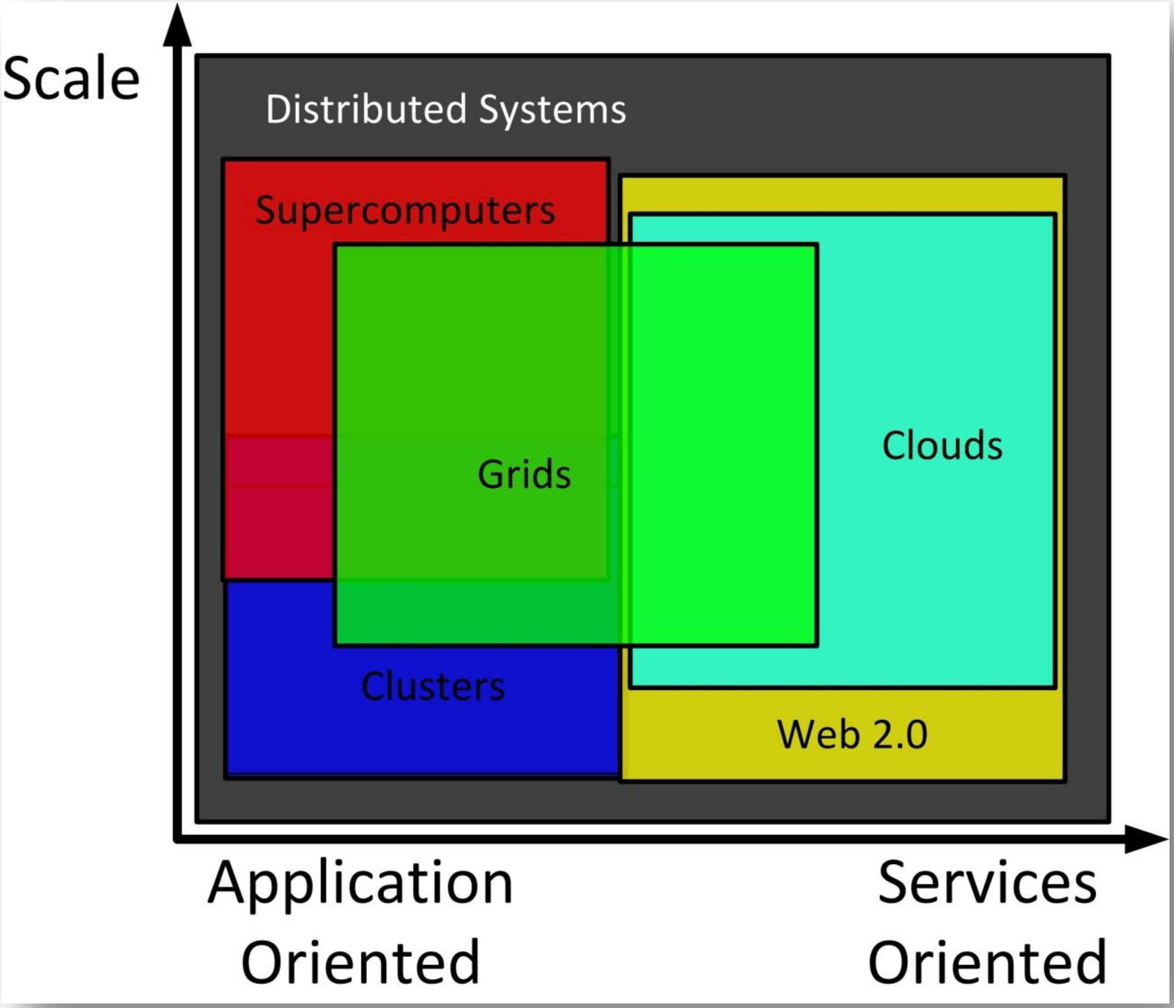
Cloud Computing and Grid Computing 360-Degree Compared

Ioan Raicu
Computer Science Department, Illinois Institute of Technology
Math and Computer Science Division, Argonne National Laboratory

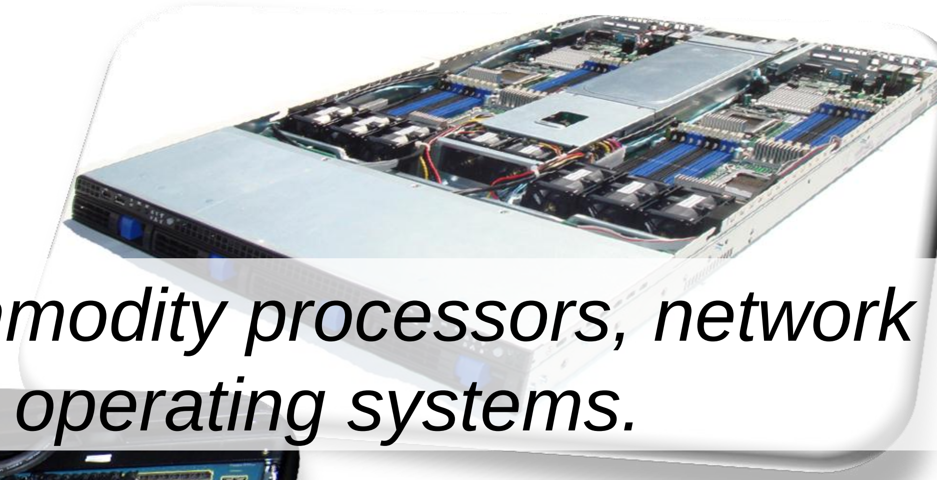
IEEE Fox Valley South Section, IIT
January 14th, 2011

Clusters, Grids, Clouds, and Supercomputers



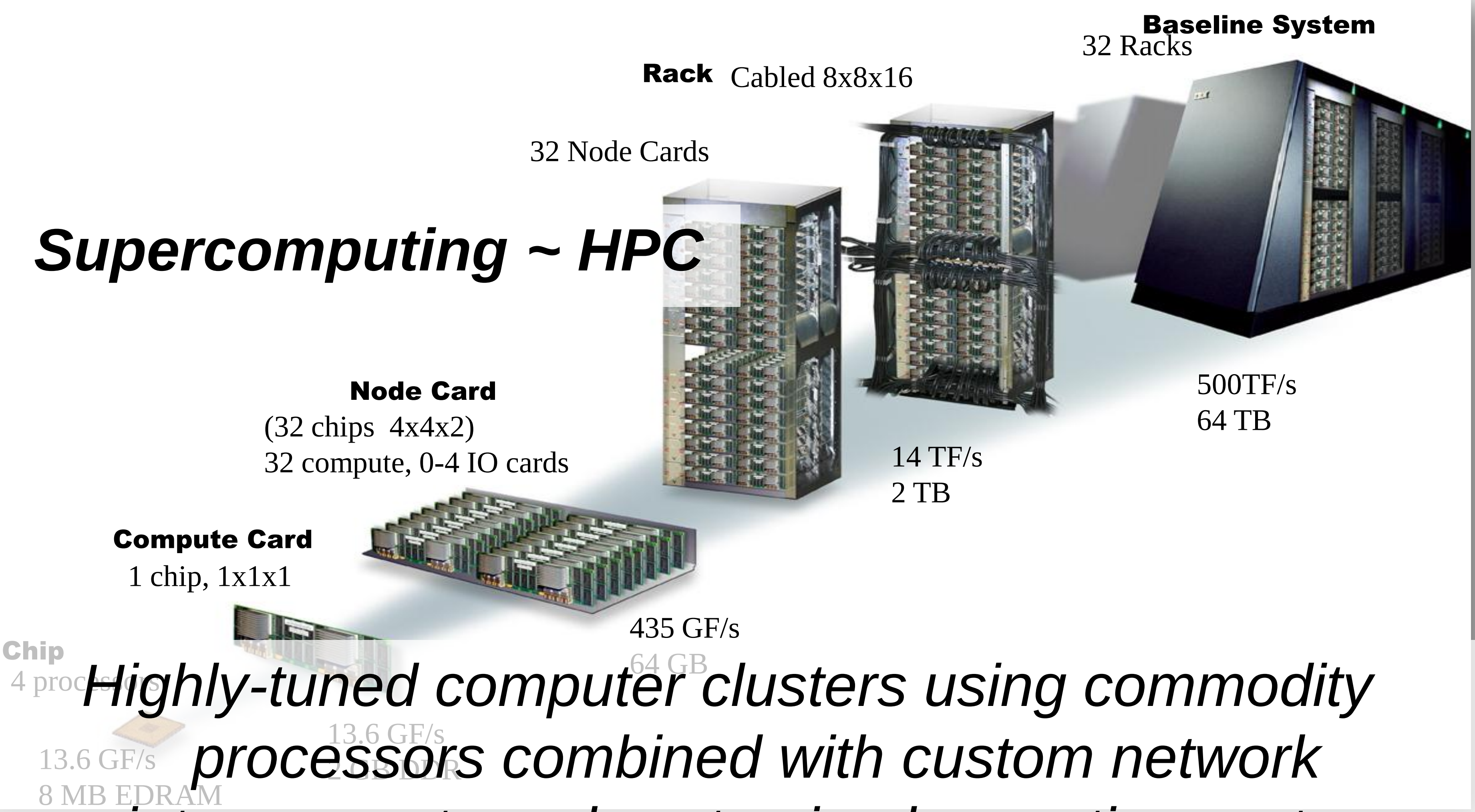
[GCE08] “Cloud Computing and Grid Computing 360-Degree Compared”

Cluster Computing



Computer clusters using commodity processors, network interconnects, and operating systems.

Supercomputing



Supercomputing ~ HPC

Highly-tuned computer clusters using commodity processors combined with custom network interconnects and customized operating system

Top 10 Supercomputers from Top500

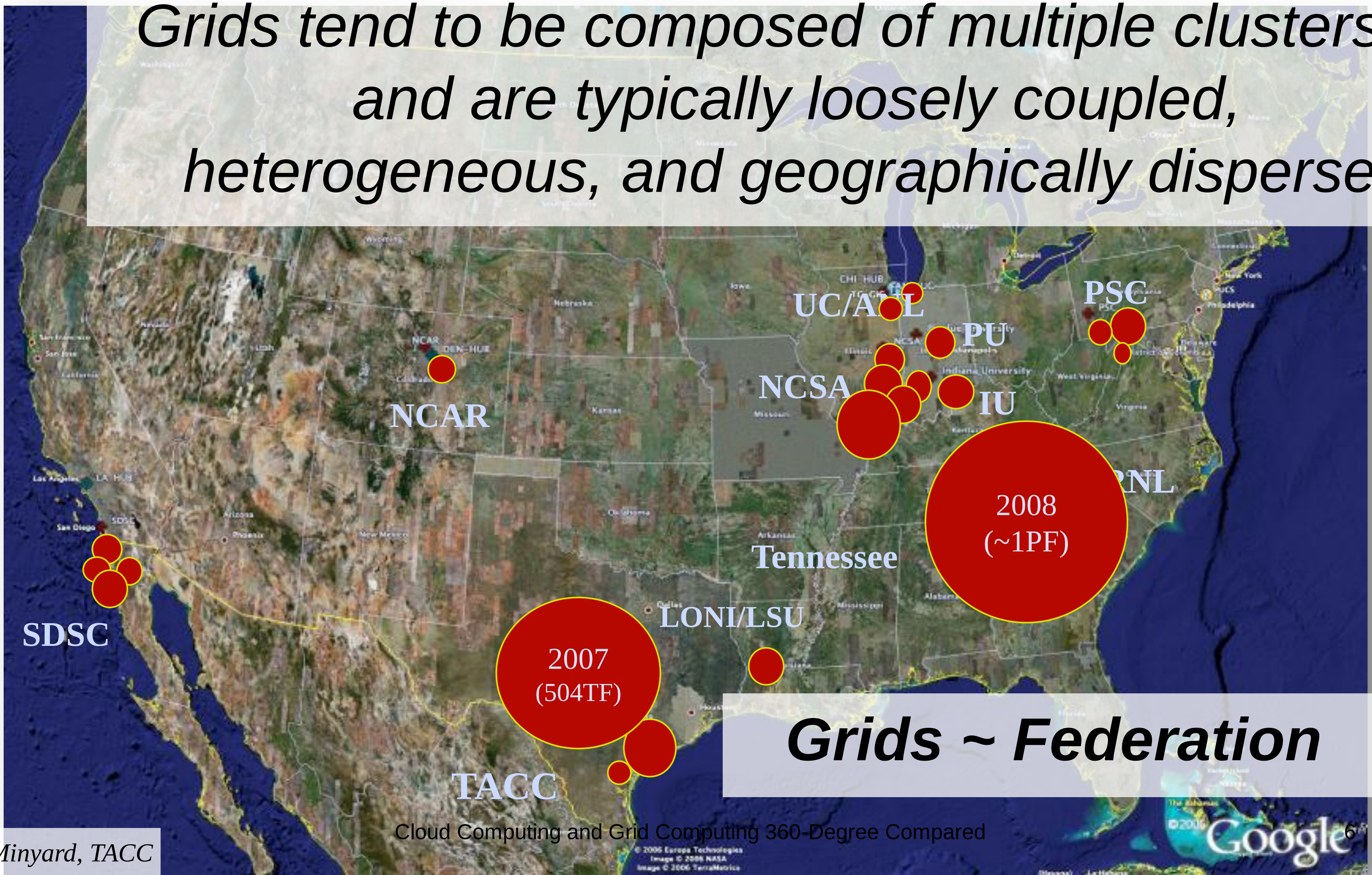
- Cray XT4 & XT5
  - Jaguar #1
  - Kraken #3
- IBM BladeCenter Hybrid
  - Roadrunner #2
- IBM BlueGene/L & BlueGene/P
  - Jugene #4
  - Intrepid #8
  - BG/L #7
- NUDT (GPU based)
  - Tianhe-1 #5
- SGI Altix ICE
  - Plaiedas #6
- Sun Constellation
  - Ranger #9
  - Red Sky #10





Grid Computing

Grids tend to be composed of multiple clusters, and are typically loosely coupled, heterogeneous, and geographically dispersed



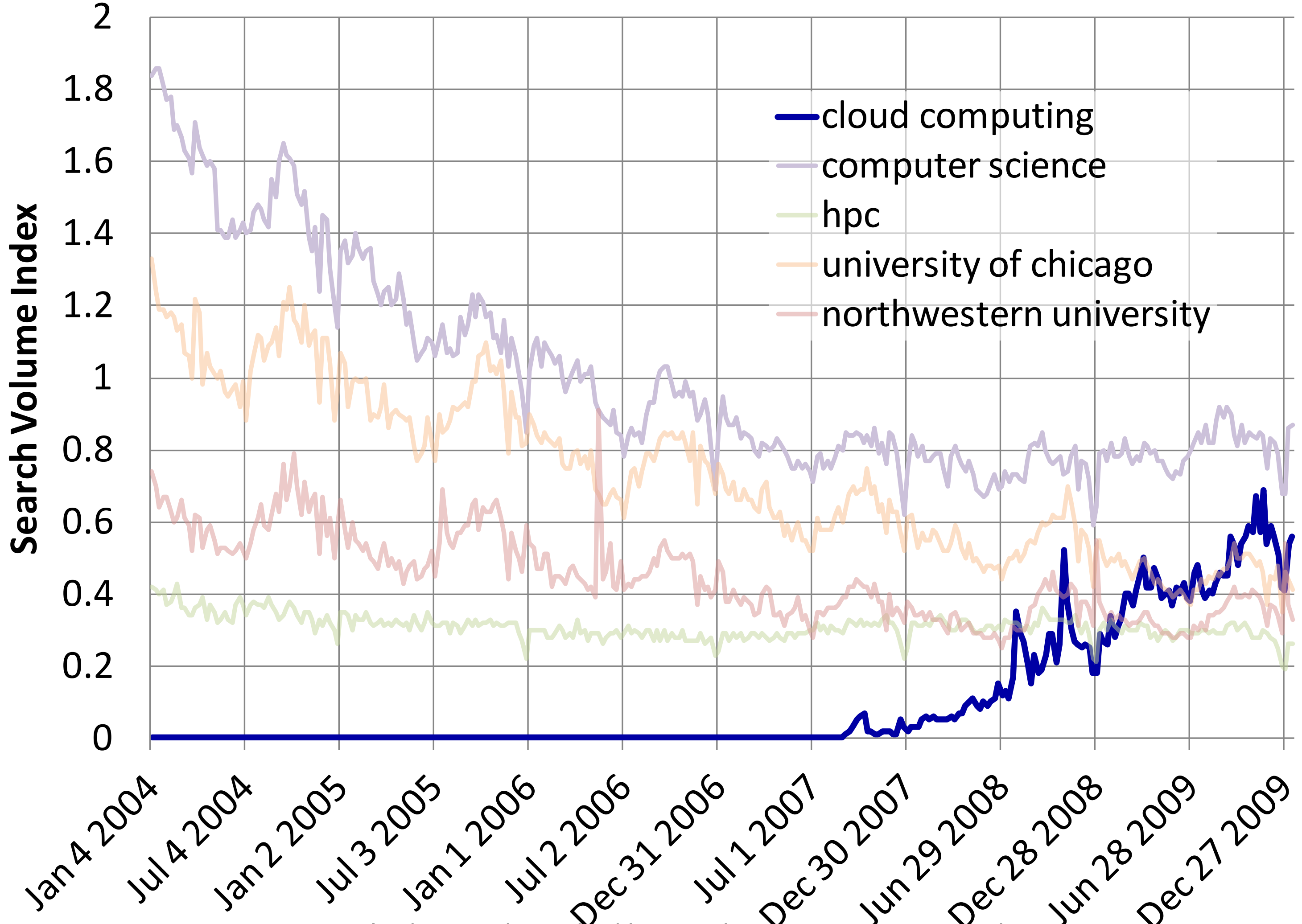
Grids ~ Federation

Tommy Minyard, TACC

Major Grids

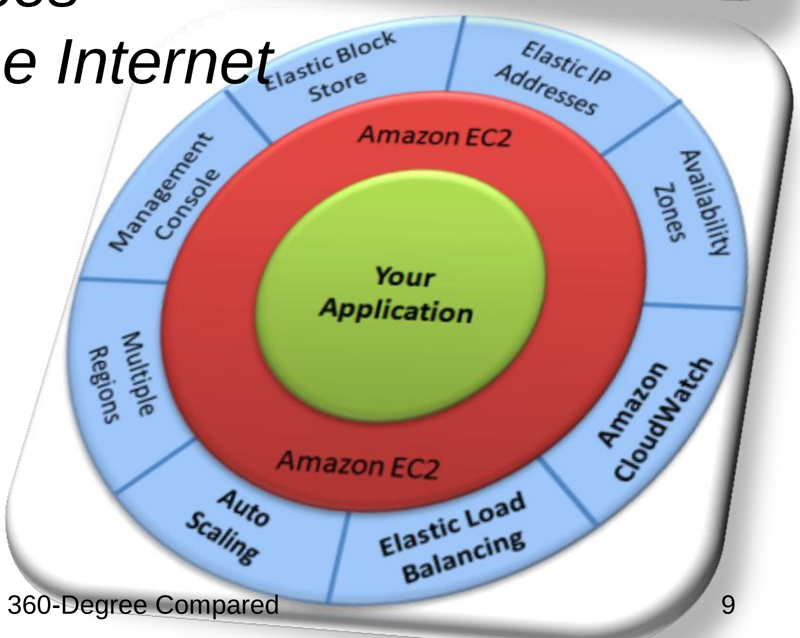
- TeraGrid (TG)
  - 200K-cores across 11 institutions and 22 systems over the US
- Open Science Grid (OSG)
  - 43K-cores across 80 institutions over the US
- Enabling Grids for E-sciencE (EGEE)
- LHC Computing Grid from CERN
- Middleware
  - Globus Toolkit
  - Unicore

Cloud Computing: An Emerging Paradigm



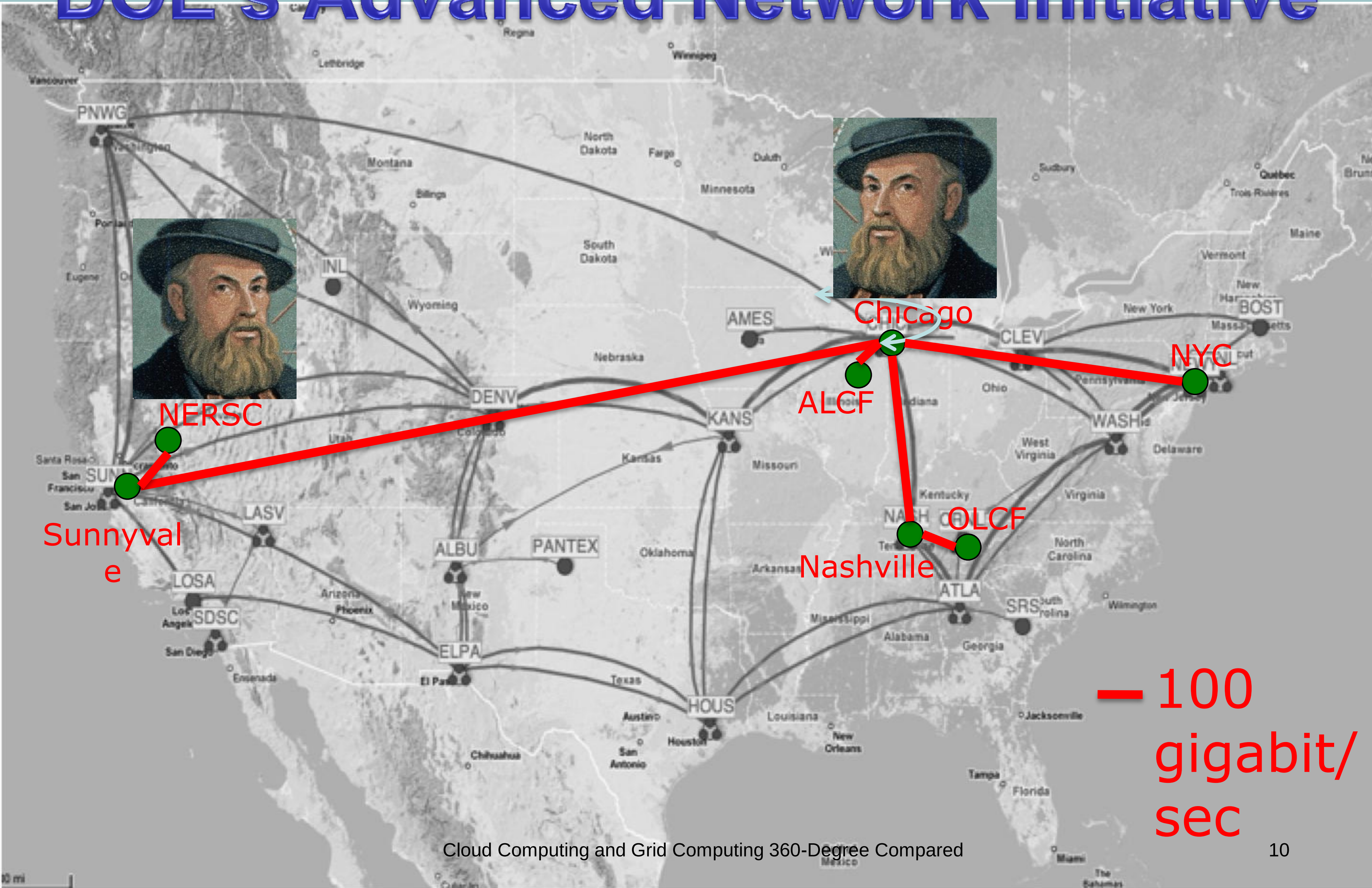
Cloud Computing

- *A large-scale distributed computing paradigm driven by:*
  1. *economies of scale*
  2. *virtualization*
  3. *dynamically-scalable resources*
  4. *delivered on demand over the Internet*



Clouds ~ hosting

Magellan + DOE's Advanced Network Initiative



Major Clouds

- Industry
  - Google App Engine
  - Amazon
  - Windows Azure
  - Salesforce
- Academia/Government
  - Magellan
  - FutureGrid
- Opensource middleware
  - Nimbus
  - Eucalyptus
  - OpenNebula

So is “Cloud Computing” just a new name for Grid?

- IT reinvents itself every five years
- The answer is complicated…

- **YES**: the vision is the same
  - to reduce the cost of computing
  - increase reliability
  - increase flexibility by transitioning from self operation to third party

So is "Cloud Computing" just a new name for Grid?

- **NO**: things are different than they were 10 years ago
  - New needs to analyze massive data, increased demand for computing
  - Commodity clusters are expensive to operate
  - We have low-cost virtualization
  - Billions of dollars being spent by Amazon, Google, and Microsoft to create real commercial large-scale systems with hundreds of thousands of computers
  - The prospect of needing only a credit card to get on-demand access to *infinite computers is exciting; *infinite<O(1000)

So is "Cloud Computing" just a new name for Grid?

- **YES:** the problems are mostly the same
  - How to manage large facilities
  - Define methods to discover, request, and use resources
  - How to implement and execute parallel computations
  - Details differ, but issues are similar

Outline

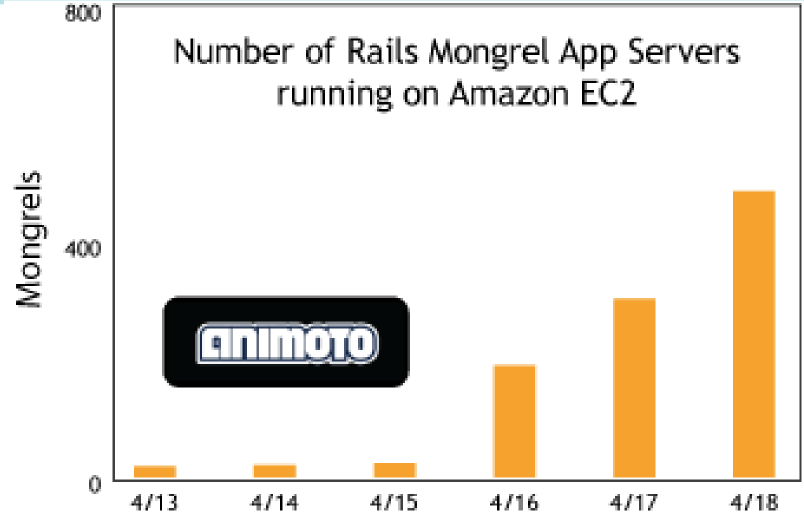
- Business model
- Architecture
- Resource management
- Programming model
- Application model
- Security model

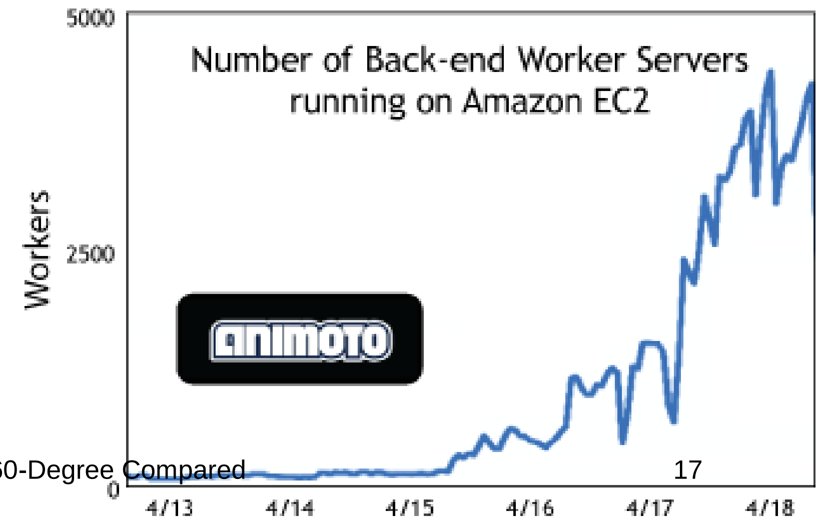
Business Model

- Grids:
  - Largest Grids funded by government
  - Largest user-base in academia and government labs to drive scientific computing
  - Project-oriented: service units
- Clouds:
  - Industry (i.e. Amazon) funded the initial Clouds
  - Large user base in common people, small businesses, large businesses, and a bit of openn science research
  - Utility computing: real money

Business Model
Why is it a big deal?

- Why is this a big deal?
  - No owned infrastructure
  - All resources rented on demand
- Critical for startups with risky business plans
- Not possible without Cloud Computing and a credit card
  - Launched in 2007/2008 timeframe



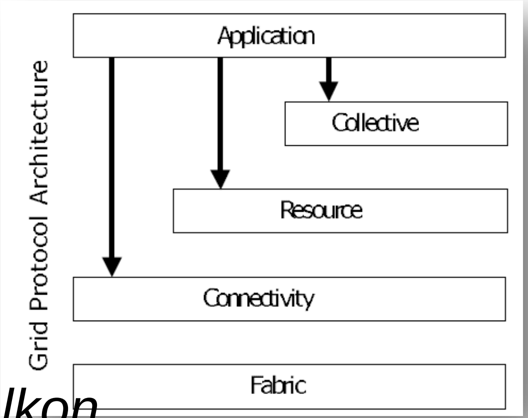


An Example of an Application in the Cloud

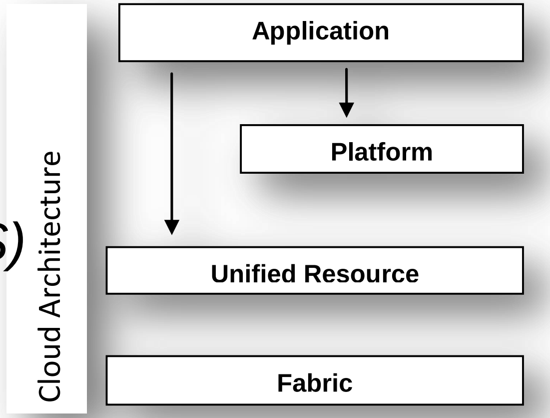
- Animoto
  - Makes it r… ate videos with t…

Architecture

- Grids:
  - Application: *Swift, Grid portals (NVO)*
  - Collective layer: *MDS, Condor-G, Nimrod-G*
  - Resource layer: *GRAM, Falkon, GridFTP*
  - Connectivity layer: *Grid Security Infrastructure*
  - Fabric layer: *GRAM, PBS, SGE, LSF, Condor, Falkon*



- Clouds:
  - Application Layer: *Software as a Service (SaaS)*
  - Platform Layer: *Platform as a Service (PaaS)*
  - Unified Resource: *Infrastructure as a Service (IaaS)*
  - Fabric: *IaaS*



Resource Management

- Compute Model
  - batch-scheduled vs. time-shared
- Data Model
  - Data Locality
  - Combining compute and data management
- Virtualization
  - Slow adoption vs. central component
- Monitoring
- Provenance

Programming and Application Model

- Grids:
  - Tightly coupled
    - High Performance Computing (MPI-based)
  - Loosely Coupled
    - High Throughput Computing
    - Workflows
  - Data Intensive
    - Map/Reduce
- Clouds:
  - Loosely Coupled, transactional oriented

Programming Model Issues

- **Multicore** processors
- Massive **task parallelism**
- Massive **data parallelism**
- Integrating **black box applications**
- Complex **task dependencies** (task graphs)
- **Failure**, and other execution management issues
- **Dynamic task graphs**
- Documenting **provenance** of data products
- **Data management**: input, intermediate, output
- **Dynamic data access** involving large amounts of data

Gateways

- Aimed to simplify usage of complex resources
- Grids
  - Front-ends to many different applications
  - Emerging technologies for Grids
- Clouds
  - Standard interface to Clouds

An Example of an Application in the Grid

Security Model

- Grids
  - Grid Security Infrastructure (GSI)
  - Stronger, but steeper learning curve and wait time
    - Personal verification: phone, manager, etc
- Clouds
  - Weaker, can use credit card to gain access, can reset password over plain text email, etc

Conclusion

- Move towards a mix of micro-production and large utilities, with load being distributed among them dynamically
  - Increasing numbers of small-scale producers (local clusters and embedded processors—in shoes and walls)
  - Large-scale regional producers
- Need to define protocols
  - Allow users and service providers to discover, monitor and manage their reservations and payments
  - Interoperability

Conclusion (cont)

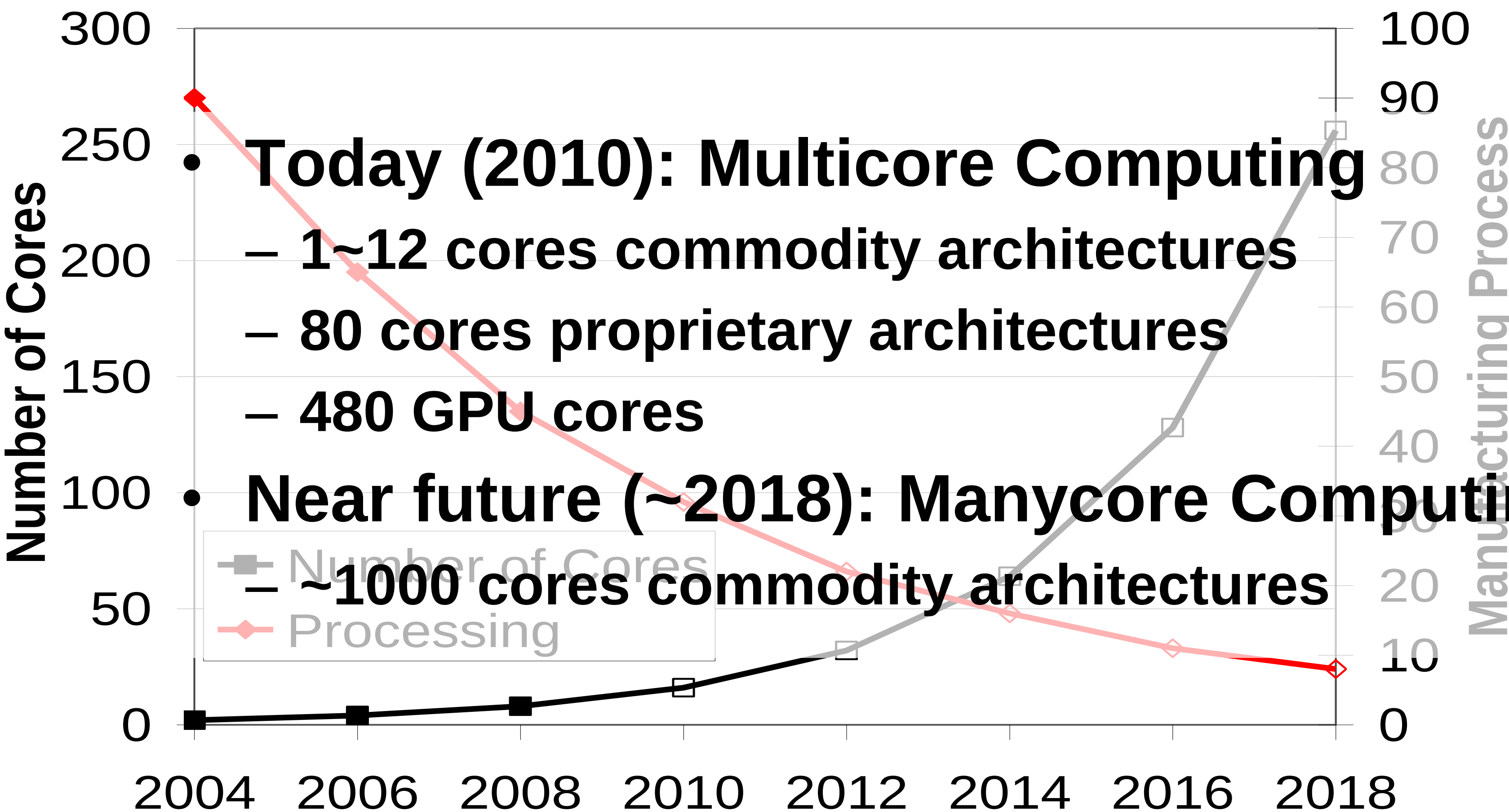
- Need to combine the centralized scale of today's Cloud utilities, and the distribution and interoperability of today's Grid facilities
- Need support for on-demand provisioning
- Need tools for managing both the underlying resources and the resulting distributed computations
- Security and trust will be a major obstacle for commercial Clouds by large companies that have in-house IT resources to host their own data centers

A Glimpse into my Own Research

- Distributed Systems
- Cluster Computing
- Grid Computing
- Supercomputing
- Cloud Computing
- Manycore Computing
- Petascale and Exascale Computing
- Data-Intensive Computing

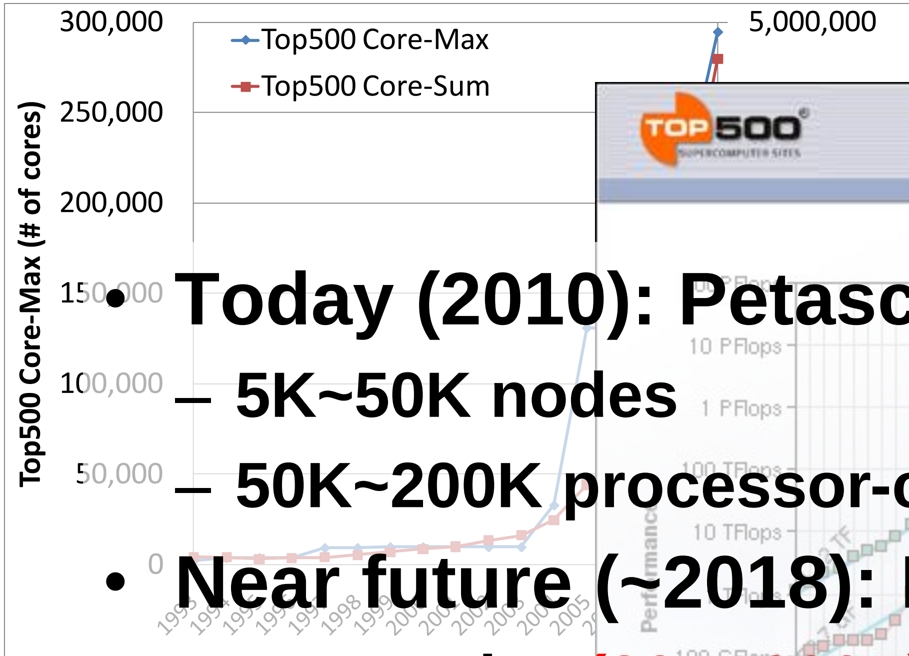
Manycore Computing

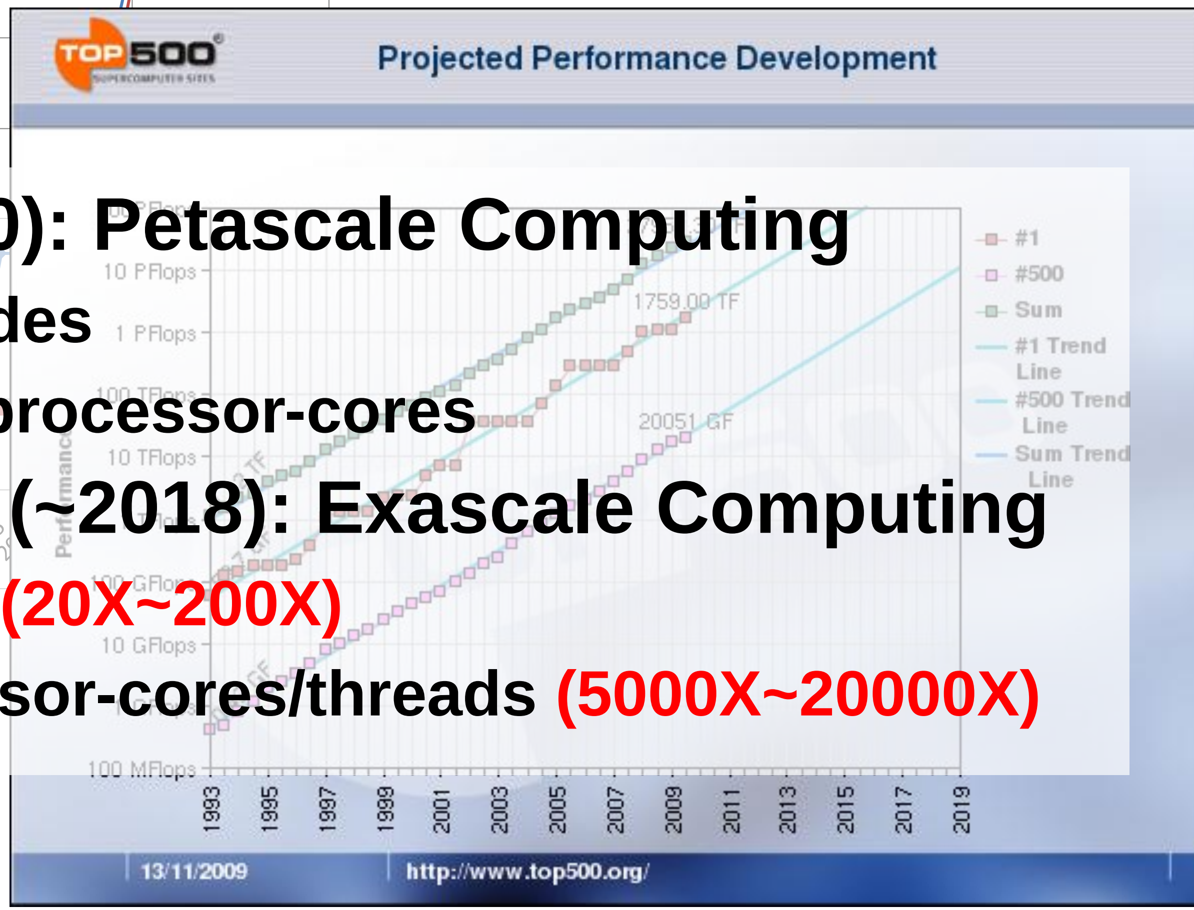
- **Today (2010): Multicore Computing**
  - **1~12 cores commodity architectures**
  - **80 cores proprietary architectures**
  - **480 GPU cores**
- **Near future (~2018): Manycore Computing**
  - **~1000 cores commodity architectures**



Pat Helland, Microsoft, The Irresistible Forces Meet the Movable Objects, November 9th, 2007

Exascale Computing





- **Today (2010): Petascale Computing**
  - **5K~50K nodes**
  - **50K~200K processor-cores**
- **Near future (~2018): Exascale Computing**
  - **~1M nodes (20X~200X)**
  - **~1B processor-cores/threads (5000X~20000X)**

Top500 Projected Development, http://www.top500.org/lists/2009/11/performance_development

Cloud Computing

- Relatively new paradigm… 3 years old
- Amazon in 2009
  - 40K servers split over 6 zones
    - 320K-cores, 320K disks
    - $100M costs + $12M/year in energy costs
    - Revenues about $250M/year
- Amazon in 2018
  - Will likely look similar to exascale computing
    - 100K~1M nodes, ~1B-cores, ~1M disks
    - $100M~$200M costs + $10M~$20M/year in energy
    - Revenues 100X~1000X of what they are today

Common Challenges

- Power efficiency
  - Will limit the number of cores on a chip (Manycore)
  - Will limit the number of nodes in cluster (Exascale and Cloud)
  - Will dictate a significant part of the cost of ownership
- Programming models/languages
  - Automatic parallelization
  - Threads, MPI, workflow systems, etc
  - Functional, imperative
  - Languages vs. Middlewares

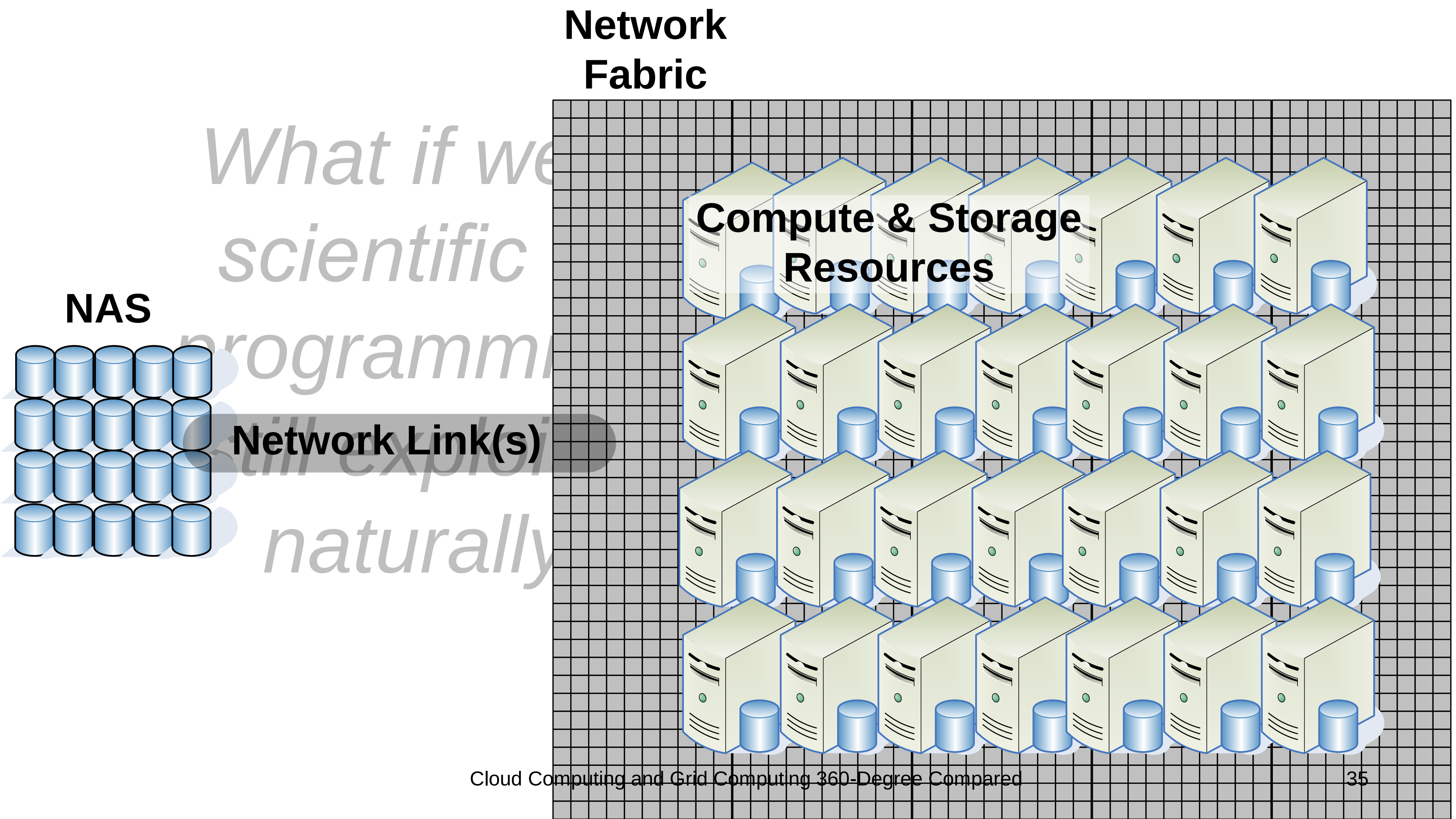
Common Challenges

- Bottlenecks in scarce resources
  - Storage (Exascale and Clouds)
  - Memory (Manycore)
- Reliability
  - How to keep systems operational in face of failures
  - Checkpointing (Exascale)
  - Node-level replication enabled by virtualization (Exascale and Clouds)
  - Hardware redundancy and hardware error correction (Manycore)

Research Directions

- ***Decentralization is critical***
  - Computational resource management (e.g. LRMs)
  - Storage systems (e.g. parallel file systems)
- ***Data locality must be maximized, while preserving I/O interfaces***
  - POSIX I/O on shared/parallel file systems ignore locality
  - Data-aware scheduling coupled with distributed file systems that expose locality is the key to scalability over the next decade

Storage System Architecture



Plan of Work

- ***Building on my own research (e.g. data-diffusion), parallel file systems (PVFS), and distributed file systems (e.g. GFS)***
- Build a distributed file system for HEC
  - It should complement parallel file systems, not replace them
- Critical issues:
  - Must mimic parallel file systems interfaces and features in order to get wide adoption
  - Must handle some workloads currently run on parallel file systems significantly better

Plan of Work (cont)

- Access Interfaces and Semantics
  - POSIX-like compliance for generality (e.g. via FUSE)
  - Relaxed semantics to increase scalability
    - Eventual consistency on data modifications
    - Write-once read-many data access patterns
- Distributed metadata management
  - Employ structured distributed hash tables like data-structures
  - Must have O(1) put/get costs
  - Can leverage network-aware topology overlays
- Distribute data across many nodes
  - Must maintain and expose data locality in access patterns

Access Patterns

- **1-many read** (all processes read the same file and are not modified)
- **many-many read/write** (each process read/write to a unique file)
- **write-once read-many** (files are not modified after it is written)
- **append-only** (files can only be modified by appending at the end of files)
- **metadata** (metadata is created, modified, and/or destroyed at a high rate).

Usage Scenarios

- **machine boot-up** (e.g. reading OS image on all nodes)
- **application loading** (e.g. reading scripts, binaries, and libraries on all nodes/processes)
- **common user data loading** (e.g. reading a common read-only database on all nodes/processes)
- **checkpointing** (e.g. writing unique files per node/process)
- **log writing** (writing unique files per node/process)
- **many-task computing** (each process reads some files, unique or shared, and each process writes unique files)

More Information

- More information:
  - http://www.cs.iit.edu/~iraicu/
  - iraicu@cs.iit.edu
- Questions?