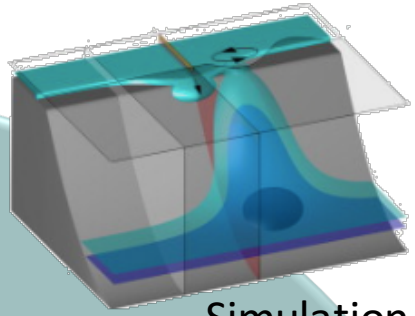

# Docker, VMs, and Cloud Architectures for HPC

*Kate Keahey*

*keahey@mcs.anl.gov*

# Why Cloud and HPC?

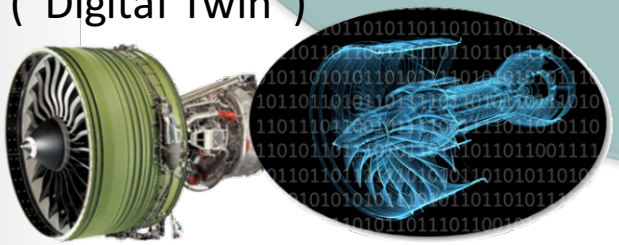
Rapid rise in experimental sciences



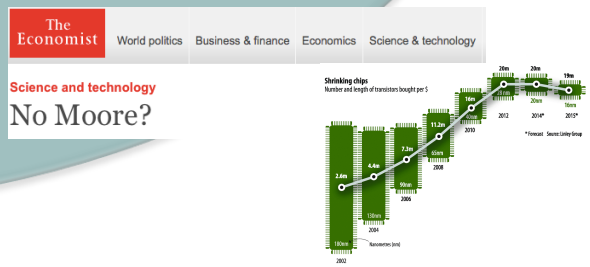
Simulation as experimental instrument

*Increased need for on-demand, reproducible, HPC computations*

Use of simulation in experiments ("Digital Twin")



End of Moore's Law



# Today's Topic: Appliances

- **Appliances**
  - Pros and Cons
  - Implementation and management
  - Virtual Machines and Containers side by side (qualitative and quantitative comparison)
- And if we have the time: on-demand availability
  - Provider and user concerns
  - Cloud and HPC models side by side
  - Combining cloud and HPC models

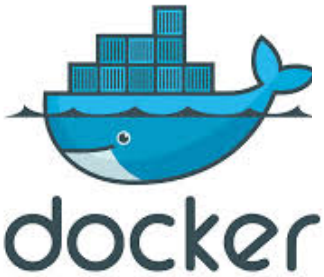
# Appliances as Abstraction

- Appliance = Application + Environment
  - Decouples resources and their configuration
- Benefits of using appliances
  - Control over environment and privilege level
  - Version management and reproducibility
  - Practical packaging, specialized installations
  - Live migration and sharing
  - Reconciling user requirements for many user groups
- Challenges of using appliances
  - Appliance configuration and management
  - Security implications
  - The need for speed: performance
- Appliance images, instances, and snapshots

# Appliance Implementations











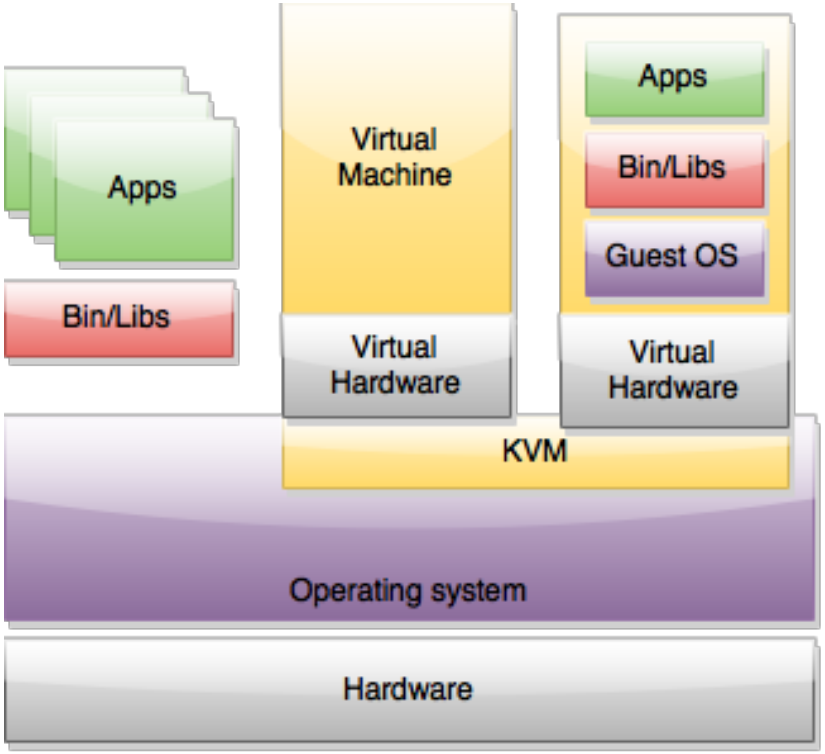
And more...

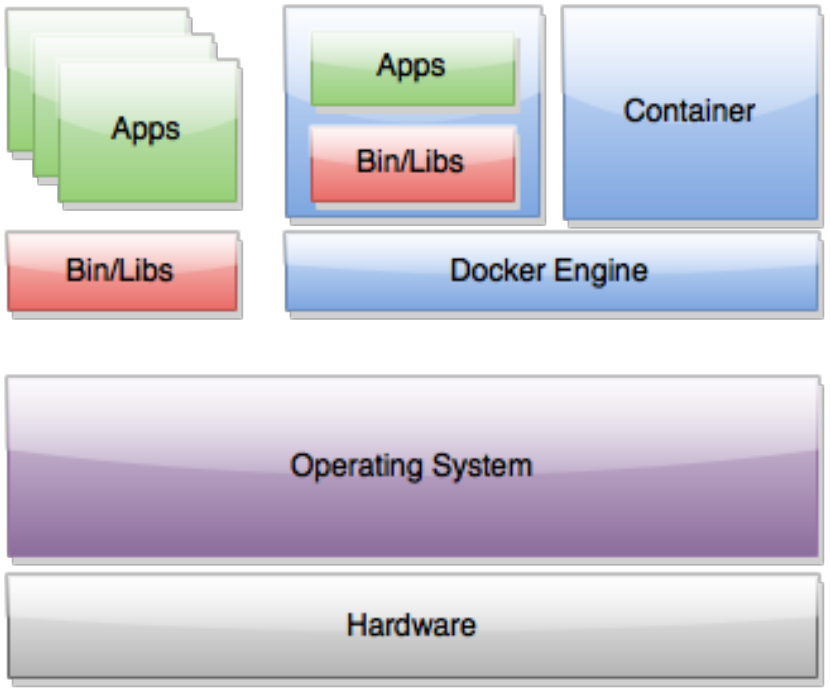
- Image type, e.g., VMs, containers, bare metal
- Provider, e.g., Chameleon, Magellan, Amazon
- Contextualization and “one-click” virtual clusters

# Appliance Management Process

- Sustainable image management
  - Automatically generate disk images for every supported platforms
  - Prevents getting “out of sync” images
- Disk image generation
  - Create a disk image offline and upload
    - Start from an existing disk image
    - Generate an image from scratch
    - OpenStack diskimage builder
  - Snapshot on cloud platform
    - Base image + automated deploy and configure (Packer) + snapshot

# KVM vs. Docker



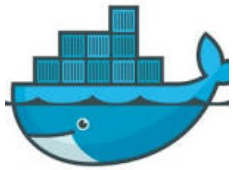

- Namespaces: pid, net, ipc, mnt, etc.
- Control groups: resource mngmt

# Feature Analysis

| Feature | KVM | Docker |
|---|---|---|
| **Guest OS** | Windows / Linux / Unix | Linux with same kernel |
| **Startup Time** | VMs take a few minutes to boot up | Containers take a few seconds to boot up |
| **Isolation and Security** | VMs are fully isolated. The attack surface is VMM | The attack surface is shared kernel |
| **Live migration support** | Yes | No (pre-alpha level support available) |
| **Integrated with OpenStack** | Yes | Yes |

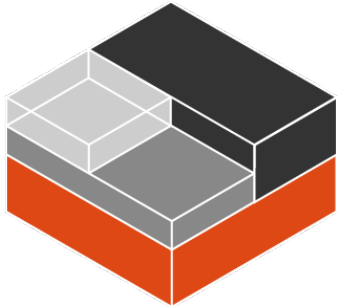
# Virtualization versus Containers

Palacios



Docker


LXD



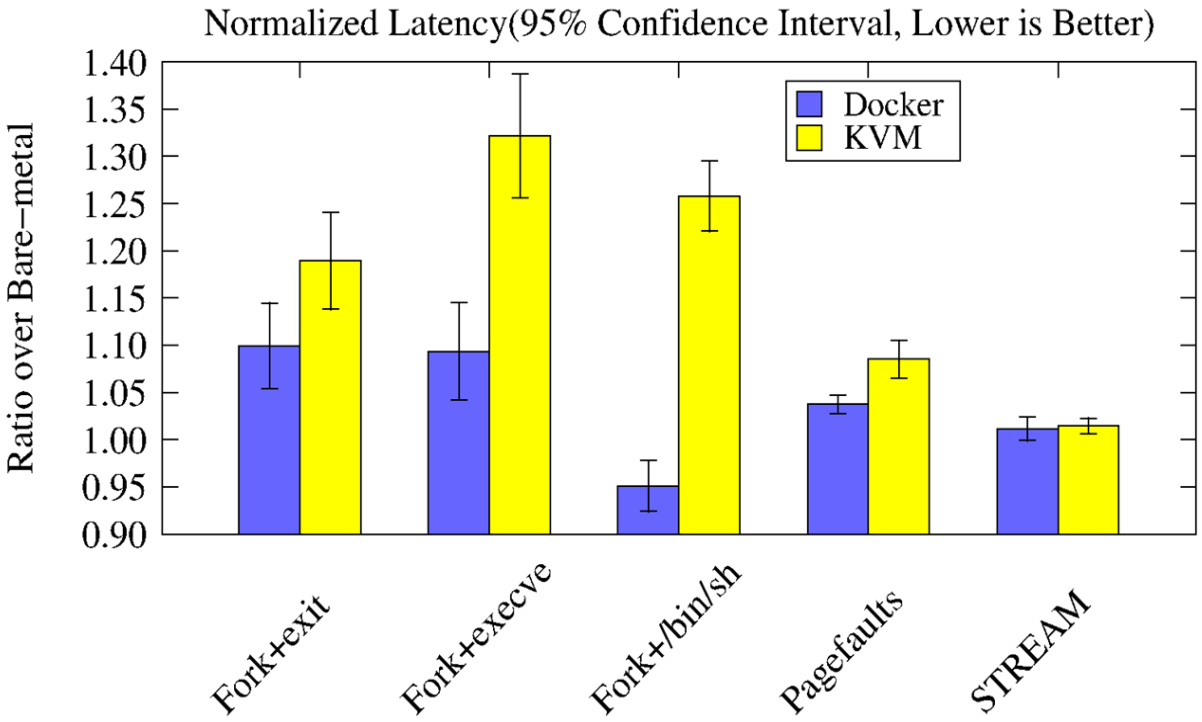
Bare Metal

# What's the Performance Like in Practice?

- Experiments comparing KVM and Docker
- Work by Yuyu Zhou and Balaji Subramaniam
- Chameleon experimental testbed
  - Total of ~600 nodes and 5 PB of storage in University of Chicago and TACC
  - Deeply reconfigurable: users can use bare metal, reboot, power on/off, console access, etc.
  - Supports use of dedicated/isolated resources
  - Is available to any U.S.-basec
  - **www.chameleoncloud.org**

# Microbenchmark: Lmbench Results

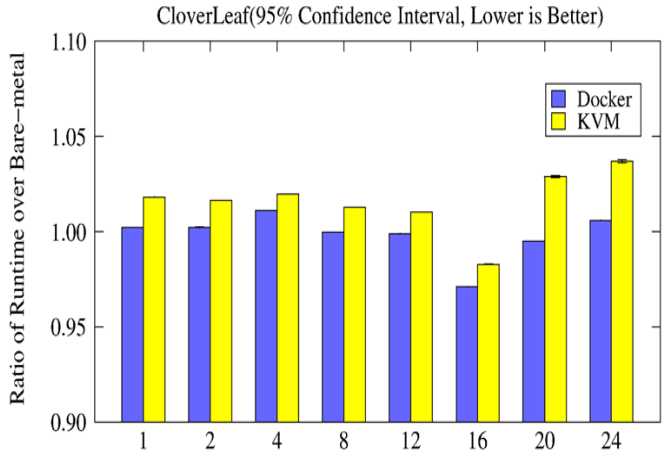
- Bandwidth tests (cached file read, memory copy, pipe)
- Latency tests (context switching, file creation and deletion, process creation, signal handling, system call overhead, memory read latency)

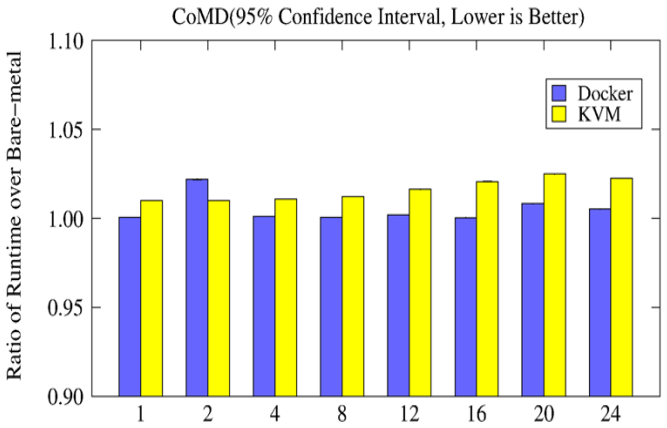



KVM has worse performance than Docker

# Scale-Up Results: Mantevo

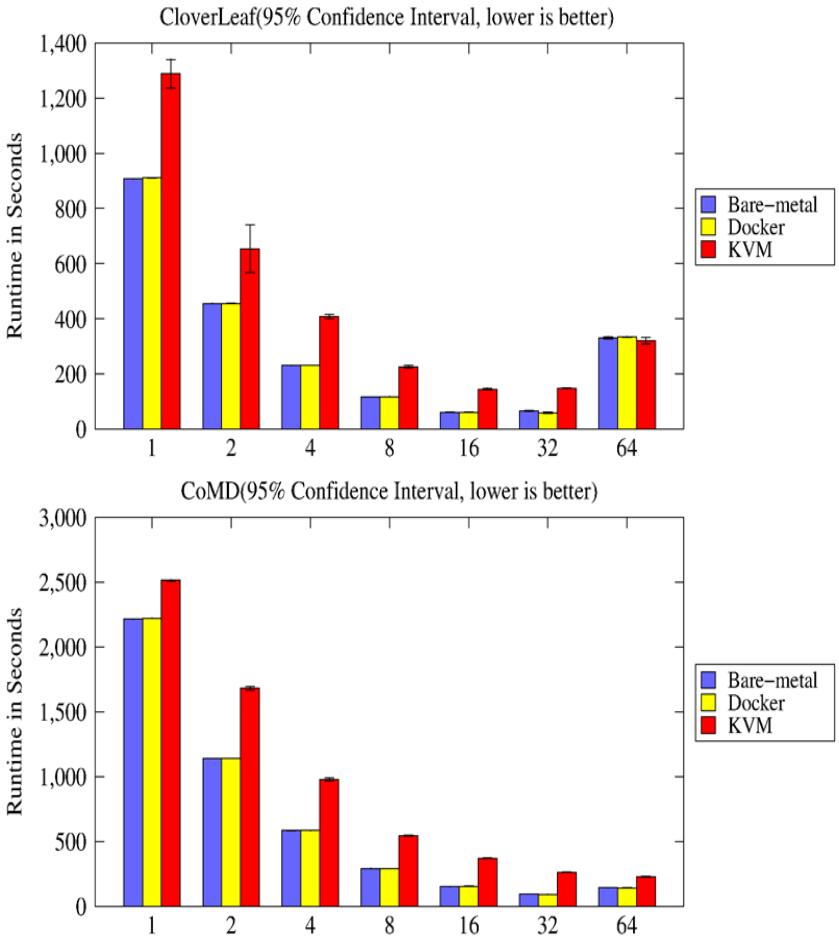
- Application Proxies
  - CloverLeaf (Hydrodynamics)
  - CoMD (molecular dynamics)
- Experimental Setup
  - Single node experiments
  - One KVM/Docker per node
  - Number of threads varies from 1 to 24



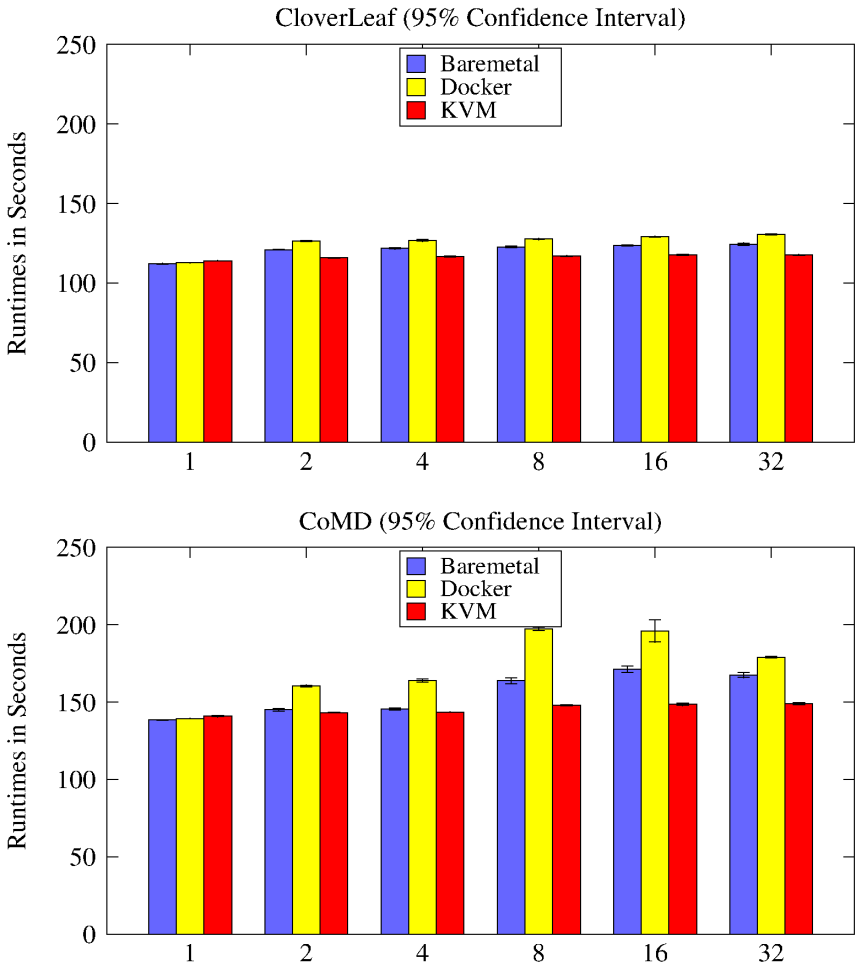
# Scale Out Results (1)

- Multi-node experiments
- One KVM VM or Docker container is run on a physical machine
- Used MPI benchmarks
- Up to 64 nodes were used



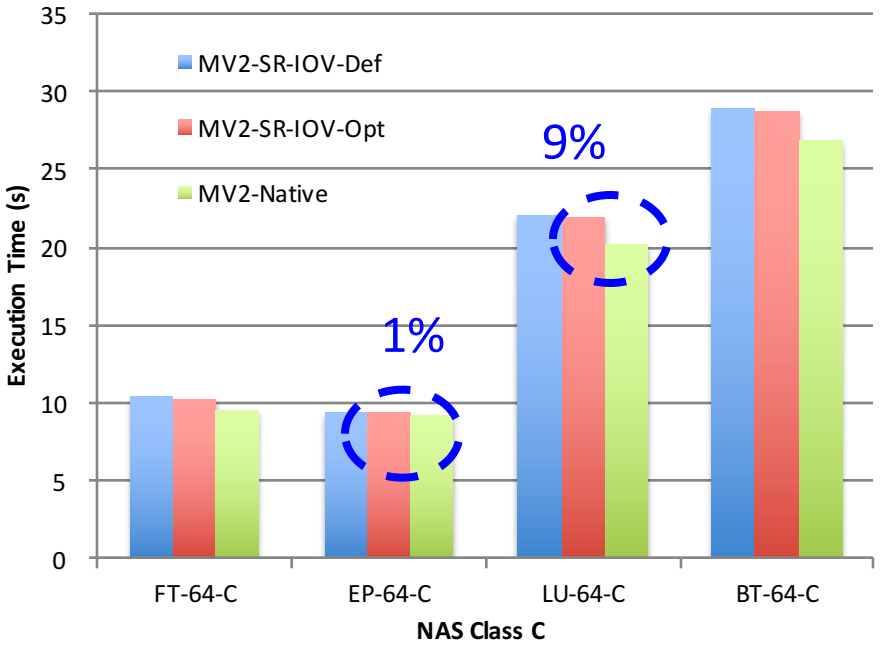
# Scale Out Results (2)

- Fixing KVM settings
  - PCI passthrough
  - Expose NUMA Topology to KVM
  - Pin VCPUs properly

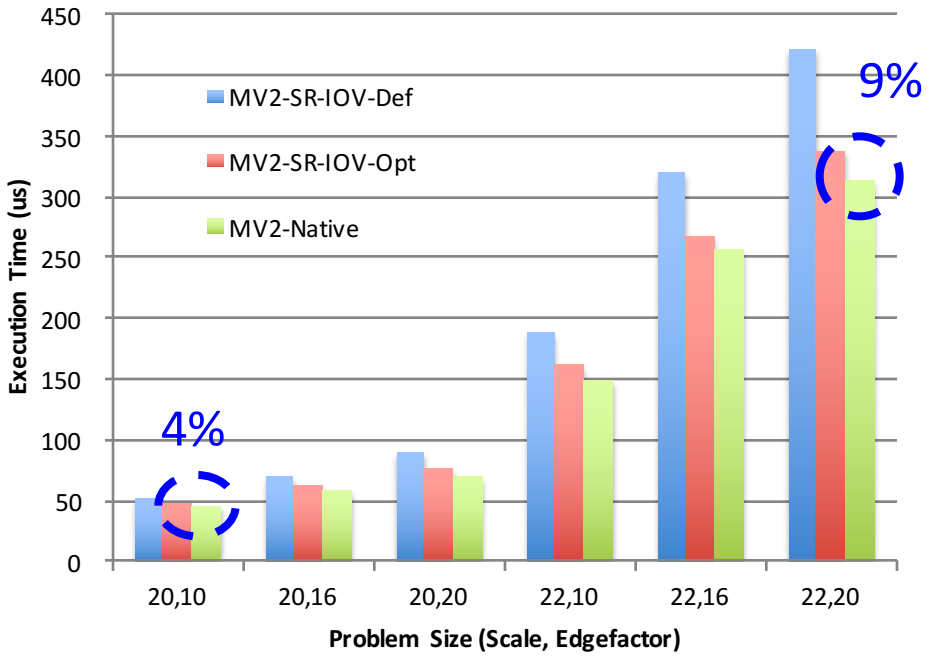



KVM was harder to set up correctly than Docker

# A Few Words about High-Performance Networks…




NAS




Graph500

- Compared to Native, 1-9% overhead for NAS
- Compared to Native, 4-9% overhead for Graph500
- Application –Level performance (8VM* 8Core/VM)

*Zhang et al., CCGrid'15*

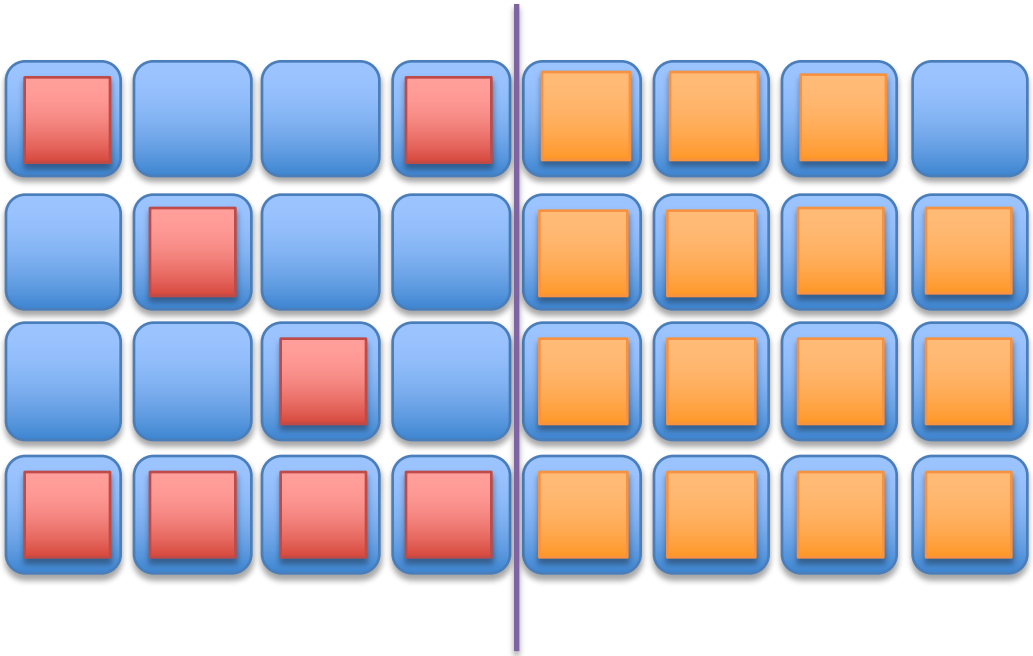
# Outline

- Appliances
  - Pros and Cons
  - Implementation and management
  - Virtual Machines and Containers side by side: qualitative and quantitative comparison
- **Apparently we do have time so: On-demand availability**
  - Provider and user concerns
  - Cloud and HPC models side by side
  - Combining cloud and HPC models

# Availability: HPC vs Cloud

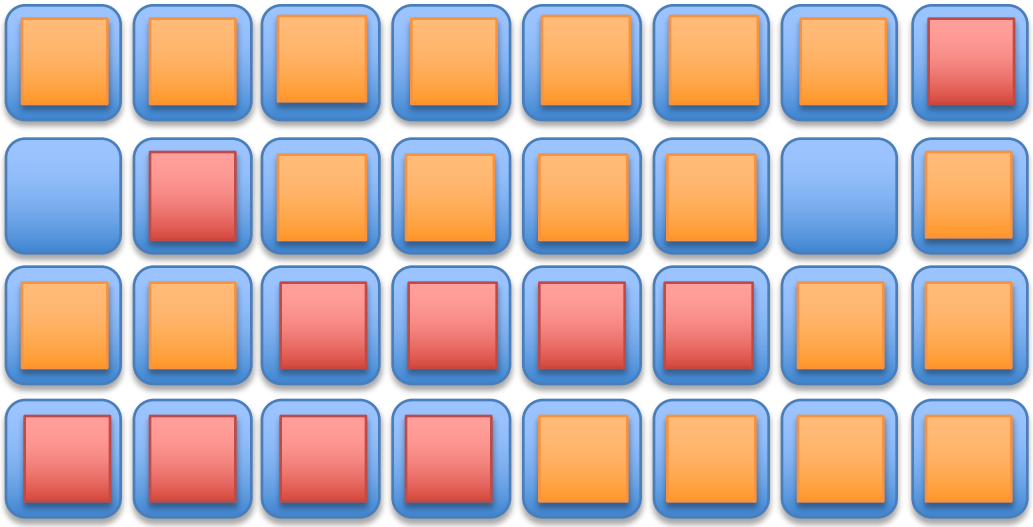
- HPC model:
  - Utilization is an important goal: expensive computational resources need to be amortized
  - **On-availability** scheduling: provider-centric, resources for a job are scheduled when there is availability
  - Optimizes provider concerns, users have no control over resource availability
- Infrastructure Cloud model:
  - **On-demand** availability supports interactive and time-sensitive computations
  - Implies keeping a proportion of resources available at, i.e., low utilization
  - Emphasizes user concerns over utilization concerns
  - Critical to many new data-driven applications

# On-Demand and HPC Resources



- Batch: Multiple HPC supercomputing centers
- On-demand: Magellan, Comet, JetStream, Bridges, Chameleon
- Some hybrid models: Shifter, high-priority models (urgent computing)

# HPC vs Cloud: Towards Dynamic Resource Sharing



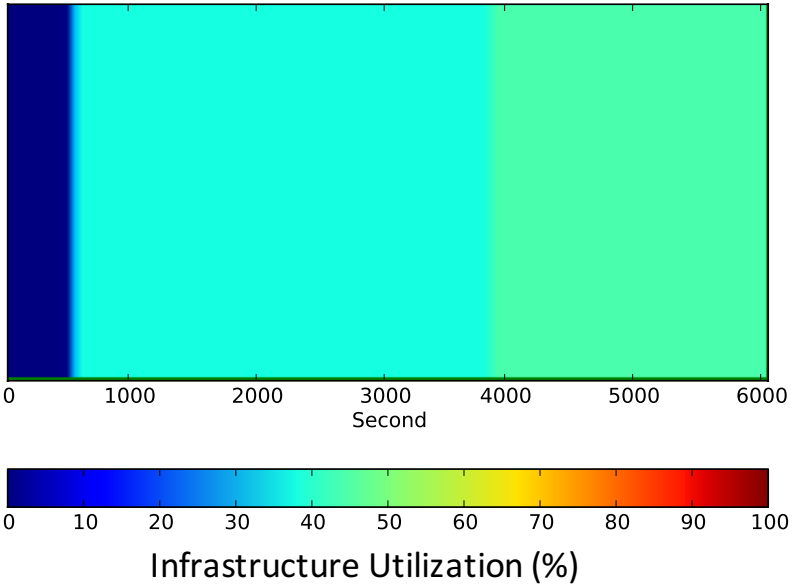
- Focus for now: availability management only
- Broader focus: sharing nodes, multi-dimensional resource match-making, etc.

# Proposal 1: OD and OA Convergence

- Approach: HTC "fills the gaps" around on-demand

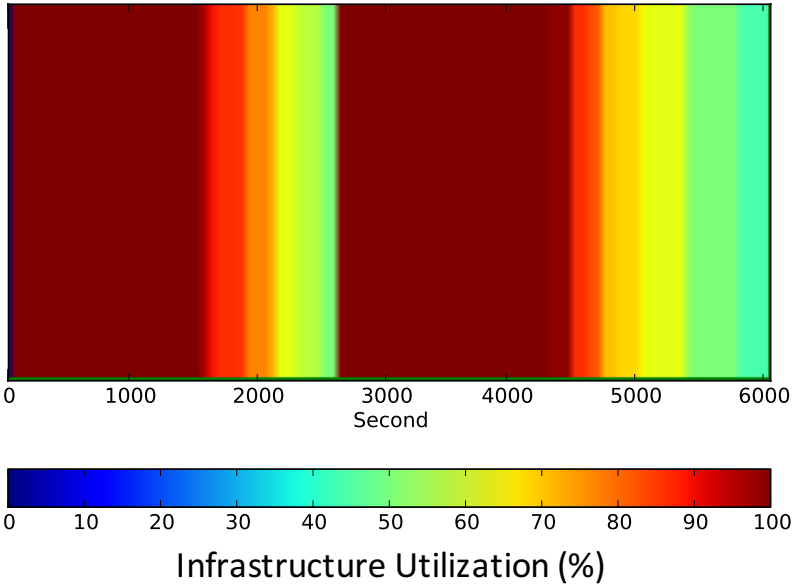
## On-Demand Only

Average utilization: 36.36%
Maximum utilization: 43.75%




## On-Demand and On-Availability

Average utilization: 83.82%
Maximum utilization: **100%**




*Paper: Marshall et al., I*mproving Utilization of Infrastructure Clouds, Marshall,*CCGrid'11*

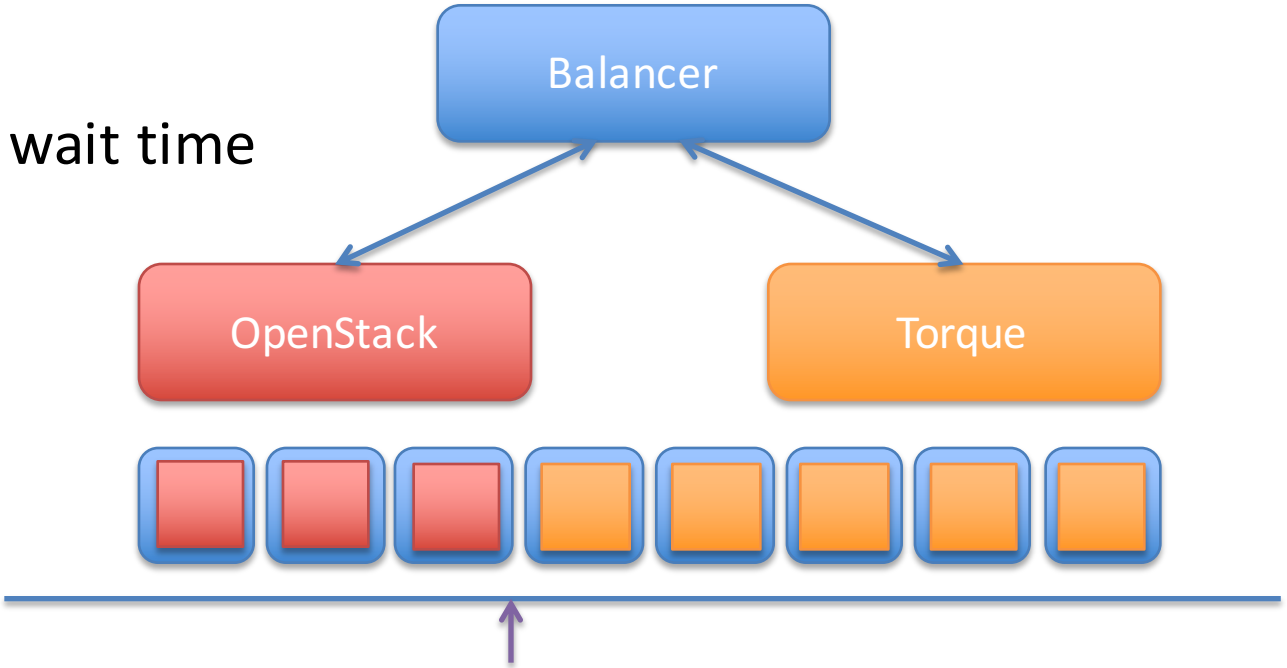
# Proposal 2: OD and OA Convergence

- Approach: Steal, don't kill! (reject requests instead)

Parameters:

w – on-demand wait time

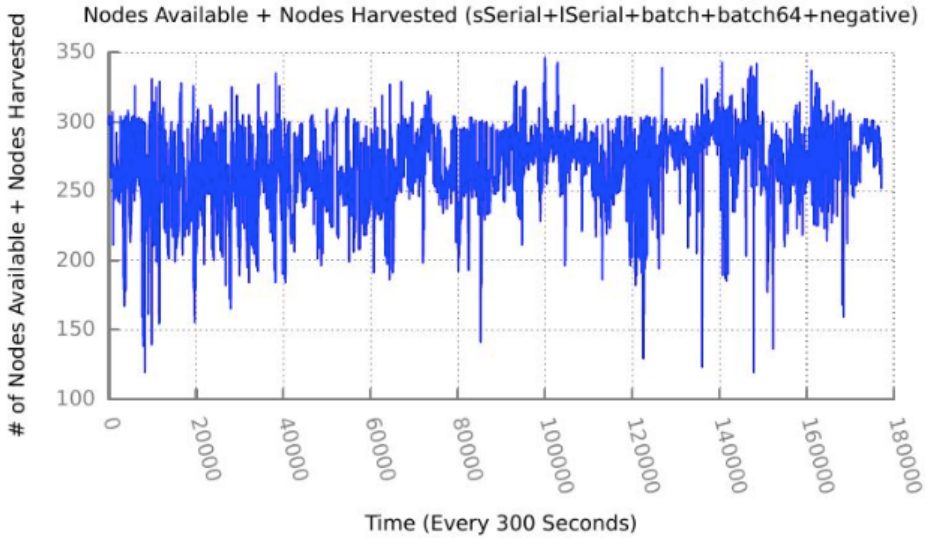
R -- reserve



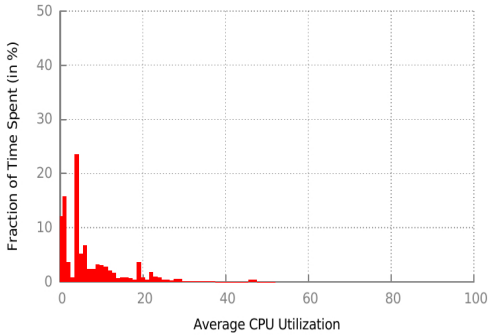
# Feasibility Analysis

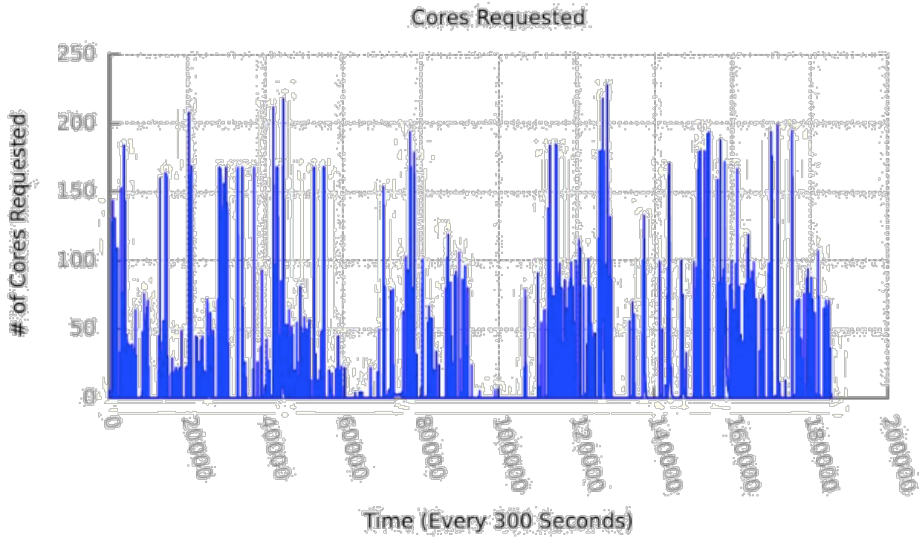
## Availability in LCRC

(01/01/14 -- 09/08/15)




## Demand in APS



# Preliminary Results

| | util% | | | Batch slowdown (lower is better) | Batch makespan (seconds) | On-demand rejection (%) |
|---|---|---|---|---|---|---|
| | batch | on-demand | overall | | | |
| 100% dynamic | 62.6% | 8.7% | 71.3% | 4.0 | 10092 | 12 (10.5%) |
| 75% batch/25% on-demand | 60.3% | 11.0% | 71.3% | 15.2 | 10489 | 45 (39.4%) |
| 50% batch/50% ondemand | 44.9% | 19.4% | 64.3% | 71.2 | 14405 | 11 (9.6%) |
| 25% batch/75% ondemand | 22.6% | 21.8% | 44.4% | 238.4 | 28151 | 1 (0.9%) |

Wait time = 30 seconds

# Summary

- Will cloud computing reach Top500?
  - In 2009 the answer was NO
  - In 2010 Amazon virtual cluster was #42 on Top500
- Appliances are only one aspect of cloud computing
  - There is also: on-demand availability, fine-grained resource management, data-focused frameworks, support for new patterns are others
- Appliances are a building block of cloud systems
  - They decouple resources and their configuration
- Appliance implementations
  - Performance is not so bad as once thought and getting better
  - Different performance, security, feature trade-offs

# Things to Try

- Try out Docker and KVM on the Chameleon testbed
  - www.chameleoncloud.org
  - One rack of Connectx3 IB
  - Bare metal appliances with KVM and Docker are provided
- Try other cloud technologies in HPC context
  - Appliances with Hadoop, OpenStack, etc.
- Using clouds: Chameleon, Jetstream, Bridges, Comet
- Share any research or tools you have developed with the community via Chameleon appliances