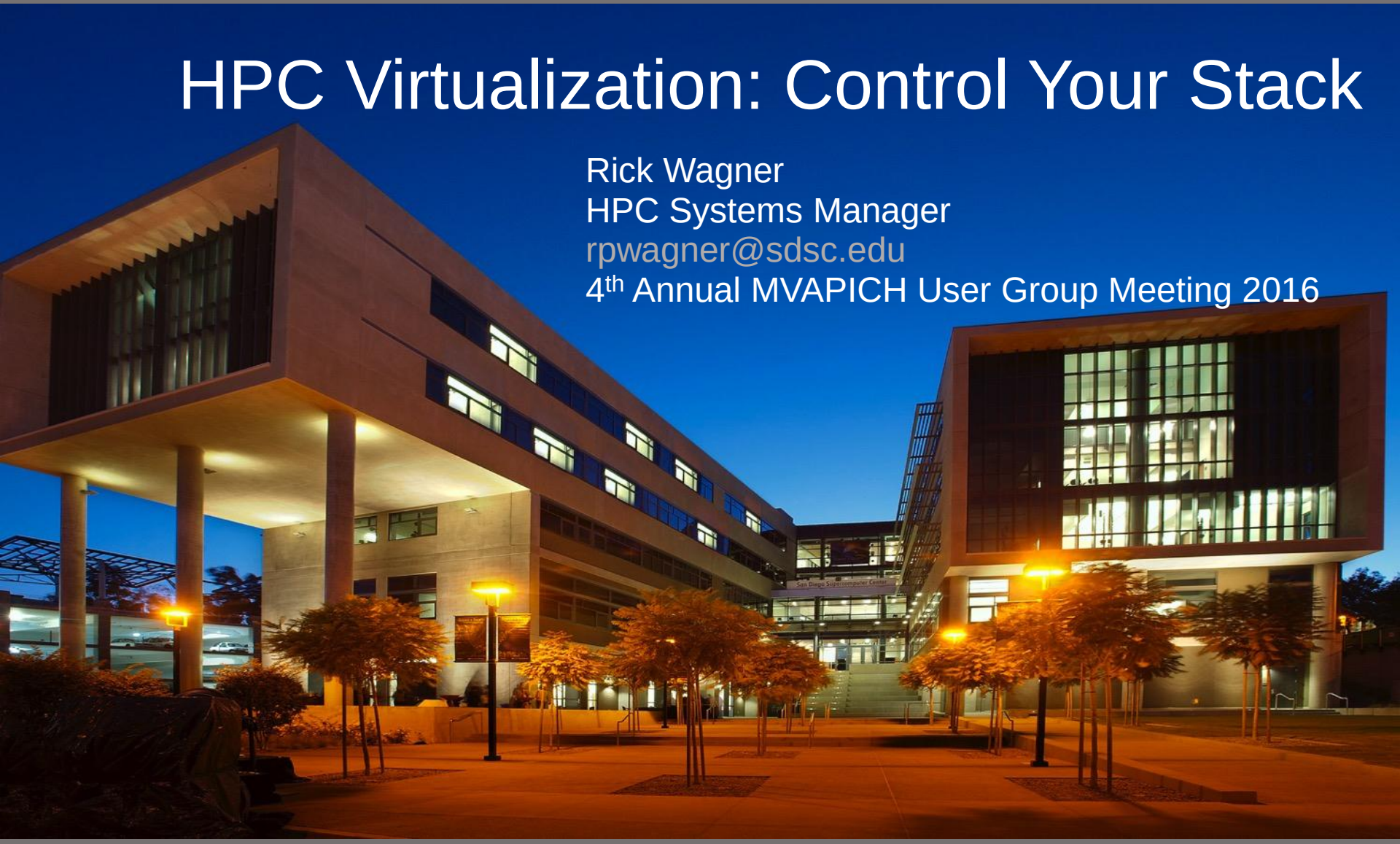
HPC Virtualization: Control Your Stack



Rick Wagner
HPC Systems Manager
rpwagner@sdsc.edu
4th Annual MVAPICH User Group Meeting 2016

comet.sdsc.edu
32 racks of awesomeness

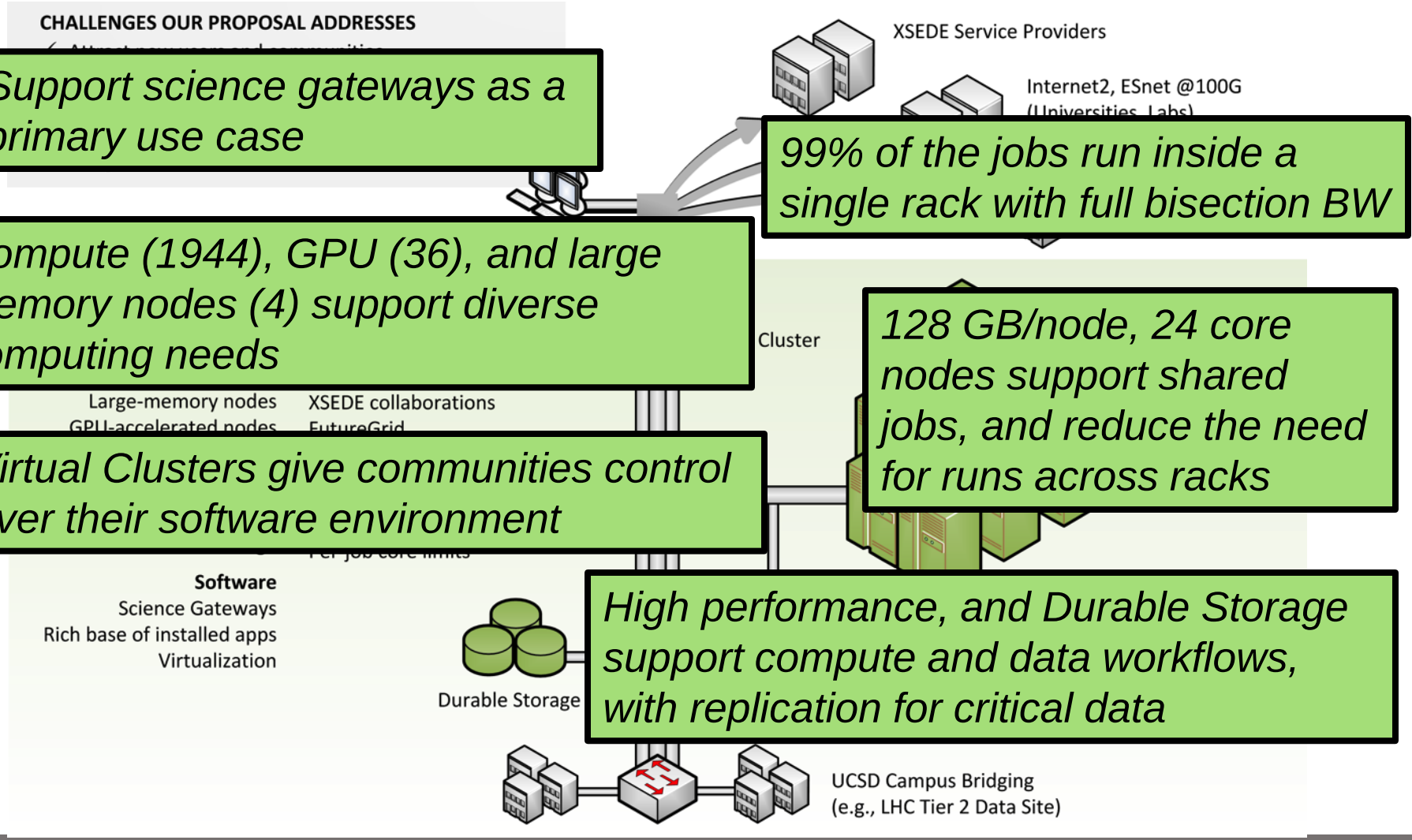
Project High Level Goals

- **"… expand the use of high end resources to a much larger and more diverse community**
- **… support the entire spectrum of NSF communities**
- **... promote a more comprehensive and balanced portfolio**
- **… include research communities that are not users of traditional HPC systems."**

NSF solicitation 13-528

The long tail of science needs HPC

Comet's integrated architecture is a platform for a wide range of computing modalities



Support science gateways as a primary use case

99% of the jobs run inside a single rack with full bisection BW

Compute (1944), GPU (36), and large memory nodes (4) support diverse computing needs

128 GB/node, 24 core nodes support shared jobs, and reduce the need for runs across racks

Virtual Clusters give communities control over their software environment

High performance, and Durable Storage support compute and data workflows, with replication for critical data

Virtualization Staffing

SDSC: Project management, system managements, systems software (Nucleus)

IU: User support, client software (Cloudmesh)

Virtual Clusters

Goal:

Provide a near bare metal HPC performance and management experience

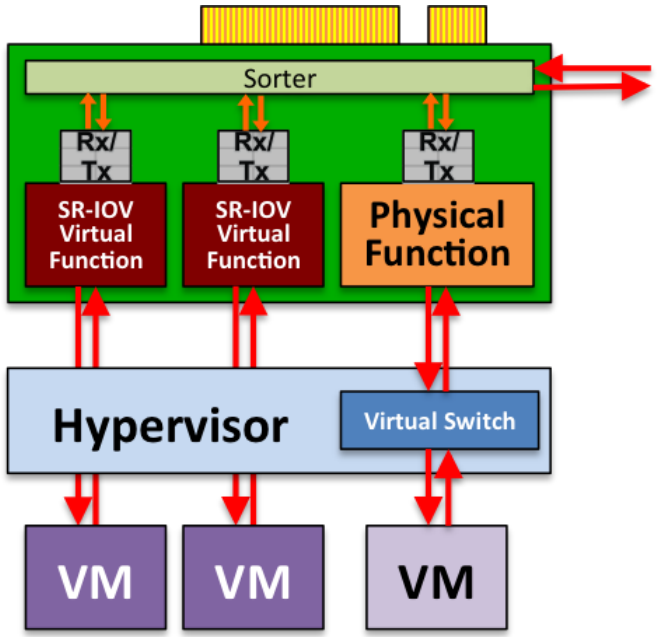
Target Use

Projects that can manage their own cluster, and:

- can't fit our batch environment, and
- don't want to buy hardware or
- have bursty or intermittent need

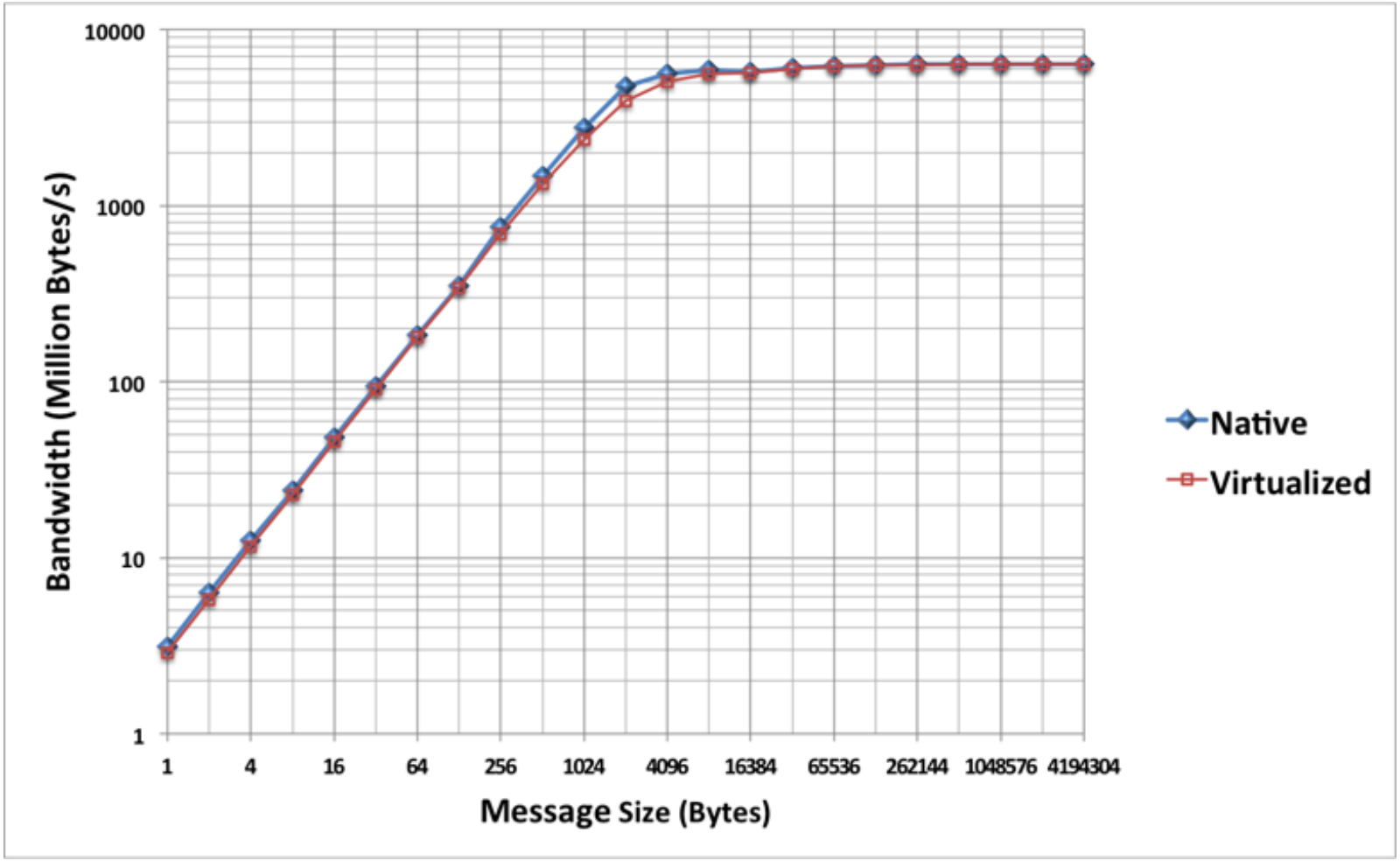
Single Root I/O Virtualization in HPC

- Problem: Virtualization generally has resulted in significant I/O performance degradation (e.g., excessive DMA interrupts)
- Solution: SR-IOV and Mellanox ConnectX-3 InfiniBand host channel adapters
  - One physical function → multiple virtual functions, each light weight but with its own DMA streams, memory space, interrupts
  - Allows DMA to bypass hypervisor to VMs
- *SRIOV enables virtual HPC cluster w/ near-native InfiniBand latency/bandwidth and minimal overhead*

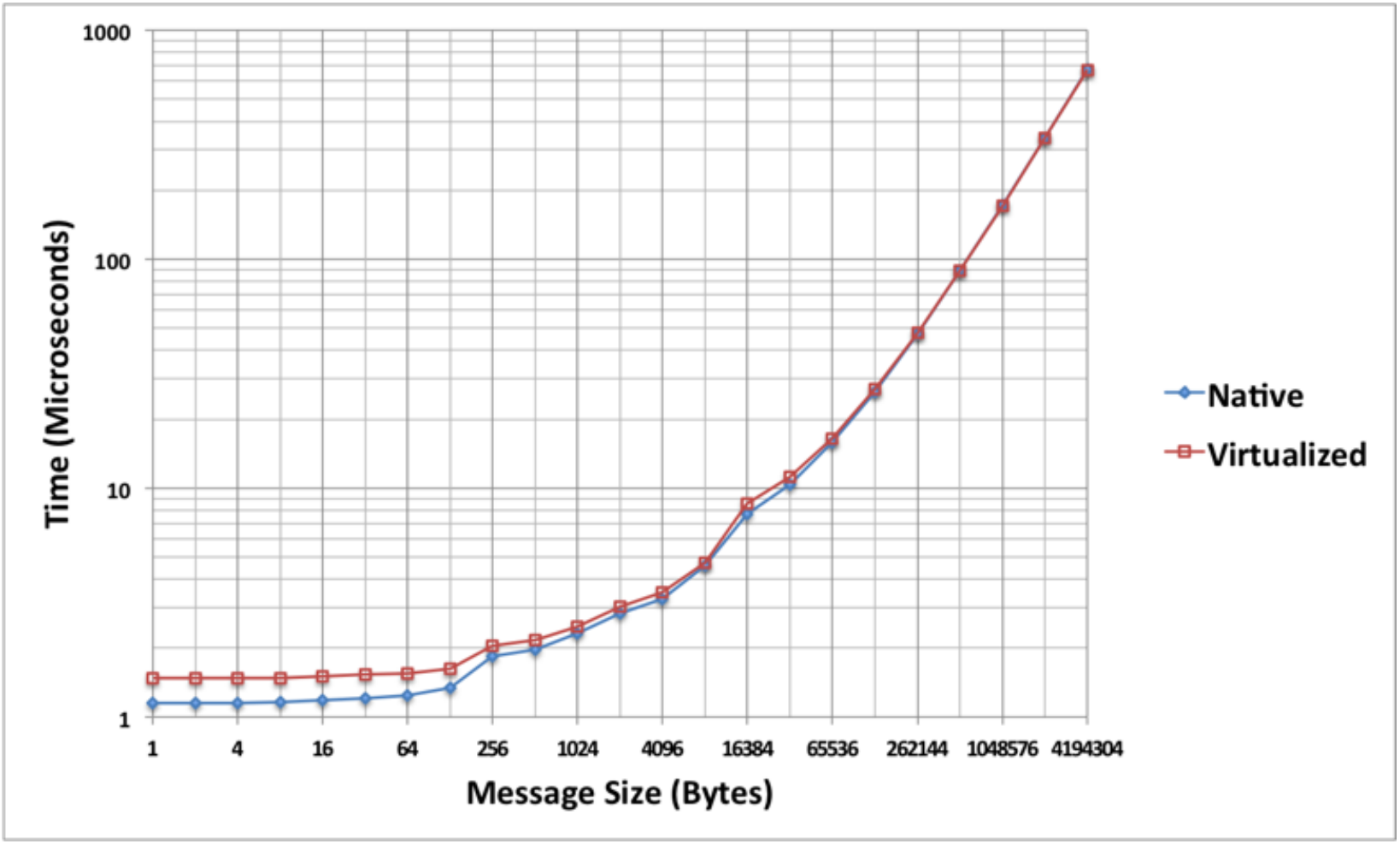


Mellanox TECHNOLOGIES

MPI bandwidth slowdown from SR-IOV is at most 1.21 for medium-sized messages & negligible for small & large ones

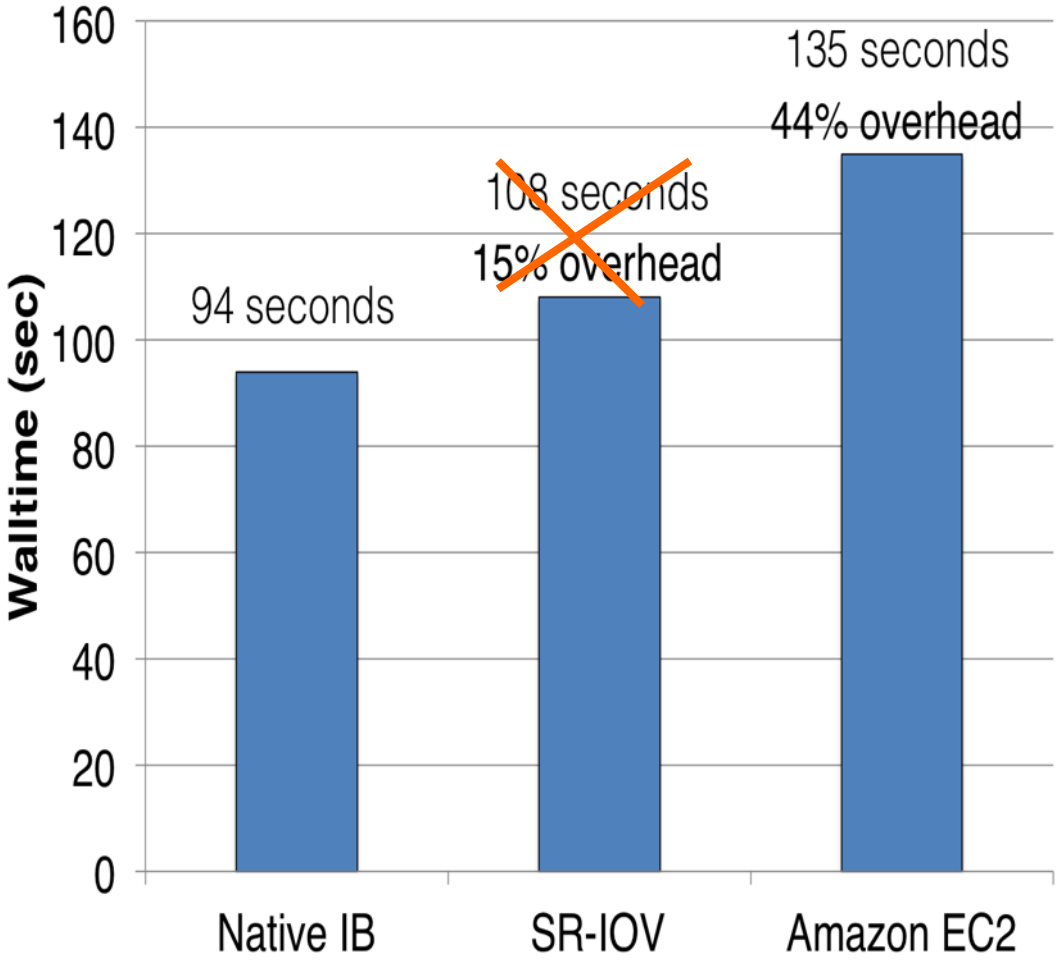


MPI latency slowdown from SR-IOV is at most 1.32 for small messages & negligible for large ones



WRF Weather Modeling – 2% Overhead with SR-IOV IB

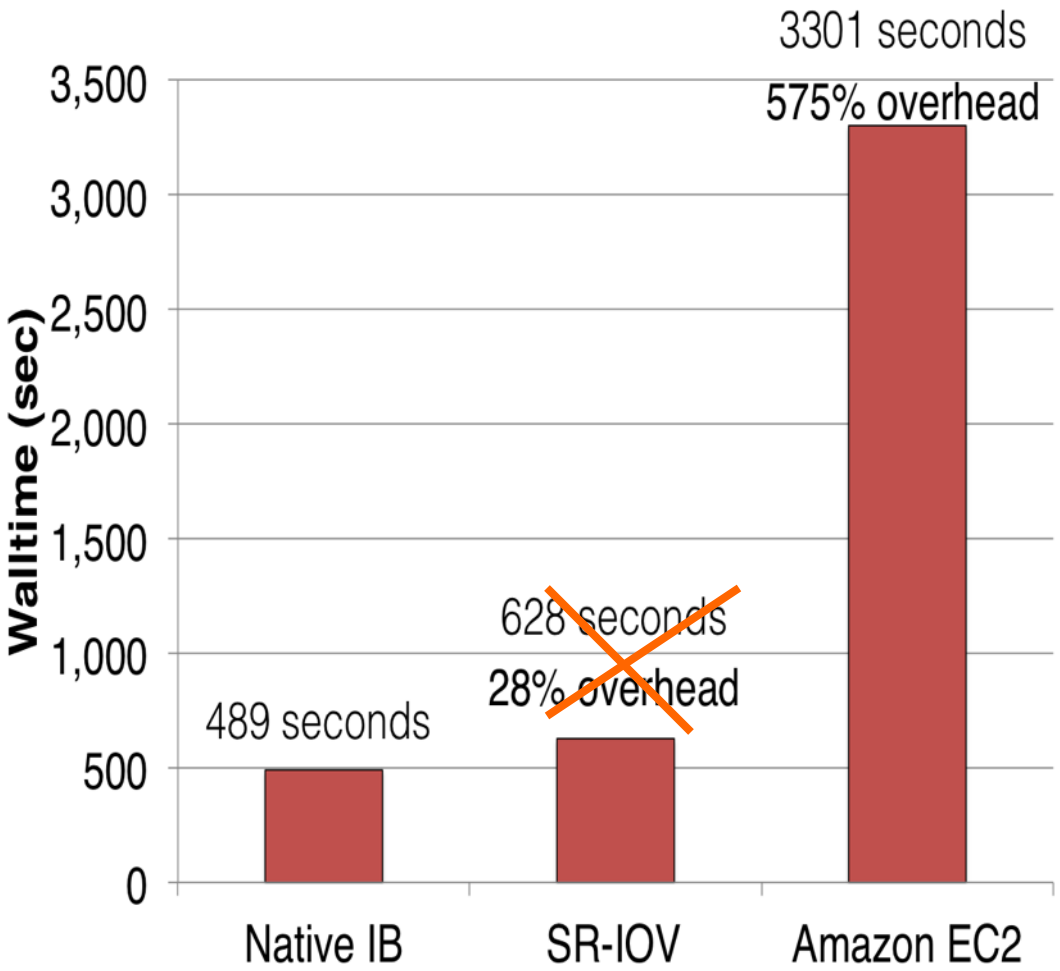
- 96-core (6-node) calculation
- Nearest-neighbor communication
- Scalable algorithms
- ~~SR-IOV incurs modest (15%) performance hit~~
- 2% slower w/ SR-IOV vs native IB!
- Still 20% faster than EC2 Despite 20% slower CPUs



WRF 3.4.1 – 3hr forecast

Quantum ESPRESSO: ~~28%~~ 8% (!) Overhead

- 48-core (3 node) calculation
- CG matrix inversion - irregular communication
- 3D FFT matrix transposes (all-to-all communication)
- ~~28% slower w/ SR-IOV vs native IB~~
- 8% slower w/ SR-IOV vs native IB!
- SR-IOV still > 500% faster than EC2 Despite 20% slower CPUs

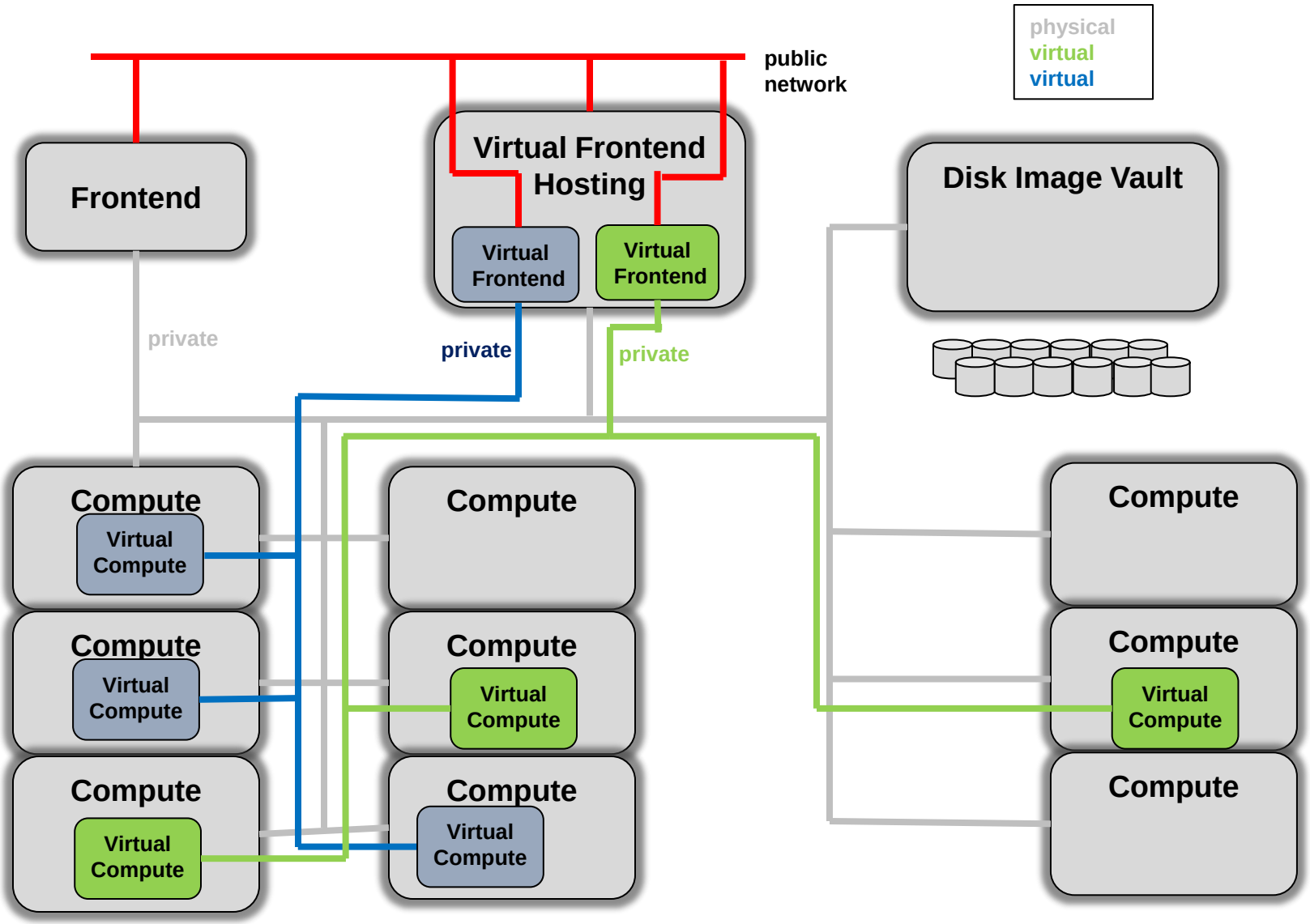


Quantum Espresso 5.0.2 – DEISA AUSURF112 benchmark

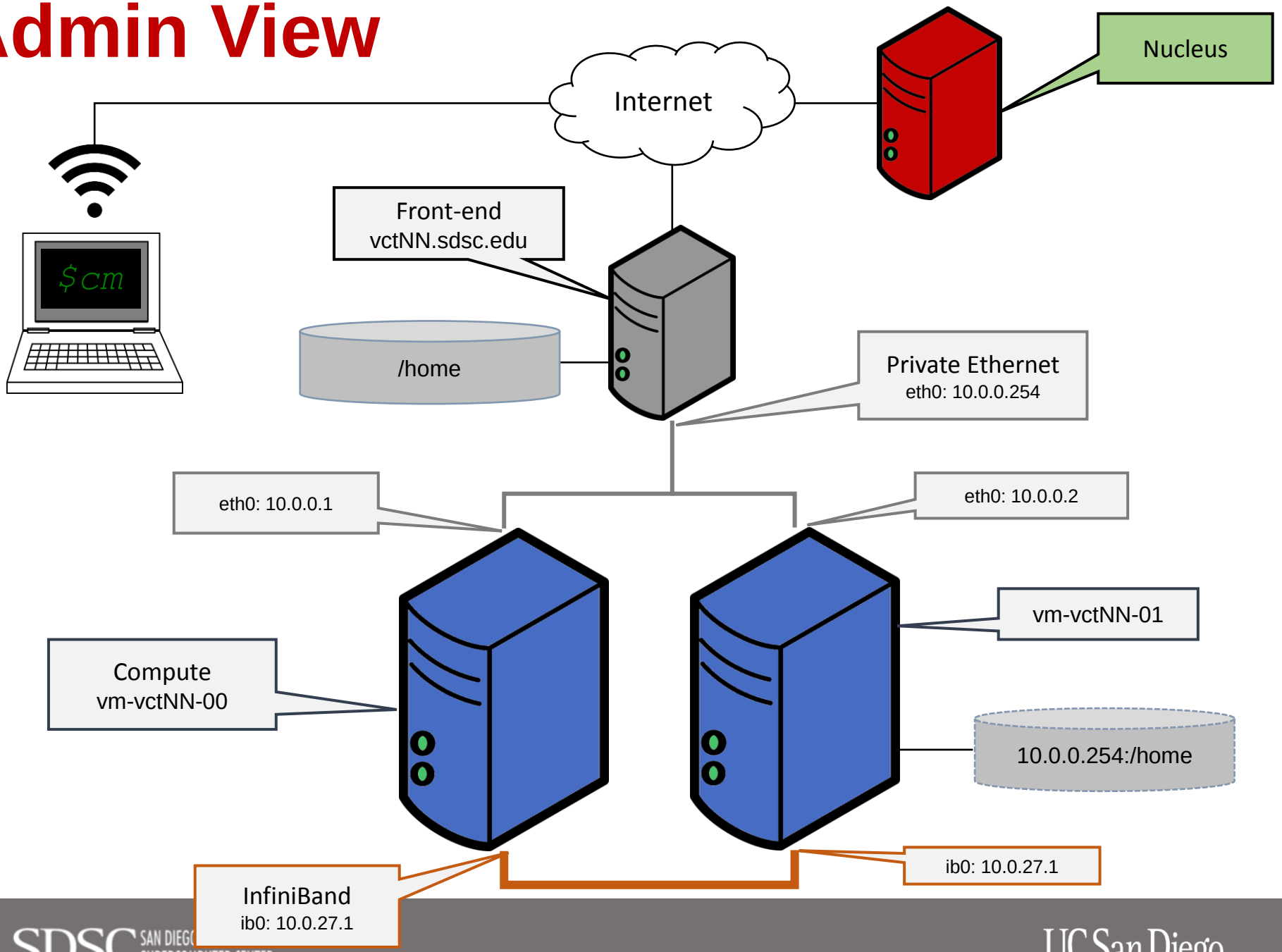
Selected Technologies: Enabling Major Impact

- KVM—Let's us run virtual machines (all processor features)
- SR-IOV—Makes MPI go fast on VMs
- Rocks—Systems management
- ZFS—Disk image management
- VLANs—Isolate virtual cluster management network
- pkeys—Isolate virtual cluster IB network
- **Nucleus—Coordination engine (scheduling, provisioning, status, etc.)**
- **Client – Cloudmesh**

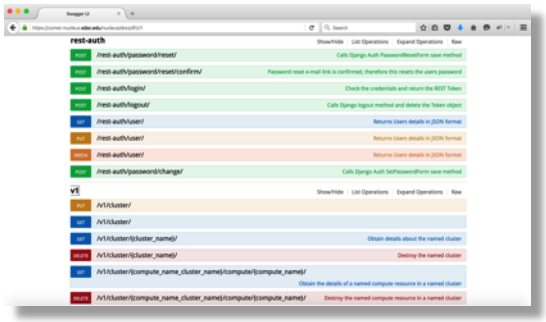
User-Customized HPC



Admin View

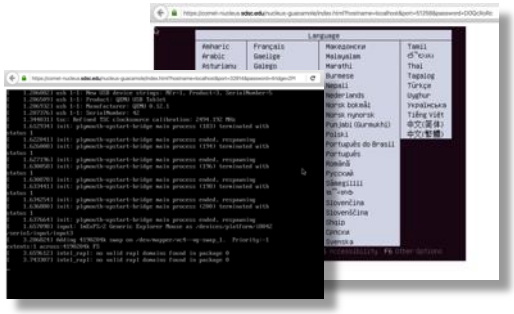

Nucleus
Internet
Front-end
vctNN.sdsc.edu
$ cm
/home
Private Ethernet
eth0: 10.0.0.254
eth0: 10.0.0.1
eth0: 10.0.0.2
vm-vctNN-01
Compute
vm-vctNN-00
10.0.0.254:/home
ib0: 10.0.27.1
InfiniBand
ib0: 10.0.27.1

Accessing Comets Virtual Cluster Capabilities



- REST API
- Command line interface
- Command shell for scripting
- Console Access
- (Portal)

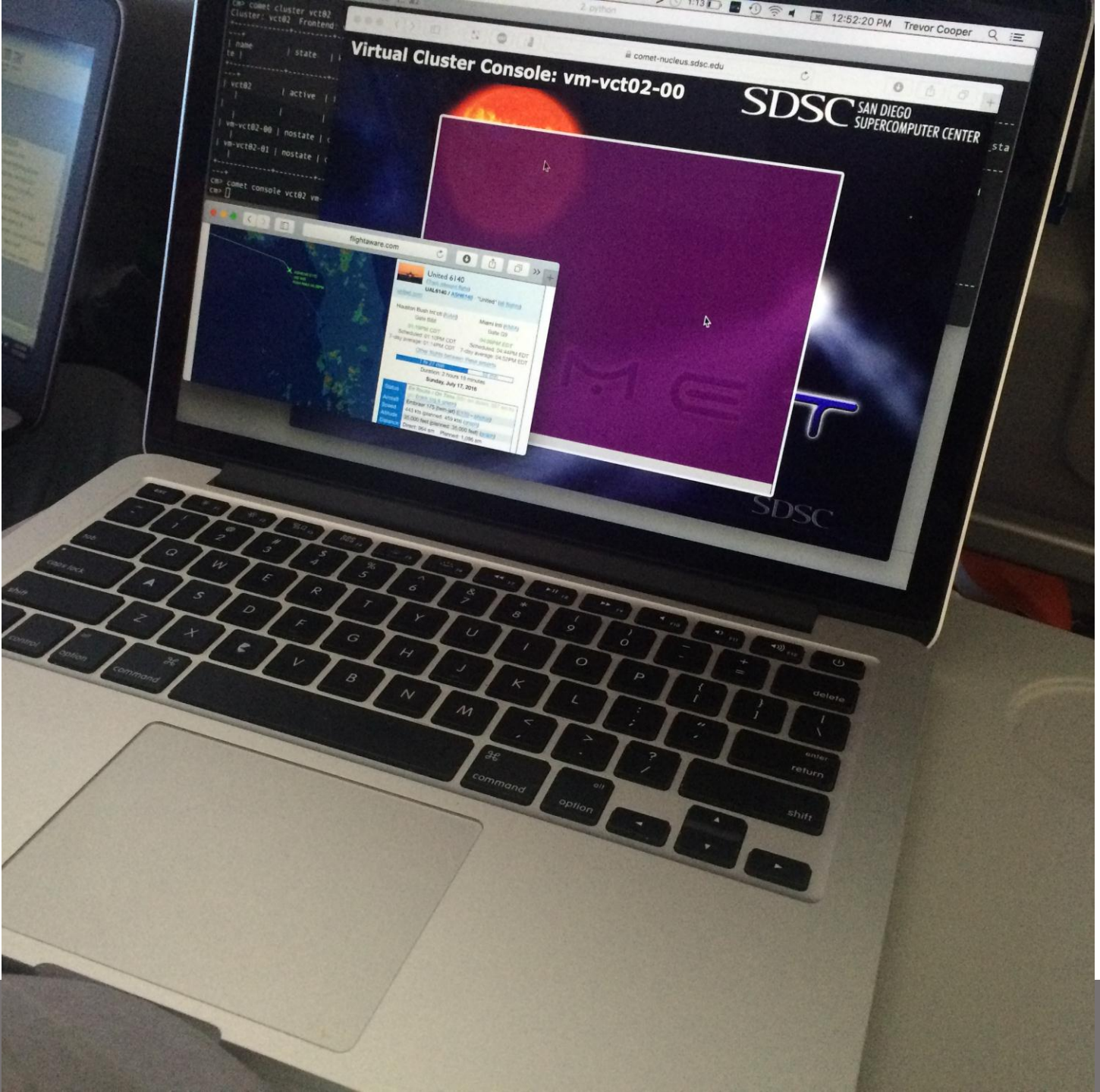
```
Usage:
   comet ll [CLUSTERID] [--format=FORMAT]
   comet cluster [CLUSTERID]
                 [--format=FORMAT]
   comet computeset [COMPUTESETID]
   comet power on CLUSTERID [--count=NUMNODES] [NODESPARAM]
                 [--allocation=ALLOCATION]
                 [--walltime=WALLTIME]
   comet power (off|reboot|reset|shutdown) CLUSTERID [NODESPARAM]
   comet console CLUSTERID [COMPUTENODEID]
   comet image list
   comet image upload [--imagename=IMAGENAME] PATHIMAGEFILE
   comet image attach IMAGENAME CLUSTERID [COMPUTENODEID]
   comet image detach CLUSTERID [COMPUTENODEID]
   comet node rename CLUSTERID OLDNAME NEWNAME
```

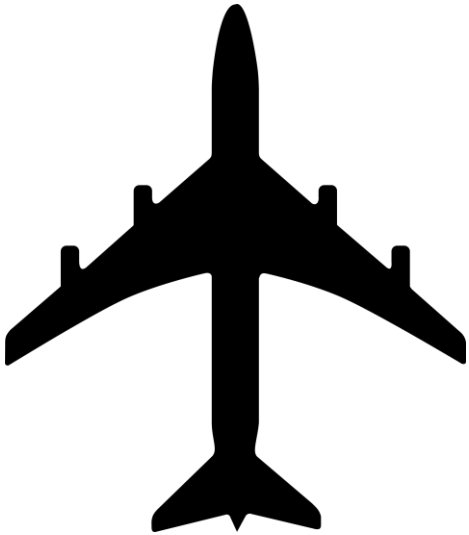


```
(ENV)big:client big$ cm help

Documented commands (type help <command>):
========================================
EOF       comet    group      key       open     refresh      server  var
banner    context  h          launcher  pause    register     shell   verbose
check     debug    help       limits    portal   reservation  ssh     version
clear     default  history    list      py       reset        submit  vm
cloud     echo     hpc        man       q        rsync        sync
cluster   exec     image      network   quit     secgroup     timer
color     flavor   inventory  nova      quota    select       usage
```

User does NOT see: Rocks, Slurm, etc.


Virtual Cluster Console: vm-vct02-00
SDSC
SAN DIEGO
SUPERCOMPUTER CENTER
12:52:20 PM Trevor Cooper
comet-nucleus.sdsc.edu
flightaware.com
United 6140



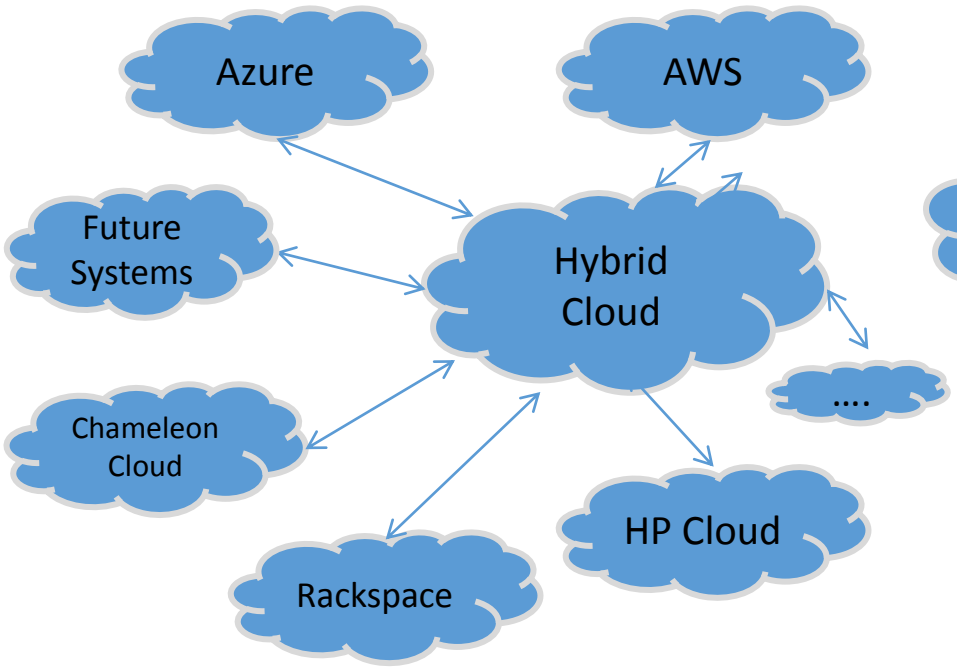
Cloudmesh

Hybrid Cloud Management

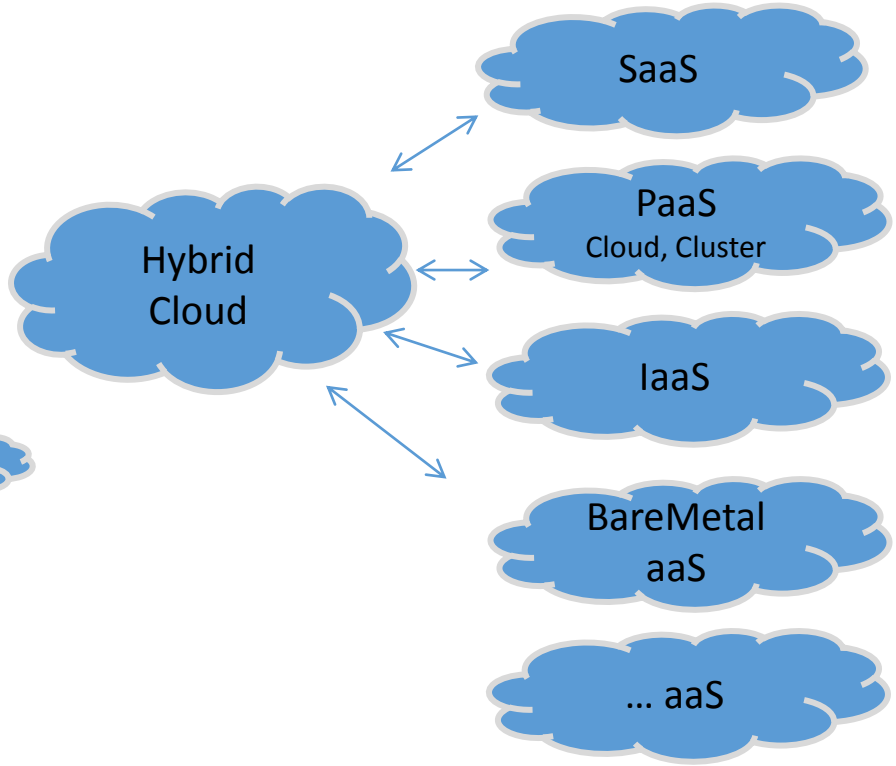
http://cloudmesh.github.io/client/

Integrated Clouds

Capacity Integration



Technology Integration



Cloudmesh Fills the Gap

- Matches user needs with multiple provider's services.
- Researchers can use one platform to manage their clouds.
- Orchestrates provisioning and allocation of cloud resources.
- Local copy of your cloud data is created, so jobs and VMs are traceable across clouds
- New clouds with similar configurations can be created easily.
- Default attributes allow easy control of cloud artefacts.
- Users can switch easily between clouds.
- Users can switch easily between HPC systems

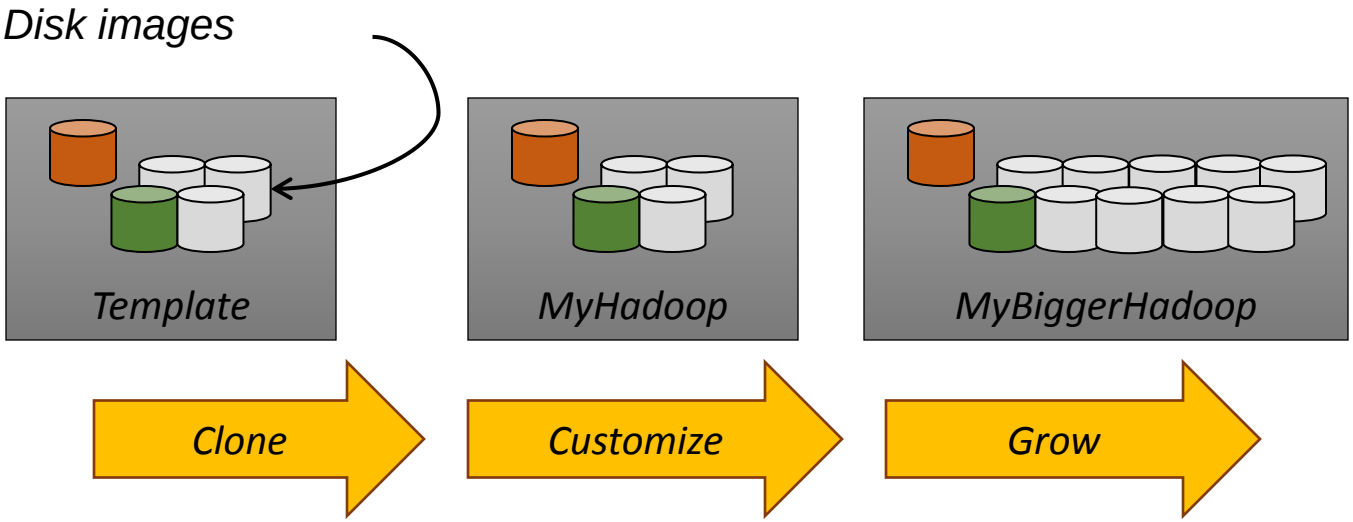
- cm default cloud=comet
- cm vm boot
- cm default cloud=chameleon
- cm vm boot

Future: Comet Cloudmesh Platform Launchers

- Customizable launchers
- Launchers available through commandline or browser

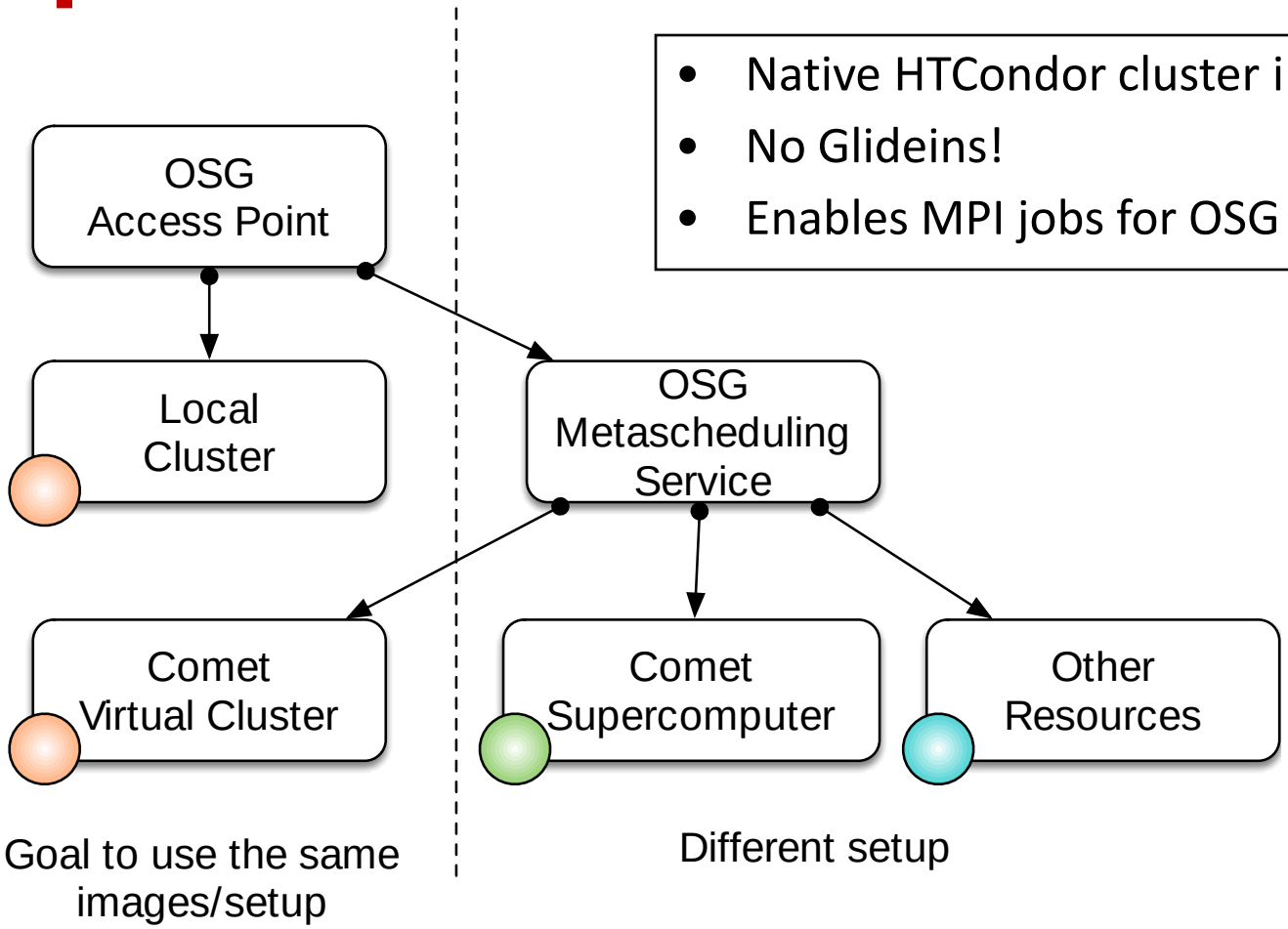
Example: Hadoop

$cm hadoop –n 10 –group=myHadoop



Early Success: Open Science Grid

- Native HTCondor cluster inside of Comet!
- No Glideins!
- Enables MPI jobs for OSG



New Ideas & Changes

Control & Responsibility

- Shared responsibility
  - SDSC: hosting environment
  - Cluster admin & PI: virtual machine stack
- Different from current HPC roles
- Considering an explicit “Acknowledgement of Responsibility” signed by PI & cluster admin
  - Based on SDSC’s “Outback Network” agreement
  - Outback is a separate VLAN & IP subnet for user-managed systems

Innovation in Utilization

- VMs co-scheduled within the batch system
- Works for automated workflows bringing up virtual compute nodes when needed
- What about responsiveness?
  - Can tune batch policies based on need
  - Reserve some physical nodes for fast launch to some scale
- Reuse existing allocations & accounting process
- Likewise, existing science gateway policies: e.g., community account for cluster

New Use Case: Training

- Training
  - Nearly full stack: OS, networking (IP & InfiniBand), applications
  - Any type of cluster: HPC; N-tier web framework; big data
  - Limits: no layer 1 (VLANs, etc.) networking

New Use Case: Campus Bursting

- Custom HPC clusters can help campuses extend a familiar environment
- How can a campus get a large compute allocation?
- Need to justify science and compute time
- Single PI proposals from large users?
- Historical campus cluster utilizations
- Don’t want to burn out Campus Champion allocations

And in Other News

- Check out Singularity: http://singularity.lbl.gov/
- User space containers for HPC
- Deployed on Comet, Gordon, and TSCC (campus cluster)
- Impromptu BoF Wednesday afternoon (location TBD)
- https://github.com/cjprybol/reproducibility-via-singularity

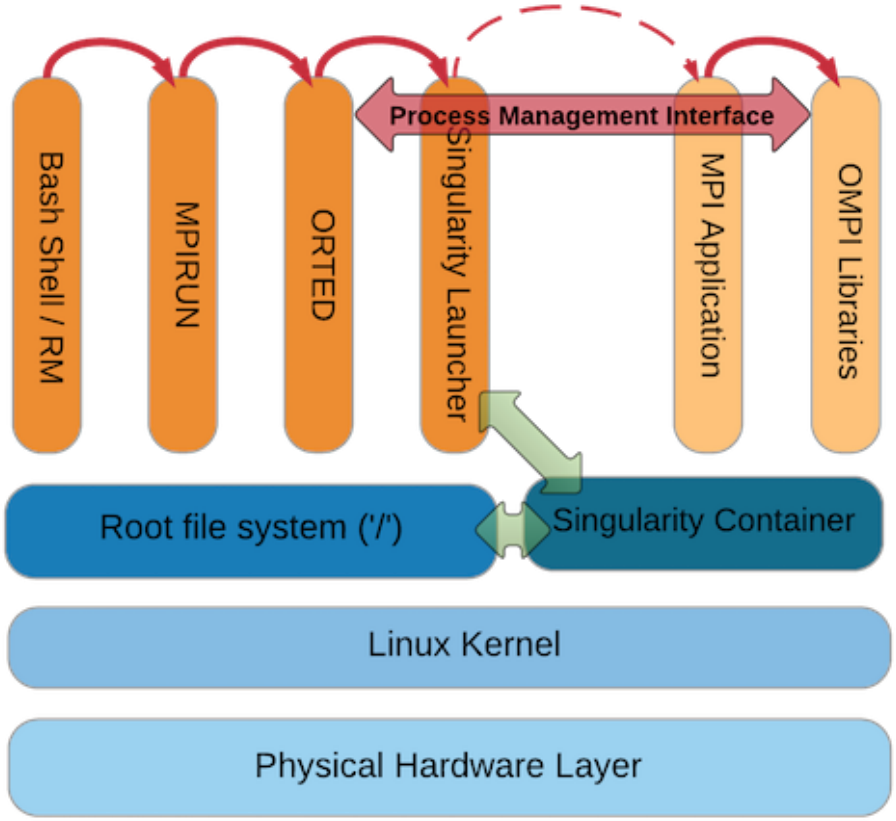
Make your research more reproducible with Singularity

Services like GitHub, Bitbucket, and GitLab have democratized access to affordable (or free!) tools that reinforce reproducibility in research, via the code repositories that these services offer. These services make it easier to backup, version control, collaborate on, and distribute code (or any other text-based file). These features make it easier for researchers to write and maintain high-quality code. These features also increase the chance that someone else will review the code for errors or bugs. These reviews can be done by some combination of reading and executing the code. Unfortunately, unless the code is run on the exact same computer, while logged in as the same user, getting the same code to run the same way can be a research project in and of itself.

Singularity

https://github.com/gmkurtzer/singularity

https://github.com/singularityware



Singularity & Open MPI