Grid Computing – Technology, Trends & Attributes

Mitesh Agarwal

IT Architect – HPC Solutions

Sun Microsystems

mitesh.agarwal@sun.com

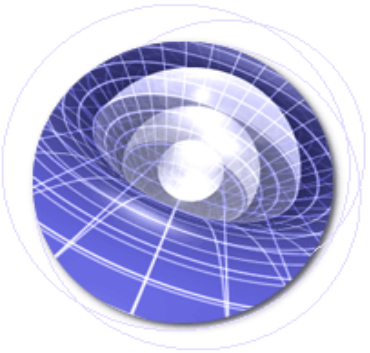
http://sun.com/grid

Sun microsystems

We make the net work.

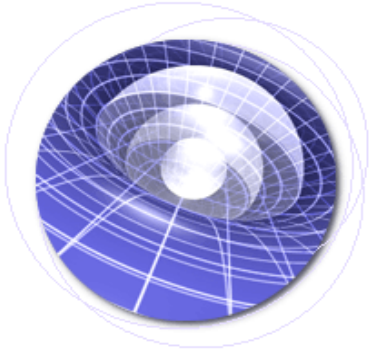
Agenda

- What is a Grid?
- How Does it Work?
- Underpinning Technologies
- Types of Grids
- Trends
- Attributes
- Discussions, Q&A



What Is Grid Computing?

- A hardware and software infrastructure that connects distributed computers, storage devices, databases and software applications through a network, and is managed by distributed resource management software



- A way of using many resources to perform many kinds of tasks, accessible from many places by many people

- A universal computing infrastructure that builds on the power of the Net and enables more efficient **computation**, **collaboration**, and **communication**

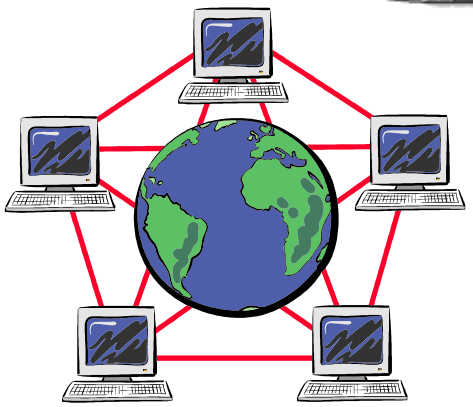
Grid Computing Explained

Problem-solving through resource pooling in virtual systems:

Virtualization of…	Resources into a dynamic, single compute resource from federated assets
Transparent scalability of…	CPU cycles, storage
Access that is...	Dependable, consistent, pervasive, inexpens

Why use Grid Computing?

- Reduce Costs
- Increase Productivity
- Increase Quality
- Shorten Time to Market
- Transparent Growth
- Do things you couldn't do before

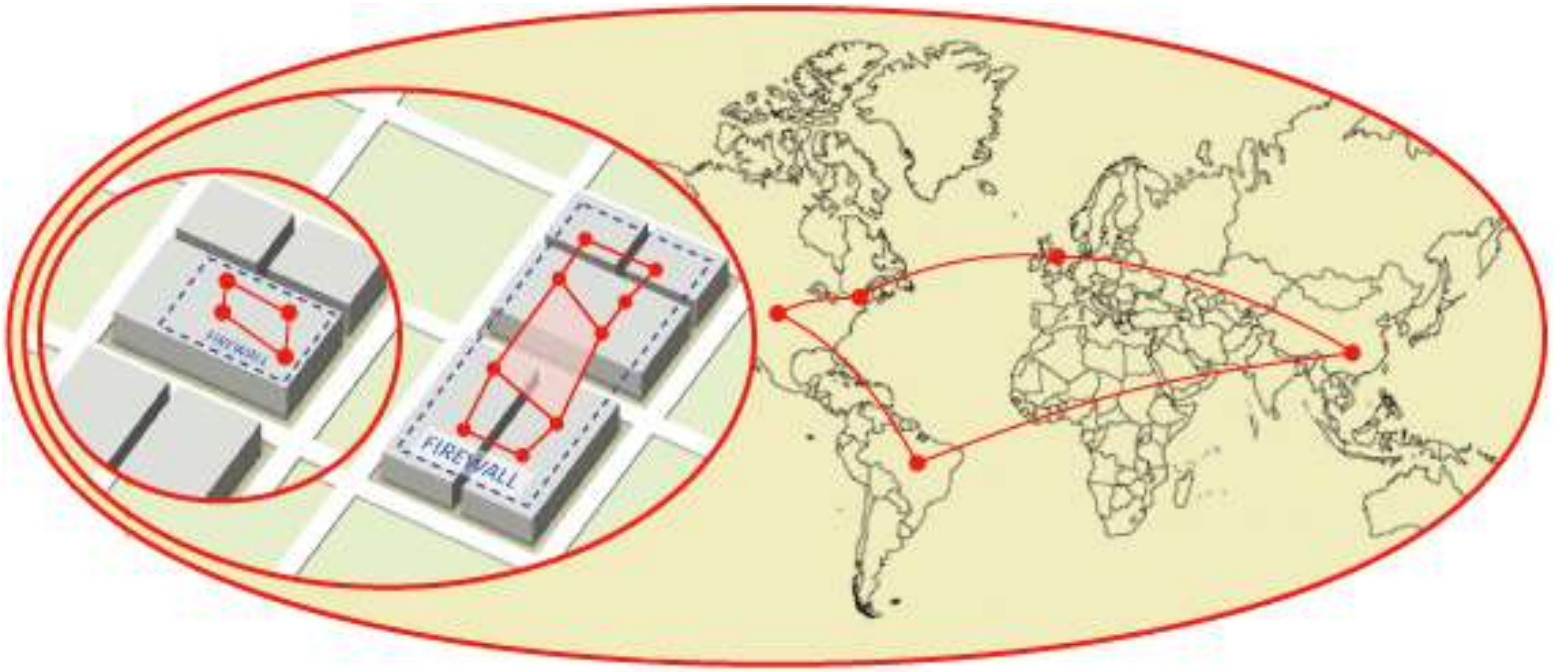


The Strategic Need for Compute Power

Across Diverse Industries

Industry	Computationally Intensive activities
Life Sciences	Genetic sequencing, database queries
Electronic Design	Simulations, verifications, regression testing
Financial Services	Risk and portfolio analysis, simulations
Automotive Manufacturing	Crash testing simulations, stress testing, aerodynamics modeling
Scientific Research	Large computational problems, collaboration
Oil and Gas Exploration	Simulations, seismic analysis, visualization
Digital Content Creation	Frame rendering

Levels of Grid Computing



Department Grids
- Single user community
- Single organization

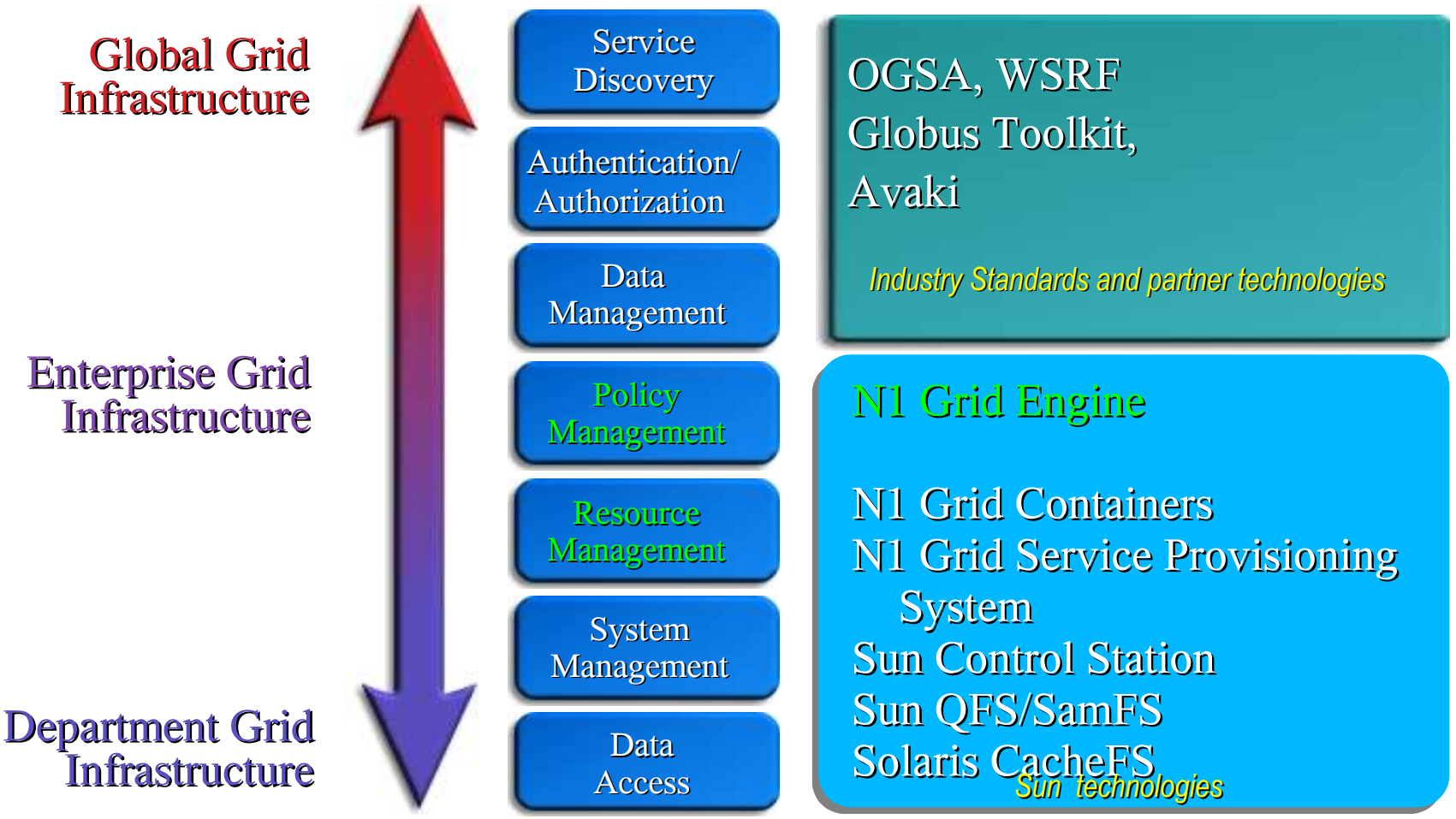
Enterprise Grids
- Multiple user communities
- Single organization

Global Grids
- Multiple user communities
- Multiple organizations

Academia & Research
Business

Grid Enabling Technologies

Across all levels of deployment



Grid Engine: Sun's Key Grid Technology

- Software to manage jobs & resources on distributed systems
- Well-established software
  - Initially developed in 1994, acquired by Sun in 2000
  - Core technology powers over 8000 Grids worldwide
- N1 Grid Engine 6 :
  - Enterprise-scale components on top of open-source core technology
  - Fully supported by Sun on: Solaris, Linux, Mac OS X, HP-UX, AIX, Irix (Windows: end of 2004)

Examples of Grids

Cycle-stealing: use resources when unoccupied

- "desktop grid"
- datacenter sharing

Compute Farms: racks of dedicated nodes

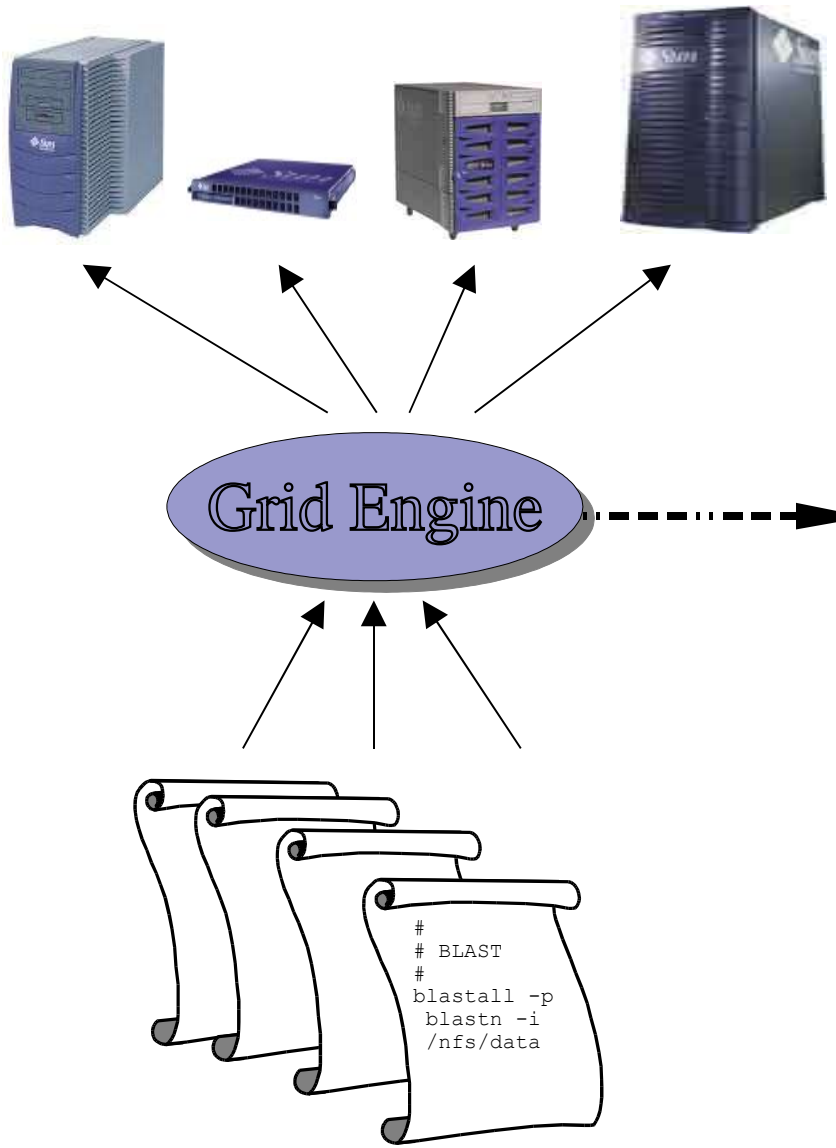
- High-throughput
- "Beowulf"

HPC Clusters: clumps of large SMP systems

- Massively-parallel jobs, eg, weather, astro

Or any combination of these

N1 Grid Engine Overview



Resource Management

Selection of Jobs

Job-based policies for resource ROI, SLA, QoS, etc

User-based policies for sharing, ranking, by departments, projects, user groups, etc

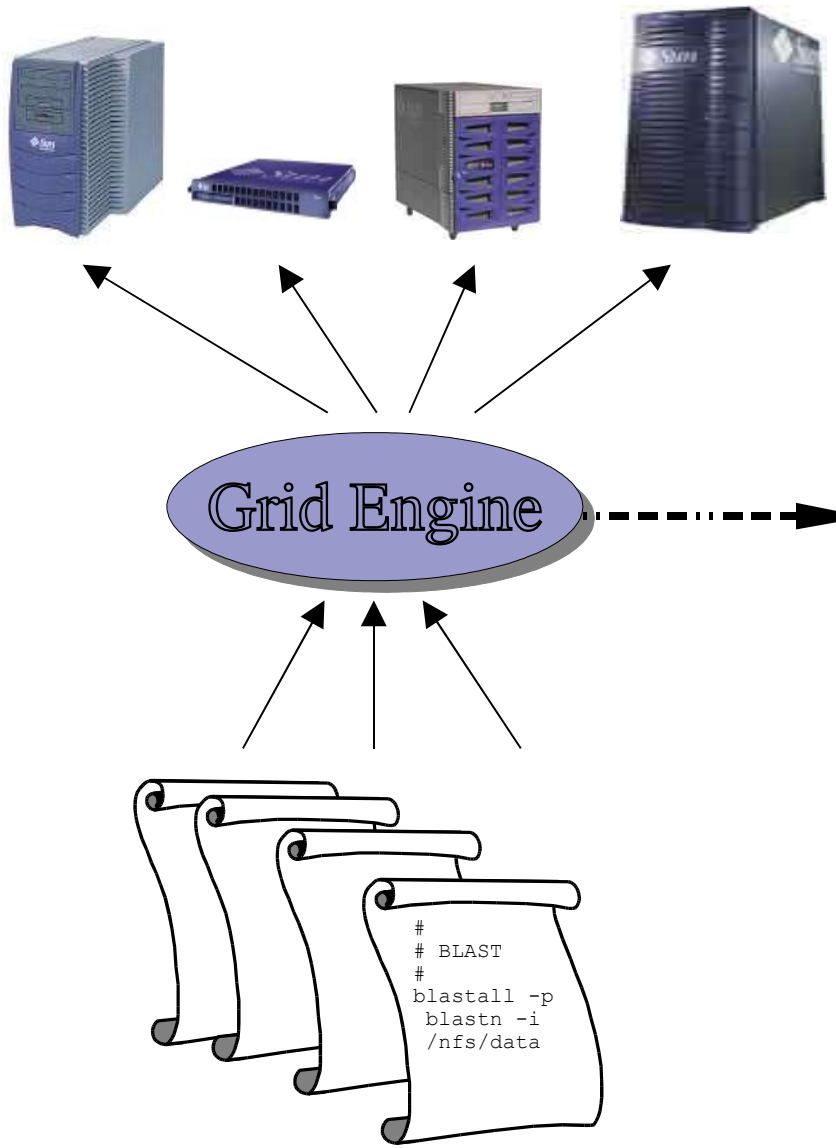
Selection of Resources

System characteristics: CPU, memory, OS, patches, etc.

Status of systems: avail. mem, load, free disk space, etc.

Status of other resources: licenses, shared storage, other software, etc.

N1 Grid Engine Overview



Resource Control

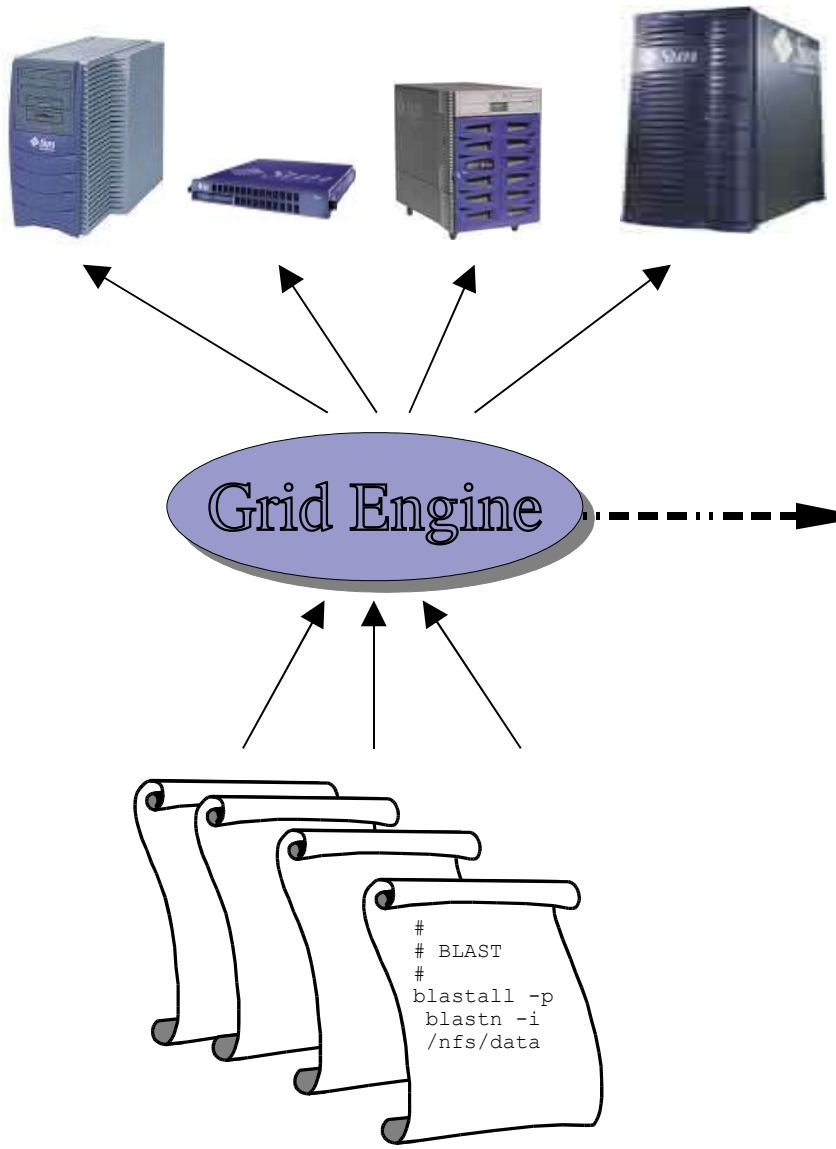
Control of jobs

- Customizable action methods for Suspend, Resume, Kill, Migrate, Restart
- Manual or automated via triggers

Control of resources

- Regulate load on systems based upon resource value thresholds
- Control access to systems via permissions, time/date, jobtype

N1 Grid Engine Overview



Resource Accounting

Accounting of jobs

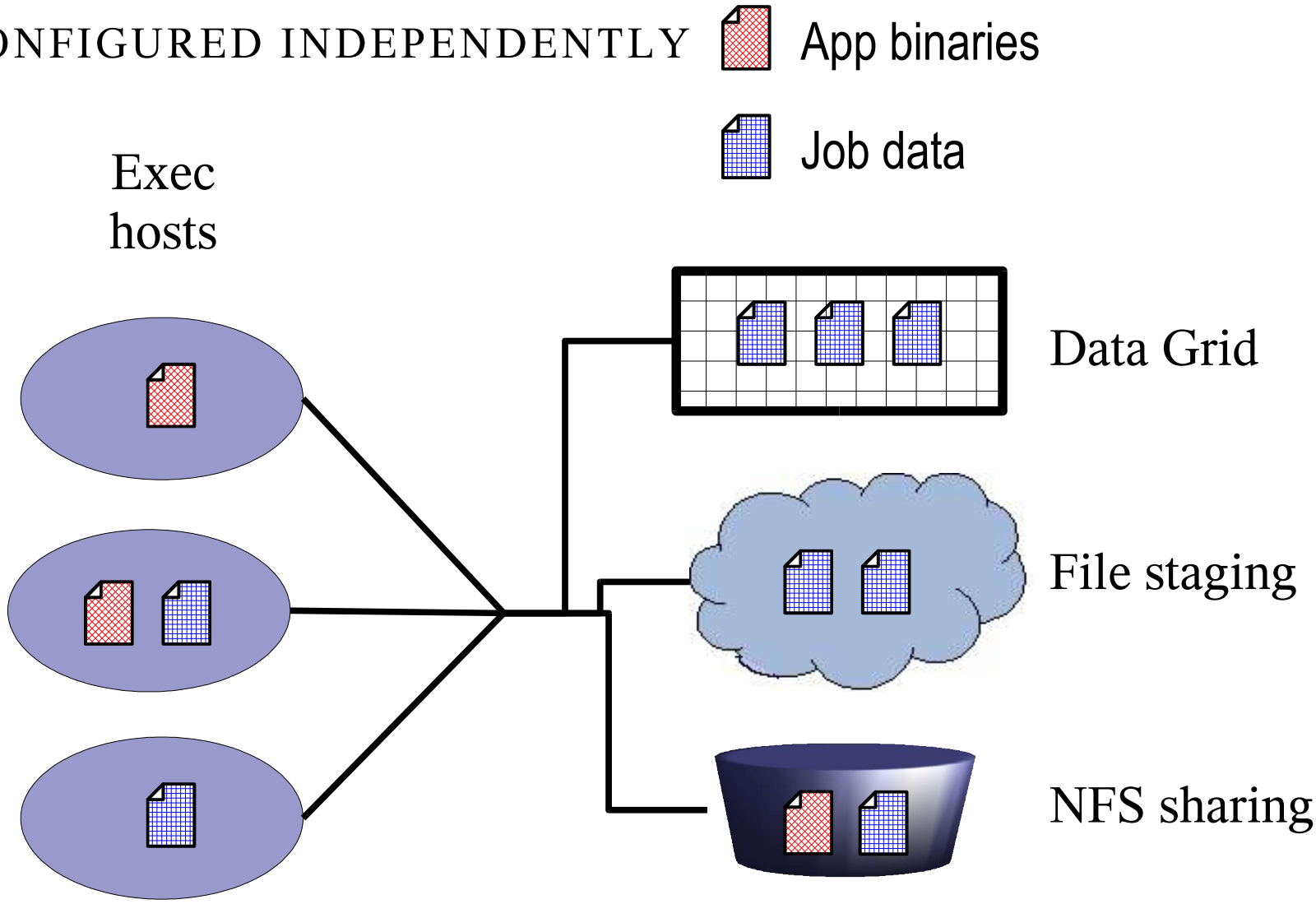
- Current resource consumption always monitored
- Total detailed consumption recorded at end of job
- Includes record of user, department, project, etc,

Accounting of resources

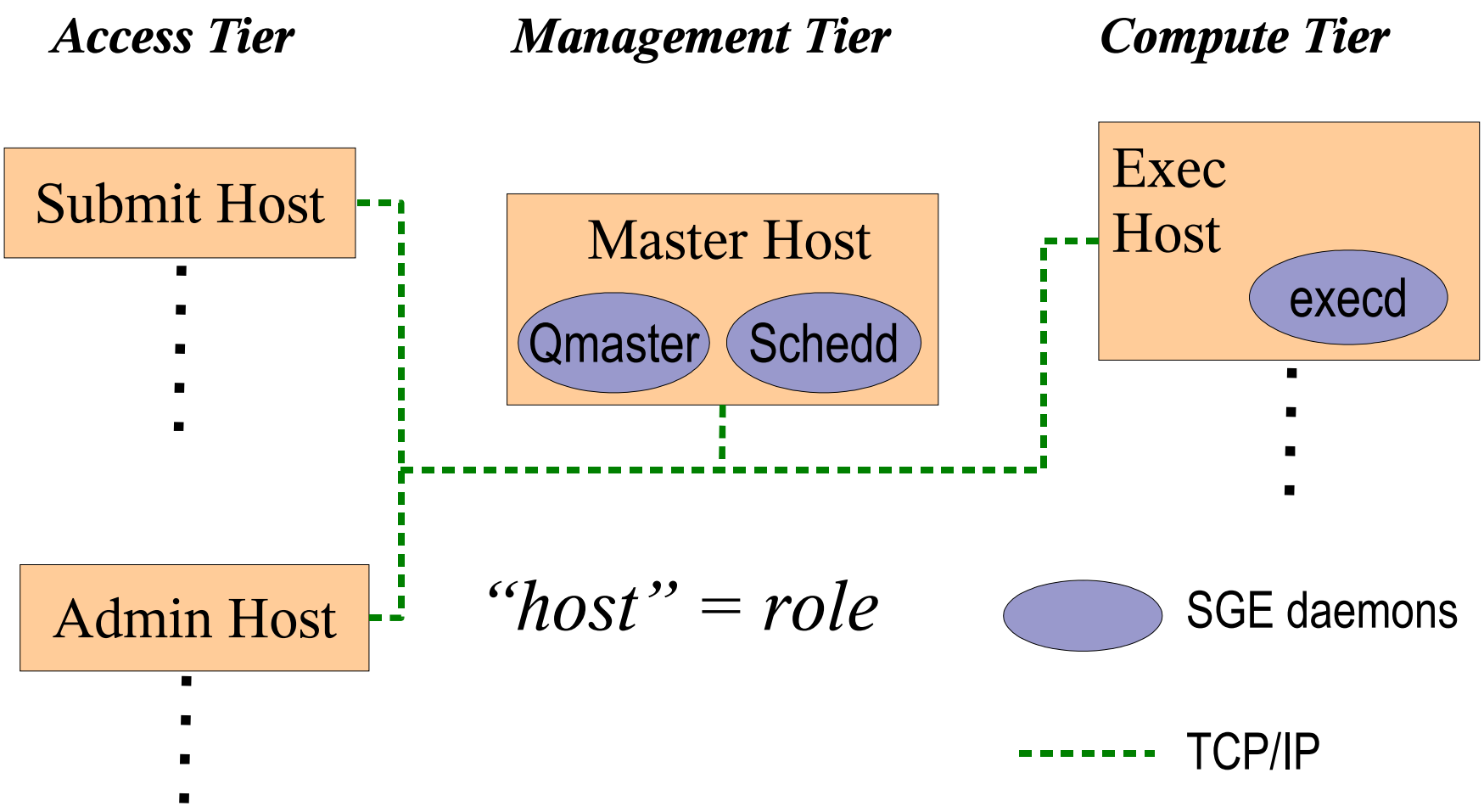
- Utilization of all resources recorded (eg, CPU, memory, license, etc)
- Usage accounting kept for users, departments, projects, etc.

Data Access

CONFIGURED INDEPENDENTLY



Grid Engine Architecture

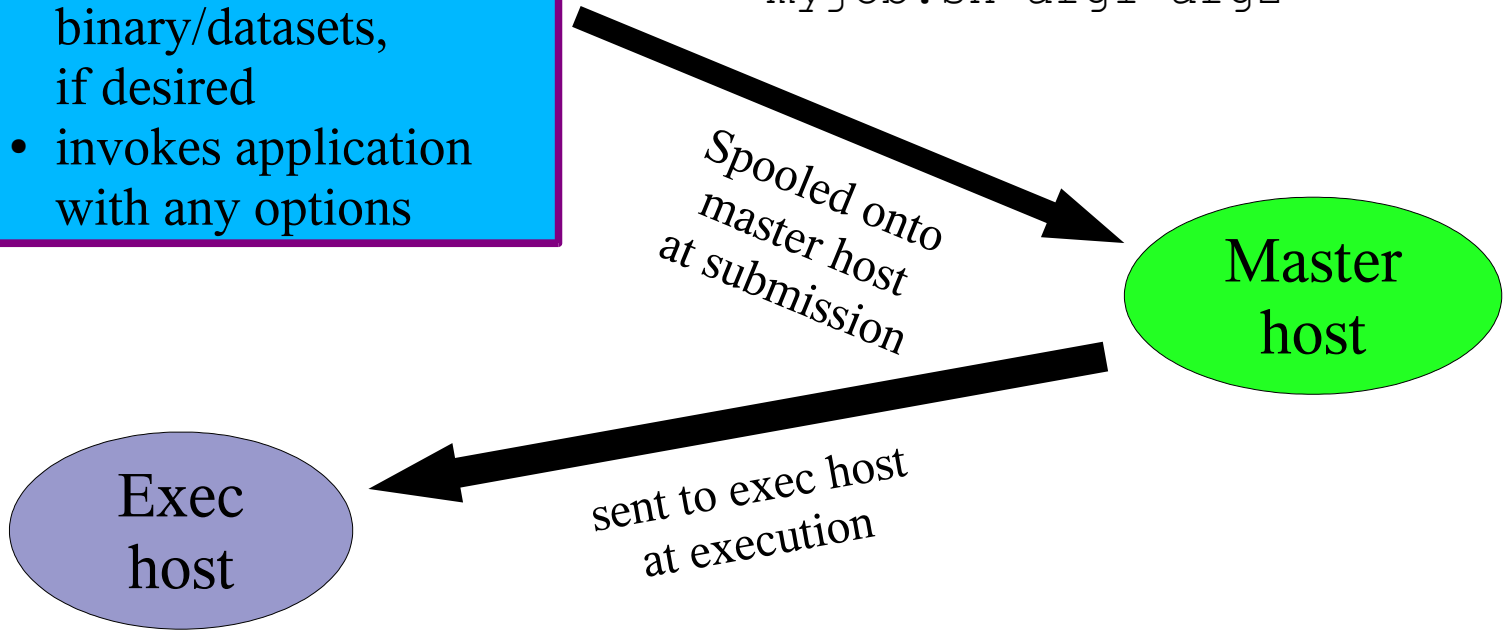


Batch Job Execution

job script
- any script, eg,
  #!/bin/csh
  #!/bin/perl
- can do pre-work,
  eg, preprocessing
- may stage in/out
  binary/datasets,
  if desired
- invokes application
  with any options

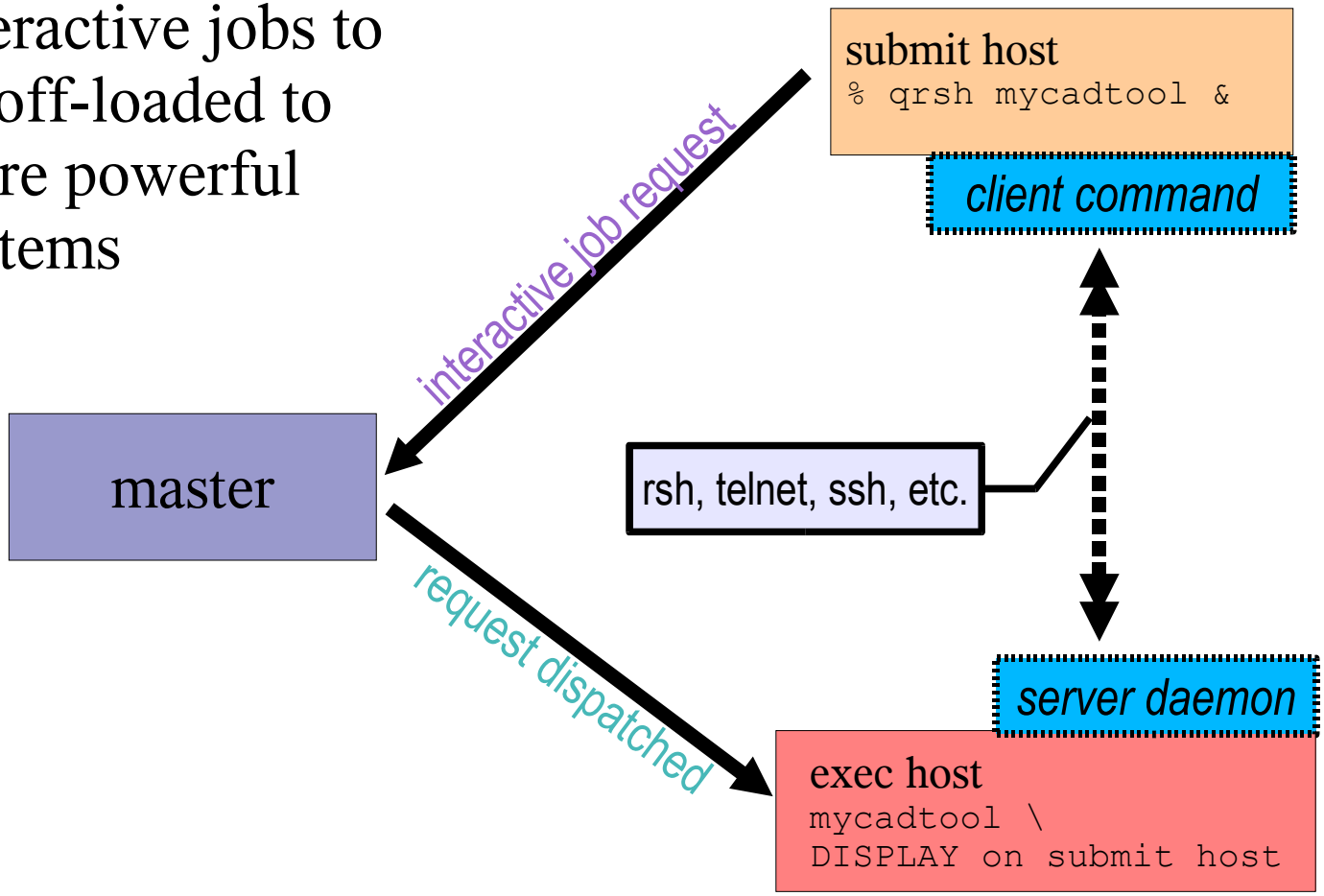
applications run without modification:
1)use job script, or
2)submit binary directly

```
% qsub -l memfree=1GB,
arch=linux, jobtype=analysis
myjob.sh arg1 arg2
```



Interactive Applications

Allows demanding interactive jobs to be off-loaded to more powerful systems



Resources

THE HEART OF GRID ENGINE MANAGEMENT

Per Host, eg,
- load_avg
- mem_free
- *OS/patch-level*

Global, eg,
- *floating licenses*
- *shared storage*

Built-in and custom resources

- Static resources: *strings, numbers, boolean*
- Countable resources: *eg, licenses, MB of memory/disk*
- Measured resources: *value provided through Load Sensor*

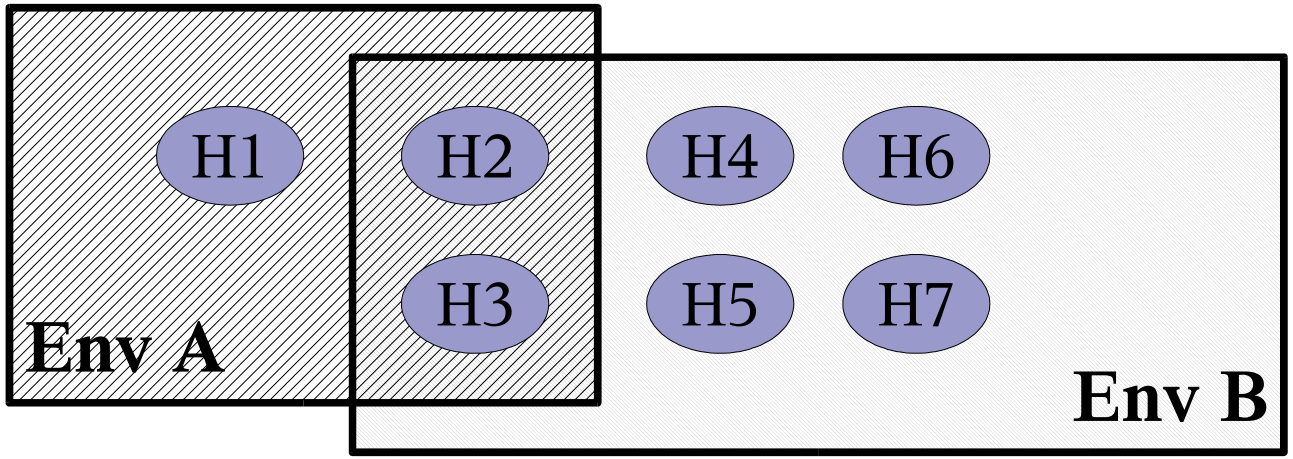
Resources used for

- Job resource request: *job A needs 1 license and 1GB*
- Action thresholds: *suspend jobs if load_avg > 1.5*
- Resource preference: *send jobs to hosts with least load; out of those, choose hosts with most free memory*

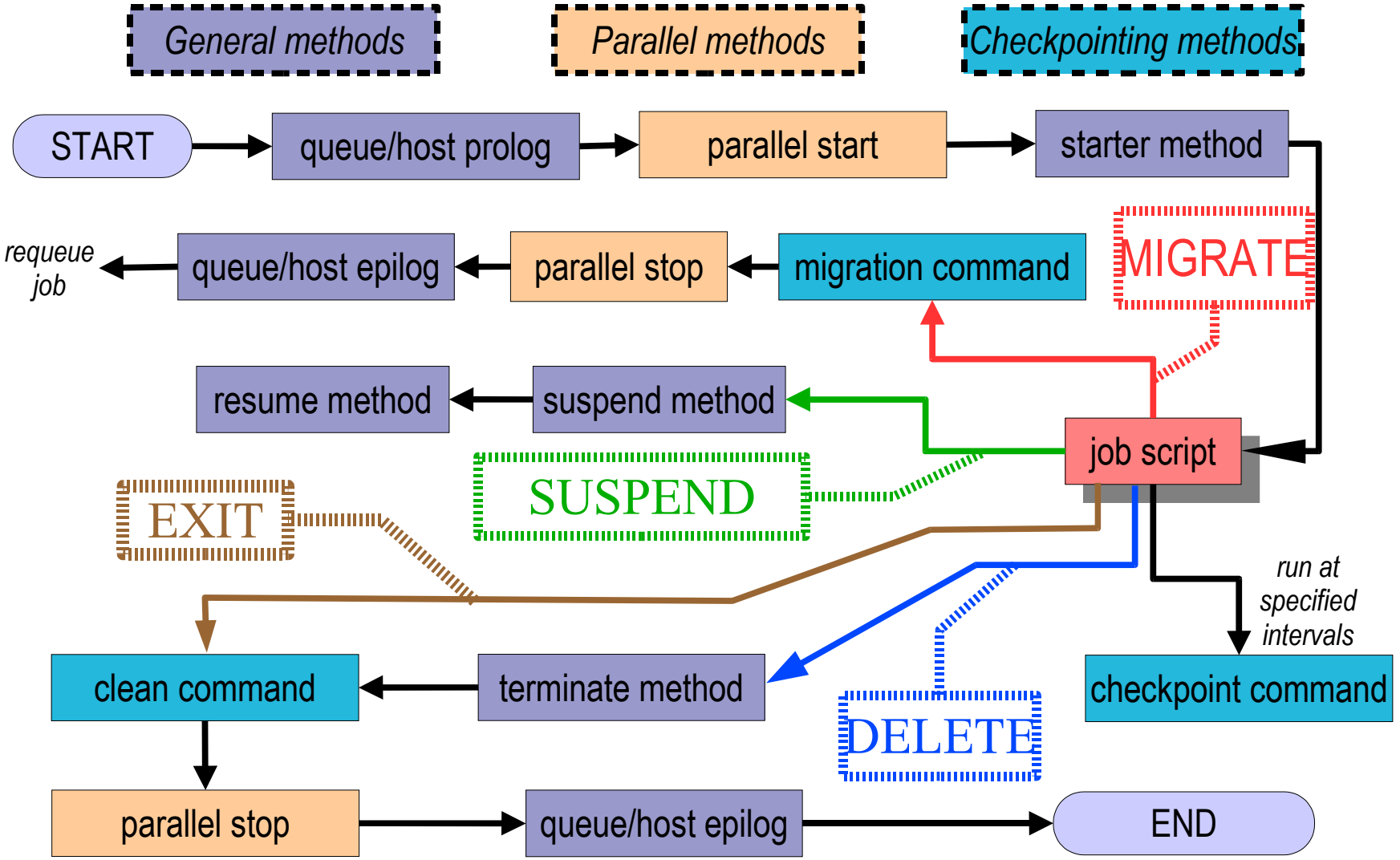
Parallel and Checkpointing Environments

Environment: a **set of hosts and associated procedures** that are used to support **parallel or checkpointing applications**

applications must inherently support parallel/checkpointing execution

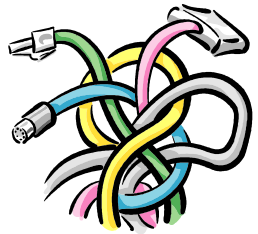


Application Integration Methods



N1GE 6 Scheduler

Next-generation Functionality

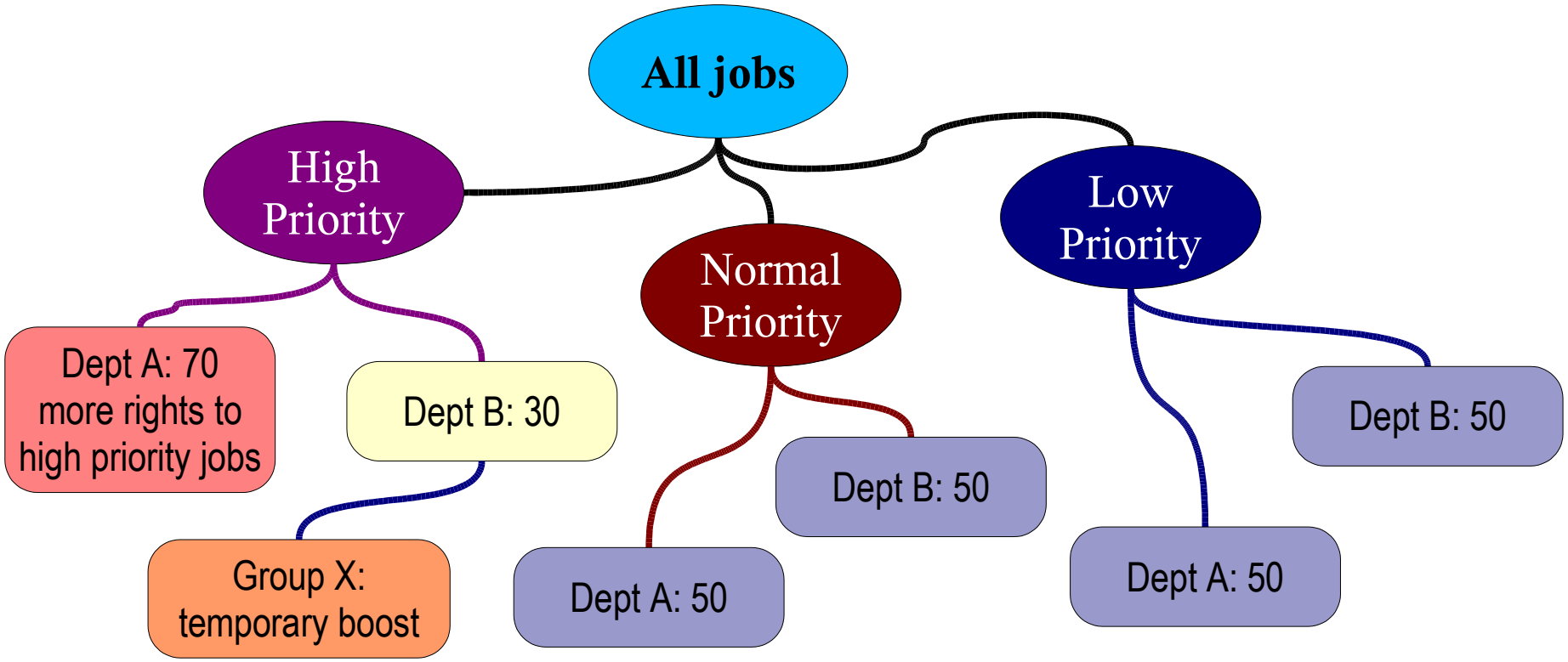


- Sophisticated Policy system
  - Align resource allocation with business priorities
  - Ability to implement SLA, QoS guarantees
  - Ensure greatest ROI for your assets
- Look-ahead planning
  - Resource Reservation / Preemption / Backfilling
    - Can reserve any resource, eg memory, CPU, license, disk
  - Ensure that important jobs get the resources they need, while maximizing overall utilization

1) Ticket-based Policies

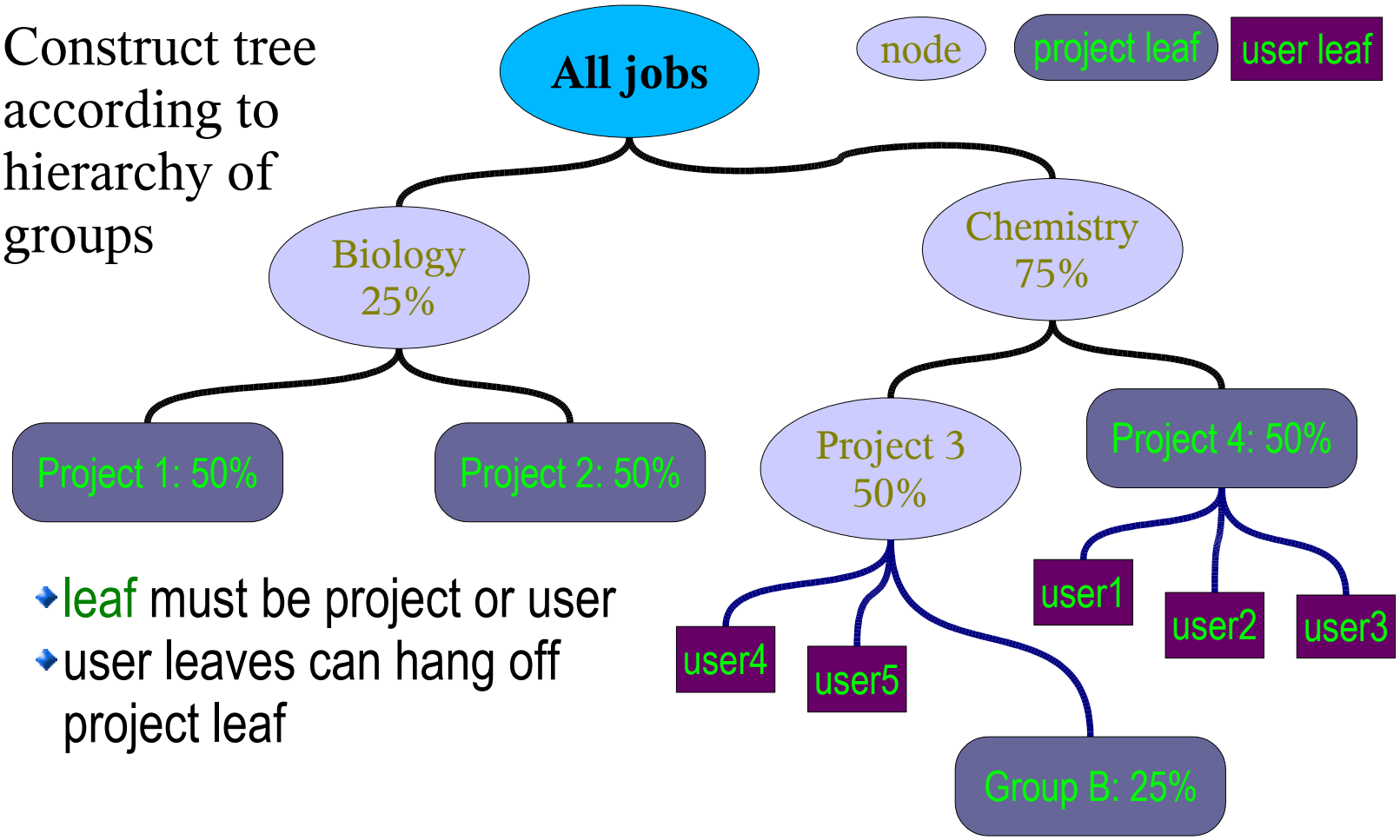
Resource allocation aligned to business priorities

- policy basis includes: department, project, user groups, cumulative utilization, category priority
- powerful, flexible, tunable, easy to configure



Ticket policy: Share Tree

Construct tree according to hierarchy of groups



- leaf must be project or user
- user leaves can hang off project leaf

Ticket Policy: Strict Order

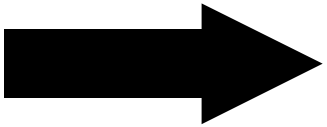
Objective: strict determination of job dispatch order

- jobs grouped according to type or grouping
- jobs from certain groups go ahead of others, regardless of submit order or current running jobs

Job Submit order

1) Project A: job1
2) Project C: job1
3) Project B: job1
4) Project C: job2
5) Project A: job2
6) Project C: job3
7) Project B: job2
8) Project A: job3
9) Project C: job4
10) Project C: job5
11) Project B: job3

"project" = type, group, etc



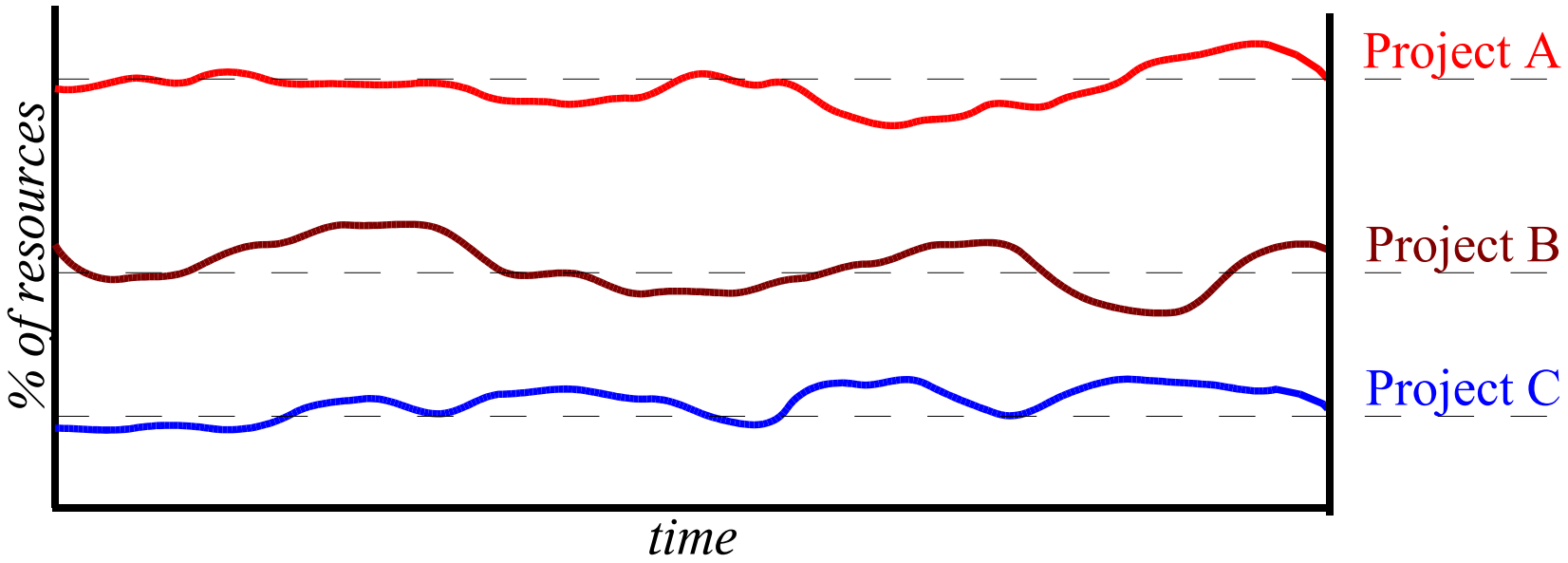
Job Dispatch order

1) Project A: job1
2) Project A: job2
3) Project A: job3
4) Project B: job1
5) Project B: job2
6) Project B: job3
7) Project C: job1
8) Project C: job2
9) Project C: job3
10) Project C: job4
11) Project C: job5

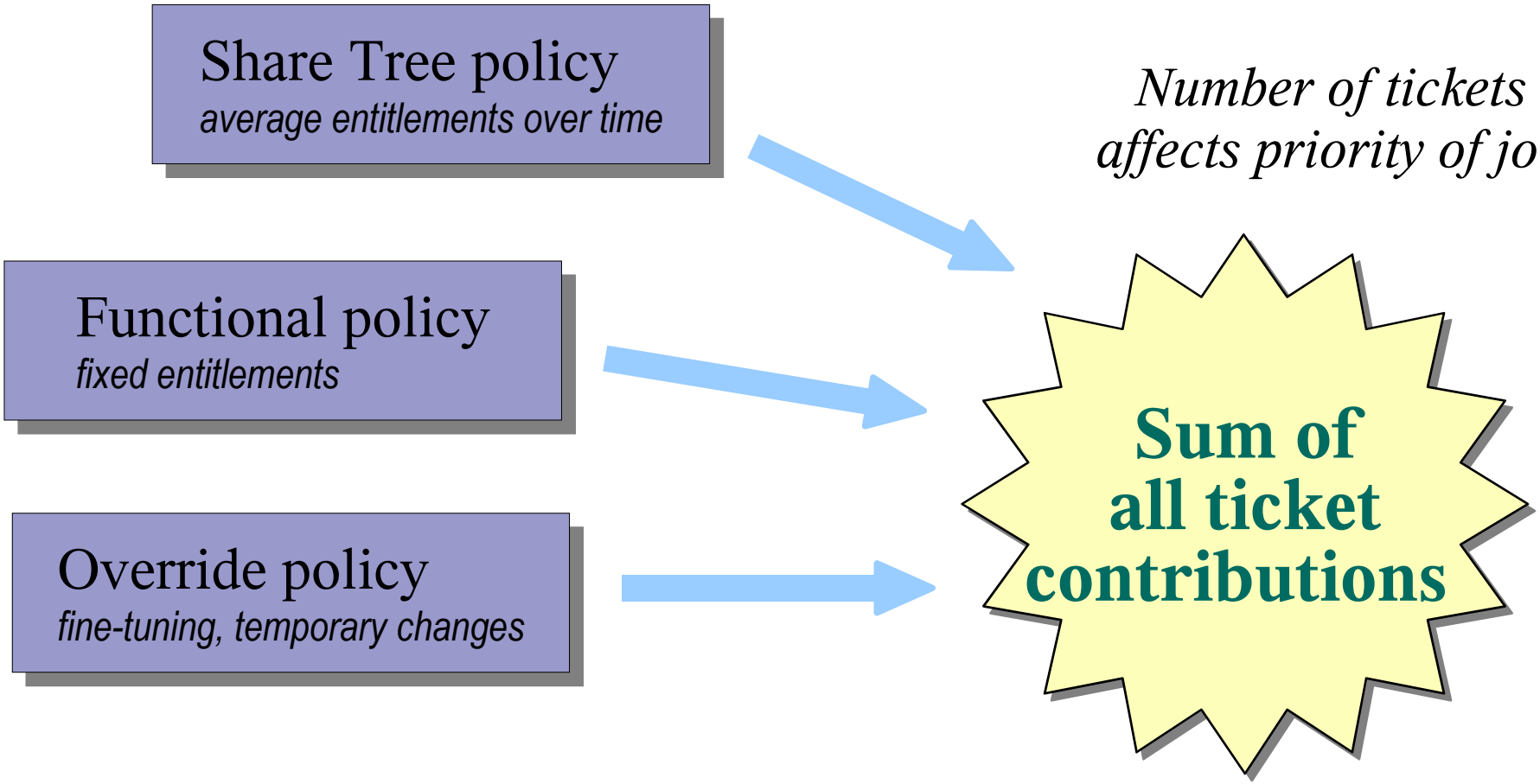
Ticket Policy: Preferential Order

Objective: preferred hierarchy of groups of jobs

- jobs grouped according to projects, users, departments
- groups are given certain target percentage of resources
- scheduler tries to implement targets globally, eg, Project B jobs might go before Project A jobs if Project B is using less than target share at that time



Unifying Ticket Policies



2) Urgency Policies

- Resource-based urgency: gives priority to
  - expensive asset (SW license, $$$ server)
  - demanding job (multi-CPU, lots of memory)
- Deadline-based urgency
  - Enables SLA-type policies
- Wait-time urgency
  - Enables quality-of-service guarantees

3) POSIX priority Sub-policy

- Priority value assigned to jobs with a simple number between -1023 and 1024
- Used to implement site-specific custom priorities (eg, external co-scheduler)
- Also enables admin override of any other policies

Combining Policies

- Final priority of jobs calculated based on unifying the three sub-policies
- Each policy normalized to $0.0 < N < 1.0$ before combining using weight factors
- Allows hierarchy of policies

$$\mathbf{prio} = W_{urg} \times N_{urg} + W_{tix} \times N_{tix} + W_{psx} \times N_{psx}$$

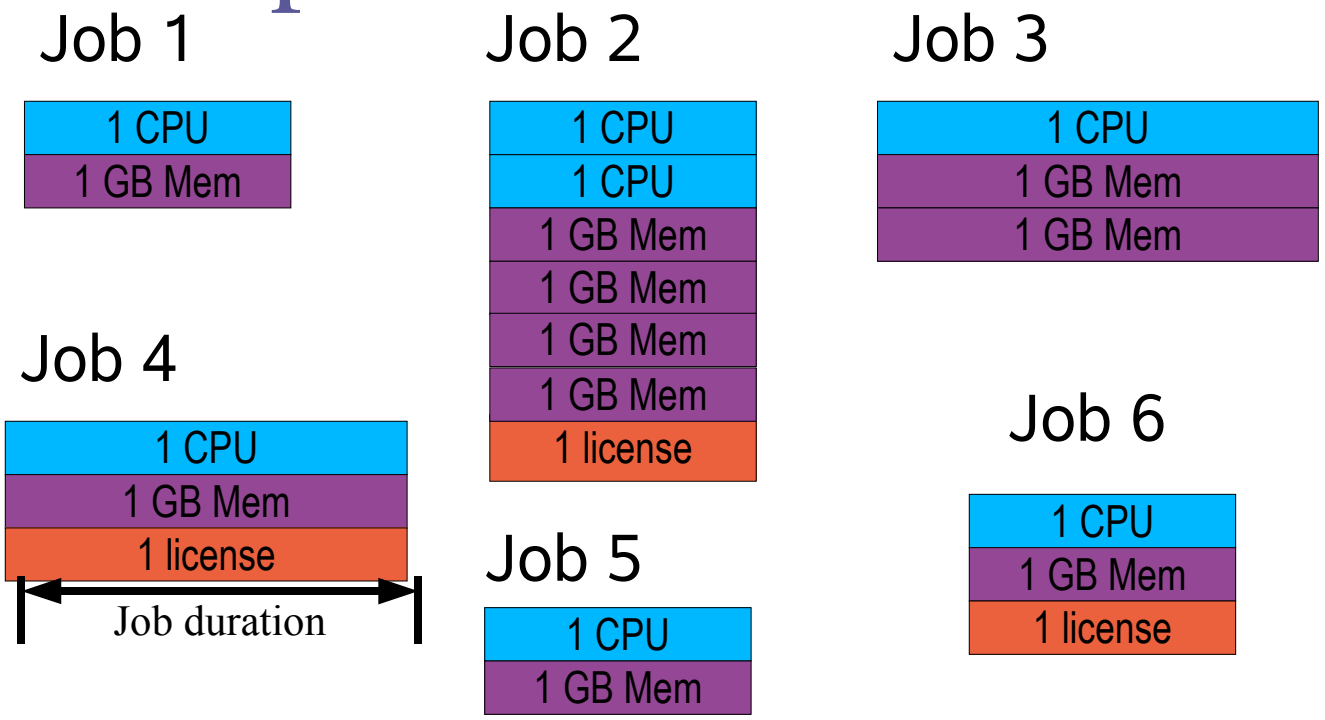
N_{urg} = normalized Urgency

N_{tix} = normalized Tickets

N_{psx} = normalized Posix

W = weighting factors

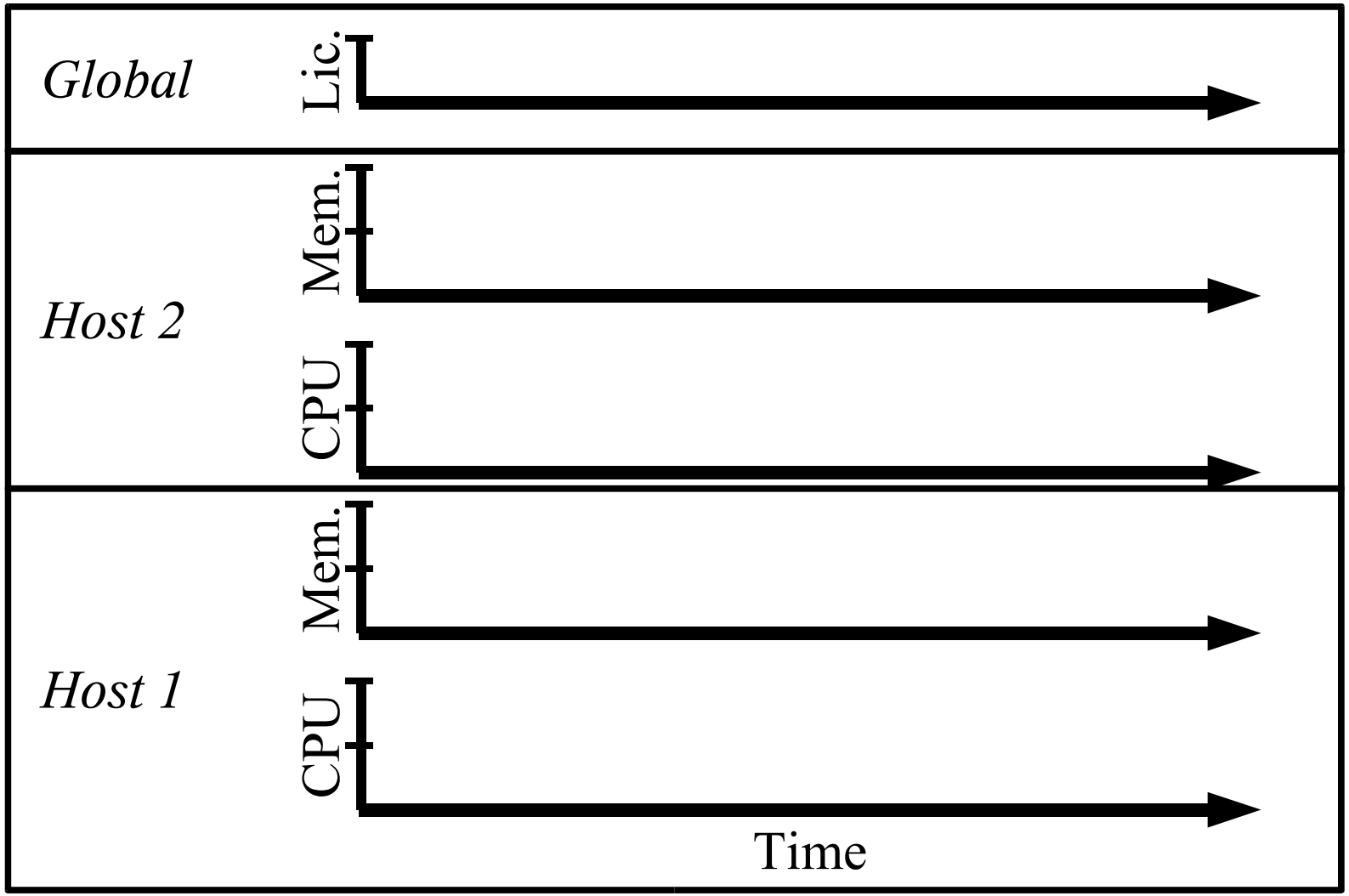
Resource Reservation: an example



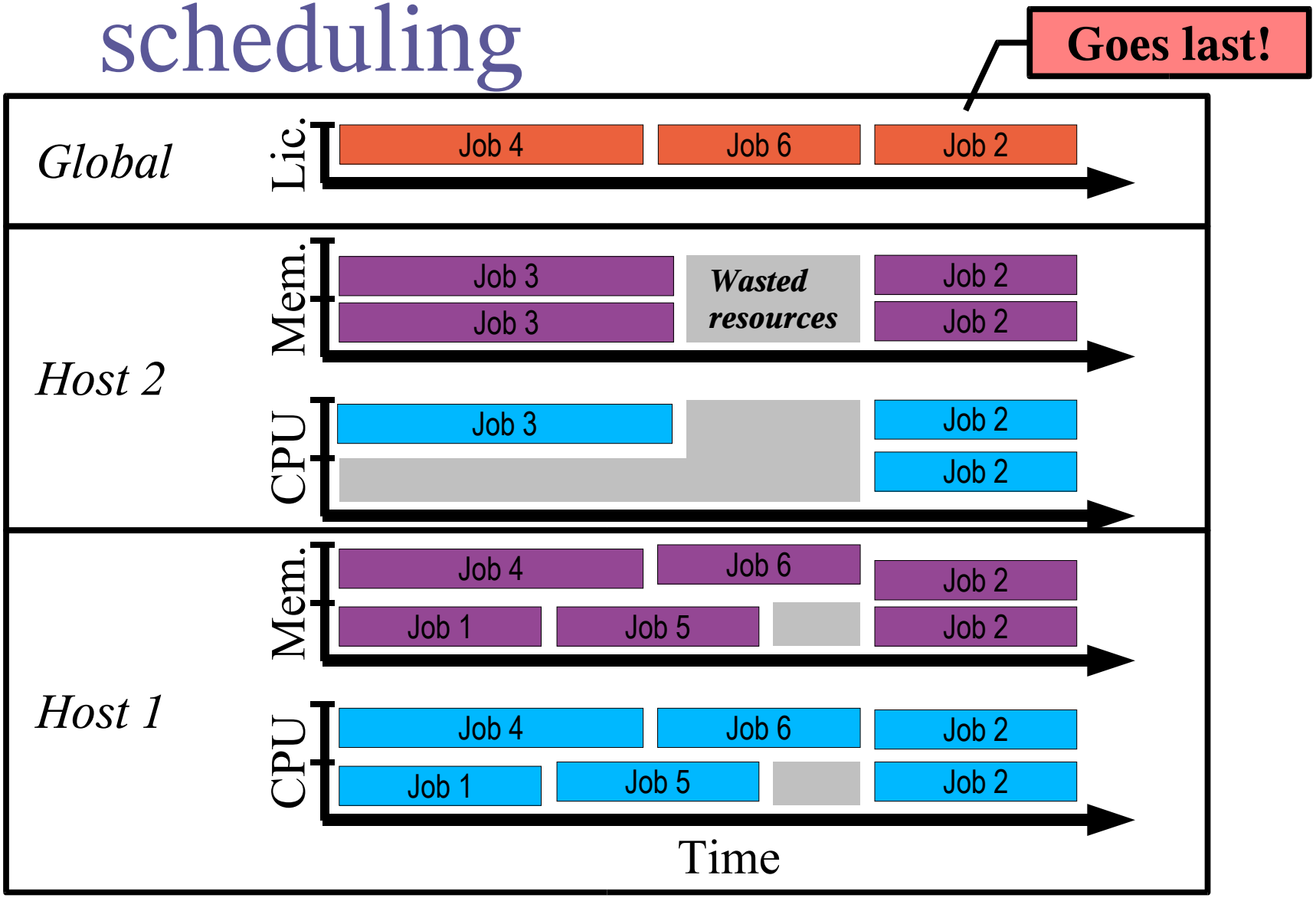
Jobs, with resource requirements

- *number indicates priority*
- *length represents duration of requirement*

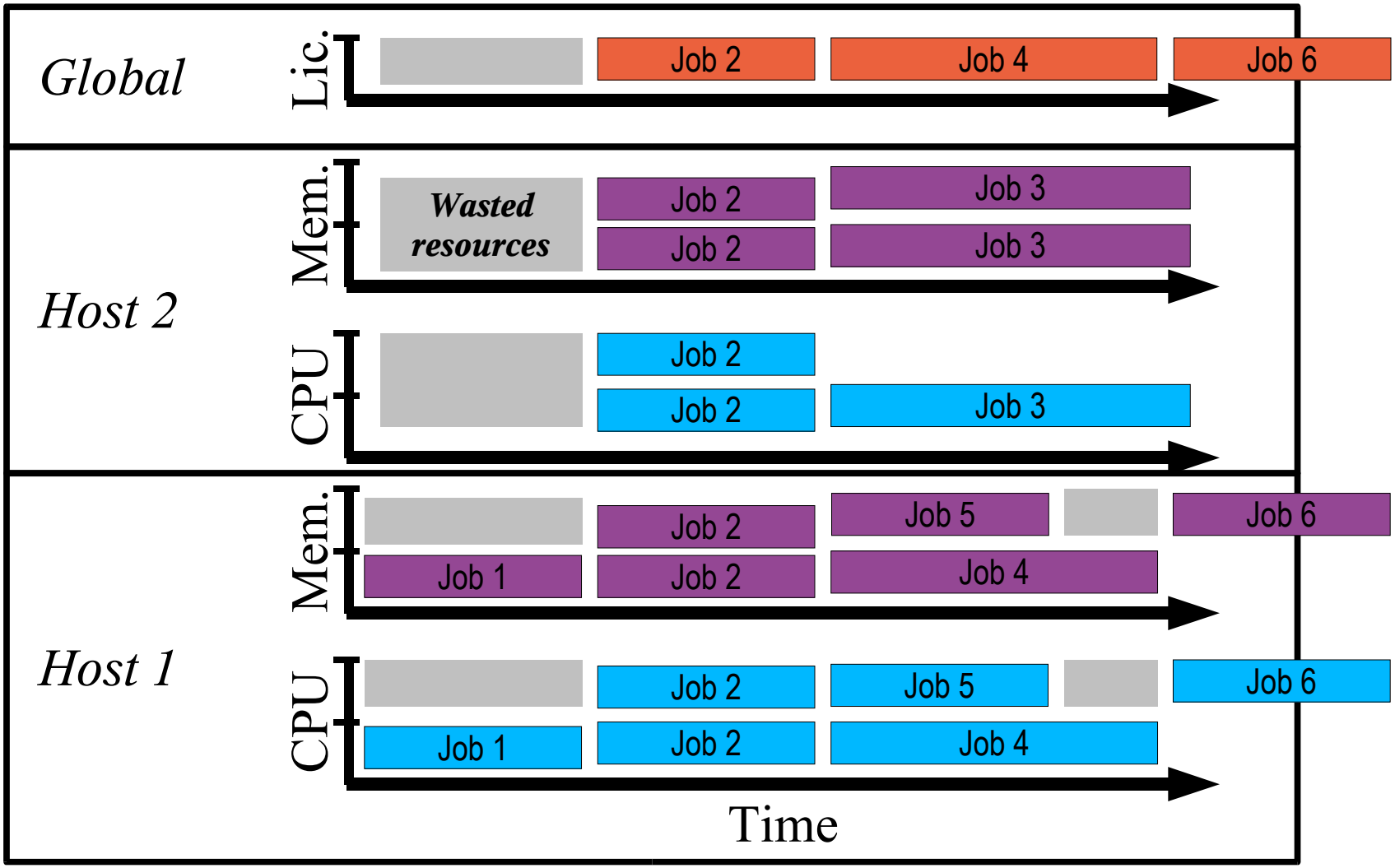
Grid Resources Representation



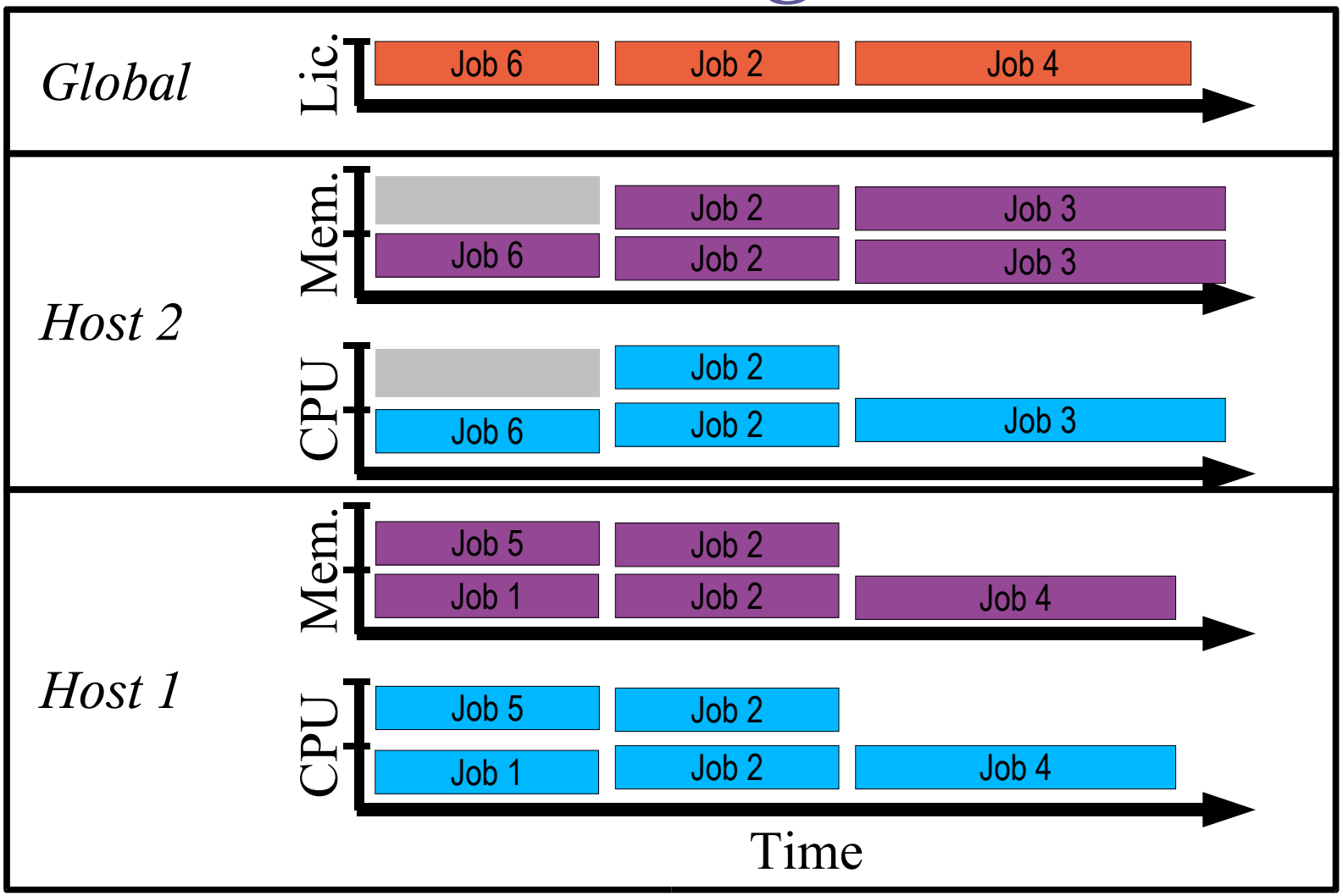
Simple, priority-based scheduling



Scheduling with Resource Reservation



Resources Reservation with Backfilling



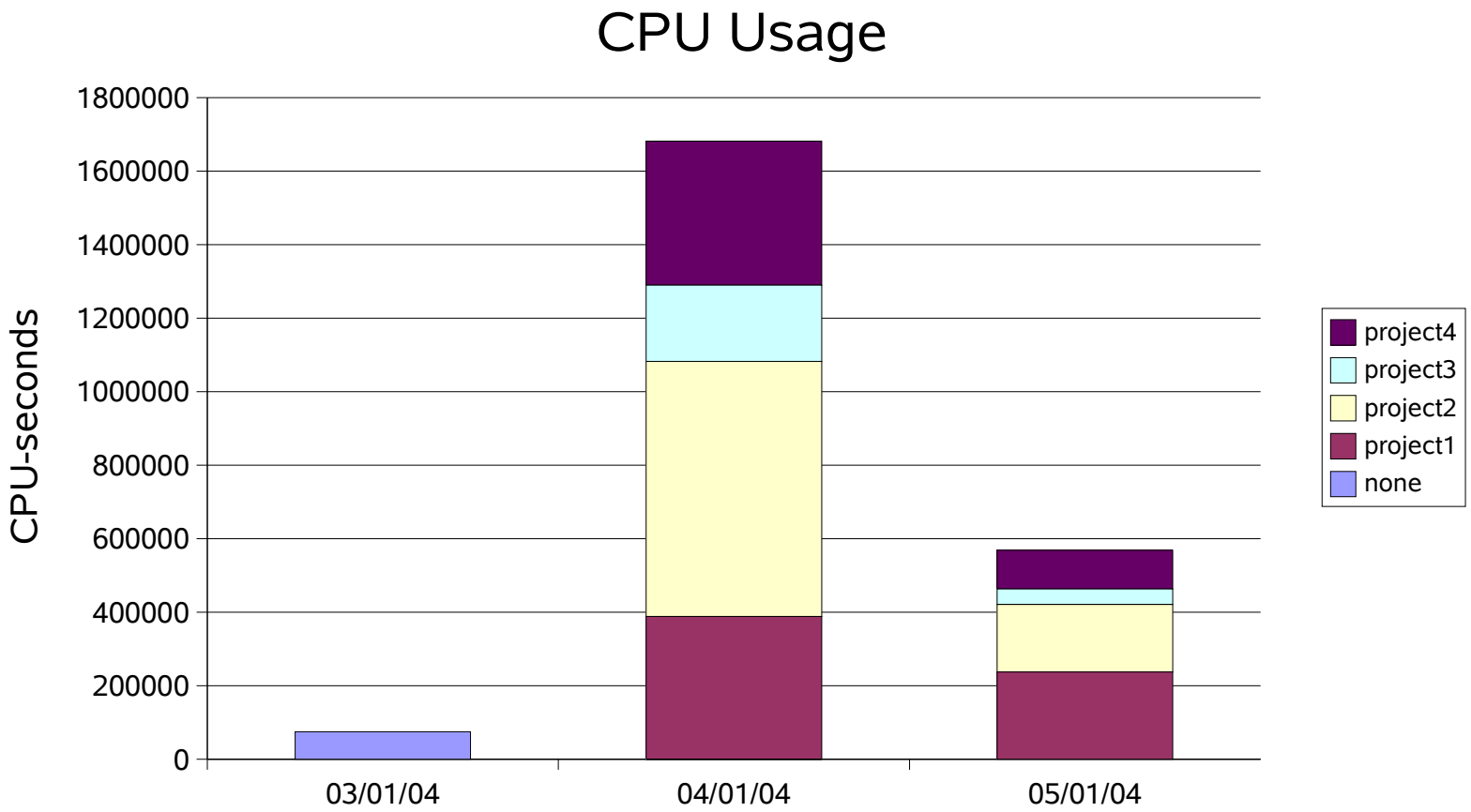
N1GE 6 Reporting

Analysis / Monitoring / Accounting

- Module for doing analysis, monitoring, accounting reports, etc.
  - Fine-grained resource recording
  - Stored in RDBMS in well-defined schema
  - provides built-in capability for analysis, reporting, chargeback, etc
  - Web-based console tool provided for generating reports, queries, etc.
  - Standard SQL access for 3rd party tools

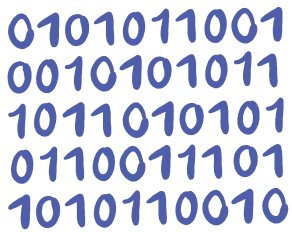
Example Analysis

DATA IMPORTED INTO STAROFFICE



N1GE 6 Enhanced Performance

Scalability



- Berkeley DB spooling
- Multi-threaded Master Daemon
- New communication system

- *Scalability goals:*
  - 10,000 unique hosts
  - 500,000 unique jobs

N1GE 6 Other Features

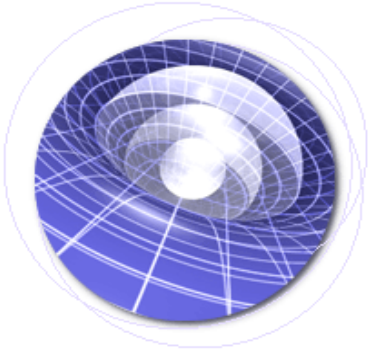
Standards Compliance

- DRMAA 1.0
  - API to submit, control, monitor jobs
  - Audience: developers who want to “grid-enable” their applications, without customizing to particular DRM/Grid system
  - C-binding, Java-binding (future: Perl)
- XML-based monitoring
  - Status command 'qstat' outputs full status info in well-defined XML DTD

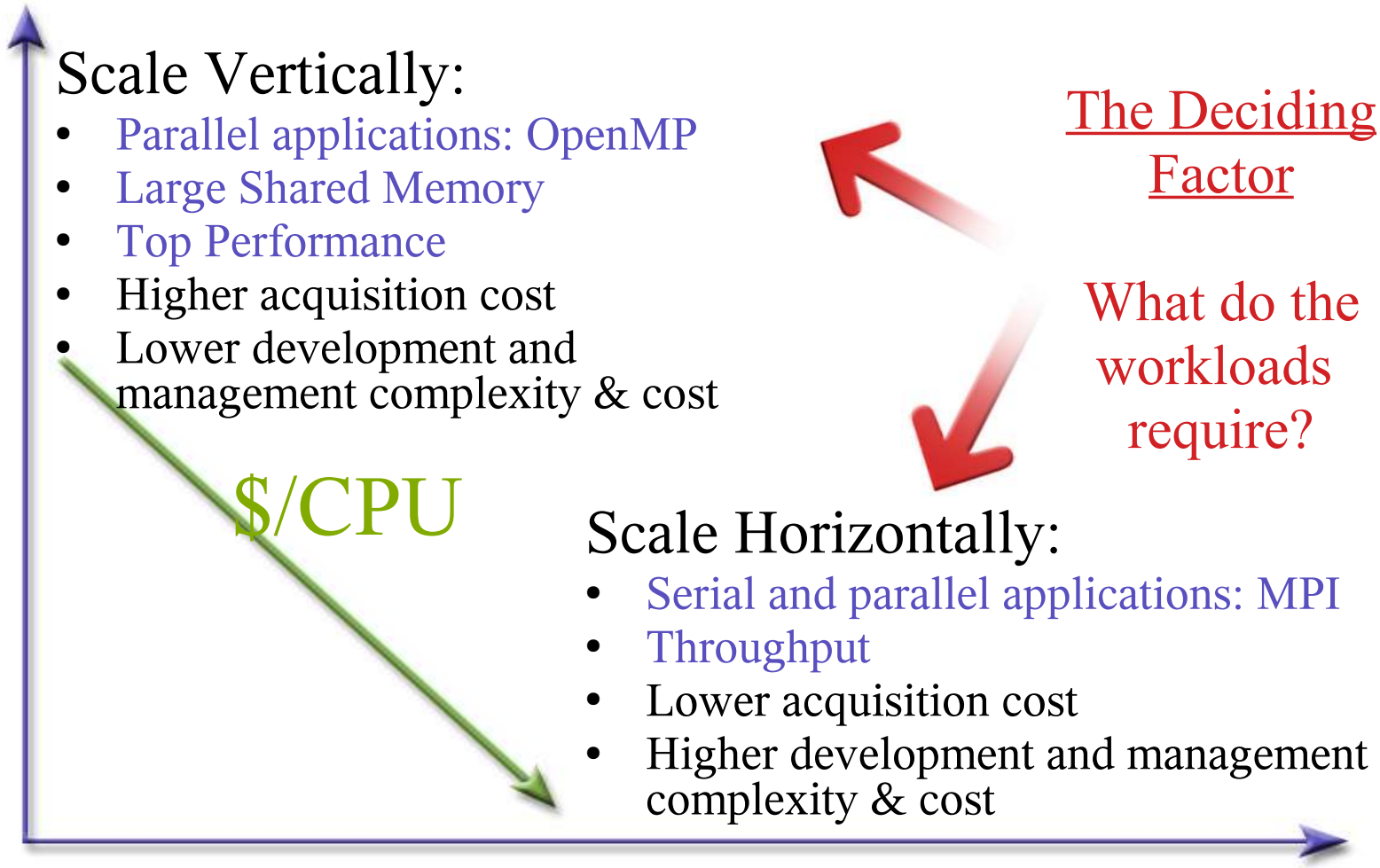
Grid Engine

Useful Links

- http://www.sun.com/grid
  - Success stories
  - Web-based training and testing
  - Grid Certification course information
  - Partner information
- http://www.sun.com/gridware
  - Sun Grid Engine products
- http://gridengine.sunsource.net
  - Open Source Project
  - Mailing Lists for users, developers (get answers from SGE experts)
  - FAQs, HOWTOs, documentation for users and developers
  - Courtesy Binaries for all major Unix platforms



The Grid Architecture Dilemma: Scale Vertically or Scale Horizontally?



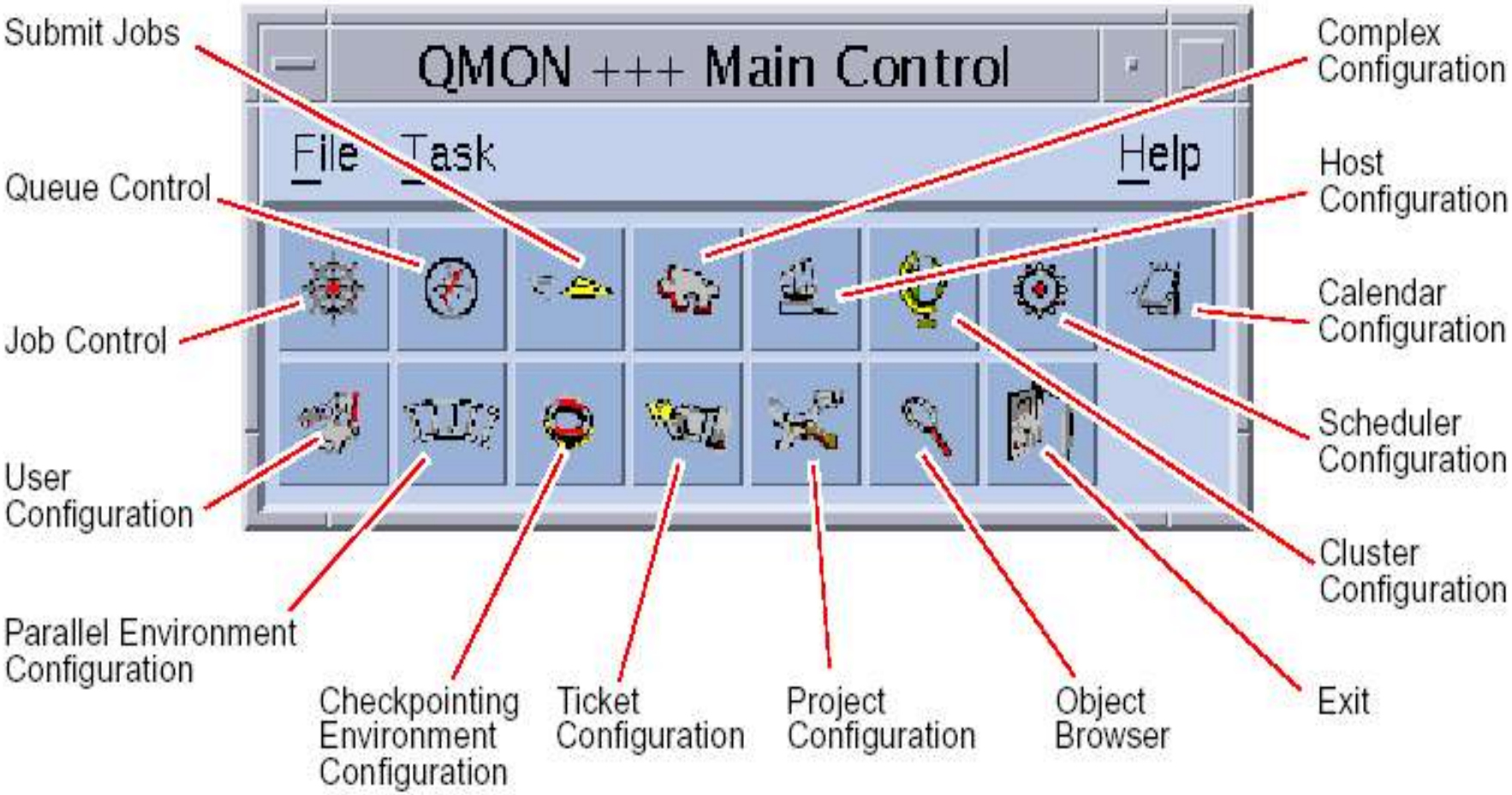
Processor Choice

	SMP	x86
Large Shared Memory	X	
32 bit x86 code		X
64 bit x86 code		X
Mixed x86 code		X
Single Threaded		X
Multi threaded	X	

Node Selection

- Understand your workload characteristics
  - Vertical or horizontal scaled
  - Interconnect requirements
  - Processor requirements
    - 32 bit v 64 bit
    - SPARC or x86 code
    - Single threaded, multi-threaded
  - Application requirements/restrictions

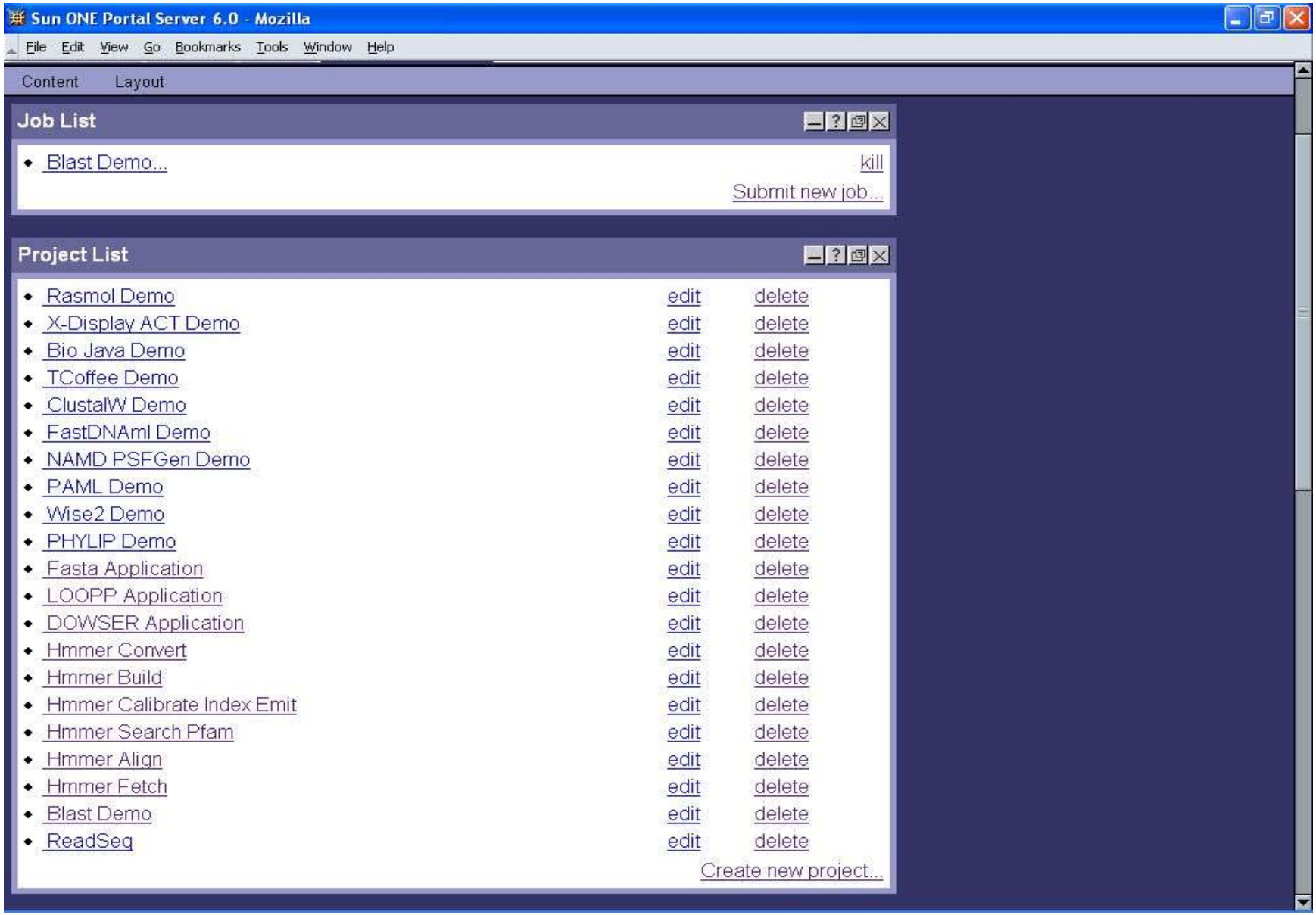
Qmon: N1 Grid Engine's GUI



Grid Engine Portal

- Web interface to Sun Grid Engine & SGE-EE
- Highly secure internet access to Grid applications
- Submit and Monitor SGE jobs (no Unix skills required)
- Securely upload input files to the Portal Server
- Securely download output files to local workstation
- View X-Windows based applications using VNC
- Register new applications to the Grid in minutes
- Dynamic Project Directory Creation
- Accounting capabilities
- Runs on Sun ONE Portal Server

Grid Engine Portal (Web Interface)



What Grid is Not

It's *not* futuristic

Grid technology is:

- Here now
- Real

Sun grid solutions are:

- Based on solid technology
- Ready to be delivered today!

What Grid is Not

It's *not* new technology

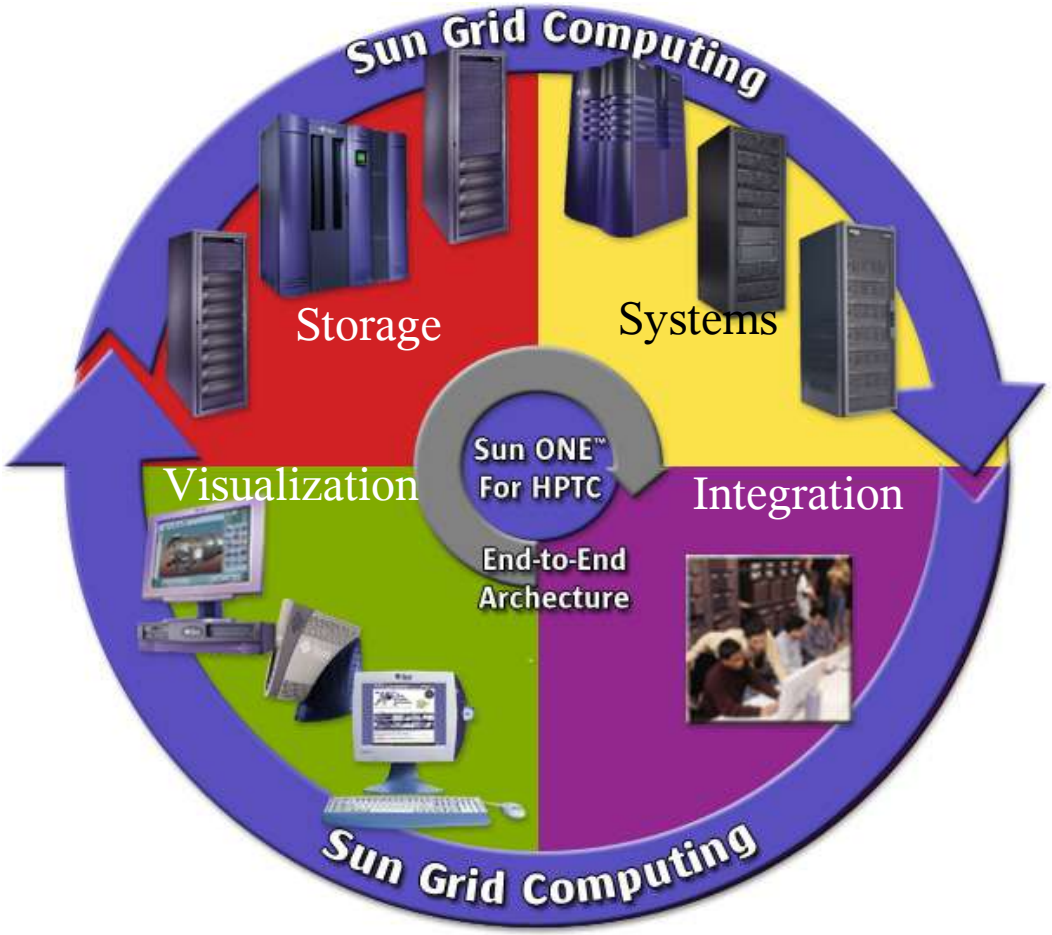
- The evolution of grid has been ongoing for many years
- Sun has been an active participant in the growth and development of grid technology
- Sun has been assisting customers deploy grid technology for several years

Sun's Philosophy
"All Grid, All the Time"

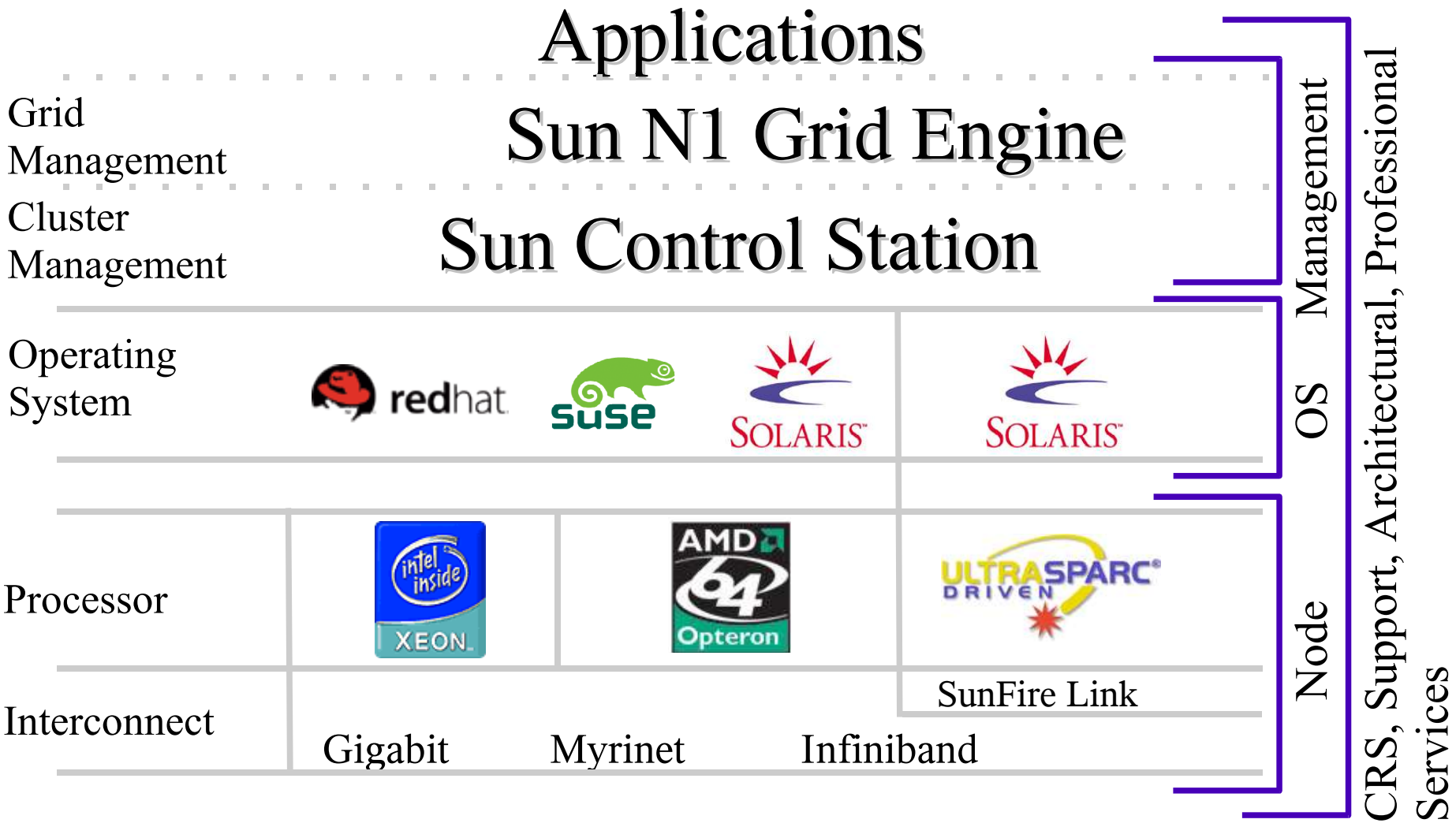
Data Grids

Compute Grids

Graphics Grids



Complete Grid Stack

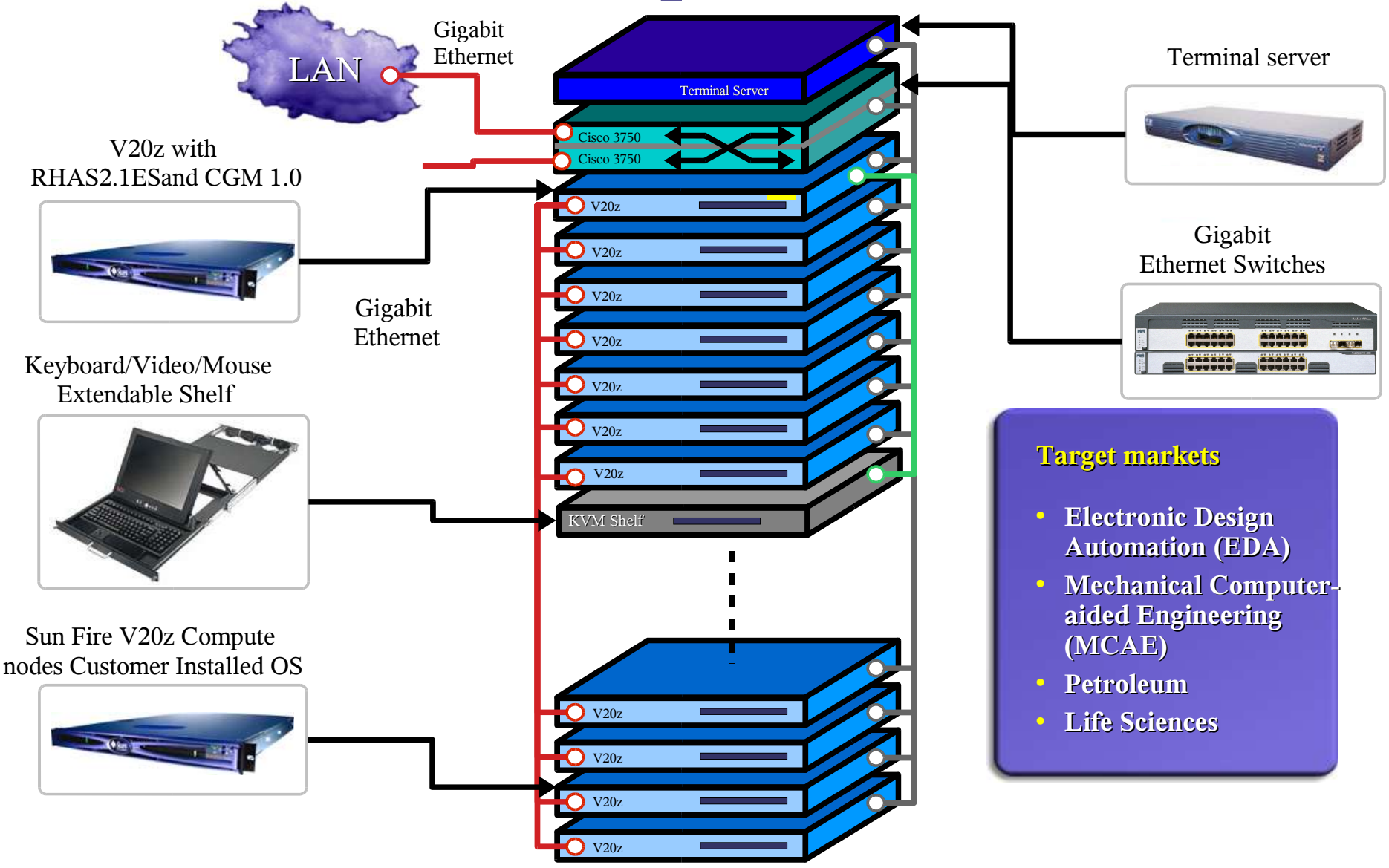


Compute Grid: Systems to Scale Horizontally

Low Cost Building blocks for Compute Grid Services

- Thin Servers
  - V20z / V40z: 32bit x86, Solaris, Linux
  - Compute Grid Rack (V20z)
  - V210 / V240: 64bit SPARC, Solaris
- Small 4 - 8 way SMP Servers
  - V480/490
  - V880/890

Horizontal Compute Grid Rack

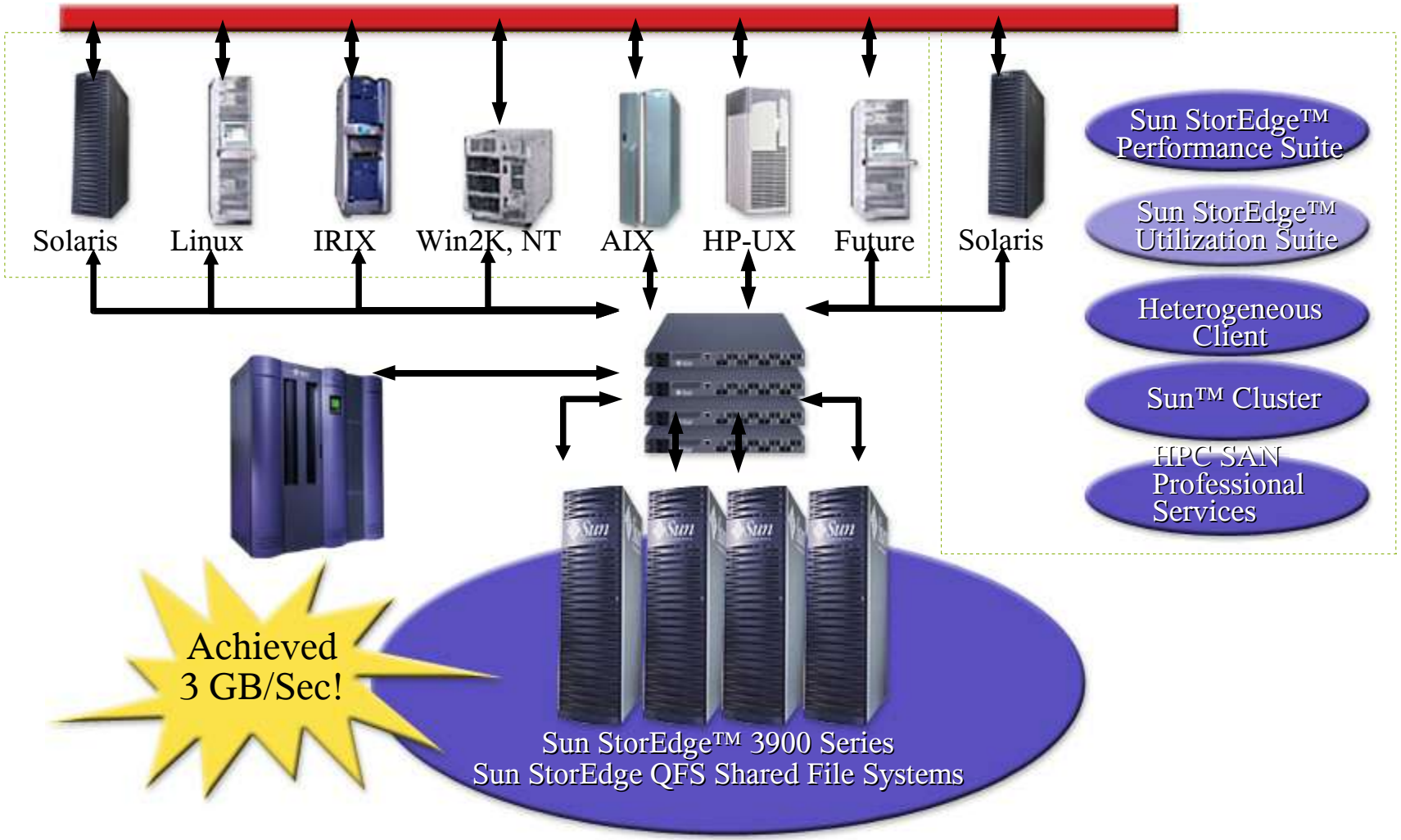


Compute Grid: Systems to Scale Vertically

Low Complexity Building blocks for Compute Grid Services

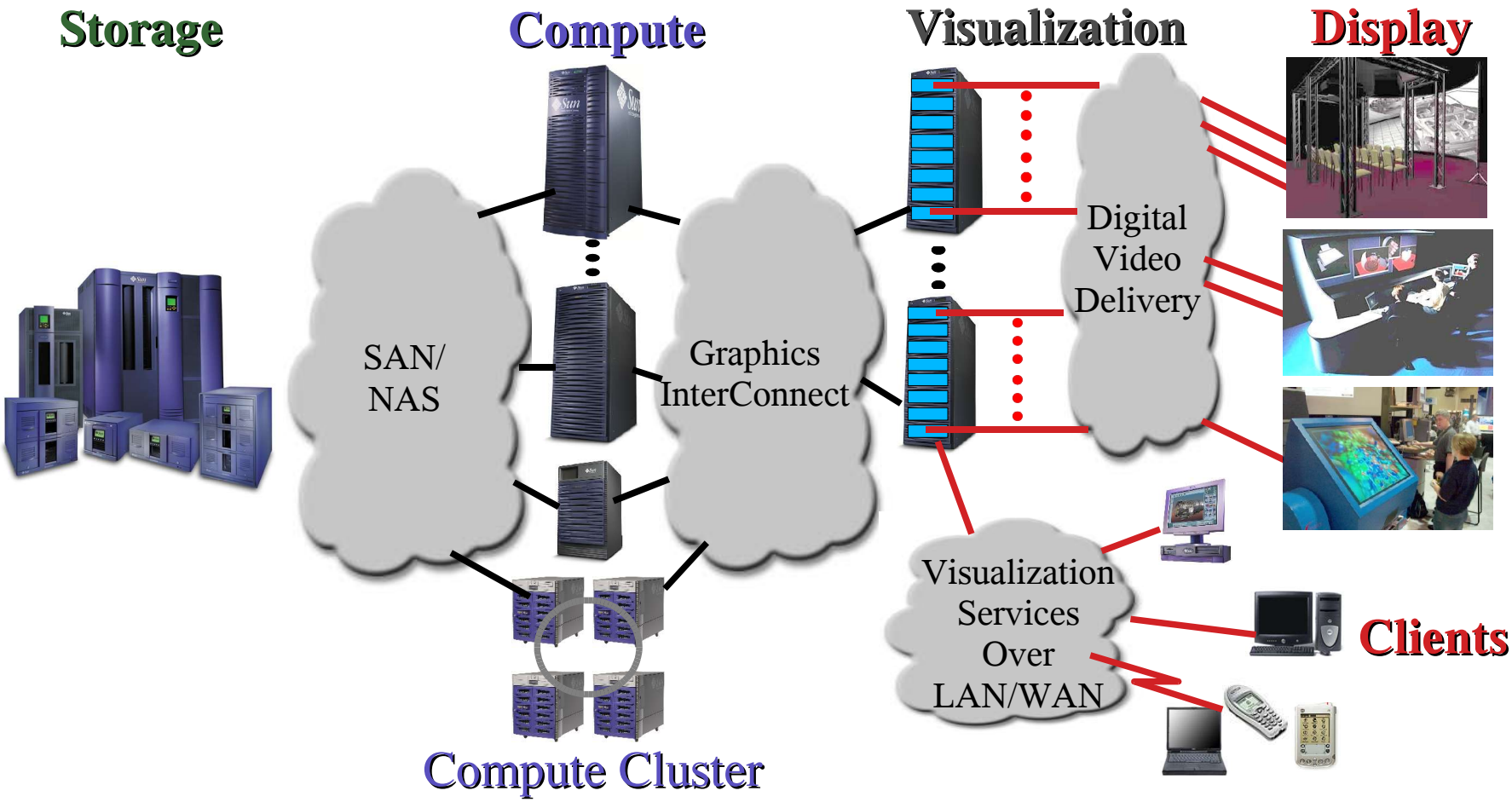
- Large SMP Servers
  - SF V1280/E2900
  - 4800/4900
  - SF6800/E6900
  - SF12K/E20K
  - SF15K/E25K

Data Grid: HPC SAN



Graphics Grid:

Access for More Users to Visualization Services at Required Visual Quality and Performance Levels



The Sun Ranch

- Sun microprocessor design
- Large scale EDA computing using Cluster Grid approach
- Best design practices for highly productive computing

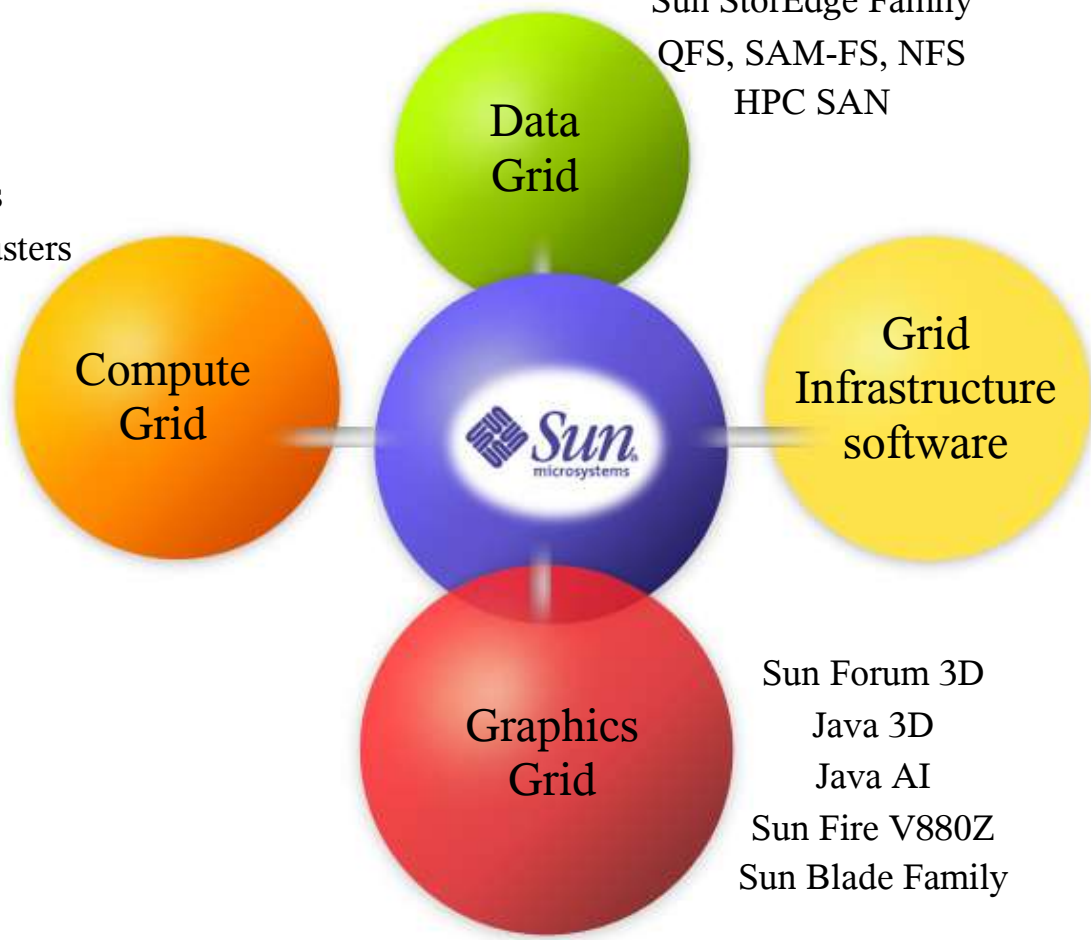
9-yrs compute time/day!
98% CPU usage 24/7/365!
10,000+ UltraSPARC CPUs in 3 grids
100% SPARC,Solaris,Storedge
30+ HA-clusters
520TB A5x00 & T3 DiskArrays
Gigabit networking
300 SunRays
250 EDA CAD Tools
Tape Robot Backups
15Ksqft space at 200Watts/sqft

Sun Differentiation:

Innovation Through IP Ownership

Sun StorEdge Family
QFS, SAM-FS, NFS
HPC SAN

Terascale Capability Clusters
SPARC & x86 Throughput Clusters
Sun Fire Link
Infiniband
Trusted Solaris
Sun Fireplane Architecture
UltraSPARC



N1 Grid Portal
N1 Grid Engine
HPC Cluster Tools
JES Studio Tools
JXTA
N1

Sun Forum 3D
Java 3D
Java AI
Sun Fire V880Z
Sun Blade Family

Q&A

Mitesh Agarwal

IT Architect

Sun Microsystems

mitesh.agarwal@sun.com

http://sun.com/grid

Sun microsystems

We make the net work.