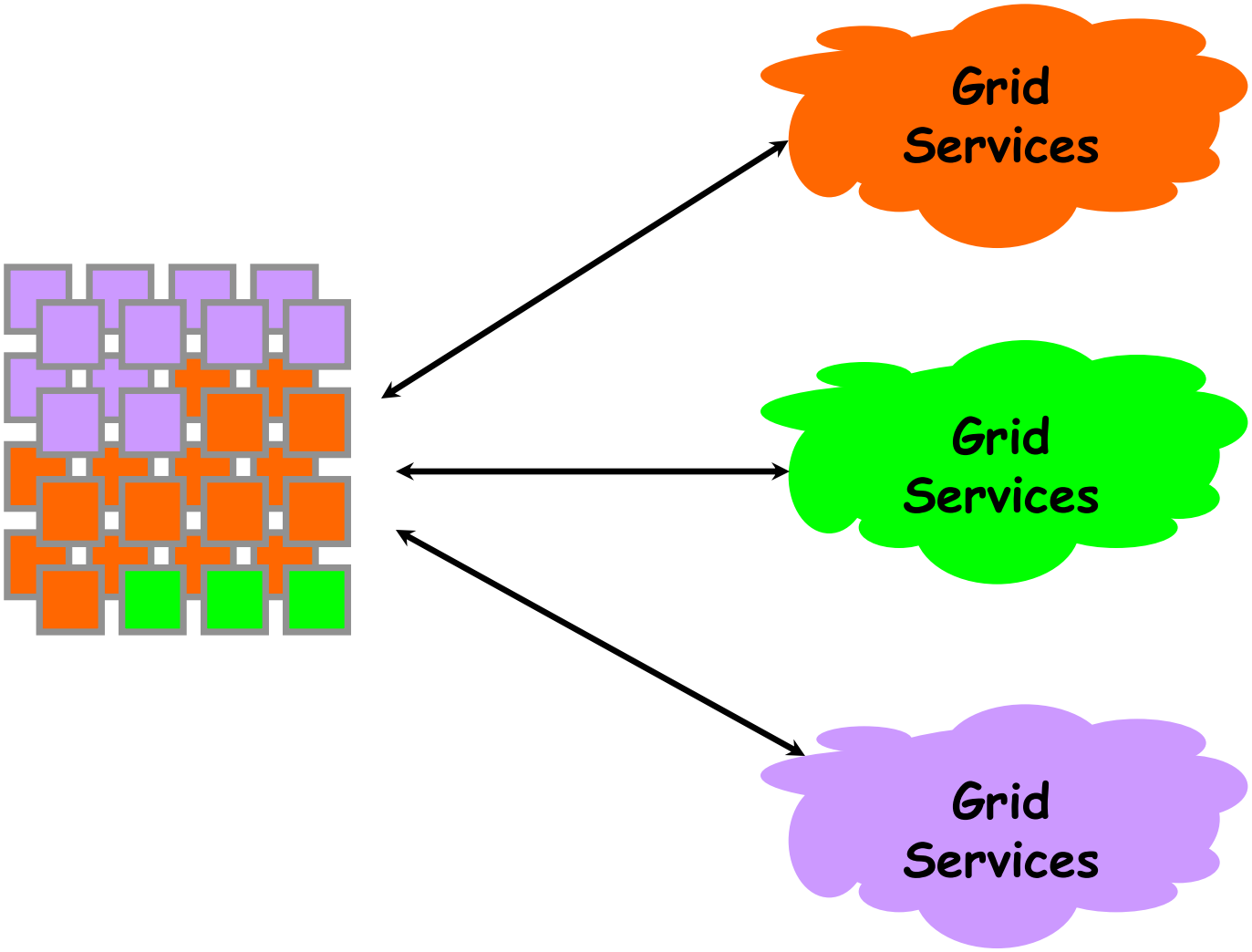
Dynamic Virtual Clusters in a Grid Site Manager

Jeff Chase, **David Irwin,** Laura Grit,

Justin Moore, Sara Sprenkle

Department of Computer Science
Duke University

Dynamic Virtual Clusters

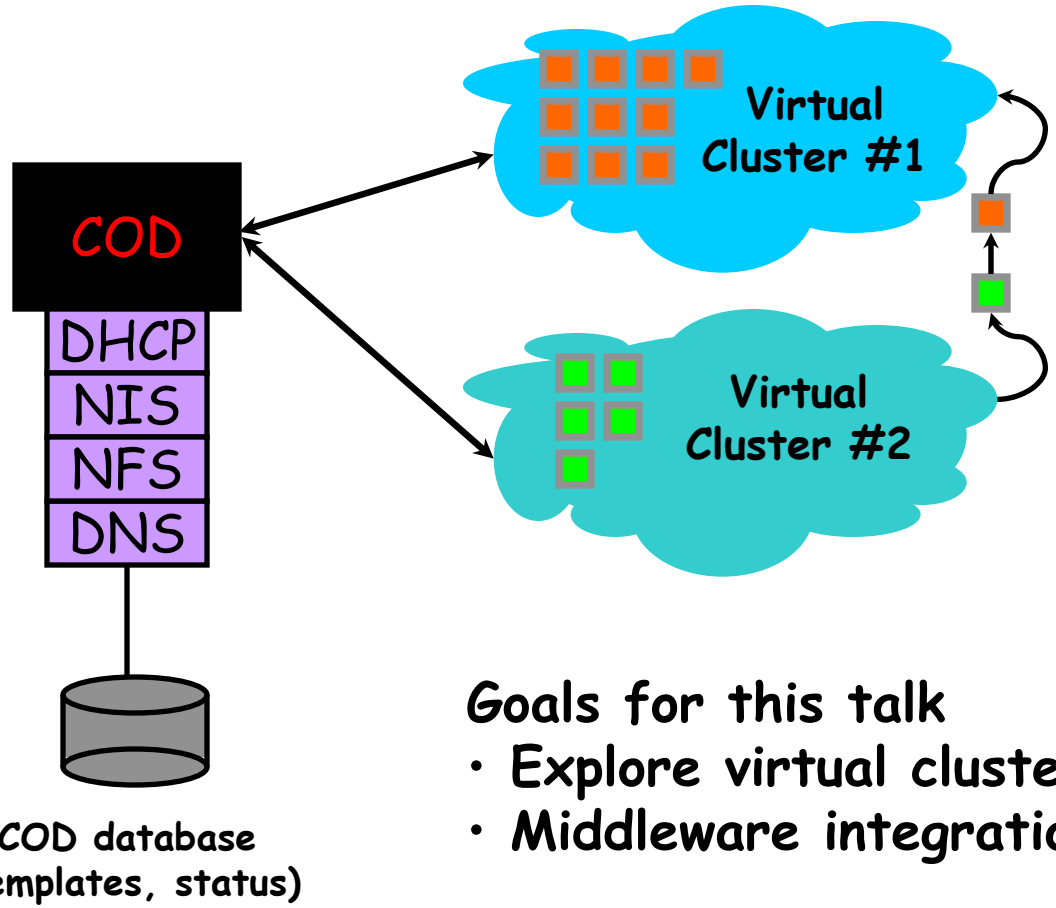


Motivation

Next Generation Grid

- Flexibility
  - Dynamic instantiation of software environments and services
- Predictability
  - Resource reservations for predictable application service quality
- Performance
  - Dynamic adaptation to changing load and system conditions
- Manageability
  - Data center automation

Cluster-On-Demand (COD)



Differences:
- OS (Windows, Linux)
- Attached File Systems
- Applications
- User accounts

Goals for this talk
- Explore virtual cluster provisioning
- Middleware integration (feasibility, impact)

Cluster-On-Demand and the Grid

Safe to donate resources to the grid

- Resource peering between companies or universities
- Isolation between local users and grid users
- Balance local vs. global use

Controlled provisioning for grid services

- Service workloads tend to vary with time
- Policies reflect priority or peering arrangements
- Resource reservations

Multiplex many Grid PoPs

- Avaki and Globus on the same physical cluster
- Multiple peering arrangements

Outline

Overview

- Motivation
- Cluster-On-Demand

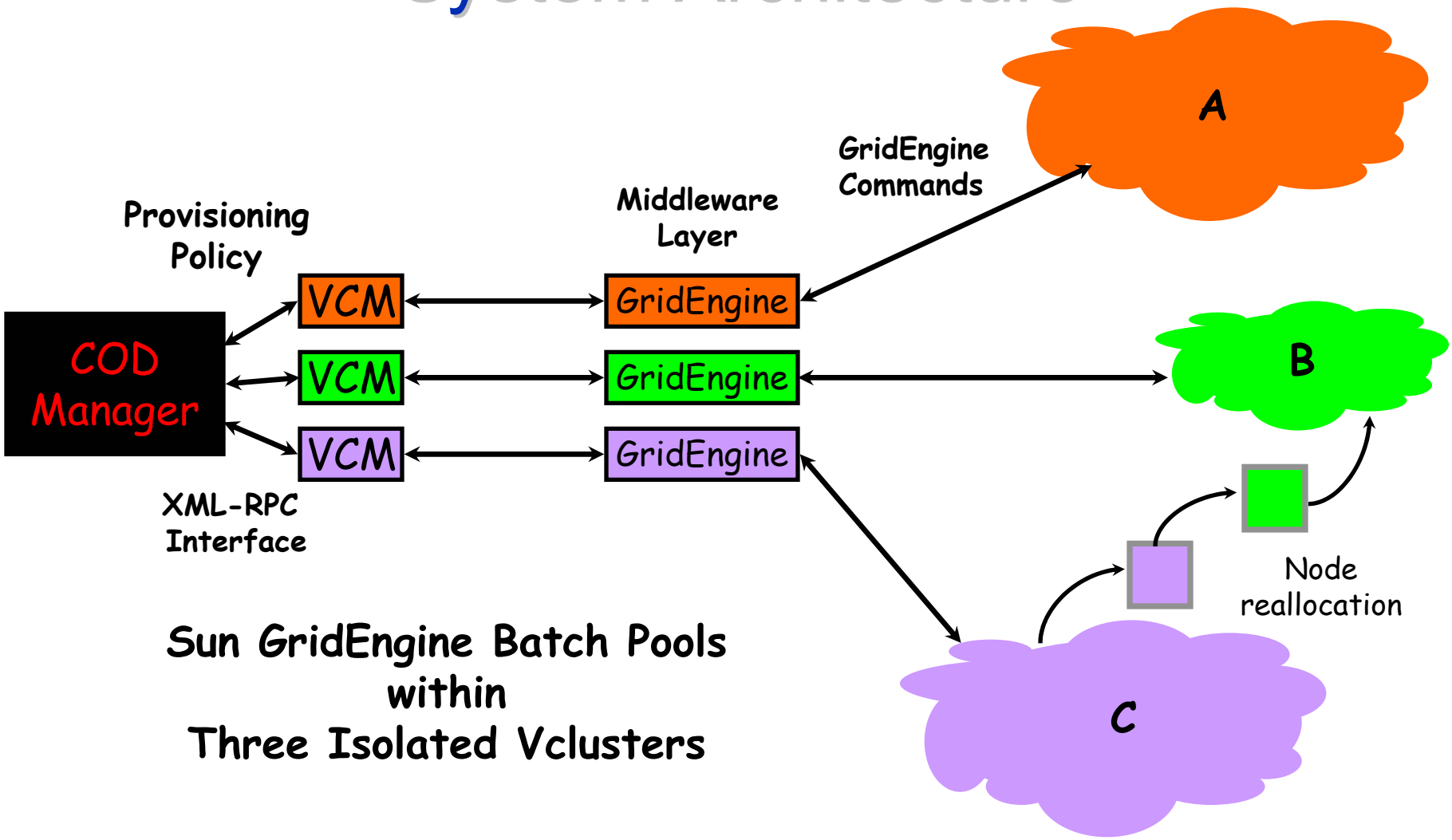
System Architecture

- Virtual Cluster Managers
- Example Grid Service: SGE
- Provisioning Policies

Experimental Results

Conclusion and Future Work

System Architecture



Sun GridEngine Batch Pools within Three Isolated Vclusters

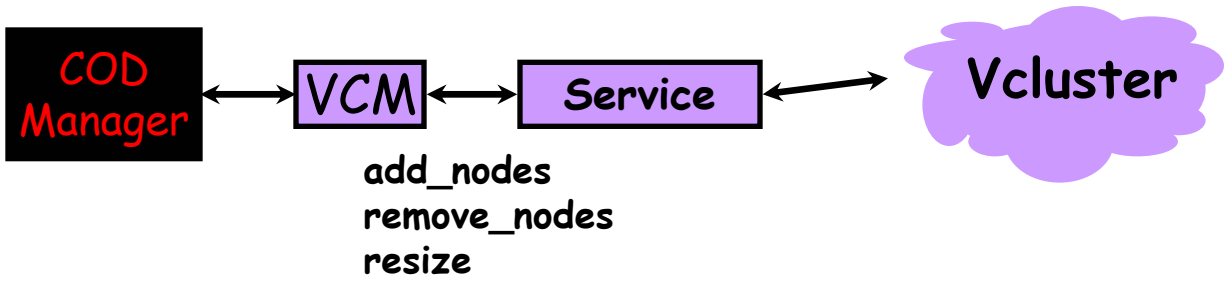
Virtual Cluster Manager (VCM)

Communicates with COD Manager

- Supports graceful resizing of vclusters

Simple extensions for well-structured grid services

- Support already present
  - Software handles membership changes
  - Node failures and incremental growth
- Application services can handle this gracefully



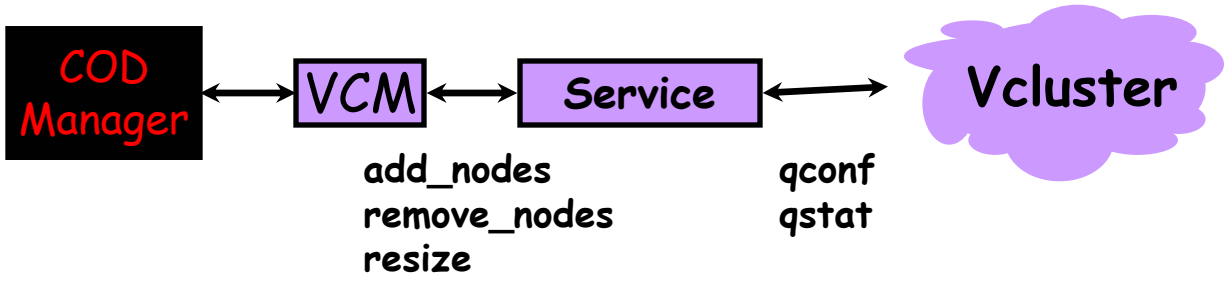
Sun GridEngine

Ran GridEngine middleware within vclusters

Wrote wrappers around GridEngine scheduler

Did not alter GridEngine

Most grid middleware can support modules



Pluggable Policies

Local Policy

- Request a node for every *x* jobs in the queue
- Relinquish a node after being idle for *y* minutes

Global Policies

- Simple Policy
  - Each vcluster has a priority
  - Higher priority vclusters can take nodes from lower priority vclusters
- Minimum Reservation Policy
  - Each vcluster guaranteed percentage of nodes upon request
  - Prevents starvation

Outline

Overview

- Motivation
- Cluster-On-Demand

System Architecture

- Virtual Cluster Managers
- Example Grid Service: SGE
- Provisioning Policies

Experimental Results

Conclusion and Future Work

Experimental Setup

Live Testbed

- Devil Cluster (IBM, NSF)
  - 71 node COD prototype
- Trace driven---sped up traces to execute in 12 hours
- Ran synthetic applications

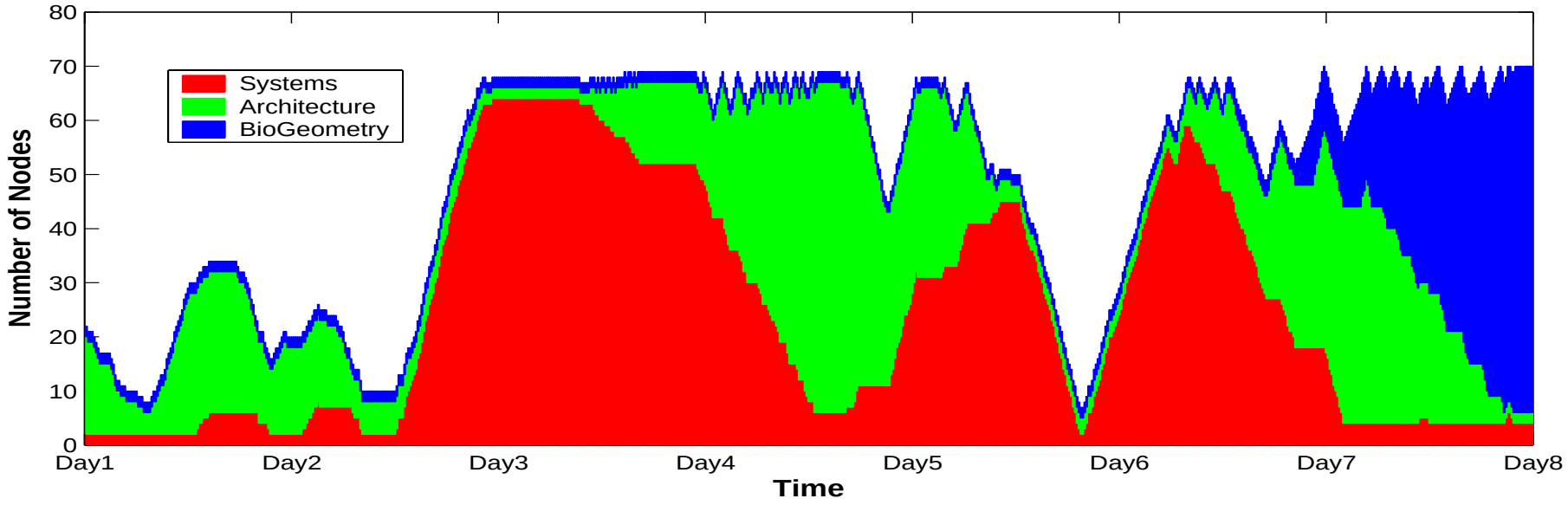
Emulated Testbed

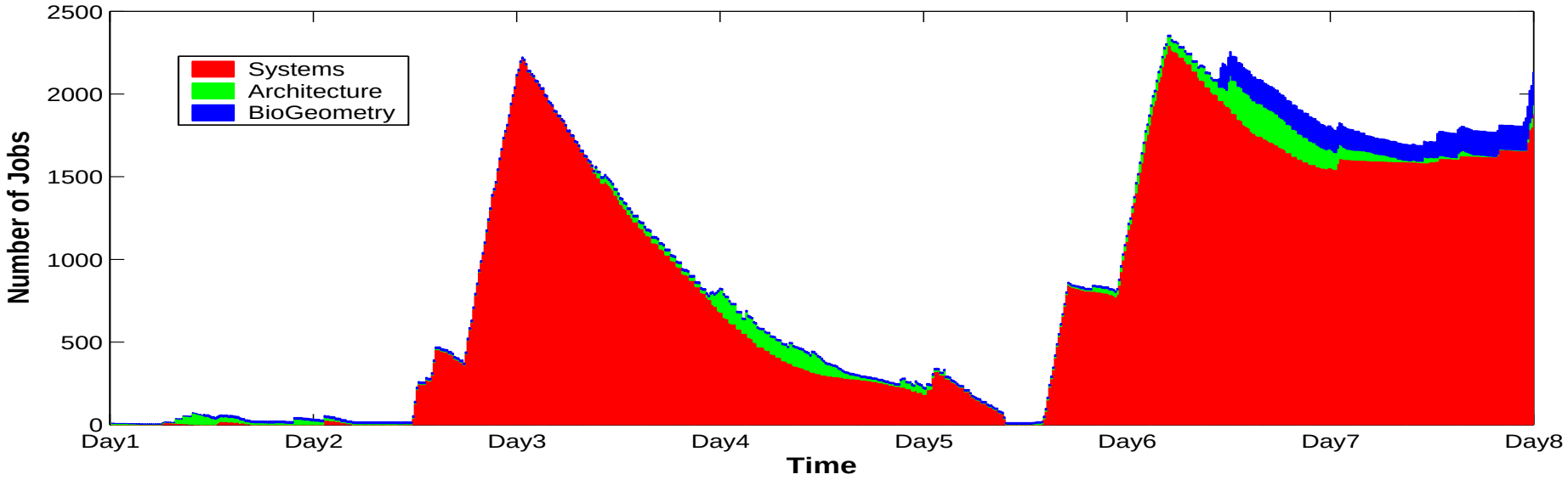
- Emulates the output of SGE commands
- Invisible to the VCM that is using SGE
- Trace driven
- Facilitates fast, large scale tests

Real batch traces

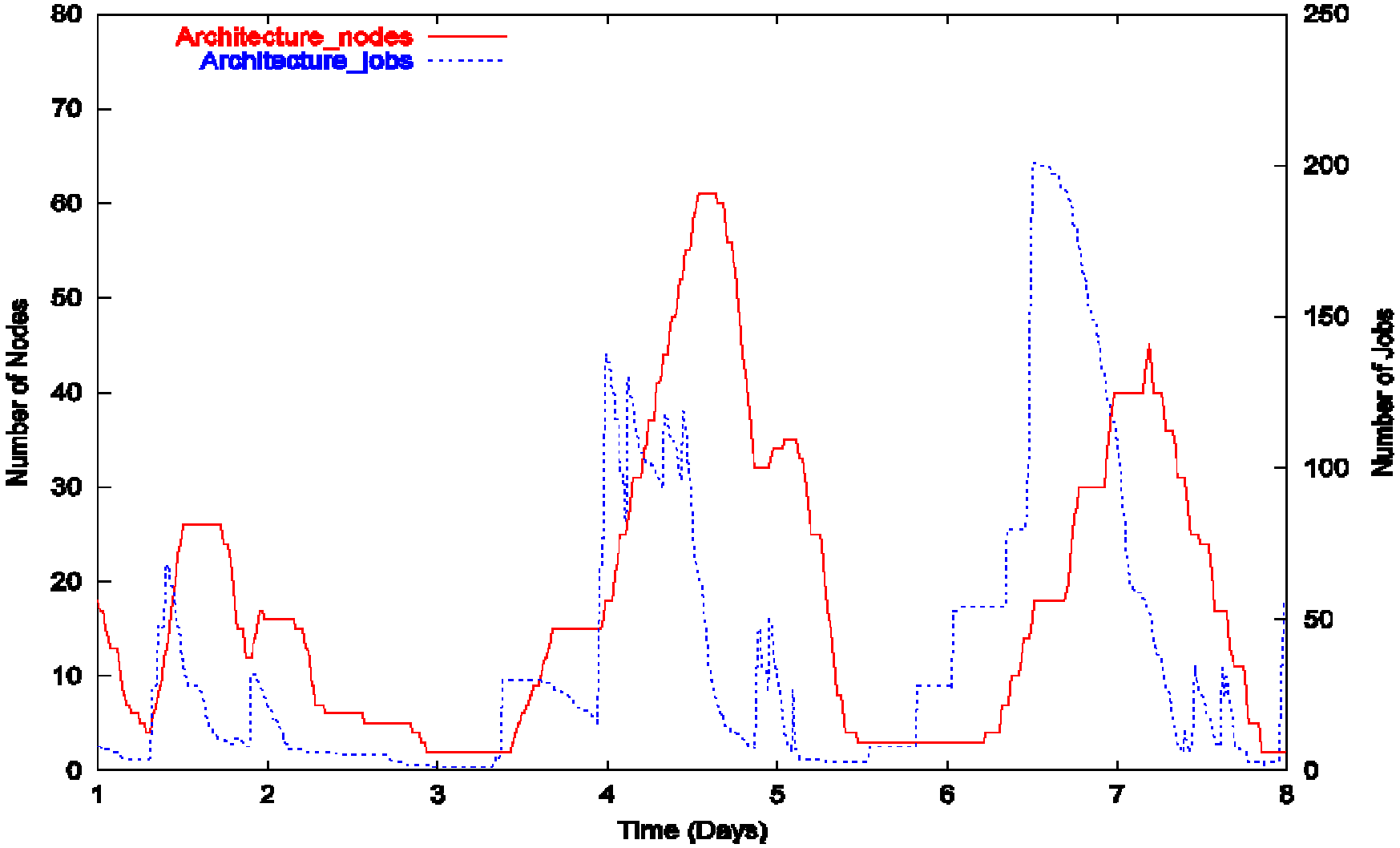
- Architecture, BioGeometry, and Systems groups

Live Test

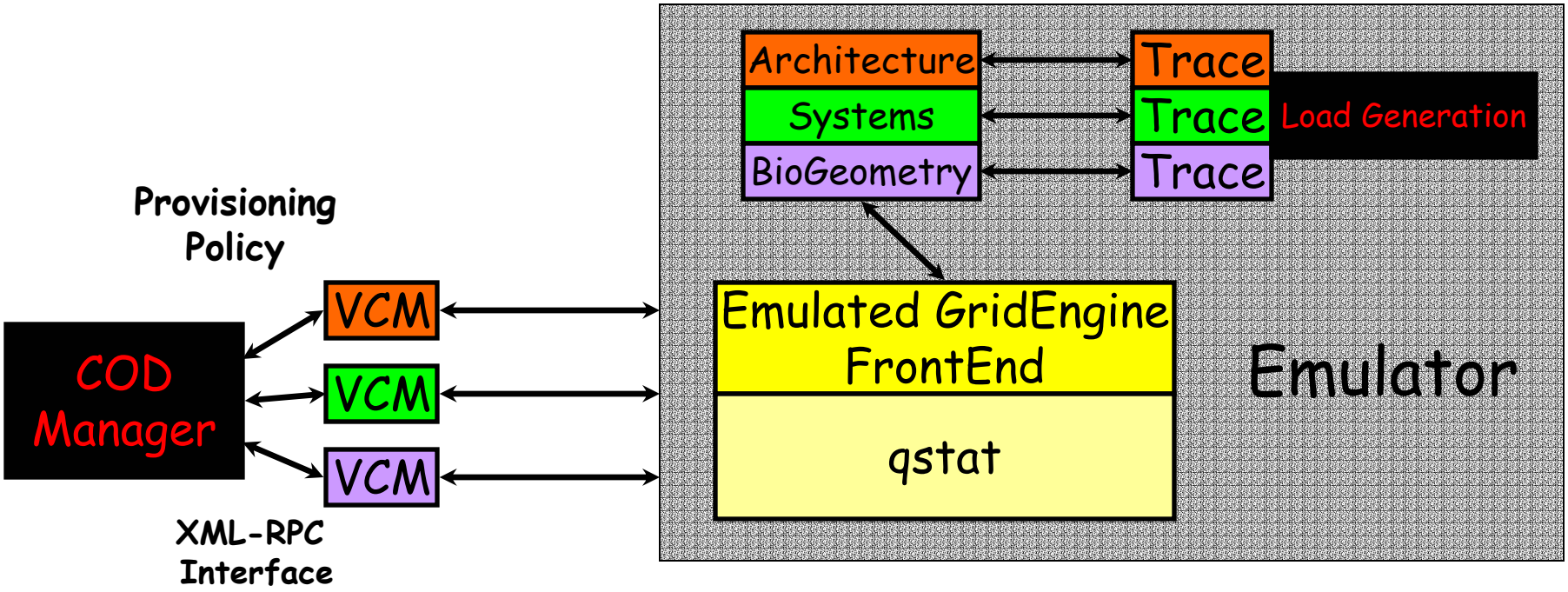

Number of Nodes
Systems
Architecture
BioGeometry
0
10
20
30
40
50
60
70
80
Day1
Day2
Day3
Day4
Day5
Day6
Day7
Day8
Time


Number of Jobs
Systems
Architecture
BioGeometry
0
500
1000
1500
2000
2500
Day1
Day2
Day3
Day4
Day5
Day6
Day7
Day8
Time

Architecture Vcluster



Emulation Architecture

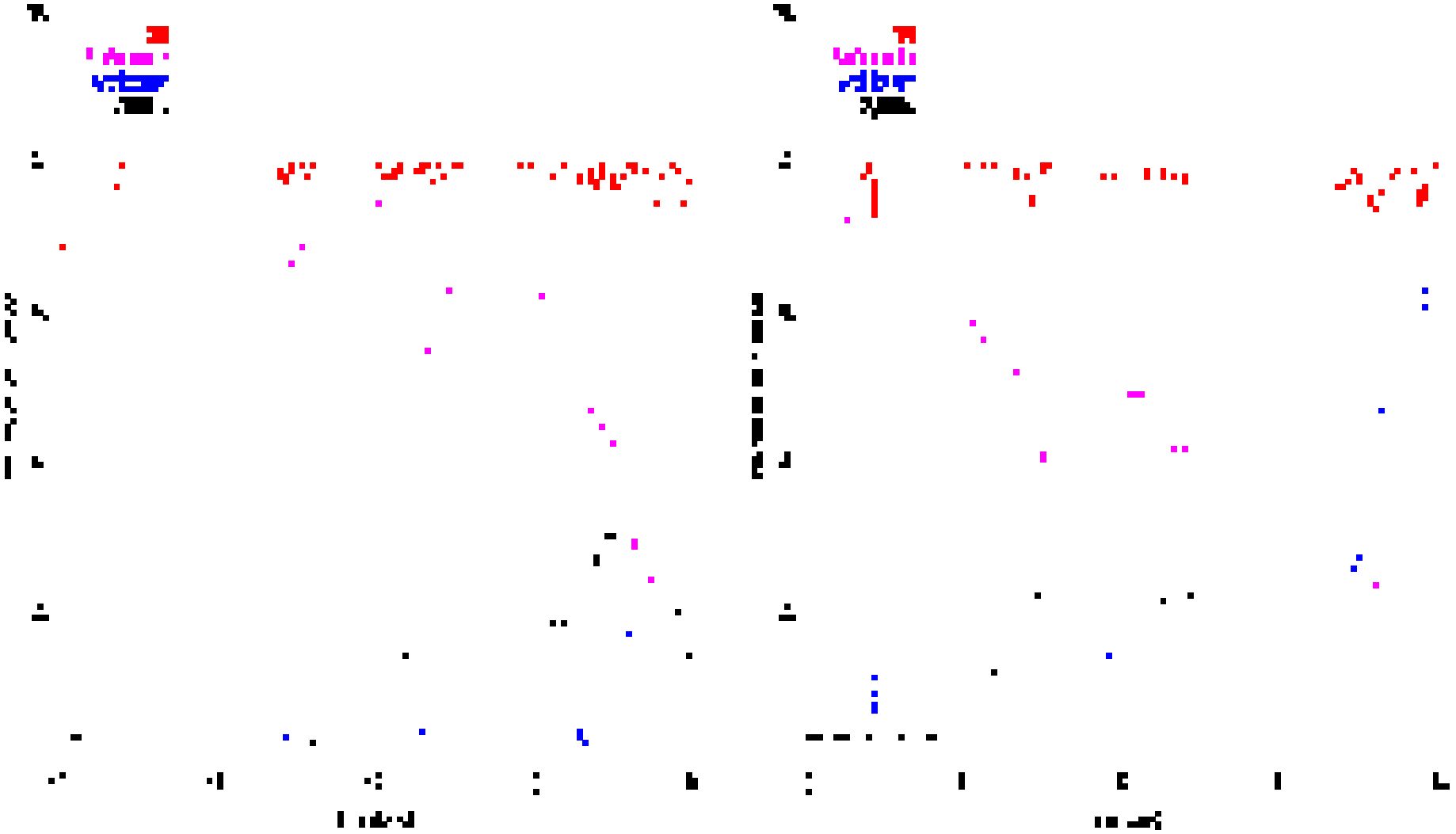


COD Manager and VCM are unmodified from real system

Each Epoch

1. Call resize module
2. Pushes emulation forward one epoch
3. qstat returns new state of cluster
4. add_node and remove_node alter emulator

Minimum Reservation Policy



Emulation Results

Minimum Reservation Policy

- Example policy change
- Removed starvation problem

Scalability

- Ran same experiment with 1000 nodes in 42 minutes making all node transitions that would have occurred in 33 days
- There were 3.7 node transitions per second resulting in approximately 37 database accesses per second.
- Database scalable to large clusters

Related Work

Cluster Management

- NOW, Beowulf, Millennium, Rocks
- Homogenous software environment for specific applications

Automated Server Management

- IBM's Oceano and Emulab
- Target specific applications (Web services, Network Emulation)

Grid

- COD can support GARA for reservations
- SNAP combines SLAs of resource components

  COD controls resources directly

Future Work

Experiment with other middleware

Economic-based policy for batch jobs

Distributed market economy using vclusters

- Maximize profit based on utility of applications
- Trade resources between Web Services, Grid Services, batch schedulers, etc.

Conclusion

No change to GridEngine middleware

Important for Grid services

- Isolates grid resources from local resources
- Enables policy-based resource provisioning

  Policies are pluggable

Prototype system

- Sun GridEngine as middleware

Emulated system

- Enables fast, large-scale tests
- Test policy and scalability

Example Epoch