Magellan: Building a cloud for technical workloads

Narayan Desai

Mathematics and Computer Science Division

Argonne National Laboratory

Magellan Project Overview: Phase 1

- DOE-ASCR funded Recovery Act project
  - Joint with LBNL/NERSC
  - 2 year term (2009-2011)
- Evaluation of cloud approaches for mid-range HPC workloads
  - Assess cloud system software (IaaS) (ANL)
  - Application performance study (LBNL)
  - Data intensive applications (both)
- Argonne Plan
  - IaaS POC system
    - Build with HPC bill of materials
  - Deploy system and software
  - Field friendly users
  - Tune software stack based on user feedback

Our Plan

- Step 1: Proof of Concept
  - Are IaaS private clouds suitable for technical workloads at all?
  - Expected overheads to be high for storage and network
- Approach: Build a private cloud
  - Use HPC cluster approach in terms of parts list
  - Fast nodes
  - High bandwidth network
  - Large scale (for private clouds at least)
    - 504 compute nodes
    - 200 active storage nodes
    - 133 GPU nodes
    - 8 fileserver nodes
    - 2 25 Tbit switches
- Evaluate technical workloads
  - Start with least I/O intensive, and work up from there

Magellan Hardware Architecture

Compute Servers

504 Compute Servers
  Nehalem Dual quad-core 2.66GHz
  24GB RAM, 500GB Disk
Totals
  4032 Cores, 40TF Peak
  12TB Memory, 250TB Disk

Active Storage Servers

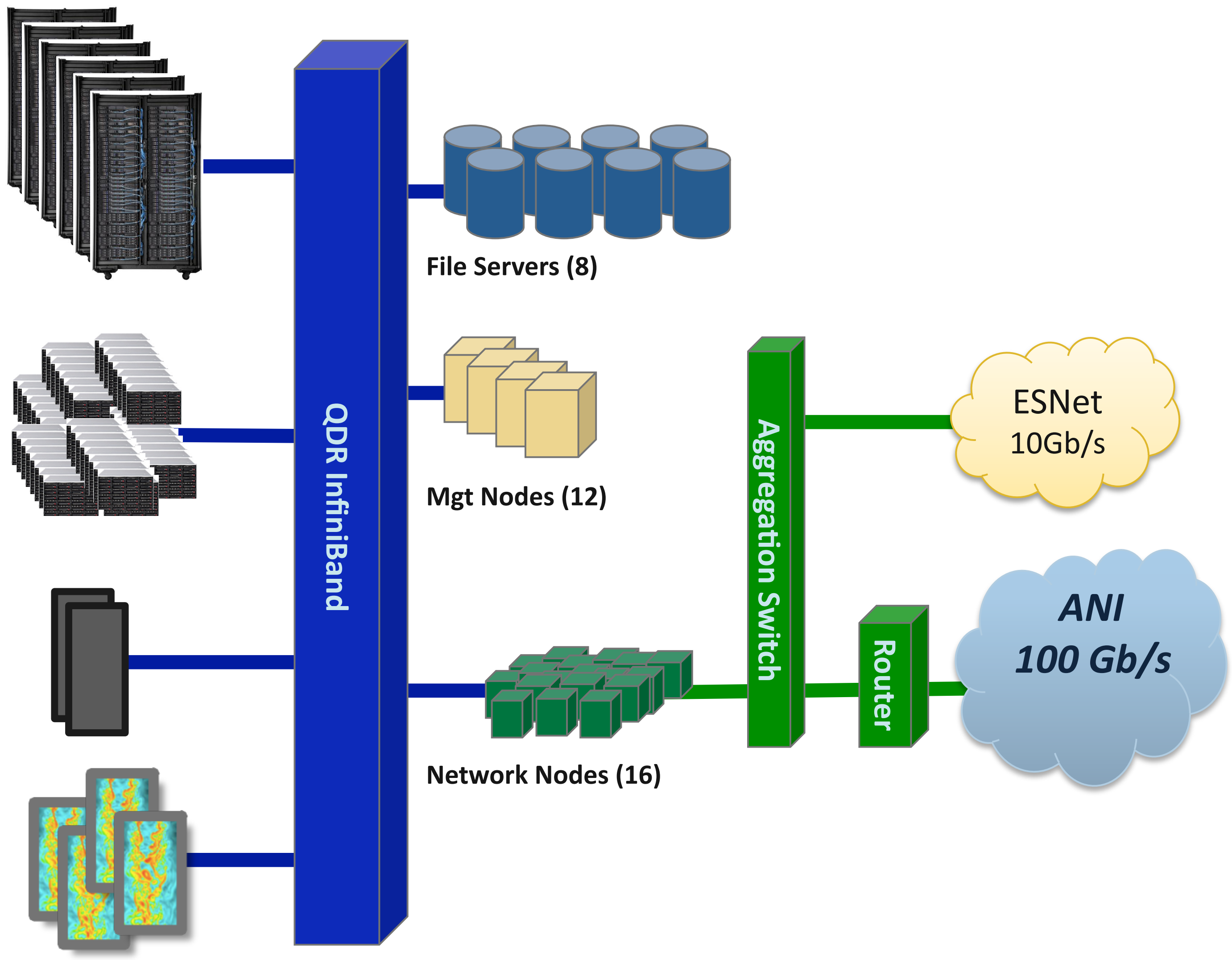
200 Compute/Storage Nodes
40TB SSD Storage
9.6TB Memory
1.6PB SATA Storage

Big Memory Servers

15 Servers
15TB Memory, 15TB Disk

GPU Servers

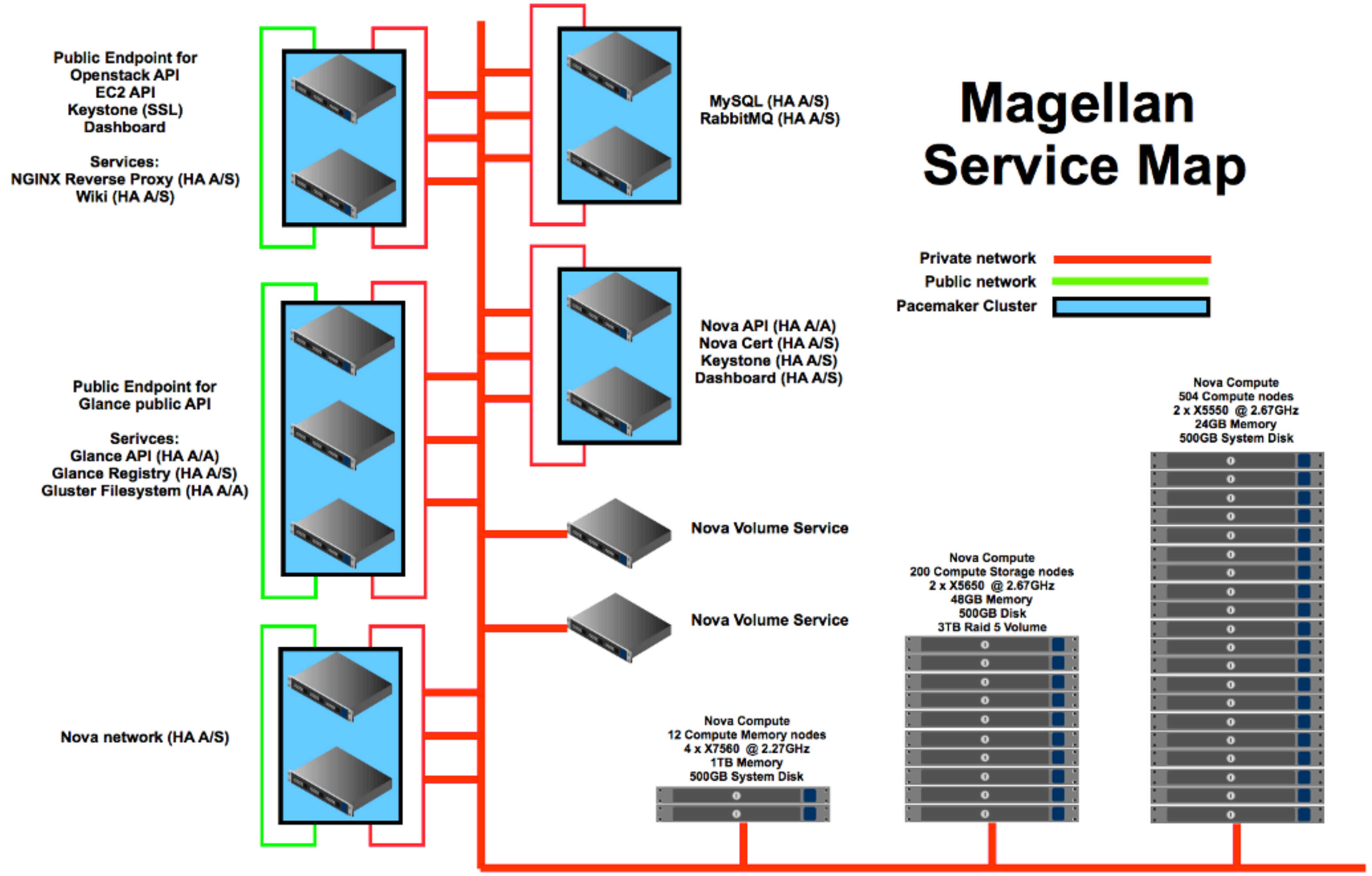
133 GPU Servers
8.5TB Memory, 133TB Disk
266 Nvidia 2070 GPU cards



IaaS Software Stack

- Interfaces for provisioning computing resources
  - VMs
  - Networks
  - Storage
- Relatively immature ecosystem
  - Early experiences with Eucalyptus had serious scalability problems
    - Even at pretty small scale (64 nodes)
  - Things got better once we switched to OpenStack
    - Scales to full system size
    - Extremely active community
    - 500 committers to most recent release
- Virtualization causes some “performance opportunities”
  - Some easy to address
    - processor flags
  - Others more difficult
    - OS bypass networking
    - Lack of drivers for GPUs

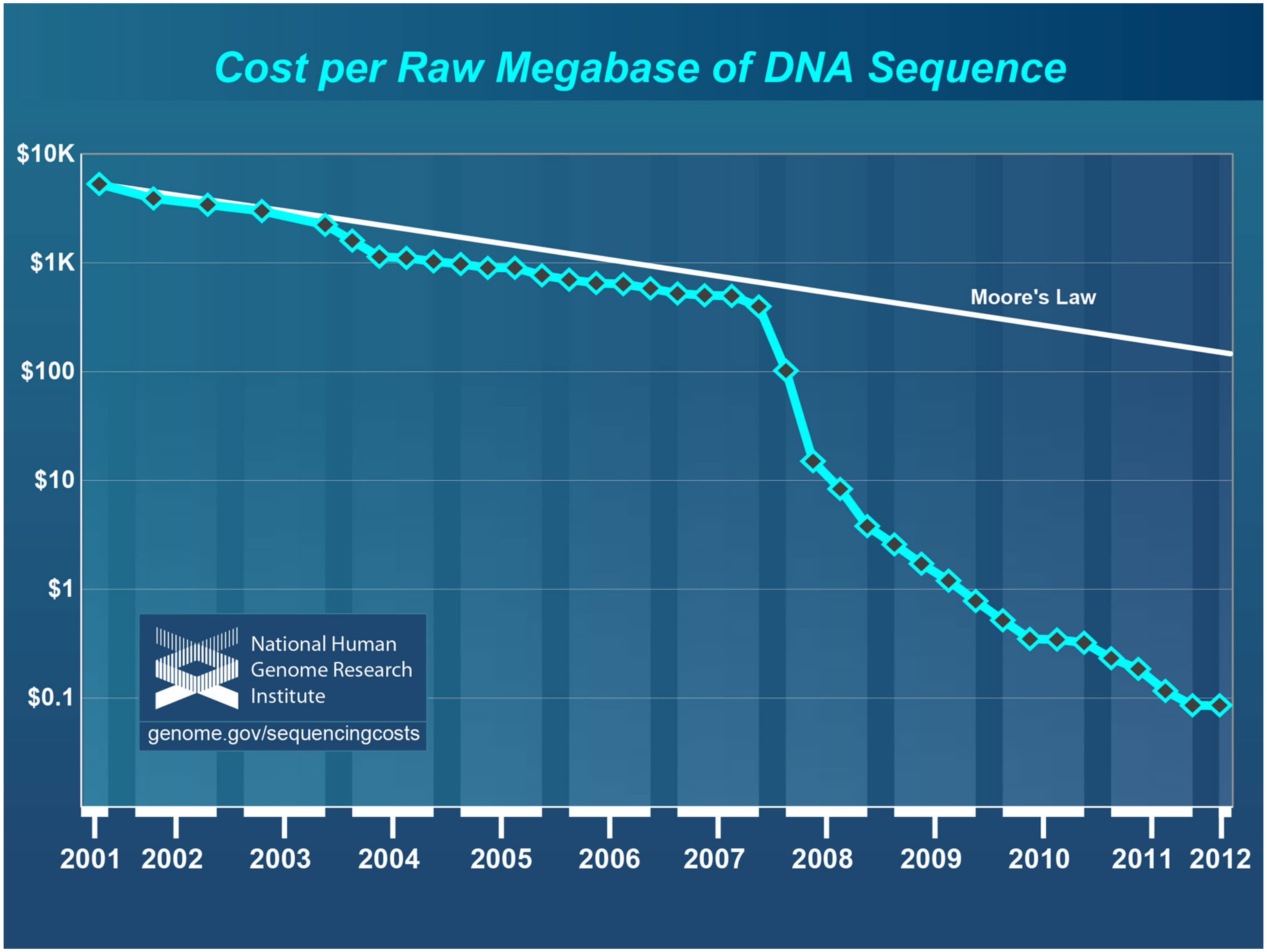
Magellan Software Architecture



Initial Application Performance Results

- Raw CPU performance good
- I/O bandwidth is pretty decent
- I/O latency is *terrible*
- Has about the effects you'd expect
  - Tightly coupled applications not really worth running
- Not so great for the "mid-range HPC workload"
- Some bright spots
  - Bioinformatics workloads
  - Data analysis workloads

The Bioinformatics Slide



Unexpected Results

- Application execution is a single component of a holistic computational science workload
  - Also includes code development
  - Experimentation and evaluation of new approaches
  - Infrastructure can also greatly improve performance (memcached)
- Disintermediated architecture enabled developers to experiment easily
  - Low cost, convenient
- Our developer productivity went through the roof
- The IaaS cloud model also directly supports service deployment
  - Which in turn enables construction of data management infrastructure
  - (Recall that as computing costs go up, the value of reusing results increases as well)
- The combination of improvements in innovation, and supporting both application-centric and data-service workloads was a substantial win over traditional HPC system architectures for bioinformatics codes

Thinking about cloud APIs

- Resource (not job) centric
  - Enables configuration of custom infrastructure on a per-workload basis
  - Scales (today) to moderate numbers of components
  - Often thin-provisioned
- Compelling for mixed-intensity workloads
- Resource management is relatively naïve
  - Immediate satisfaction/denial of requests
  - Requests are resource commitments without a fixed term
    - Most parallel job scheduling techniques don't apply for this reason
  - Overcommit is generally the path to good utilization
- Availability is generally a first order design goal
  - HTTP protocol/LAMP stack helps with this somewhat
- Fast-moving ecosystem
  - Lots of (other) people writing code

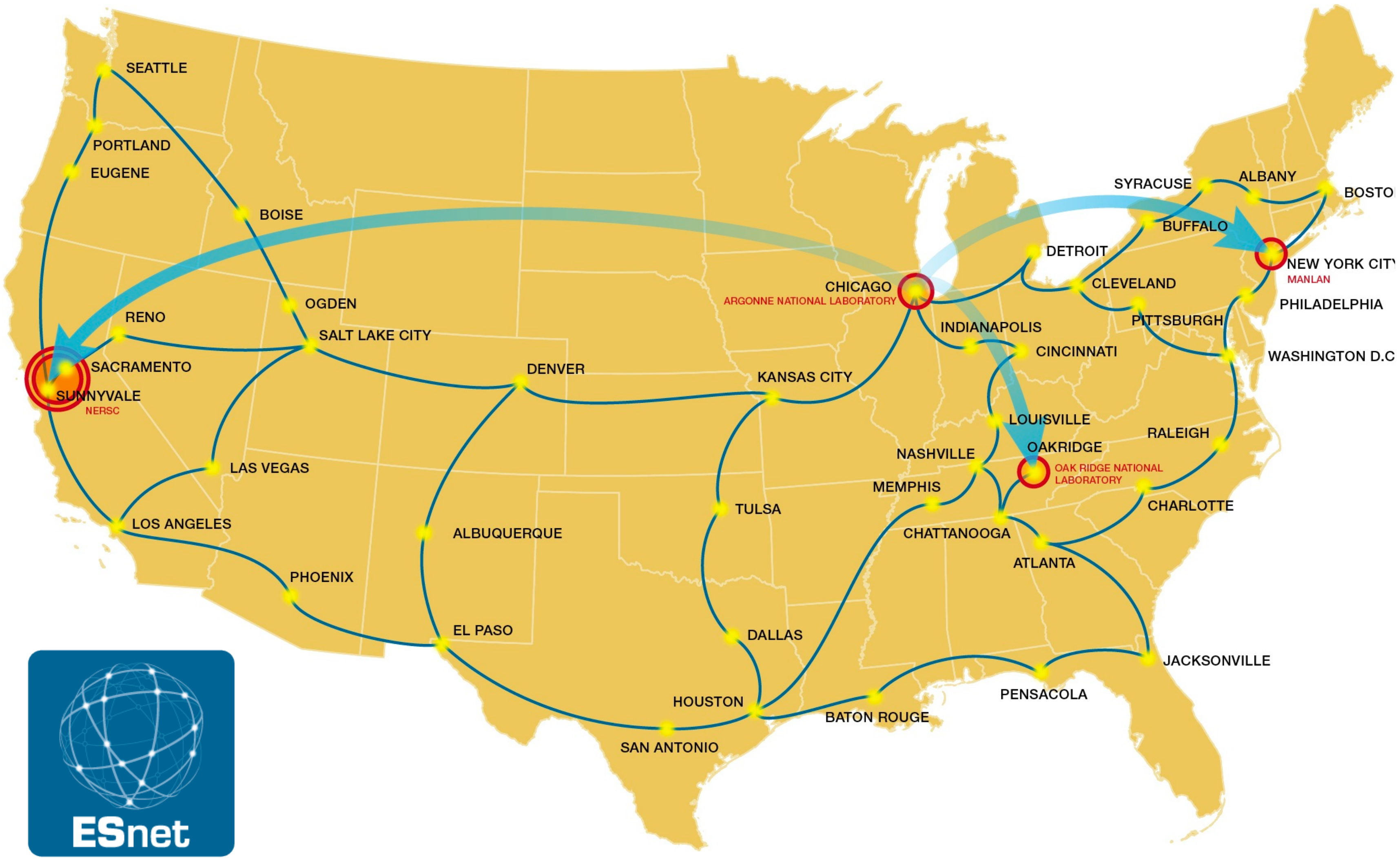
Magellan Phase II: DOE Kbase and Generalization

- Upon completion of the ARRA project, we moved the system to support application areas that are well suited to its architecture
  - Bioinformatics (DOE Kbase and other projects)
  - Local data intensive applications
- Areas of active work (collaborators welcome!)
  - Reducing performance cost of virtualization
    - SR-IOV
    - Openflow
    - VM tuning
  - Generalization to other application domains
    - Cosmology next
  - Building high-performance infrastructure
    - On-demand data transfer nodes
    - Dynamic clusters

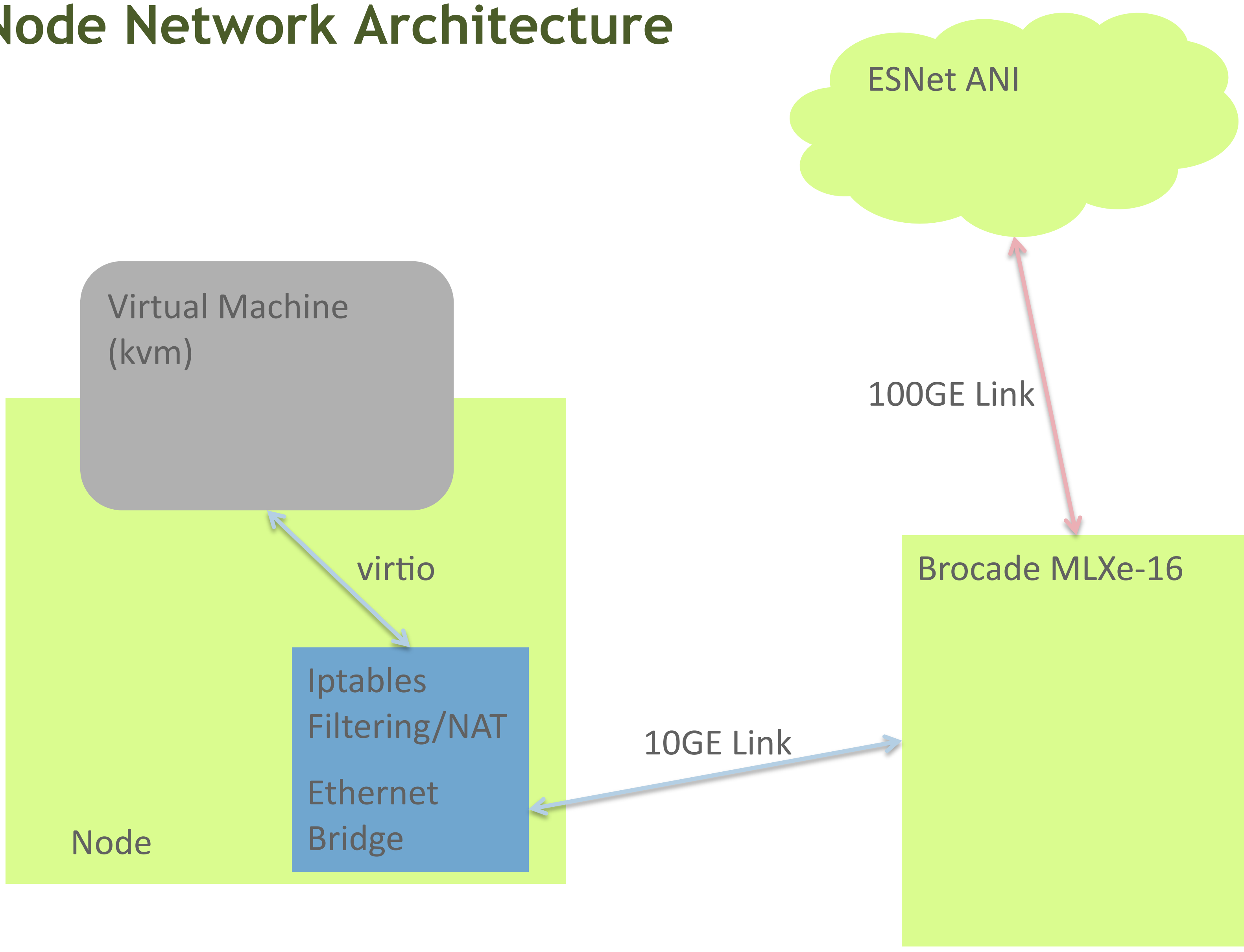
Network Performance Expedition

- Goal: To determine the limits of Openstack infrastructure for wide area network transfers
  - Want small numbers of large flows as opposed to large numbers of slow flows
- Kbase will eventually need to support movement of 10-100's of TB of data per day
- Built a new Essex test deployment
  - 15 compute nodes, with 1x10GE link each
  - Had 15 more in reserve
  - Expected to need 20 nodes
  - KVM hypervisor
- Used FlatManager network setup
  - Multi-host configuration
  - Each hypervisor ran ethernet bridging and ip firewalling for its guest(s)
- Nodes connected to the DOE ESNet Advanced Networking Initiative

ESNet Advanced Networking Infrastructure



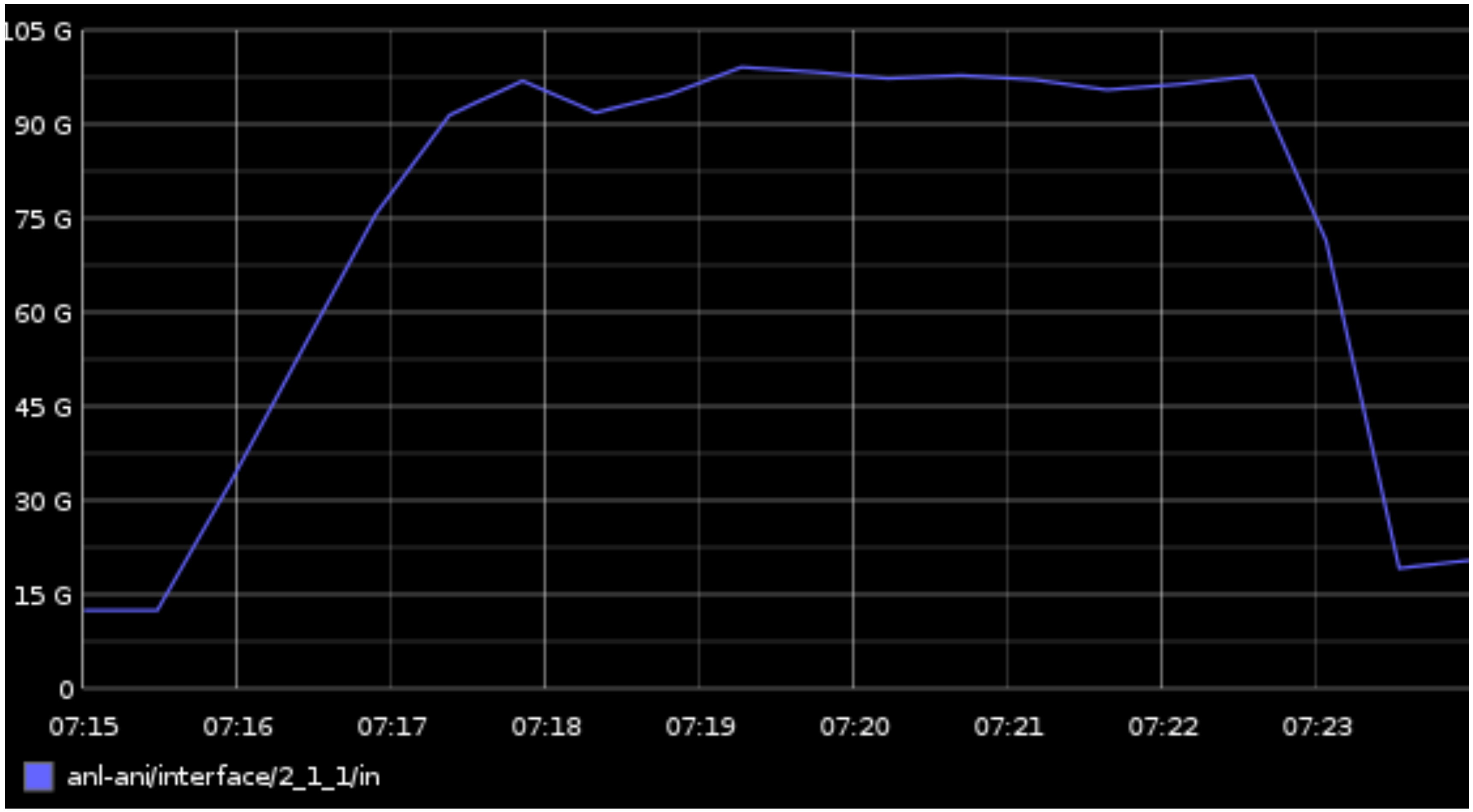
VM/Node Network Architecture



Setup and Tuning

- Standard instance type
  - 8 vcpus
  - 4 vnics bridged to the same 10GE ethernet
  - virtio
- Standard tuning for wide area high bandwidth transfers
  - Jumbo frames (9K MTU)
  - Increased TX queue length on the hypervisor
  - Buffer sizes on the guest
  - 32-64 MB window size on the guest
  - Fasterdata.es.net rocks!
- Remote data sinks
  - 3 nodes with 4x10GE
  - No virtualization
- Settled on 10 VMs for testing
  - 4 TCP flows each (ANL -> LBL)
  - Memory to memory

Network Performance Results



Expedition Results and Comments

- 95 gigabit consistently
  - 98 peak!
  - ~12 GB/s across 50 ms latency!
  - Only 10 nodes, with 1 vm/node
- Single node performance was way higher than we expected
  - CPU utilization even suggests we could handle more bandwidth (5-10 more?)
  - Might be able to improve more with EoIB or SR-IOV
- Single stream performance was worse than native
  - Topped out at 3.5-4 gigabits
- Exotic tuning wasn't really required
- Openstack performed beautifully
  - Was able to cleanly configure this networking setup
  - All of the APIs are usable in their intended ways
  - No duct tape involved!

Where to next?

- Network performance is looking promising, but performance still lags in other areas
  - Particularly to support end to end wide area data movement
  - Block storage is next
  - Our initial results demonstrate virtual disks (using iSCSI/iSer) performing at 1.8 Gbyte/s
  - If these numbers scale, we could potentially push 100 gigabit across the wide area and land it on disk
    - From a dynamically instantiated system
- Hardware support is still lagging
  - GPU/accelerator support is missing, aside from coarse-grained PCI device handoff
  - Makes OS-virtualization approaches (like LXC) compelling
- Need to move to a distributed scalable storage system from monolithic servers
  - While our current approach can be made to perform over time, the servers function as single points of failure
  - Approaches like Triton/Ceph might be better

Acknowledgements

- Magellan (ARRA) team at MCS/ALCF
- Jason Hedden
- Linda Winkler
- ESNet