Hálózatba kapcsolt erőforrás platformok és alkalmazásaik

Simon Csaba
TMIT

2017

Parelell computing

Multiprocessing

- Flynn's Taxonomy of Parallel Machines
  - How many Instruction streams?
  - How many Data streams?
- SISD: Single I Stream, Single D Stream
  - A uniprocessor
- SIMD: Single I, Multiple D Streams
  - Each "processor" works on its own data
  - But all execute the same instrs in lockstep
  - E.g. a vector processor or MMX

Flynn’s Taxonomy

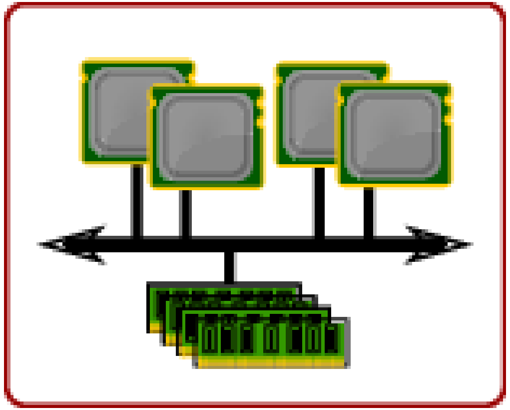
- MISD: Multiple I, Single D Stream
  - Not used much
  - Stream processors are closest to MISD
- MIMD: Multiple I, Multiple D Streams
  - Each processor executes its own instructions and operates on its own data
  - This is your typical off-the-shelf multiprocessor (made using a bunch of “normal” processors)
  - Includes multi-core processors

Multiprocessors

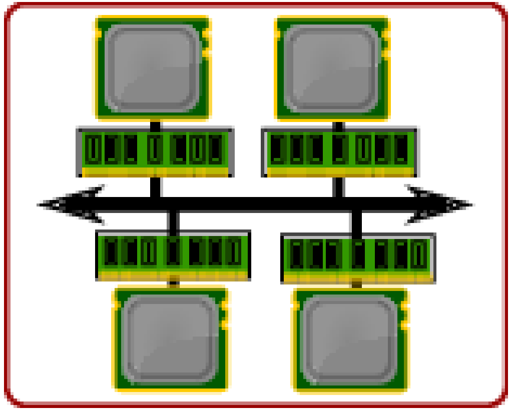
- Why do we need multiprocessors?
  - Uniprocessor speed keeps improving
  - But there are things that need even more speed
    - Wait for a few years for Moore's law to catch up?
    - Or use multiple processors and do it now?
- Multiprocessor software problem
  - Most code is sequential (for uniprocessors)
    - MUCH easier to write and debug
  - Correct parallel code very, very difficult to write
    - ***Efficient*** and correct is even harder
    - Debugging even more difficult (Heisenbugs)

MIMD Multiprocessors

Centralized Shared Memory

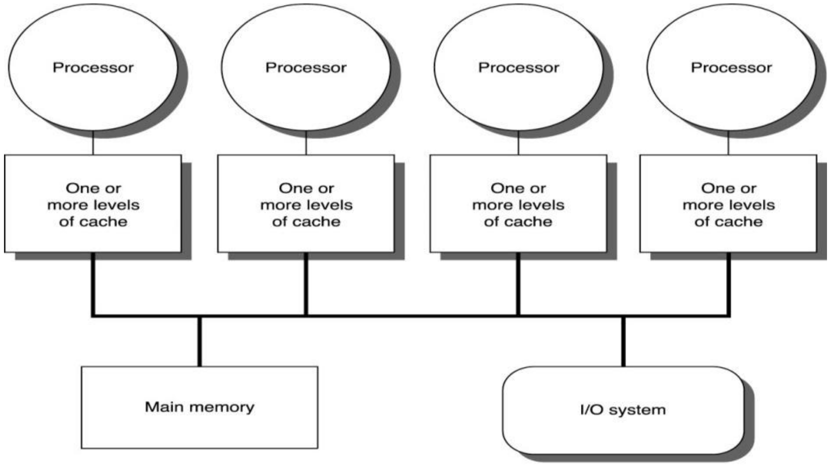


Distributed Memory

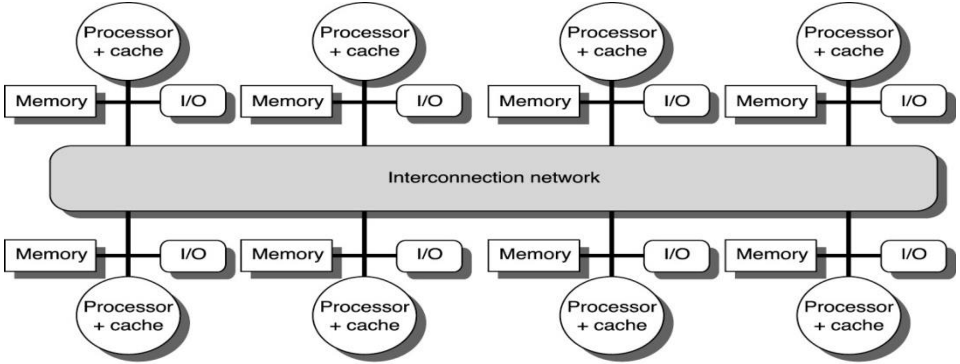


MIMD Multiprocessors

Centralized Shared Memory



Distributed Memory



Centralized-Memory Machines

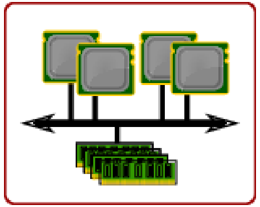
- Also "Symmetric Multiprocessors" (SMP)
- "Uniform Memory Access" (UMA)
  - All memory locations have similar latencies
  - Data sharing through memory reads/writes
  - P1 can write data to a physical address A,
    P2 can then read physical address A to get that data

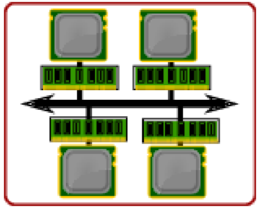
- Problem: Memory Contention
  - All processor share the one memory
  - Memory bandwidth becomes bottleneck
  - Used only for smaller machines
    - Most often 2,4, or 8 processors

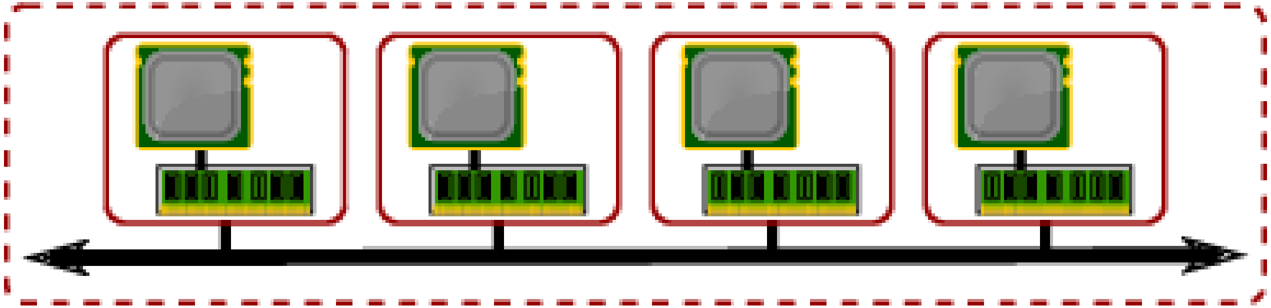
Distributed-Memory Machines

- Two kinds
  - Distributed Shared-Memory (DSM)
    - All processors can address all memory locations
    - Data sharing like in SMP
    - Also called **NUMA** (non-uniform memory access)
    - Latencies of different memory locations can differ (local access faster than remote access)
  - Message-Passing
    - A processor can directly address only local memory
    - To communicate with other processors, must explicitly send/receive messages
    - Also called multicomputers or clusters
- Most accesses local, so less memory contention (can scale to well over 1000 processors)

Message-Passing Machines





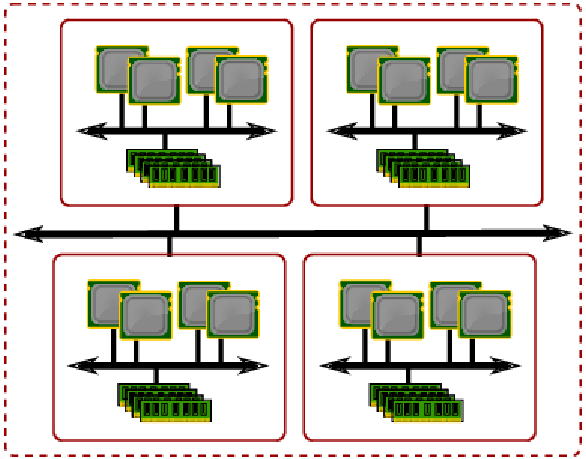


Message-Passing Machines

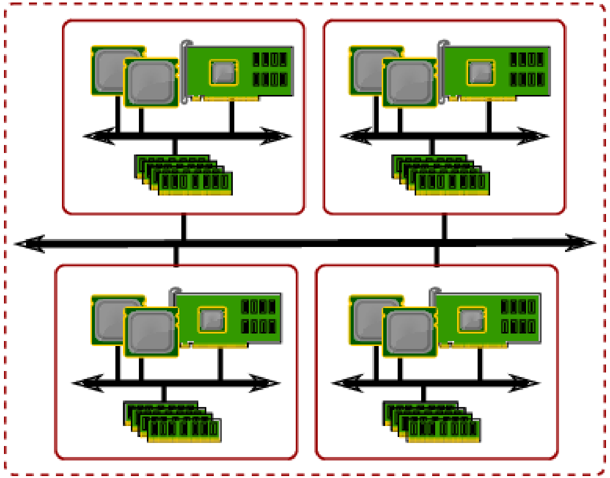
- A cluster of computers
  - Each with its own processor and memory
  - An interconnect to pass messages between them
  - Producer-Consumer Scenario:
    - P1 produces data D, uses a SEND to send it to P2
    - The network routes the message to P2
    - P2 then calls a RECEIVE to get the message
  - Two types of send primitives
    - Synchronous: P1 stops until P2 confirms receipt of message
    - Asynchronous: P1 sends its message and continues
  - Standard libraries for message passing:
    Most common is MPI – Message Passing Interface

Hybrid architectures

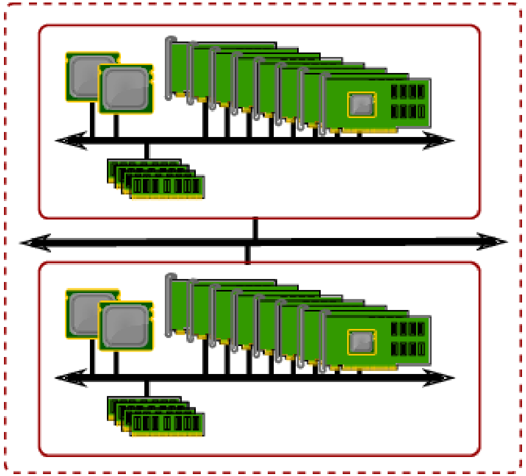
Fat cluster



GPU-accelerated



GPU cluster



Communication Performance

- Metrics for Communication Performance
  - Communication Bandwidth
  - Communication Latency
    - Sender overhead + transfer time + receiver overhead
  - Communication latency hiding
- Characterizing Applications
  - Communication to Computation Ratio
    - Work done vs. bytes sent over network
    - Example: 146 bytes per 1000 instructions

Parallel Performance

- Serial sections
  - Very difficult to parallelize the entire app
  - Amdahl's law

$$Speedup_{Overall} = \frac{1}{(1 - F_{Parallel}) + \frac{F_{Parallel}}{Speedup_{Parallel}}}$$

$$Speedup_{Parallel} = 1024,\quad F_{Parallel} = 0.5 \;\Rightarrow\; Speedup_{Overall} = 1.998$$

$$Speedup_{Parallel} = 1024,\quad F_{Parallel} = 0.99 \;\Rightarrow\; Speedup_{Overall} = 91.2$$

- Large remote access latency (100s of ns)
  - Overall IPC goes down

$$CPI = CPI_{Base} + RemoteRequestRate \times RemoteRequestCost$$

This cost reduced with multi-core

$$CPI_{Base} = 0.4 \qquad RemoteRequestCost = \frac{400ns}{0.33ns/Cycle} = 1200\ Cycles \qquad RemoteRequestRate = 0.002$$

$$CPI = 2.8$$

We need at least 7 processors just to break even!

Message Passing Pros and Cons

- Pros
  - Simpler and cheaper hardware
  - Explicit communication makes programmers aware of costly (communication) operations
- Cons
  - Explicit communication is painful to program
  - Requires manual optimization
    - If you want a variable to be local and accessible via LD/ST, you must declare it as such
    - If other processes need to read or write this variable, you must explicitly code the needed sends and receives to do this

Message Passing: A Program

- Calculating the sum of array elements

```
#define ASIZE 1024
#define NUMPROC 4
double myArray[ASIZE/NUMPROC];
double mySum=0;
for(int i=0;i<ASIZE/NUMPROC;i++)
  mySum+=myArray[i];
if(myPID=0){
  for(int p=1;p<NUMPROC;p++){
    int pSum;
    recv(p,pSum);
    mySum+=pSum;
  }
  printf("Sum: %lf\n",mySum);
}else
  send(0,mySum);
```

Must manually split the array

"Master" processor adds up partial sums and prints the result

"Slave" processors send their partial results to master

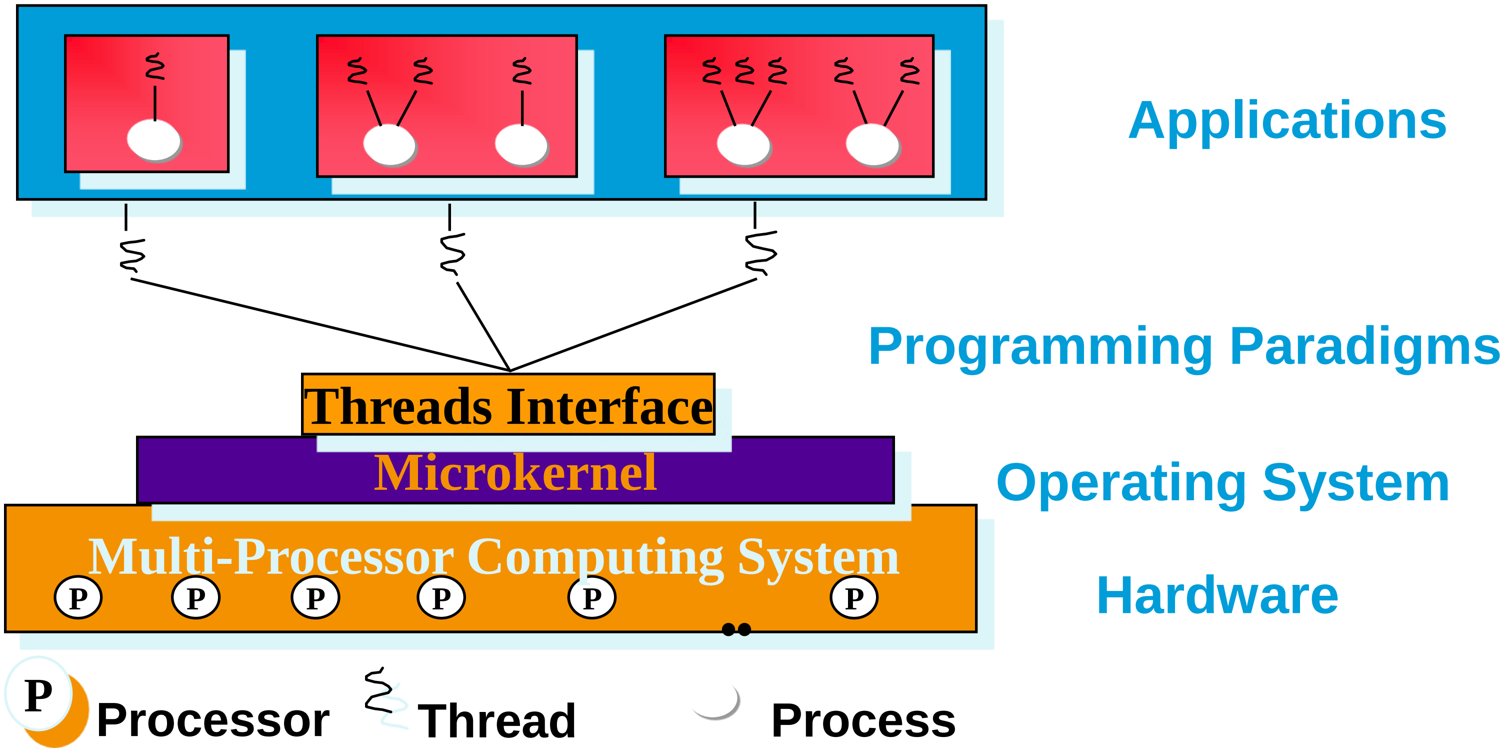
MPI programming example

- https://hpcc.usc.edu/support/documentation/examples-of-mpi-programs

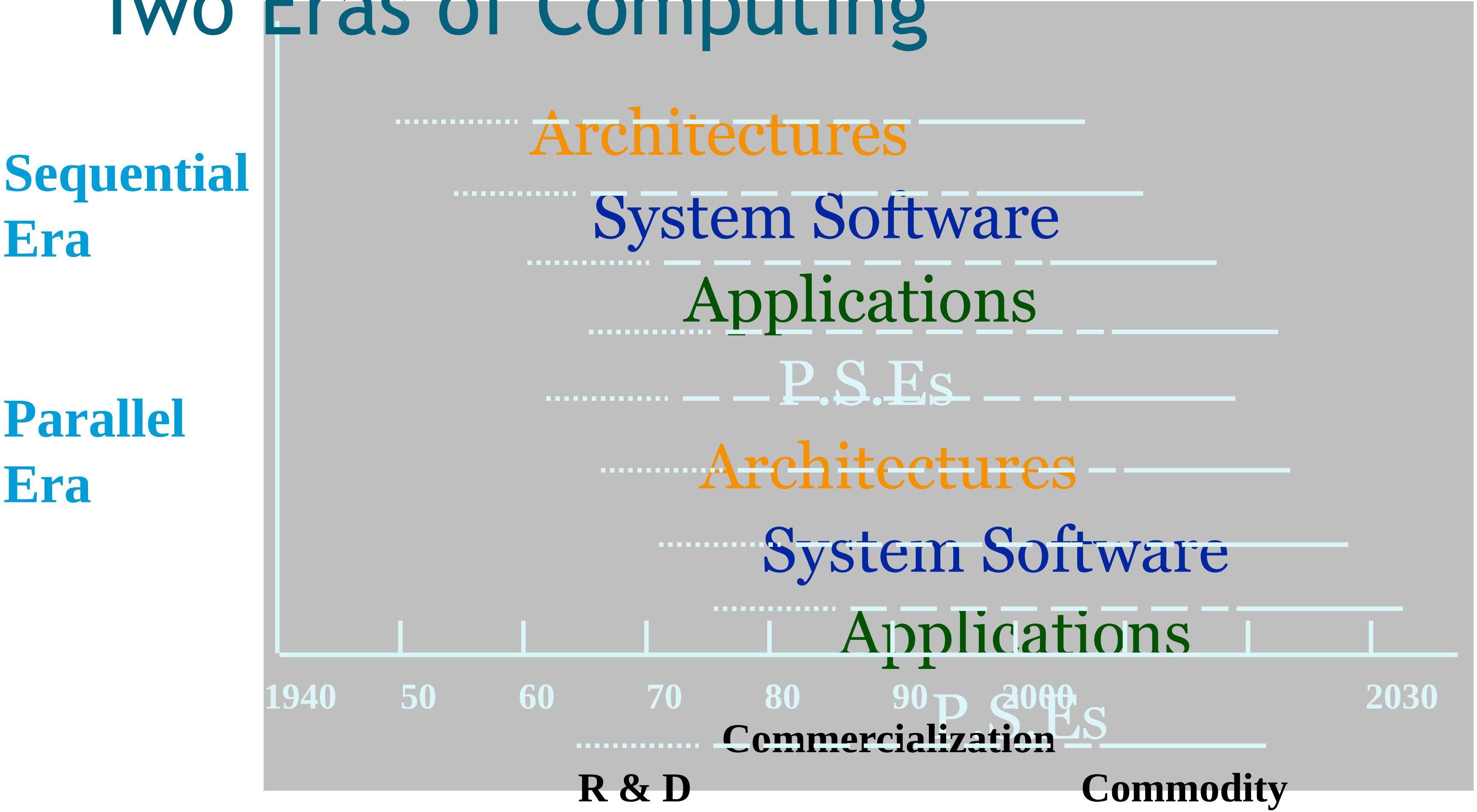
Shared Memory Pros and Cons

- Pros
  - Communication happens automatically
  - More natural way of programming
    - Easier to write correct programs and gradually optimize them
  - No need to manually distribute data
    (but can help if you do)
- Cons
  - Needs more hardware support
  - Easy to write correct, but inefficient programs
    (remote accesses look the same as local ones)

Computing Elements



Two Eras of Computing



High-Performance Computing / Introduction

Source: James R. Knight/Yale Center for Genome Analysis

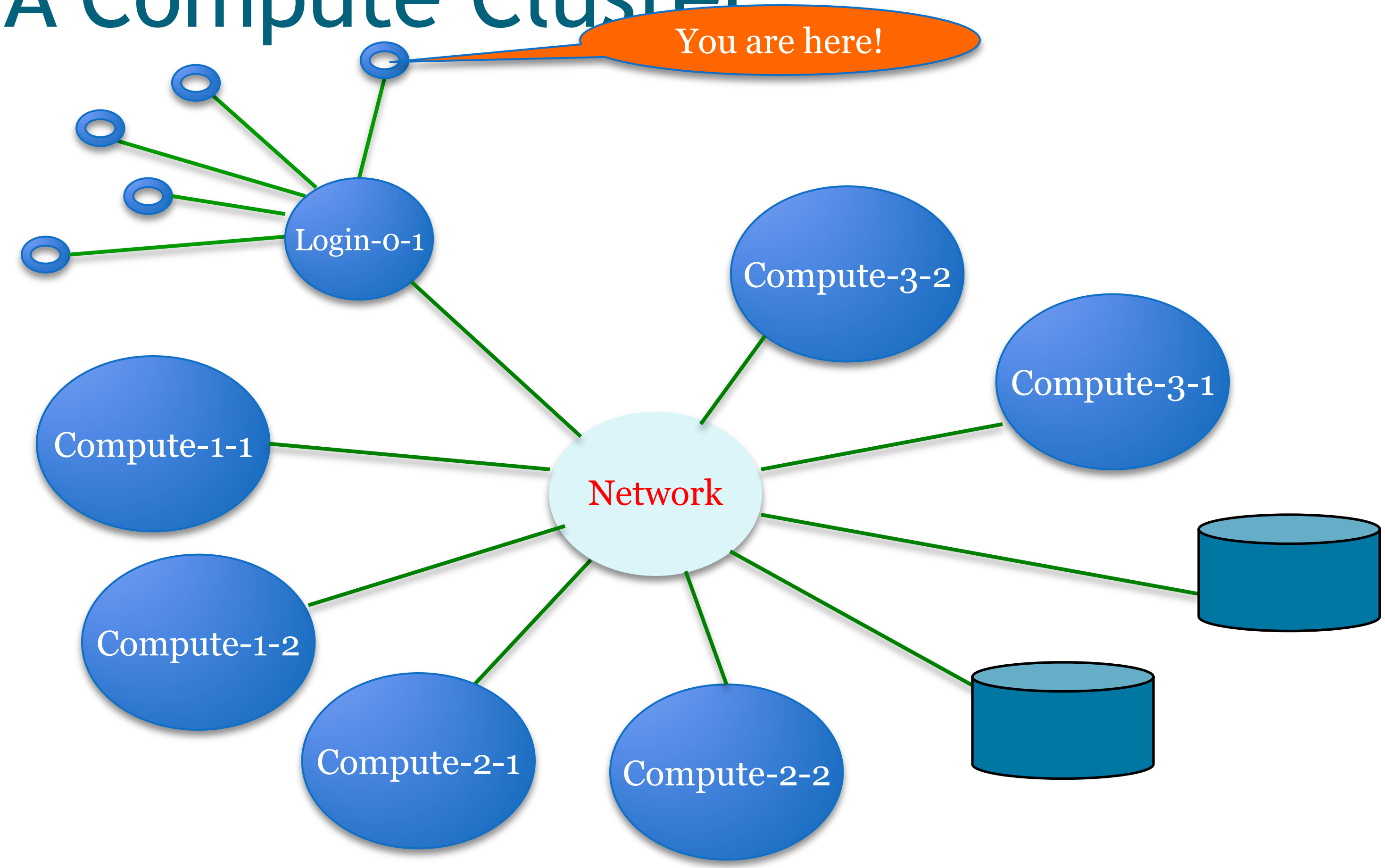
1950’s – The Beginning...

2016 – Looking very similar…

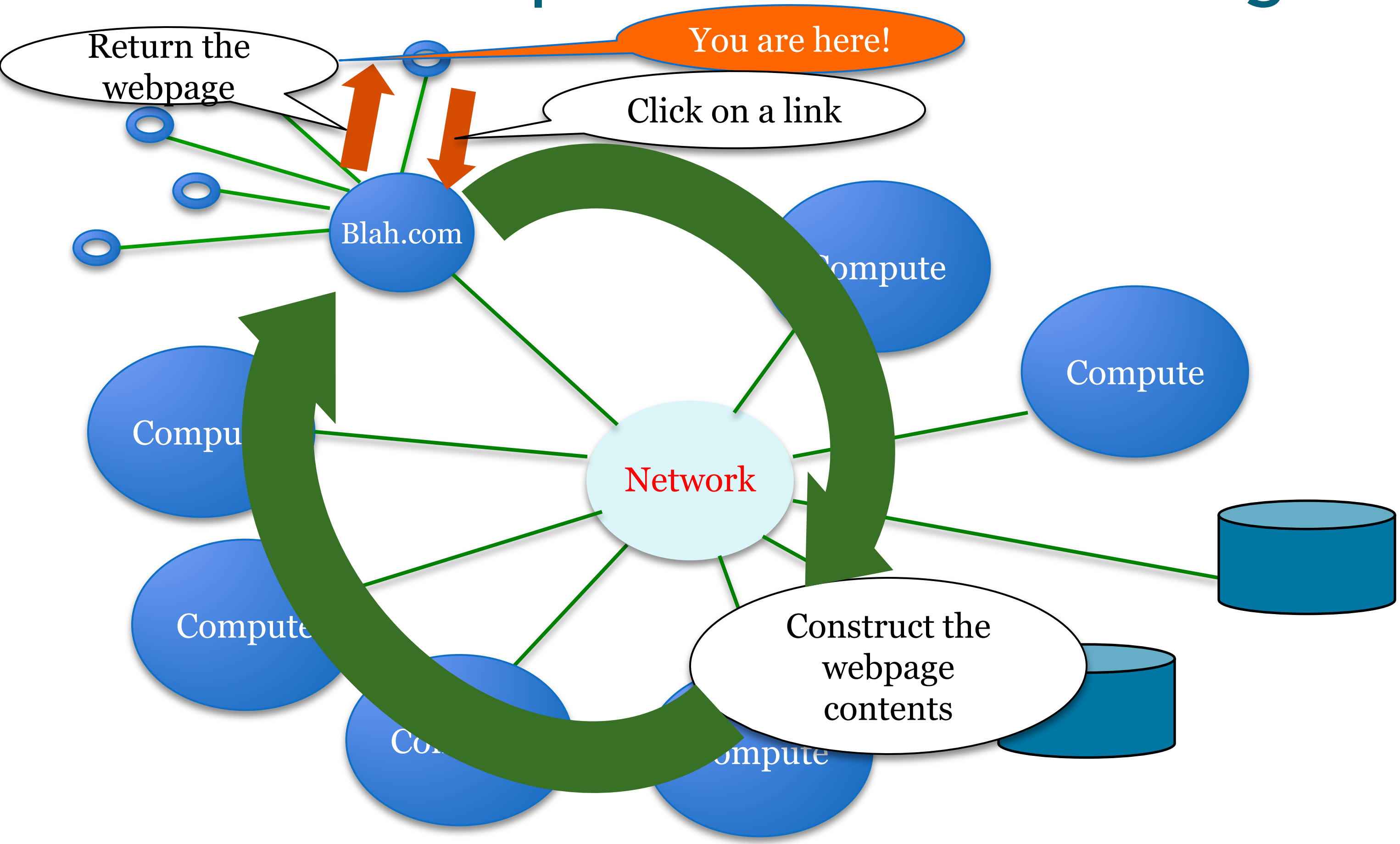
…but there are differences

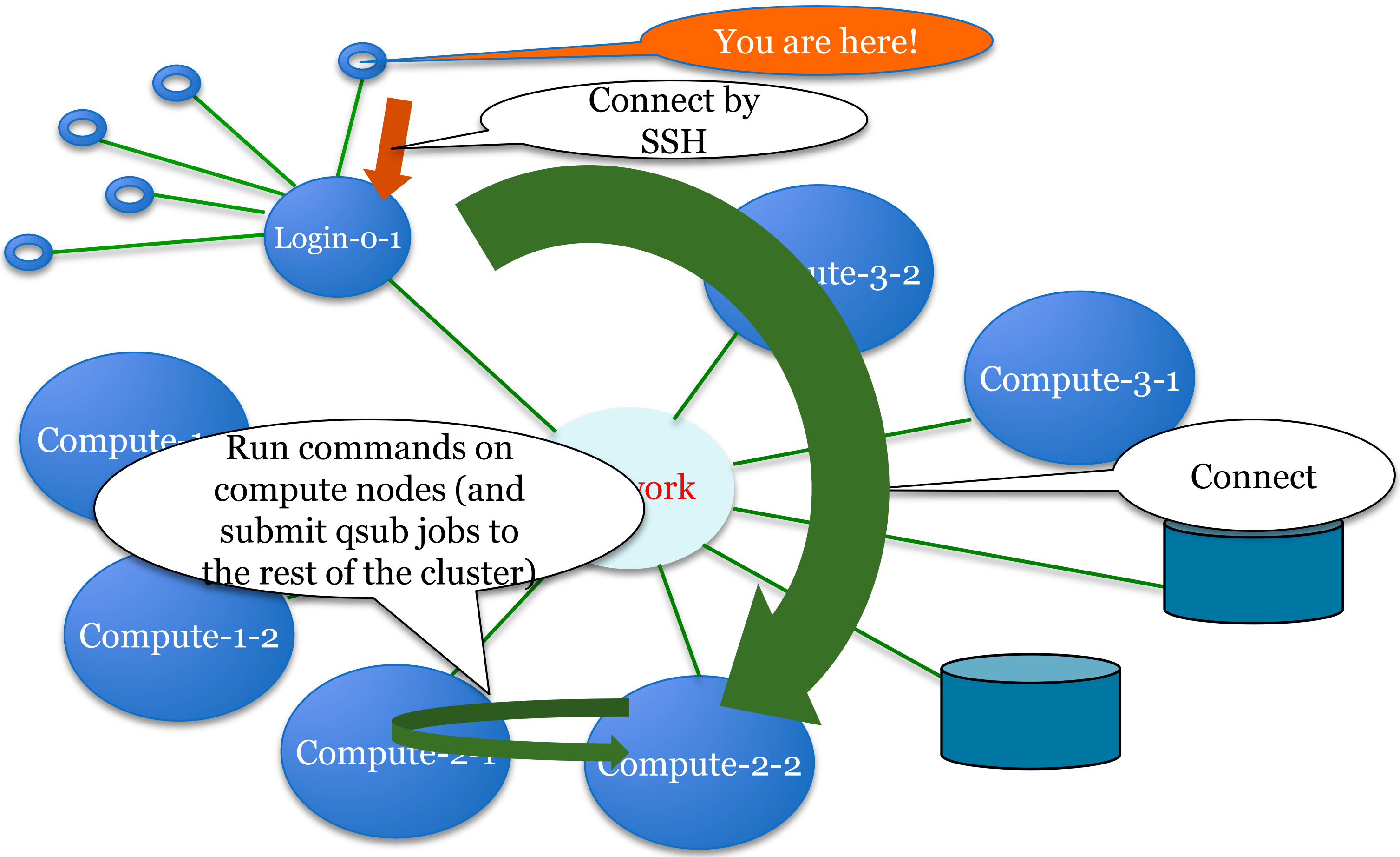
- Not a single computer but thousands of them, called a **cluster**
    - Hundreds of physical "computers", called **nodes**
    - Each with 4-64 CPU's, called **cores**
- Nobody works in the server rooms anymore
    - IT is there to fix what breaks, not to run computations (or help you run computations)
    - Everything is done by remote connections
- Computation is performed by submitting **jobs** for running
    - This actually hasn't changed…but how you run jobs has…

A Compute Cluster



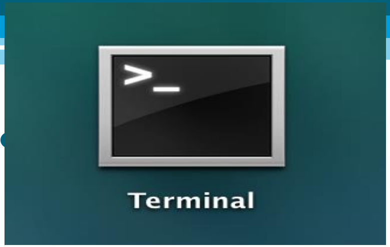
You Use a Compute Cluster! Surfing the Web



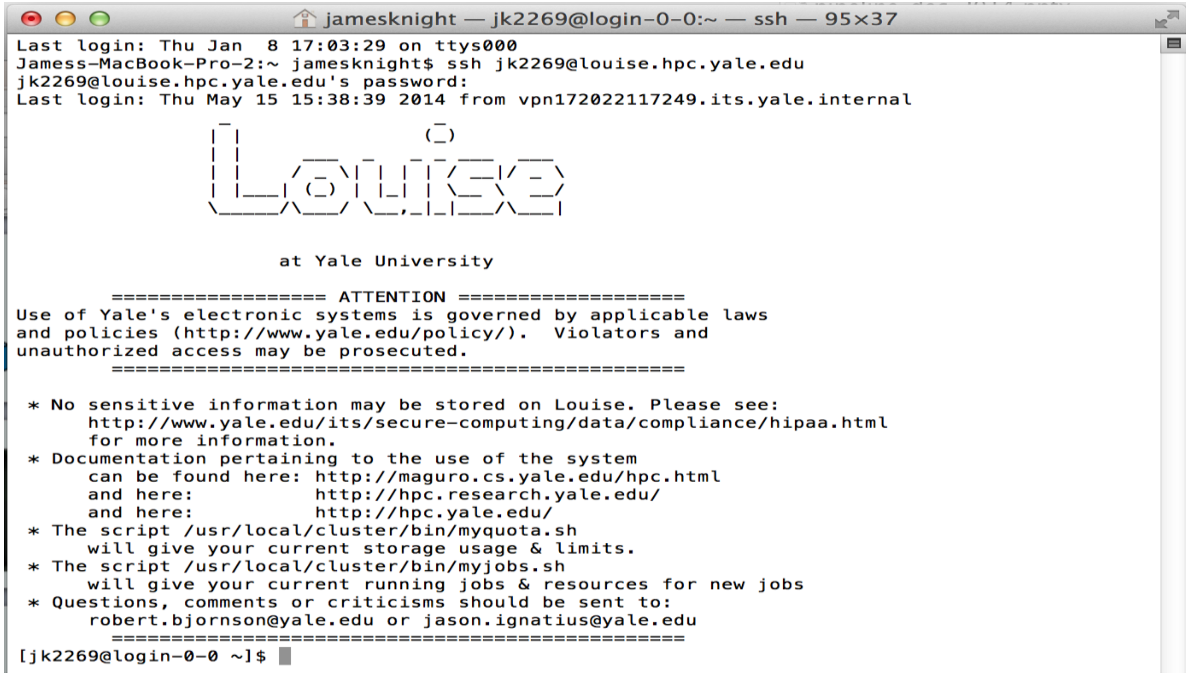

You are here!
Connect by SSH
Login-0-1
ute-3-2
Compute-3-1
Connect
Compute-1
Run commands on compute nodes (and submit qsub jobs to the rest of the cluster)
ork
Compute-1-2
Compute-2-1
Compute-2-2

1970's – Terminals, In the Beginning…

2016 – Pretty much the same.



- Terminal app on Mac
- Look in the “Other” folder in Launchpad



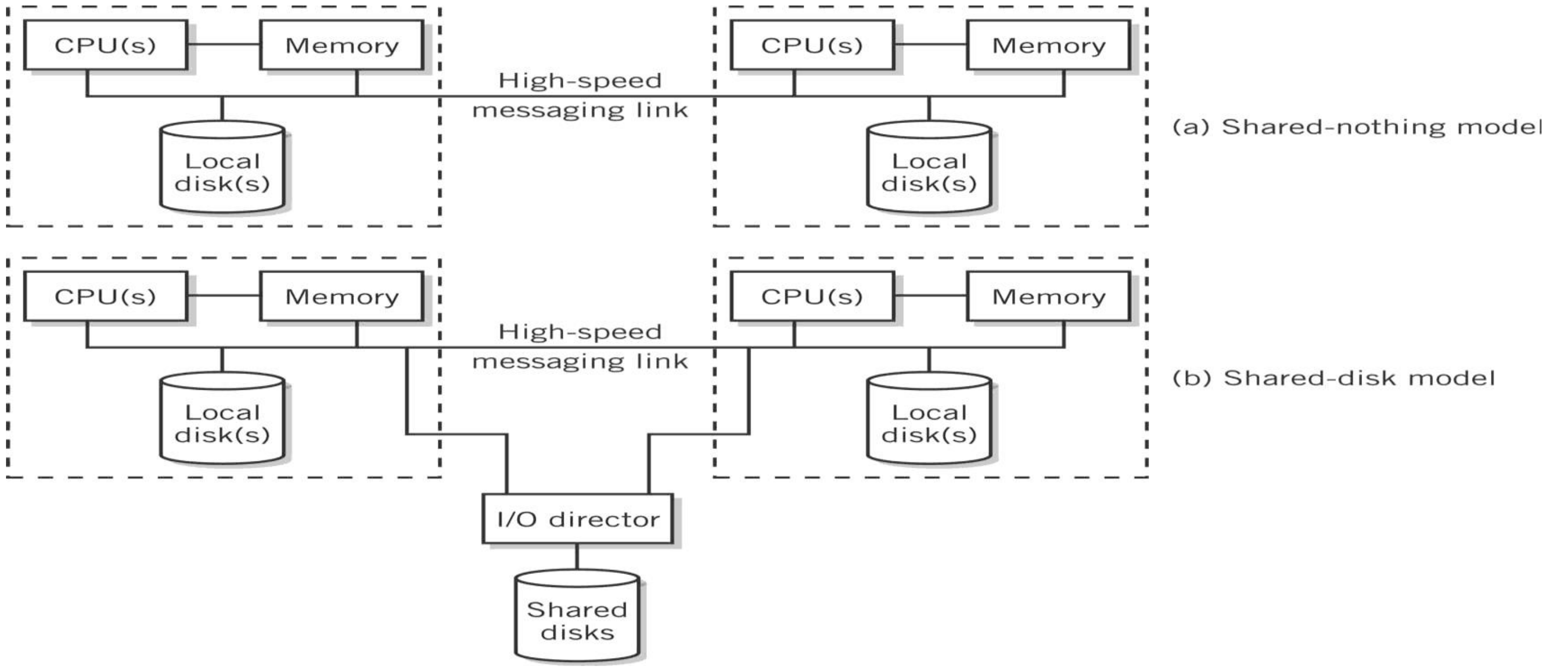
```
jamesknight — jk2269@login-0-0:~ — ssh — 95×37
Last login: Thu Jan  8 17:03:29 on ttys000
Jamess-MacBook-Pro-2:~ jamesknight$ ssh jk2269@louise.hpc.yale.edu
jk2269@louise.hpc.yale.edu's password:
Last login: Thu May 15 15:38:39 2014 from vpn172022117249.its.yale.internal
                 _                 _
                | |               (_)
                | |     ___  _   _ _ ___  ___
                | |    / _ \| | | | / __|/ _ \
                | |___| (_) | |_| | \__ \  __/
                \_____/\___/ \__,_|_|___/\___|

                      at Yale University

        ================== ATTENTION ===================
Use of Yale's electronic systems is governed by applicable laws
and policies (http://www.yale.edu/policy/).  Violators and
unauthorized access may be prosecuted.
        ================================================

 * No sensitive information may be stored on Louise. Please see:
      http://www.yale.edu/its/secure-computing/data/compliance/hipaa.html
      for more information.
 * Documentation pertaining to the use of the system
      can be found here: http://maguro.cs.yale.edu/hpc.html
      and here:          http://hpc.research.yale.edu/
      and here:          http://hpc.yale.edu/
 * The script /usr/local/cluster/bin/myquota.sh
      will give your current storage usage & limits.
 * The script /usr/local/cluster/bin/myjobs.sh
      will give your current running jobs & resources for new jobs
 * Questions, comments or criticisms should be sent to:
      robert.bjornson@yale.edu or jason.ignatius@yale.edu
        ================================================
[jk2269@login-0-0 ~]$
```

Cluster Models



Beowulf Clusters

- Simple and highly configurable
- Low cost
- Networked
  - Computers connected to one another by a private Ethernet network
  - Connection to an external network is through a single gateway computer
- Configuration
  - COTS – Commodity-off-the-shelf components such as inexpensive computers
  - Blade components – computers mounted on a motherboard that are plugged into connectors on a rack
  - Either shared-disk or shared-nothing model

Blade and Rack of Beowulf Cluster

Cluster computing concept

Cluster Computing - Research Projects

- Beowulf (CalTech and NASA) - USA
- CCS (Computing Centre Software) - Paderborn, Germany
- Condor - Wisconsin State University, USA
- DQS (Distributed Queuing System) - Florida State University, US.
- EASY - Argonne National Lab, USA
- HPVM -(High Performance Virtual Machine),UIUC&now UCSB,US
- *far* - University of Liverpool, UK
- Gardens - Queensland University of Technology, Australia
- MOSIX - Hebrew University of Jerusalem, Israel
- MPI (MPI Forum, MPICH is one of the popular implementations)
- NOW (Network of Workstations) - Berkeley, USA
- NIMROD - Monash University, Australia
- NetSolve - University of Tennessee, USA
- PBS (Portable Batch System) - NASA Ames and LLNL, USA
- PVM - Oak Ridge National Lab./UTK/Emory, USA

Cluster Computing - Commercial Software

- Codine (Computing in Distributed Network Environment) - GENIAS GmbH, Germany
- LoadLeveler - IBM Corp., USA
- LSF (Load Sharing Facility) - Platform Computing, Canada
- NQE (Network Queuing Environment) - Craysoft Corp., USA
- OpenFrame - Centre for Development of Advanced Computing, India
- RWPC (Real World Computing Partnership), Japan
- Unixware (SCO-Santa Cruz Operations,), USA
- Solaris-MC (Sun Microsystems), USA
- ClusterTools (A number for free HPC clusters tools from Sun)
- A number of commercial vendors worldwide are offering clustering solutions including IBM, Compaq, Microsoft, a number of startups like TurboLinux, HPTI, Scali, BlackStone.....)

Motivation for using Clusters

- Surveys show utilisation of CPU cycles of desktop workstations is typically <10%.
- Performance of workstations and PCs is rapidly improving
- As performance grows, percent utilisation will decrease even further!
- Organisations are reluctant to buy large supercomputers, due to the large expense and short useful life span.

Motivation for using Clusters

- <u>The development tools</u> for workstations are more mature than the contrasting proprietary solutions for parallel computers - mainly due to the non-standard nature of many parallel systems.
- <u>Workstation clusters are a cheap</u> and readily available alternative to specialised High Performance Computing (HPC) platforms.
- Use of clusters of workstations as a distributed compute resource is very cost effective - incremental growth of system!!!

Cycle Stealing

- Usually a workstation will be *owned* by an individual, group, department, or organisation - they are dedicated to the exclusive use by the *owners*.
- This brings problems when attempting to form a cluster of workstations for running distributed applications.

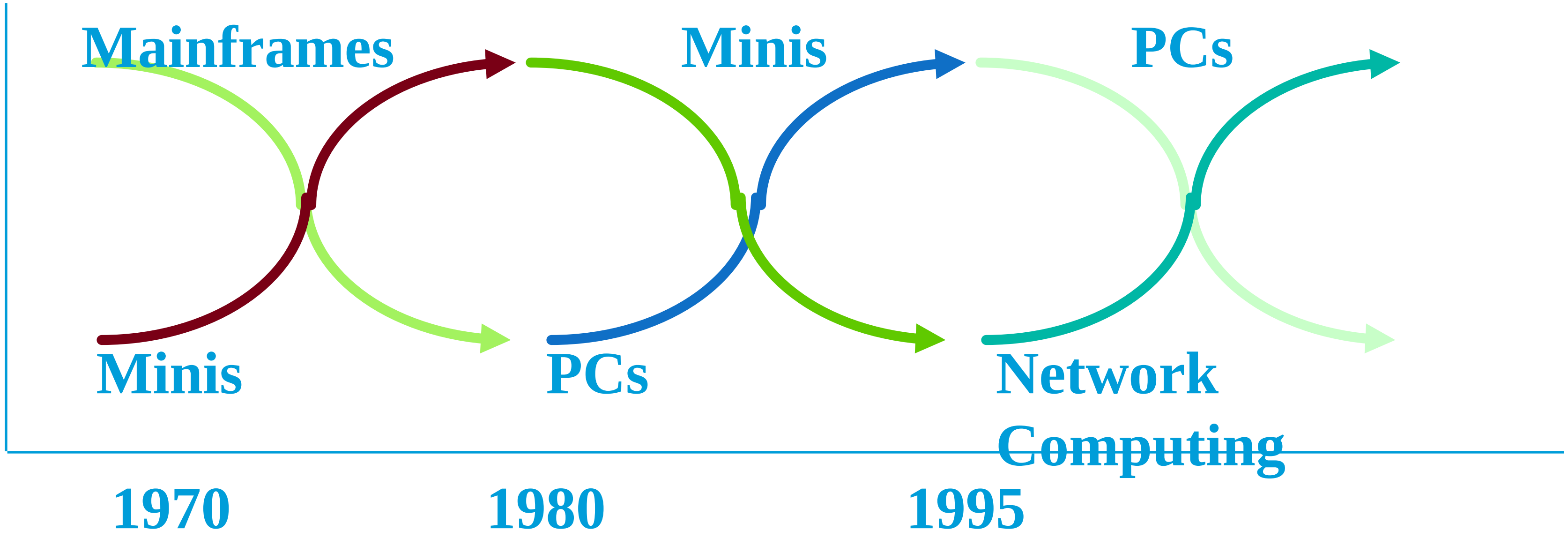
Cycle Stealing

- Typically, there are three types of owners, who use their workstations mostly for:
  1. Sending and receiving email and preparing documents.
  2. Software development - edit, compile, debug and test cycle.
  3. Running compute-intensive applications.

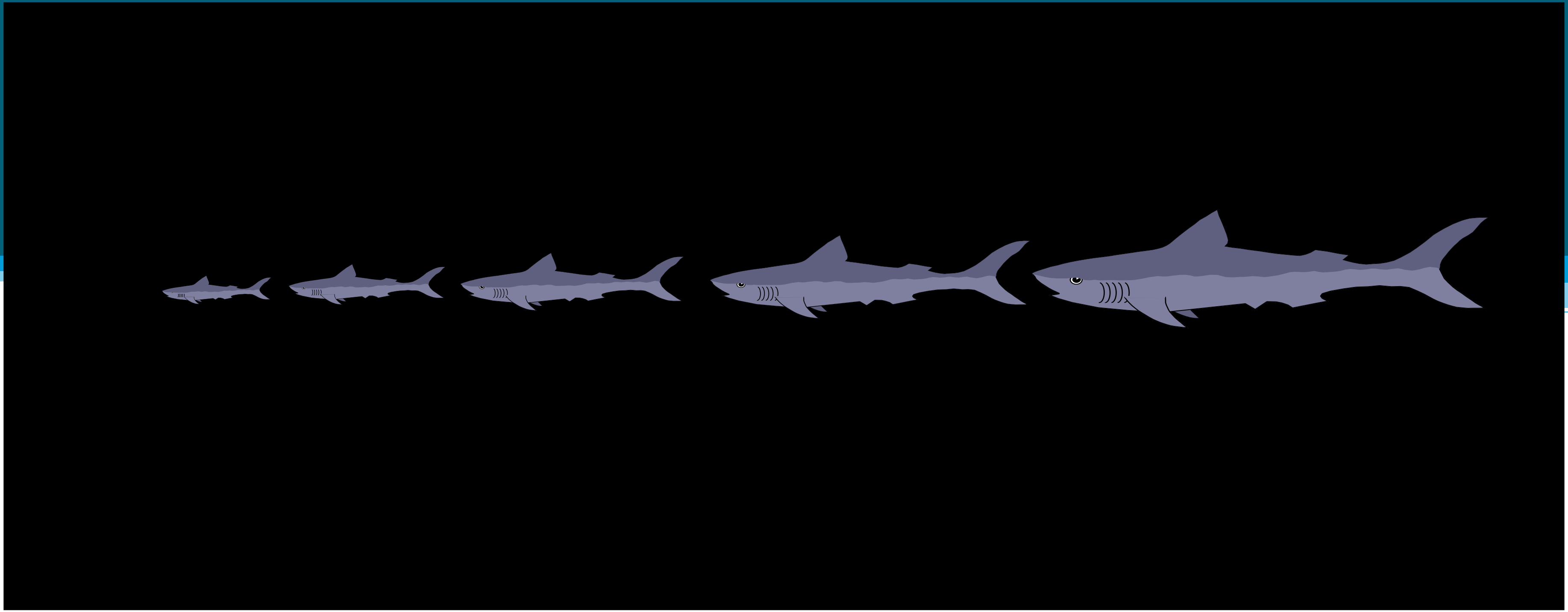
Cycle Stealing

- Cluster computing aims to steal spare cycles from (1) and (2) to provide resources for (3).
- However, this requires overcoming the *ownership hurdle* - people are very protective of *their* workstations.
- Usually requires organisational mandate that computers are to be used in this way.
- Stealing cycles outside standard work hours (e.g. overnight) is easy, stealing idle cycles during work hours without impacting interactive use (both CPU and memory) is much harder.

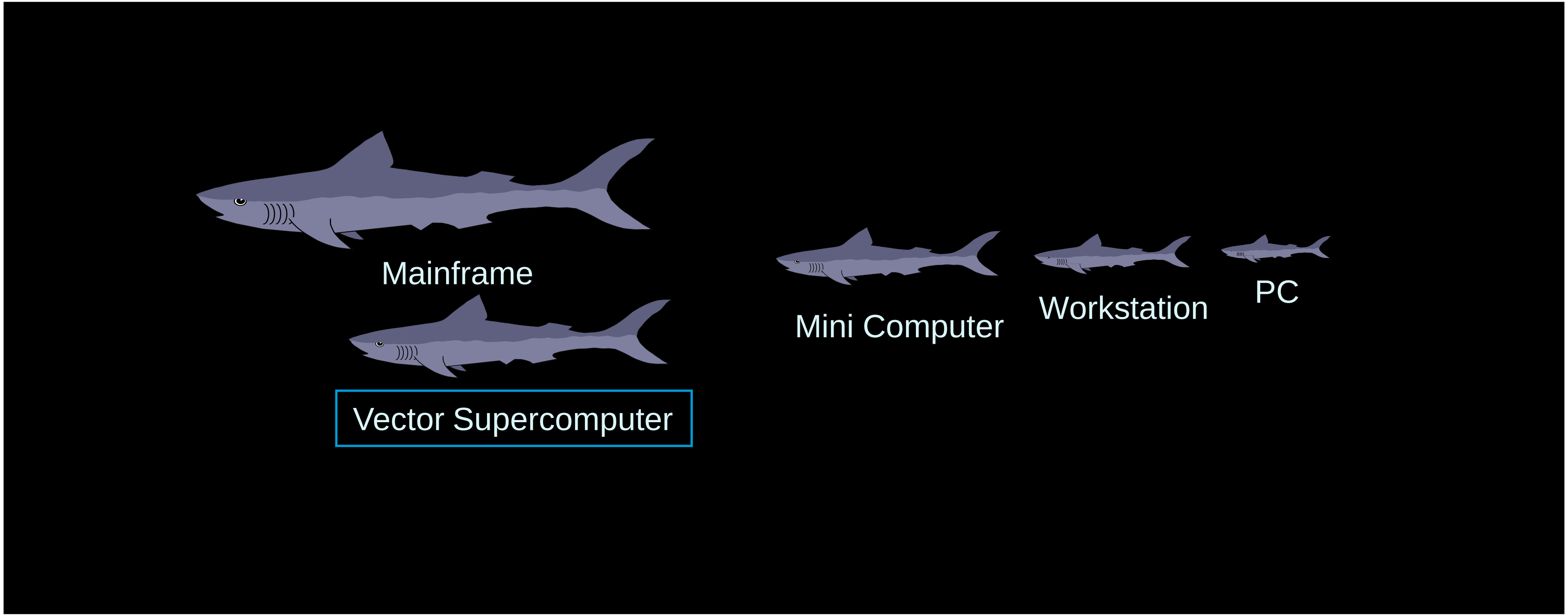
Rise & Fall of Computing Technologies



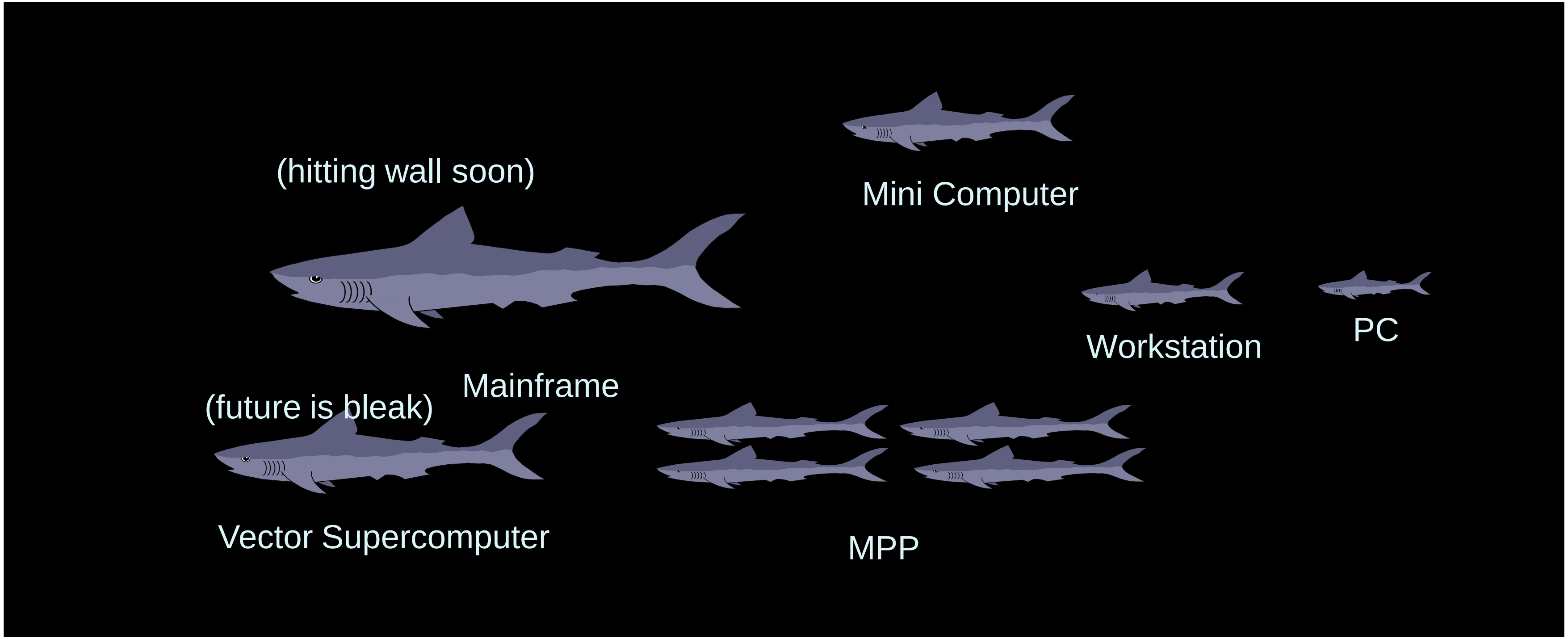
Original Food Chain Picture



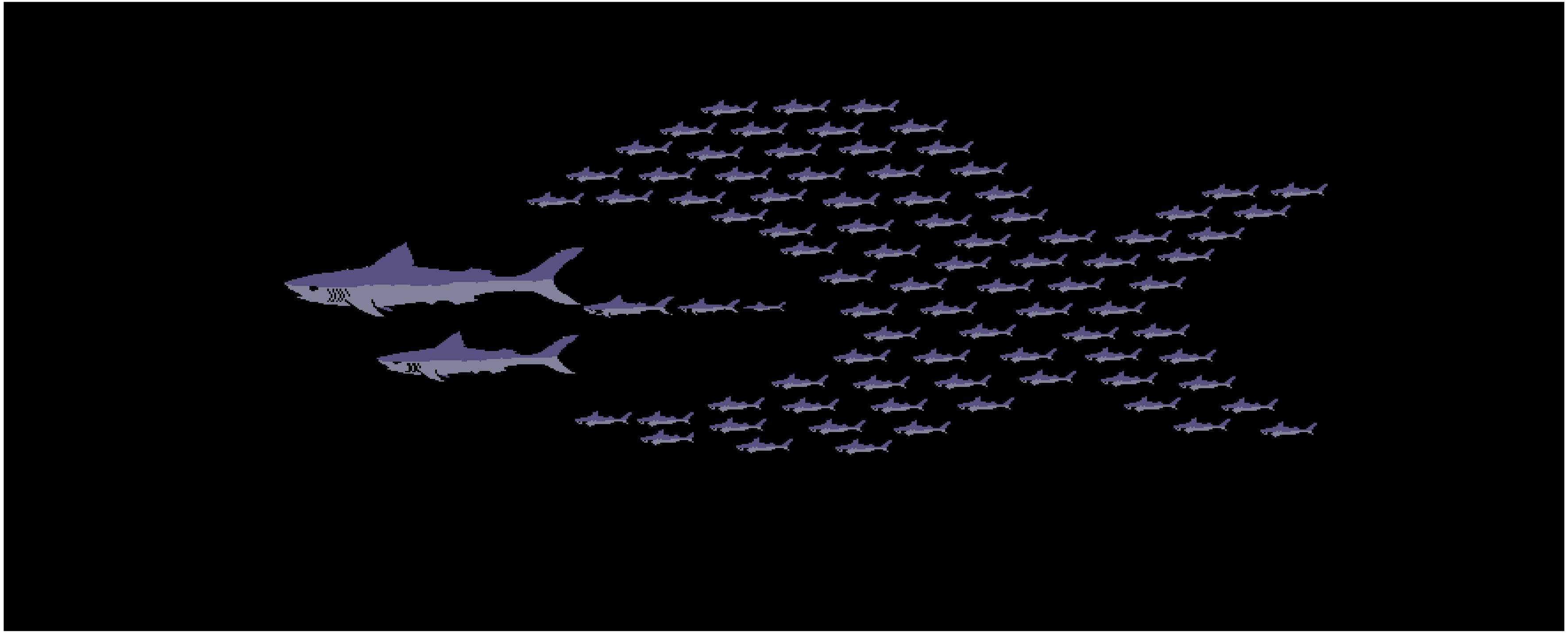
1984 Computer Food Chain



1994 Computer Food Chain



Computer Food Chain (Now and Future)



What is a cluster?

- A cluster is a type of parallel or distributed processing system, which consists of a collection of interconnected stand-alone/complete computers cooperatively working together as a single, integrated computing resource.
- A typical cluster:
  - Network: Faster, closer connection than a typical network (LAN)
  - Low latency communication protocols
  - Looser connection than SMP

Why Clusters now? (Beyond Technology and Cost)

- Building block is big enough
  - complete computers (HW & SW) shipped in millions: killer micro, killer RAM, killer disks, killer OS, killer networks, killer apps.
- Workstations performance is doubling every 18 months.
- Networks are faster
- Higher link bandwidth (v 10Mbit Ethernet)
  - Switch based networks coming (ATM)
  - Interfaces simple & fast (Active Msgs)
- Striped files preferred (RAID)
- Demise of Mainframes, Supercomputers, & MPPs

Architectural Drivers…(cont)

- Node architecture dominates performance
  - processor, cache, bus, and memory
  - design and engineering $ => performance
- Greatest demand for performance is on large systems
  - must track the leading edge of technology without lag
- MPP network technology => mainstream
  - system area networks
- System on every node is a powerful enabler
  - very high speed I/O, virtual memory, scheduling, …

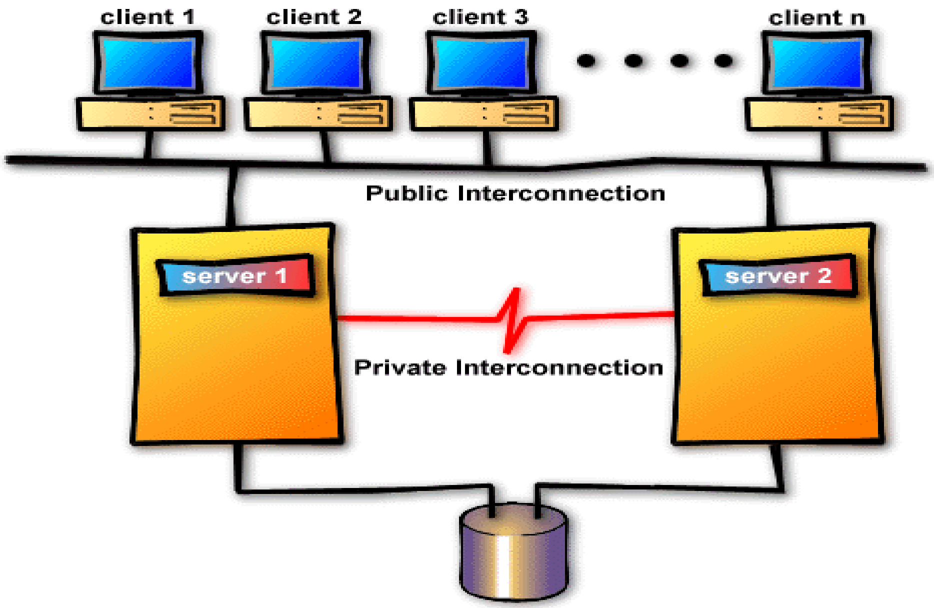
…Architectural Drivers

- Clusters can be grown: Incremental scalability (up, down, and across)
  - Individual nodes performance can be improved by adding additional resource (new memory blocks/disks)
  - New nodes can be added or nodes can be removed
  - Clusters of Clusters and Metacomputing
- Complete software tools
  - Threads, PVM, MPI, DSM, C, C++, Java, Parallel C++, Compilers, Debuggers, OS, etc.
- Wide class of applications
  - Sequential and grand challenging parallel applications

Example Clusters: Berkeley

- 100 Sun UltraSparcs
  - 200 disks
- Myrinet SAN
  - 160 MB/s
- Fast comm.
  - AM, MPI, ...
- Ether/ATM switched external net
- Global OS
- Self Config

HA Cluster: Server Cluster with "Heartbeat" Connection



Jobs

Distributed Supercomputing

- Combining multiple high-capacity resources on a computational grid into a single, virtual distributed supercomputer.
- Tackle problems that cannot be solved on a single system.

High-Throughput Computing

- Uses the grid to schedule large numbers of loosely coupled or independent tasks, with the goal of putting unused processor cycles to work.

On-Demand Computing

- Uses grid capabilities to meet short-term requirements for resources that are not locally accessible.
- Models real-time computing demands.

Collaborative Computing

- Concerned primarily with enabling and enhancing human-to-human interactions.
- Applications are often structured in terms of a virtual shared space.

Data-Intensive Computing

- The focus is on synthesizing new information from data that is maintained in geographically distributed repositories, digital libraries, and databases.
- Particularly useful for distributed data mining.

Logistical Networking

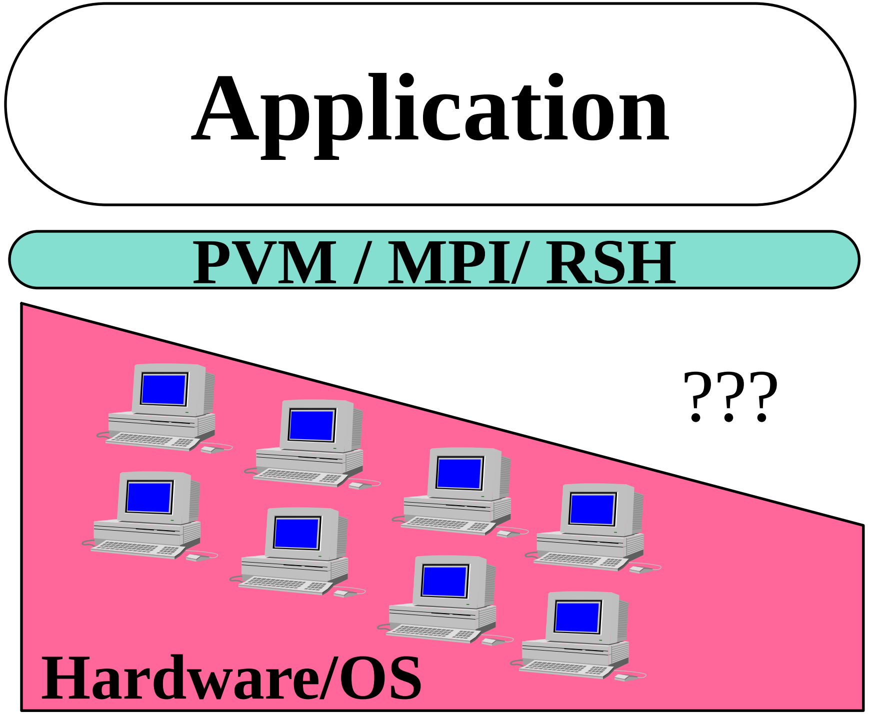
- Logistical networks focus on exposing storage resources inside networks by optimizing the global scheduling of data transport, and data storage.
- Contrasts with traditional networking, which does not explicitly model storage resources in the network.
- high-level services for Grid applications
- Called "logistical" because of the analogy it bears with the systems of warehouses, depots, and distribution channels.

P2P Computing vs Grid Computing

- Differ in Target Communities
- Grid system deals with more complex, more powerful, more diverse and highly interconnected set of resources than P2P.
- VO

Cluster Work Schedulers

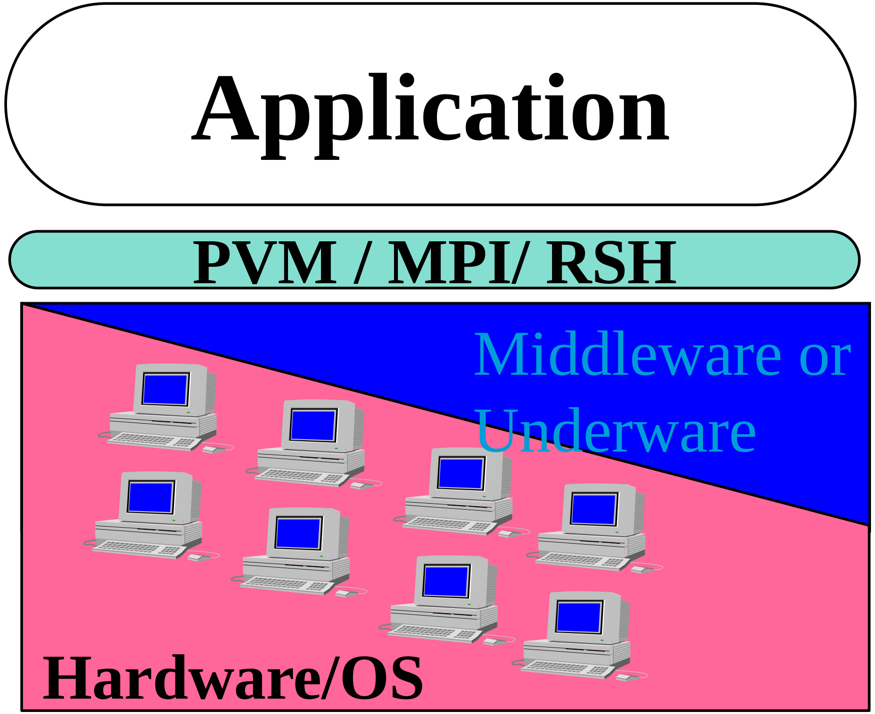
A typical Cluster Computing Environment



CC should support

- Multi-user, time-sharing environments
- Nodes with different CPU speeds and memory sizes (heterogeneous configuration)
- Many processes, with unpredictable requirements
- Unlike SMP: insufficient "bonds" between nodes
  - Each computer operates independently
  - Inefficient utilization of resources

The missing link is provide by cluster middleware/underware



SSI Clusters--SMP services on a CC

“*Pool Together*” the “*Cluster-Wide*” resources

- Adaptive resource usage for better performance
- Ease of use - almost like SMP
- Scalable configurations - by decentralized control

Result: *HPC/HAC at PC/Workstation prices*

What is Cluster Middleware ?

- An interface between between use applications and cluster hardware and OS platform.
- Middleware packages support each other at the management, programming, and implementation levels.
- Middleware Layers:
  - SSI Layer
  - Availability Layer: It enables the cluster services of
    - Checkpointing, Automatic Failover, recovery from failure,
    - fault-tolerant operating among all cluster nodes.

Middleware Design Goals

- Complete Transparency (Manageability)
  - Lets the see a single cluster system..
    - Single entry point, ftp, telnet, software loading...
- Scalable Performance
  - Easy growth of cluster
    - no change of API & automatic load distribution.
- Enhanced Availability
  - Automatic Recovery from failures
    - Employ checkpointing & fault tolerant technologies
  - Handle consistency of data when replicated..

Work schedulers - requirements

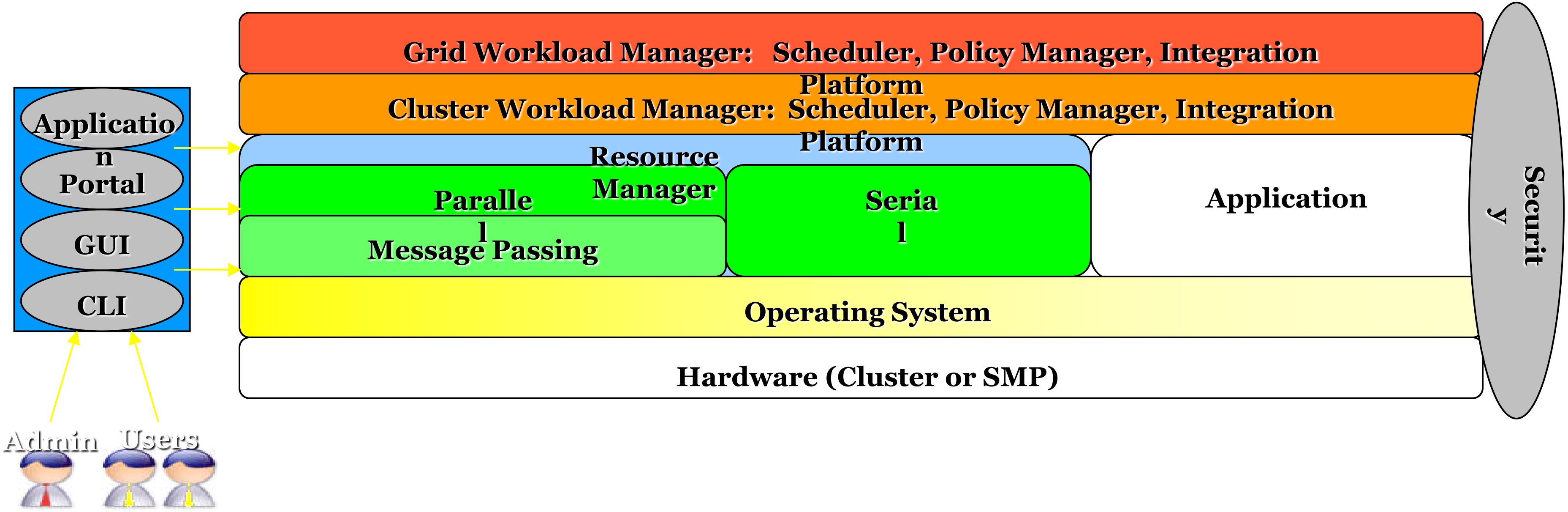
- Interactive or batch

- Stable
- Robust
- Efficient resource management
- Lightweigth
- Fair
- Avoids starvation

- SGE - Sun Grid Engine (Oracle Grid Engine, Open Grid Scheduler)
- SLURM (Simple Linux Utility for Resource Management)
- MOAB + Torque
- HTCondor
- …

Redirect: MOAB

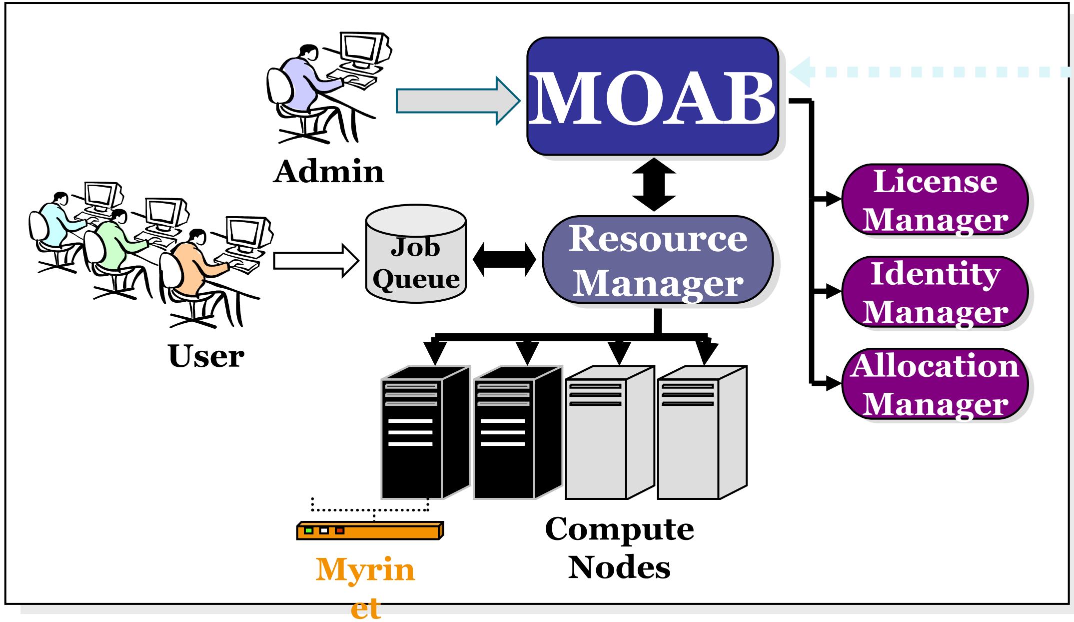
Cluster Stack / Framework:

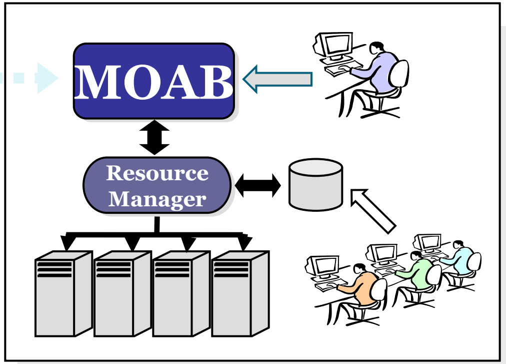


Resource Manager (RM)

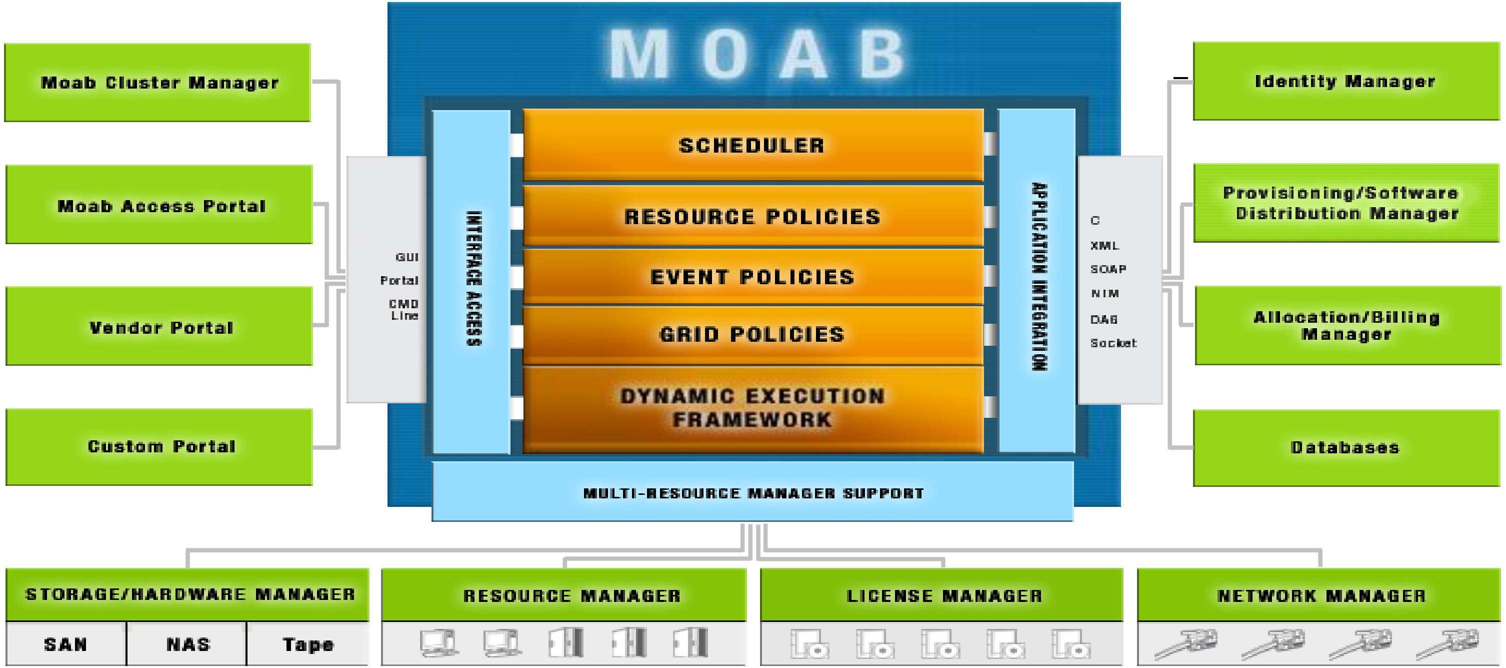
- While other systems may have more strict interpretations of a resource manager and its responsibilities, Moab's *multi-resource manager* support allows a much more liberal interpretation.
  - In essence, any object which can provide environmental information and environmental control can be utilized as a resource manager.
- Moab is able to aggregate information from multiple unrelated sources into a larger more complete *world view* of the cluster which includes all the information and control found within a standard resource manager such as TORQUE including:
  - Node
  - Job
  - Queue management services.

The Evolved Cluster





Moab Architecture



What Moab Does

- Optimizes Resource Utilization with Intelligent Scheduling and Advanced Reservations
- Unifies Cluster Management across Varied Resources and Services
- Dynamically Adjusts Workload to Enforce Policies and Service Level Agreements
- Automates Diagnosis and Failure Response

What Moab Does Not Do

- Does not does do resource management (usually)
- Does not install the system (usually)
- Not a storage manager
- Not a license manager
- Does not do message passing

Supported Platforms/Environments

- Resource Managers
  - TORQUE, OpenPBS, PBSPro, LSF, Loadleveler, SLURM, BProc, clubMASK, S3, WIKI
- Operating Systems
  - RedHat, SUSE, Fedora, Debian, FreeBSD, (+ all known variants of Linux), AIX, IRIX, HP-UX, OS/X, OSF/Tru-64, SunOS, Solaris, (+ all known variants of UNIX)
- Hardware
  - Intel x86, Intel IA-32, Intel IA-64, AMD x86, AMD Opteron, SGI Altix, HP, IBM SP, IBM x-Series, IBM p-Series, IBM i-Series, Mac G4 and G5

Redirect: SLURM

https://www.open-mpi.org/video/slurm/Slurm_EMC_Dec2012.pdf

Role of SLURM resource manger

- The “glue” for a parallel computer to execute parallel jobs
- It should make a parallel computer as almost easy to use as a PC

On a PC.	On a cluster.
Execute program “a.out”:	Execute 8 copies of “a.out”:
a.out	srun -n8 a.out

- MPI would typically be used to manage communications within the parallel program

Role of SLURM resource manger

- Allocate resources within a cluster
  - Nodes (typically a unique IP address)
    - NUMA boards
      - Sockets
        - Cores
          - Hyperthreads
      - Memory
    - Interconnect/switch resources
    - Generic resources (e.g. GPUs)
  - Licenses
- Launch and otherwise manage jobs

Can require extensive knowledge about the hardware and system software (e.g. to alter network routing or manage switch window)

SLURM in a glance

- Simple Linux Utility for Resource Management
- Development started in 2002 at Lawrence Livermore National Laboratory as a simple resource manager for Linux clusters
- Simple Linux Utility for Resource Management, used in many large computers
- Small and simple (depends upon configuration, used by Intel for their “cluster on a chip”)
- Highly scalable (managing 1.6 million core IBM BlueGene/Q, tested to 33 million cores using emulation)
- Fast (throughput up to 600 jobs per second and up to 1000 job submissions per second)
- No kernel modifications

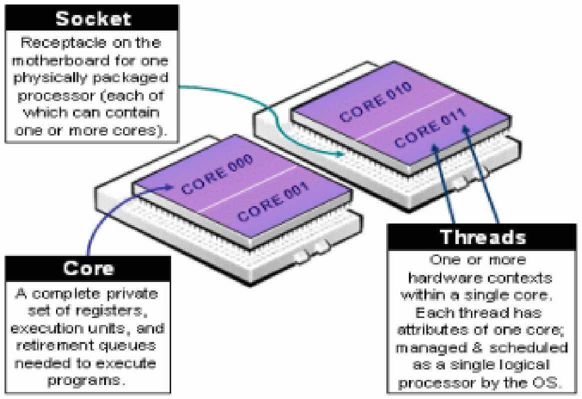
SLURM modularity

- Dynamically linked objects loaded at run time based upon configuration file and/or user options
- 80 plugins of 20 different varieties currently available
  - Accounting storage: MySQL, PostgreSQL, text file
  - Network topology: 3D-torus, tree
  - MPI: OpenMPI, MPICH1, MVAPICH, MPICH2, etc.

SLURM Kernel				
Authentication Plugin	MPI Plugin	Checkpoint Plugin	Topology Plugin	Accounting Storage Plugin
Munge	mvapich	BLCR	Tree	MySQL

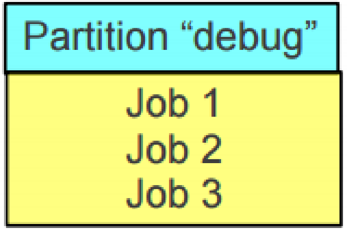
SLURM Entities

- Jobs: Resource allocation requests
- Job steps: Set of (typically parallel) tasks
- Partitions: Job queues with limits and access controls
- Nodes
  - NUMA boards
    - Sockets
      - Cores
        - Hyperthreads
    - Memory
    - Generic Resources (e.g. GPUs)



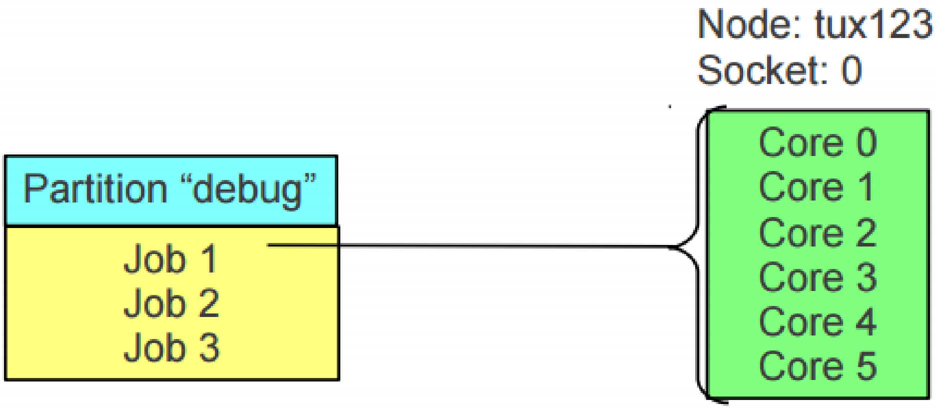
SLURM Entities Example

- Users submit jobs to a partition (queue)



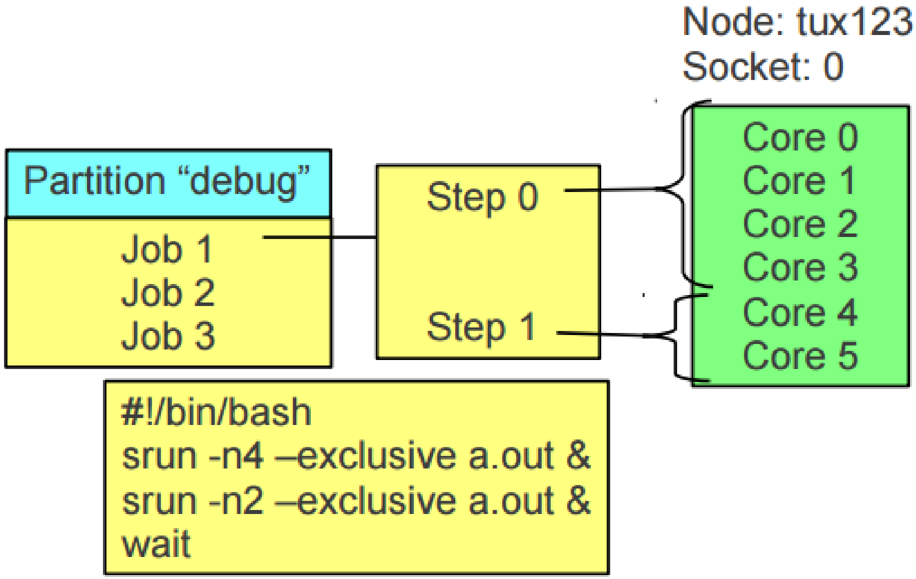
SLURM Entities Example

- Jobs are allocated resources

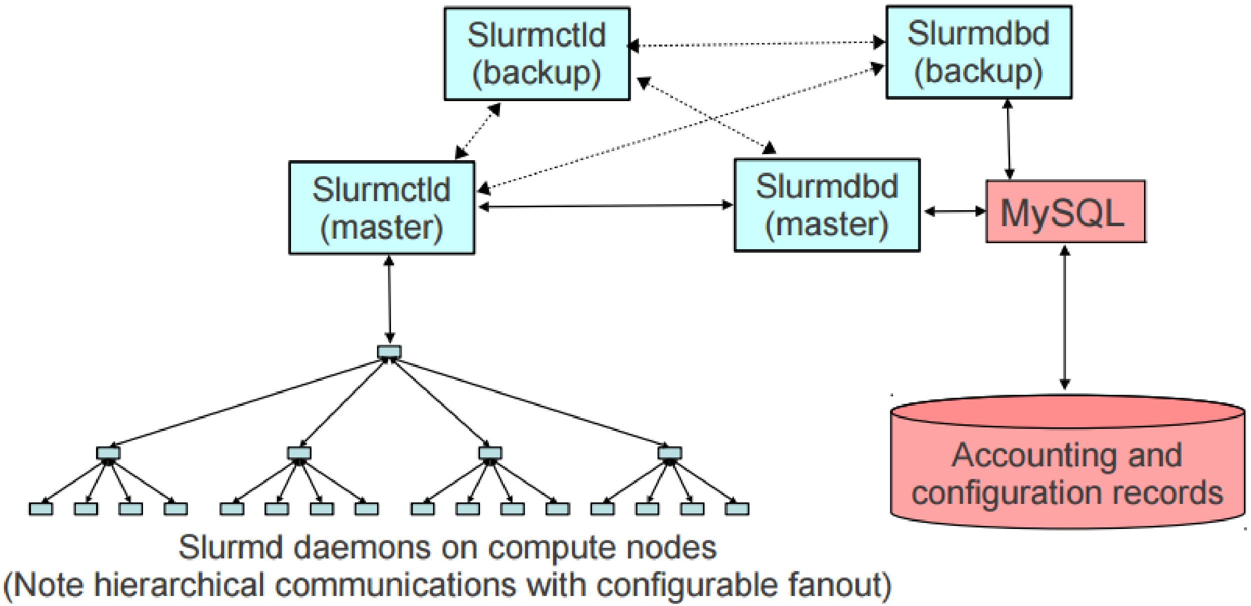


SLURM Entities Example

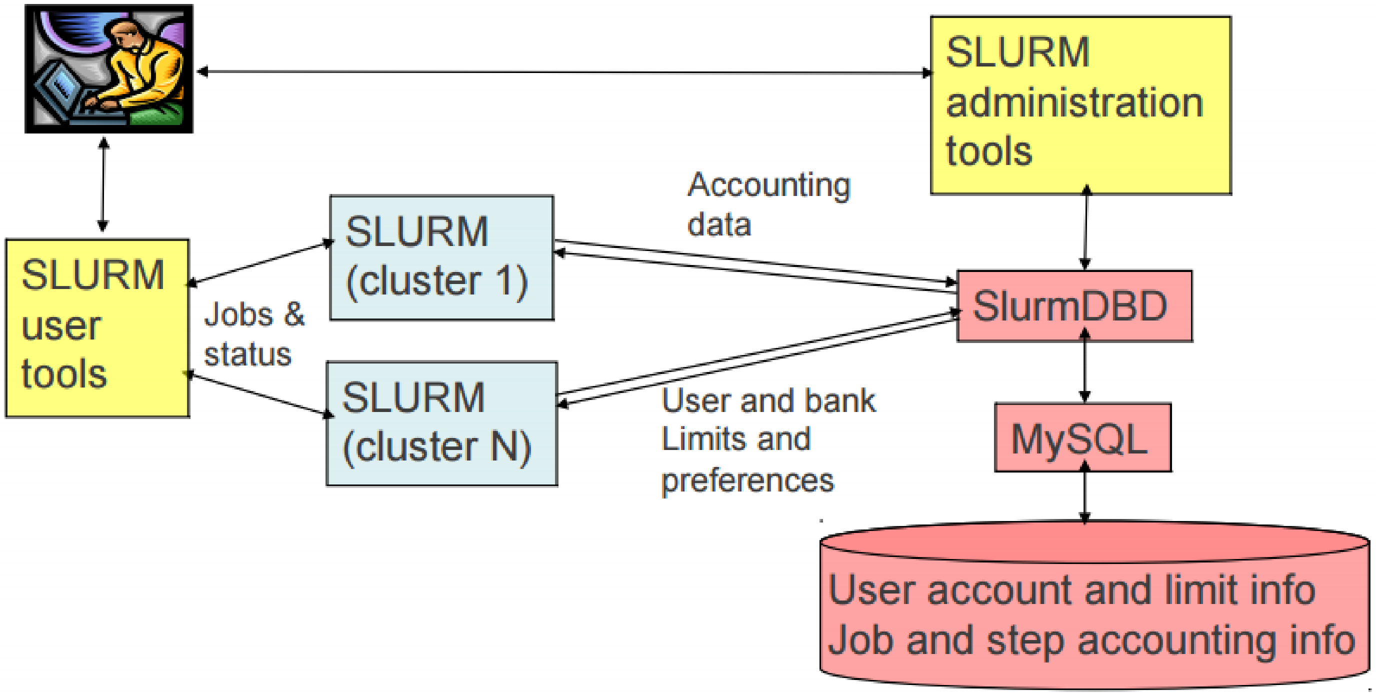
- Jobs spawn steps, which are allocated resources from within the job's allocation



Linux cluster architecture



Enterprise architecture



Summary

- Clusters – networked commodity hardware
- Very high computation power
- Message Passing Interface
- Work scheduler