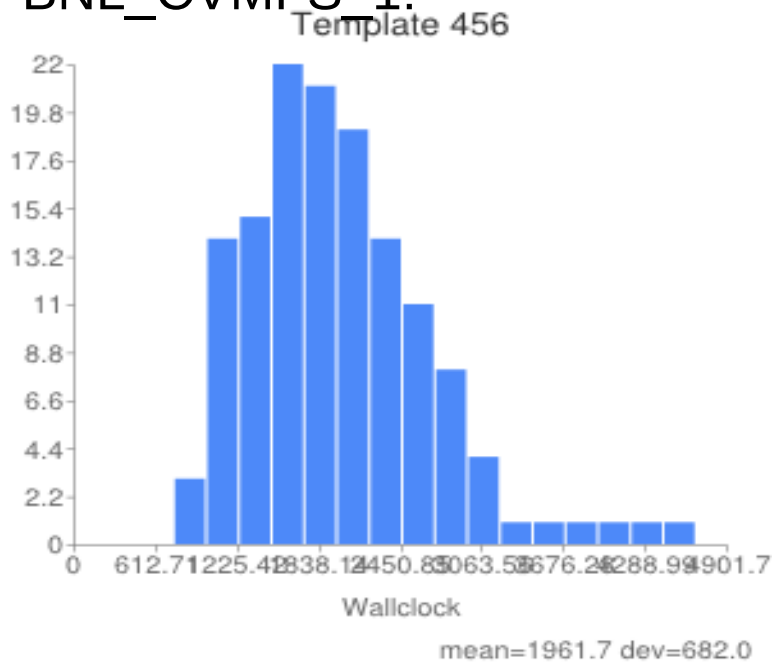
mernst@bnl.gov

Performance

- BNL_CLOUD:



BNL_CVMFS_1:



- Result: BNL_CLOUD actually faster (1740s vs. 1960s)
  - Hammercloud (ATLASG4_trf_7.2...)
  - Setup time (no AFS)? No shared filesystem?
  - Similar spread for other tests (e.g., PFT Evgen 16.6.5)
  - Anticipate using Iljia's HC framework to conduct more tests

mernst@bnl.gov

Plans (1/3)

- Cross-group Puppet group
  - Initiated at last S&C week. Includes CERN, ATLAS, and USATLAS people. Built around puppet-users@cern.ch
  - Shared configuration repo set up at https://svn.usatlas.bnl.gov/svn/atlas-puppet
- Refine setup and management
  - Generalize Puppet classes for node setup for use by other sites.
  - Finish automating controller setup via Puppet.
  - Facilitate wider use of BNL instance by other US ATLAS groups.

mernst@bnl.gov

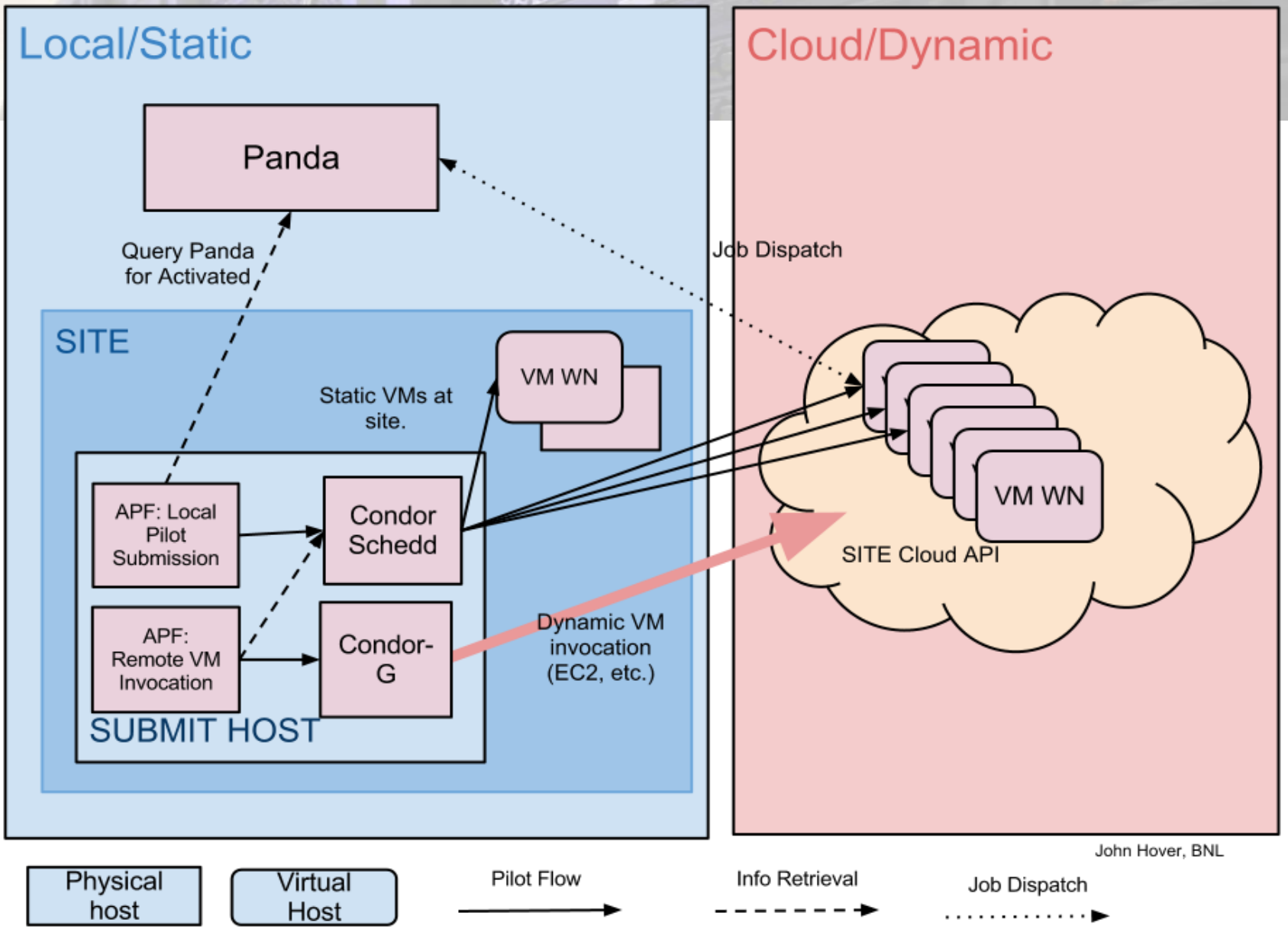
Plans (2/3)

- Decide on final UserData/Runtime contextualization:
  - Will allow the Tier 2/3 Cloud group to use our base images for their work.
  - Completely parameterize our current VM
- AutoPyFactory (APF) development
  - Full support for VM lifecycle, with flexible algorithms to decide when to terminate running VMs.
  - Cascading hierarchies of Cloud targets: Will allow programmatic scheduling of jobs on site-local->other private->and commercial clouds based on job priority and cloud cost.

mernst@bnl.gov

AutoPyFactory: Elasticity

- Static Condor schedd
  - Standalone, used only for Cloud work
- 2 AutoPyFactory (APF) Queues
  - 1 monitors a local Condor schedd,
    - When job slots are Idle, submits WN VMs to IaaS (up to defined limit).
    - When WNs are unclaimed, shuts them down.
  - 2 monitors a Panda queue
    - when there are activated jobs, submits pilots to local Condor queue.
- Worker Node VMs
  - Generic Condor startds connect back to static Condor cluster.
  - All VMs are identical
- Panda site
  - Associated with BNL/Tier-1 SE, LFC, CVMFS-based releases.
  - But no site-internal configuration (NFS, file transfer, etc).



mernst@bnl.gov

AutoPyFactory: VM Lifecycle

- Instantiation
  - When we want to *expand* the resource, a VM is instantiated, for as long as there is sufficient work to keep it busy.
- Association
  - In order to manage the lifecycle, we must track the association between a particular VM and a particular machine in the cluster.
    - The cloud API does not provide info as to which VMs are running jobs
  - Done via embedded DB (with Euca tools) or a ClassAd attribute (Condor-G)
- Retirement
  - When we want to *contract* the cluster, APF instructs VM to *retire* all batch slots, i.e. finish the current job but accept no new.
- Termination
  - Once all batch slots on a VM are idle the VM is terminated.

AutoPyFactory: Cloud Interactions

- **APF uses a plugin architecture to use Cloud APIs on EC2 and OpenStack**.
  - Current development uses Eucalyptus CLI client tools. (Euca2ools)
  - A future version of Condor will support both EC2 and OpenStack, at which point interaction can be via Condor-G (currently a Release Candidate).
- **APF will support *hierarchies* and *weighting***
  - We can create/establish multiple Cloud resources in order of preference, and expand and contract preferentially, e.g.,
    - Local OpenStack (free of charge, local)
    - Another ATLAS facility OpenStack (free of charge, remote)
    - Academic cloud at another institution (free of charge, remote)
    - Amazon EC2 via spot pricing. (low cost, remote)
    - Amazon EC2 via guaranteed instance. (costly, remote)
  - “Weighting”: Create VMs proportional to jobs waiting in the queue
  - Prototype by end February

Plans (3/3)

- Expand usage and scale
  - Scale up to a size comparable to main Grid site.
  - Dynamically re-partitionable between groups.
  - Option to transparently expand to other private clouds, and/or commercial clouds.
  - Run large-scale
- Fully document dynamic Fabric and VM configuration
  - To allow replication/customization of both site fabric deployment and job workflow by other ATLAS sites, and by other OSG VOs.
  - To allow specialized usage of our Cloud by users.

mernst@bnl.gov

Summary of Mid Range Plans

- A lot to be decided …
- Have started to deploy virtualized resources for analysis at MWT2
  - More expected to follow after evaluation by users
- Expect to grow cloud resources in Facilities
  - at BNL to 1k – 2k VMs in the next 3 – 6 months
    - Scalable network architecture
  - Cloud resources at 1 US Tier-2 in Q3/4 2013
    - Prerequisite: Install everything programmatically
- Need to address storage in the cloud

Extra Slides

mernst@bnl.gov

Overall Rationale Recap

- Why Cloud vs. current Grid?
  - Common interface for end-user virtualization management, thus..
  - Easy expansion to external cloud resources - same workflow to expand to:
    - Commercial and academic cloud resources.
  - Dynamic repartitioning of local resources, easier for site admins
  - Includes all benefits of non-Cloud virtualization: customized OS environments for reliable opportunistic usage.
  - Flexible facility management:
    - Reboot host nodes without draining queues.
    - Move running VMs to other hosts.
  - Flexible VO usage:
    - Rapid prototyping and testing of platforms for experiments.

mernst@bnl.gov

BNL OpenStack Cloud (cont)

- Ancillary supporting resources
  - SVN Repositories for configuration and development
    - E.g. boxgrinder appliance definitions.
  - See:
- http://svn.usatlas.bnl.gov/svn/griddev/boxgrinder/
  - YUM repos:
    - snapshots (OSG, EPEL dependencies, etc):
    - mirrors
    - grid dev, test, release
  - See:
- http://dev.racf.bnl.gov/yum/snapshots/rhel5/
    - http://dev.racf.bnl.gov/yum/grid/

mernst@bnl.gov

Simple WN Recipe

- Build and upload
  - svn co http://svn.usatlas.bnl.gov/svn/griddev/boxgrinder
  - <edit boxgrinder to point to your Condor, with any auth required>
  - boxgrinder-build -f boxgrinder/sl5-x86_64-wn-atlas-bnlcloud.appl
  - . ~/nova-essex2/novarc
  - glance add name=sl5-atlas-wn-mysite is_public=true disk_format=raw container_format=bare --host=cldext03.usatlas.bnl.gov --port=9292 < build/appliances/x86_64/sl/5/sl5-x86_64-wn-atlas-bnlcloud/3.0/sl-plugin/sl5-x86_64-wn-atlas-bnlcloud-sda.raw
- Describe and run images
  - euca-describe-images --config=~/nova-essex2/novarc
  - euca-run-instances --config ~/nova-essex2/novarc -n 5 ami-00000002
  - euca-describe-instances --config ~/nova-essex2/novarc

mernst@bnl.gov

Elastic Prod Cluster: Components

- Static Condor schedd
  - Standalone, used only for Cloud work.
- 2 AutoPyFactory (APF) Queues
  - One observes a local Condor schedd, when jobs are Idle, submits WN VMs to IaaS (up to some limit). When WNs are Unclaimed, shuts them down. Another observes a Panda queue, when jobs are activated, submits pilots to local cluster Condor queue.
- Worker Node VMs
  - Generic Condor startds associated connect back to local Condor cluster. All VMs are identical, don't need public Ips, and don't need to know about each other.
- PanDA site
  - Associated with BNL SE, LFC, CVMFS-based releases.
  - But no site-internal configuration (NFS, file transfer, etc).

mernst@bnl.gov

Programmatic Repeatability, Extensibility

- The ***key*** feature of our work has been to make ***all*** our process and configs general and public, so others can use it. Except for pilot submission (AutoPyFactory), we have used only ***standard***, widely used technology (RHEL/SL, Condor, Boxgrinder, Openstack).
  - Our boxgrinder definitions are published.
  - All source repositories are public and usable over the internet,e.g.:
    - Snapshots of external repositories, for consistent builds:
      - http://dev.racf.bnl.gov/yum/snapshots/
      - http://dev.racf.bnl.gov/yum/grid/osg-epel-deps/
    - Custom repo:
      - http://dev.racf.bnl.gov/yum/grid/testing
  - Our Openstack host configuration Puppet manifests are published and will be made generic enough to be borrowed.
  - Our VM contextualization process will be documented and usable by other OSG VOs/ ATLAS groups.

mernst@bnl.gov