Distributed Systems

23. Clusters

Paul Krzyzanowski

Rutgers University

Spring 2020

© 2014-2020 Paul Krzyzanowski

Computer System Design Options

- **Highly Available Systems**
  - Incorporate elements of fault-tolerant design
  - Component replication, high-quality components
  - A fully fault tolerant system will offer non-stop availability … *but you can't have this!*
  - Problem: ↑ in availability ⇒ ↑ $$
- **High Performance Systems**
  - SMP architecture
  - Shared memory, shared clock, multiple processors
  - Problems:
    - Performance gain as f(# processors) is sublinear
    - Contention for resources (bus, memory, devices)
    - The solution is also expensive!
- **Commodity off-the-shelf Systems (COTS)**
  - Inexpensive
  - Problem: Not reliable and not high performance

© 2014-2020 Paul Krzyzanowski

Clustering

Achieve reliability and scalability by interconnecting multiple independent systems

Cluster:
A group of standard, autonomous servers configured so they appear on the network as a single machine

Single system image

© 2014-2020 Paul Krzyzanowski

Ideally…

- Bunch of off-the shelf machines
- Interconnected on a high-speed LAN
- Appear as one system to users
- Processes are load-balanced across the cluster
  - May migrate
  - May run on different systems
  - All IPC mechanisms and file access available
- Fault tolerant
  - Components may fail
  - Machines may be taken down

© 2014-2020 Paul Krzyzanowski

We don’t get all that (yet)

… at least not in one general purpose package

© 2014-2020 Paul Krzyzanowski

Clustering types

- High availability (HA)
  - Failover cluster
- Supercomputing (HPC)
  - Includes batch processing
- Load balancing
- Storage

© 2014-2020 Paul Krzyzanowski

High Availability (HA) Clustering

© 2014-2020 Paul Krzyzanowski

Cluster Components

© 2014-2020 Paul Krzyzanowski

Cluster Components

- Cluster membership
- Heartbeat & heartbeat network
- Quorum
- Configuration & service management
- Storage

© 2014-2020 Paul Krzyzanowski

Cluster membership

Software to manage cluster membership

- What are the nodes in the cluster?
- Which nodes in the cluster are currently *alive* (active)?

We saw this:

- Group Membership Service in virtual synchrony
- GFS master, HDFS NameNode
- Bigtable master
- Pregel master
- MapReduce Master & Spark Cluster Manager

Currently, we mostly have custom clustering solutions for specific frameworks. Some components, such as Chubby (Apache Zookeeper) have been adopted by multiple frameworks.

© 2014-2020 Paul Krzyzanowski

Quorum

Some members may be dead or disconnected

Quorum

- Number of elements that must be online for the cluster to function
- Voting algorithm to determine whether the set of nodes has quorum (a majority of nodes to keep running)
- We saw this with Raft consensus (& Paxos)
  - Forcing a majority avoids **split-brain**

Quorum disk

- Shared storage: whichever node can reserve the disk owns it
- Enables systems to resolve who runs a service in small clusters even if the network becomes partitioned

© 2014-2020 Paul Krzyzanowski

Types of Quorum

- **Node Majority**
  - Each available node can vote
  - Need majority (over 50%) of votes for the cluster to continue running
  - Best for odd number of nodes, larger clusters
- **Node & Disk Majority** (Microsoft *Disk Witness*)
  - Designated shared disk = *disk witness*: counts as a vote
  - Need majority of votes to continue running
  - Best for an even # of nodes in one site
- **Node & File Share Majority** (Microsoft *File Share Witness*)
  - Shared file system = *file share witness* : counts as a vote
  - Need majority of votes to continue running
  - Windows Server 2019: File Share Witness on USB stick
    - Shared USB storage on router
  - Best for an even # of nodes in a multi-site cluster
- **No majority**
  - Cluster has quorum if one node is available and can communicate with a specific disk in the cluster
  - Best for an even # of nodes (e.g., 2) with no shared storage

© 2014-2020 Paul Krzyzanowski

Cluster configuration & service management

- **Cluster configuration system & manager**
  - UI to manage configuration of systems and software in a cluster
  - Administrator has a single point of control
- **Cluster management agent**
  - Runs in each cluster node
    - Changes propagate to all nodes
  - Tracks cluster membership – removes failed nodes
  - Keeps track of quorum – stops cluster when ≤ nodes not active
- **Service management**
  - Identify which applications run where
  - Specify how failover occurs
    - Active: system runs a service
    - Standby: Which system(s) can run the service if the active dies
  - E.g., MapReduce, Pregel, Spark all use coordinators

© 2014-2020 Paul Krzyzanowski

Disks

© 2014-2020 Paul Krzyzanowski

Shared storage access

- If an application can run on any machine, how does it access file data?
- If an application fails over from one machine to another, how does it access its file data?
- Can applications on different machines share files?

© 2014-2020 Paul Krzyzanowski

Network (Distributed) File Systems

One option:

- Network file systems: NFS, SMB, AFS, AFP, etc.
- Works great for many applications

Concerns

Availability

- Address with replication (most file systems offer little)

Performance

- Remote systems on a LAN vs. local bus access
- Overhead of remote operating system & network stack
- Point of congestion
- Look at GFS/HDFS to distribute file data across lots of servers
  … or other parallel file systems, such as Lustre, GlusterFS, or Ceph

© 2014-2020 Paul Krzyzanowski

Shared disks & Cluster file systems

Shared disk

- Allows multiple systems to share access to disk drives
- Works well if there isn't much contention
  … but you can't have multiple systems
  reading/writing/caching the same disk blocks

Cluster File System

- Client runs a file system accessing a shared disk at the **block level**
  - *vs. a distributed file system, which access at a file-system level*
- No client/server roles, no disconnected modes
- All nodes are peers and access a shared disk(s)
- **Distributed Lock Manager (DLM)**
  - Process to ensure mutual exclusion for disk access
  - Provides inode-based locking and caching control
  - Not needed for local file systems on a shared disk

© 2014-2020 Paul Krzyzanowski

Cluster File Systems

Examples:

- IBM General Parallel File System (GPFS)
- Microsoft Cluster Shared Volumes (CSV)
- Oracle Cluster File System (OCFS)
- Red Hat Global File System (GFS2)

Linux GFS2 (no relation to Google GFS)

- Cluster file system accessing storage at a **block level**
- Cluster Logical Volume Manager (CLVM):
  volume management of cluster storage
- Global Network Block Device (GNBD): block level storage access over ethernet:
  cheap way to access block-level storage

© 2014-2020 Paul Krzyzanowski

The alternative: shared nothing

Shared nothing

- No shared devices
- Each system has its own storage resources
- No need to deal with DLMs
- If a machine A needs resources on B, A sends a message to B
  - If B fails, storage requests have to be switched over to a live node

Need exclusive access to shared storage

- Multiple nodes may have access to shared storage
- Only one node is granted exclusive access at a time – *one owner*
- Exclusive access changed on failover

© 2014-2020 Paul Krzyzanowski

SAN: Computer-Disk interconnect

SAN = Storage Area Network

- Separate network between nodes and storage arrays
  - Fibre channel
  - iSCSI
- Any node can be configured to access any storage through a fibre channel switch

Acronyms

- **DAS**: Direct Attached Storage
- **SAN**: block-level access to a disk via a network
- **NAS**: file-level access to a remote file system (NFS, SMB,…)

© 2014-2020 Paul Krzyzanowski

Failover

© 2014-2020 Paul Krzyzanowski

HA issues

- How do you detect failover?
- How long does it take to detect?
- How does a dead application move/restart?
- Where does it move to?

© 2014-2020 Paul Krzyzanowski

Heartbeat network

- Machines need to detect faulty systems
  - **Heartbeat**: Periodic "ping" mechanism
  - An "are you alive" message
- Need to distinguish ***system faults*** from ***network faults***
  - Useful to maintain redundant networks
  - Avoid split-brain issues in systems without quorum (e.g., a 2-node cluster)
- Once you know who is dead or alive, then determine a course of action

© 2014-2020 Paul Krzyzanowski

Failover Configuration Models

- Active/Passive
  - Requests go to active system
  - Passive nodes do nothing until they're needed
  - Passive nodes maintain replicated state (e.g., SMR/Virtual Synchrony)
  - Example: Chubby

- Active/Active
  - Any node can handle a request
  - Failed workload goes to remaining nodes
  - Replication must be *N*-way for *N* active nodes
  - Example: GFS chunks

- Active/Passive: *N*+*M*
  - *M* dedicated failover node(s) for *N* active nodes

© 2014-2020 Paul Krzyzanowski

Design options for failover

- **Cold failover**
  - Application restart
  - *Example: map and reduce workers in MapReduce*
- **Warm failover**
  - Restart last checkpointed image
  - Relies on application checkpointing itself periodically
  - *Example: Pregel*
- **Hot failover**
  - Application state is synchronized across systems
    - E.g., replicated state machines or lockstep synchronization at the CPU level
  - Spare is ready to run immediately
  - May be difficult at a fine granularity, prone to software faults (e.g., what if a specific set of inputs caused the software to die?)
  - *Example: Chubby*

© 2014-2020 Paul Krzyzanowski

Design options for failover

With either type of failover …

Multi-directional failover

- Failed applications migrate to or restart on available systems

And possibly

Cascading failover

- If the backup system fails, application can be restarted on another surviving system

© 2014-2020 Paul Krzyzanowski

IP Address Takeover (IPAT)

Depending on the deployment:

- **Ignore**
  - IP addresses of services don't matter. A load balancer, name server, or coordinator will identify the correct machine
- **Take over IP address**
  - A node in an active/passive configuration may need to take over the IP address of a failed node
- **Take over MAC address**
  - MAC address takeover may be needed if we cannot guarantee that other nodes will flush their ARP cache
- **Listen on multiple addresses**
  - A node in an active/active configuration may need to listen on multiple IP addresses

© 2014-2020 Paul Krzyzanowski

Hardware support for High Availability

- Hot-pluggable components
  - Minimize downtime for component swapping
  - E.g., disks, power supplies, CPU/memory boards
- Redundant devices
  - Redundant power supplies
  - Parity on memory
  - Mirroring on disks (or RAID for HA)
  - Switchover of failed components
- Diagnostics
  - On-line identification & service

© 2014-2020 Paul Krzyzanowski

Fencing

- **Fencing**: method of isolating a node from a cluster
  - Apply to failed node
  - Disconnect I/O to ensure data integrity
  - Avoid problems with Byzantine failures
  - Avoids problems with *fail-restart*
    - Restarted node has not kept up to date with state changes

- Types of fencing
  - **Power fencing**: shut power off a node
  - **SAN fencing**: disable a Fibre Channel port to a node
  - **System service fencing**: disable access to a global network block device (GNBD) server
  - **Software fencing**: remove server processes from the group
    - E.g., virtual synchrony

© 2014-2020 Paul Krzyzanowski

Cluster software hierarchy

Example: Windows Server cluster abstractions

Top tier: Cluster abstractions

- Failover manager (what needs to be started/restarted?)
- Resource monitor (what's going on?)
- Cluster registry (who belongs in the cluster?)

Middle tier: Distributed operations

- Global status update
- Membership
- Quorum (and leader election)

Bottom tier: OS and drivers

- Cluster disk driver, cluster network drivers
- IP address takeover

© 2014-2020 Paul Krzyzanowski

High Performance Computing (HPC)

© 2014-2020 Paul Krzyzanowski

Supercomputers

2018's Most powerful supercomputer:
IBM AC922 – Summit at *Oak Ridge National Laboratory*

- 189 petaflops, >10PB memory
- 4,608 nodes
  - 6 NVIDIA Volta V100s GPUs
  - 2 IBM POWER9™ CPUs
  - 512 GB DDR4 + 96GB HBM2 RAM
  - 1600GB NV memory
  - 42 teraflops per node

>27,000 GPUs
>9,000 CPUs

- 100G InfiniBand interconnect
- 250 PB 2.5 TB/s file system
- OS: Red Hat Enterprise Linux
- Peak power consumption: 13 MW

See https://www.olcf.ornl.gov/summit/

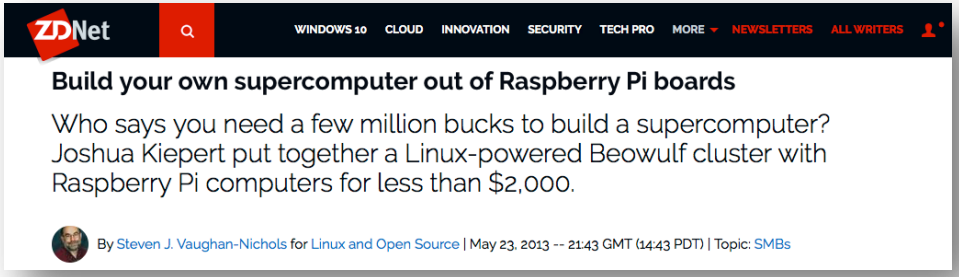
© 2014-2020 Paul Krzyzanowski

- Supercomputers are *not* distributed computers
- Lots of processors connected by high-speed networks
- Shared memory access
- Shared operating system (all TOP500 run Linux)

© 2014-2020 Paul Krzyzanowski

Supercomputing clusters

- Supercomputing cluster
  - Build a supercomputer from commodity computers & networks
  - A distributed system



- Target complex, typically scientific, applications:
  - Large amounts of data
  - Lots of computation
  - Parallelizable application

- Many custom efforts
  - Typically Linux + message passing software + remote exec + remote monitoring

© 2014-2020 Paul Krzyzanowski

Cluster Interconnects

© 2014-2020 Paul Krzyzanowski

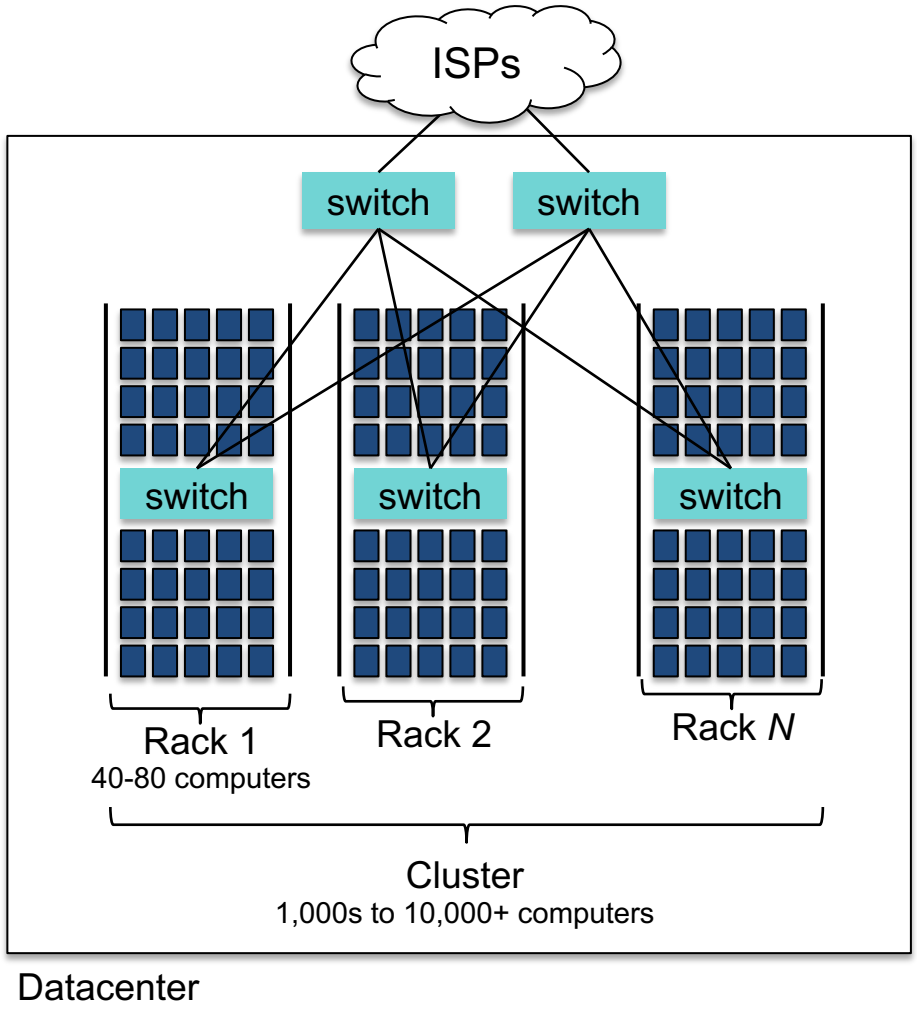
Cluster Interconnect

Provide communication between nodes in a cluster

Goals

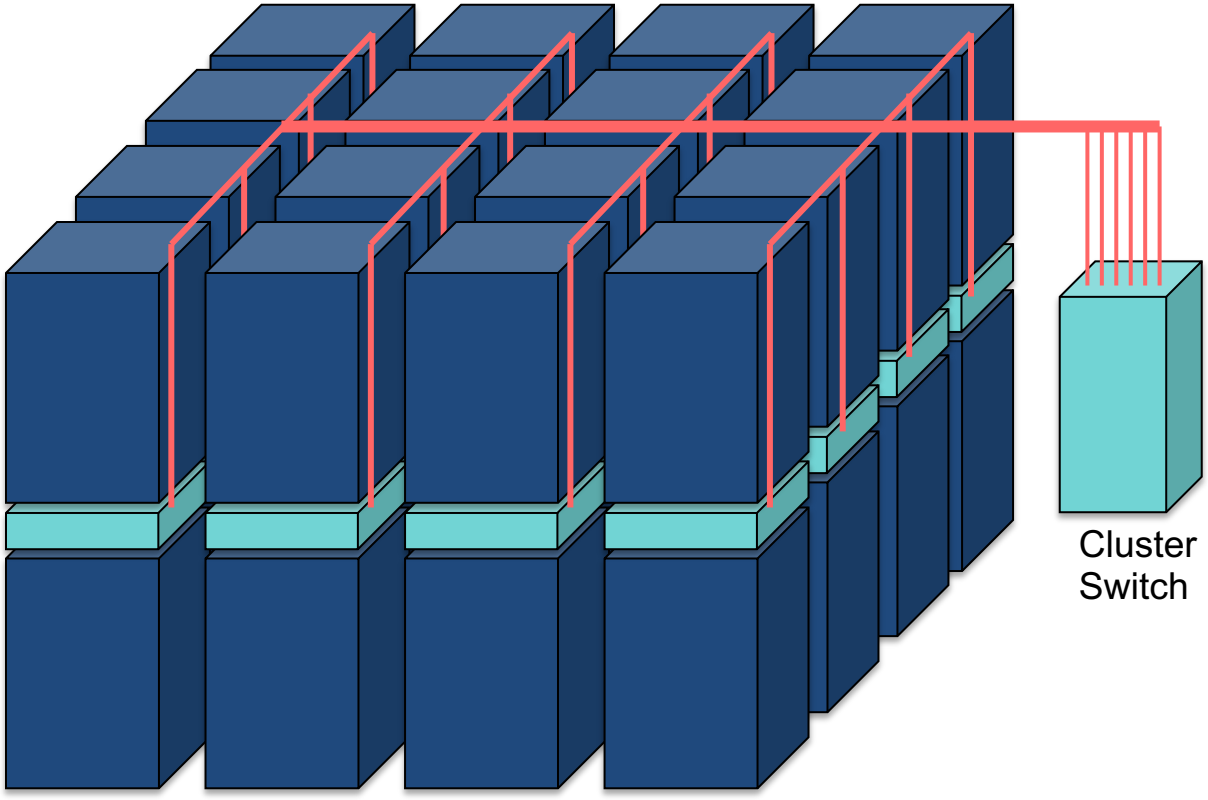
- Low latency
  - Avoid OS overhead, layers of protocols, retransmission, etc.
- High bandwidth
  - High bandwidth, switched links
  - Avoid overhead of sharing traffic with non-cluster data
- Low CPU overhead
- Low cost
  - Cost usually matters if you're connecting thousands of machines

- Usually a LAN is used: best $/performance ratio



© 2014-2020 Paul Krzyzanowski

Cluster Interconnect



Cluster of 4×4 racks

Assume:
- 10 Gbps per server
- 40 servers per rack
- ⇒ 400 Gbps/rack

16 racks
- ⇒ 8 Tbps

Max switch capacity currently ~ 5 Tbps
- ⇒ Need at least two cluster switches

© 2014-2020 Paul Krzyzanowski

Switches add latency

Within one rack

- One **switch latency** ≈ <1…8 µs for a 10 Gbps switch
- Two links (to switch + from switch) @ 1-2 meters of cable
  - **Propagation time** in copper ≈ 2×10^8 m/s ≈ 5 ns/m

Between racks in a cluster

- Three switch latency (≈ <3…24 µs)
- 4 links (to rack switch + to cluster switch + back to target rack)
- ~10-100 meters distance (50 … 500 ns)

Add to that the normal latency of sending & receiving packets:

- System latency of processing the packet, OS mode switch, queuing the packet, copying data to the transceiver, …
- **Serialization delay** = time to copy packet to media ≈ 1 µs for a 1KB packet on a 10 Gbps link

© 2014-2020 Paul Krzyzanowski

Dedicated cluster interconnects

TCP adds latency

- Operating system overhead, queueing, checksums, acknowledgements, congestion control, fragmentation & reassembly, …
- Lots of interrupts
- Consumes time & CPU resources

How about using a high-speed LAN without the overhead?

- LAN dedicated for intra-cluster communication
  - Sometimes known as a System Area Network (SAN)
- Dedicated network for storage: Storage Area Network (SAN)

© 2014-2020 Paul Krzyzanowski

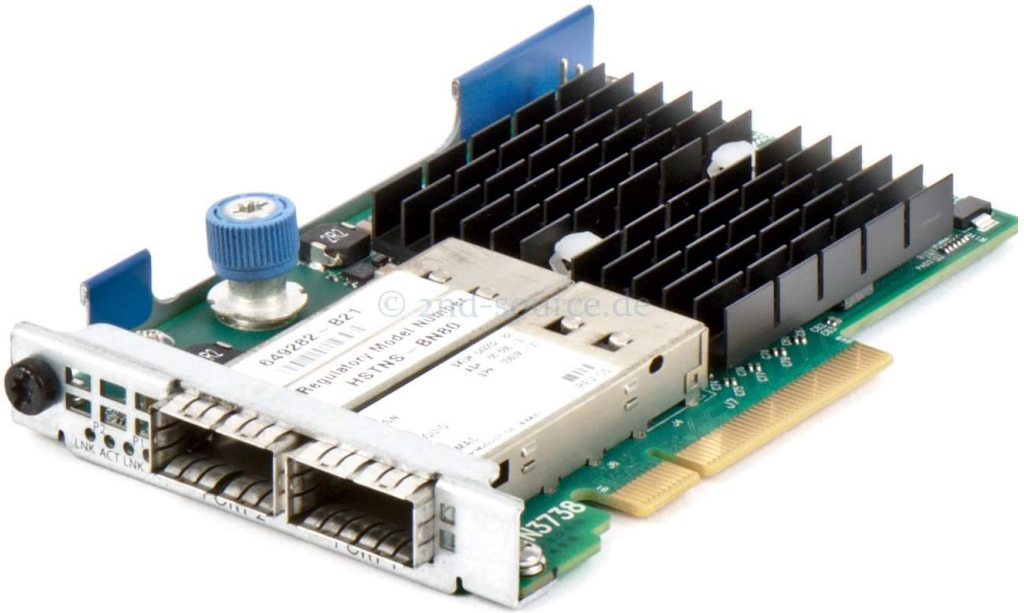
Example High-Speed Interconnects

Common traits

- **TCP/IP Offload Engines** (TOE)
  - TCP stack at the network card
- **Remote Direct Memory Access** (RDMA)
  - memory copy with no CPU involvement

Intel I/O Acceleration Technology (I/OAT) – combines TOE & RDMA

- Data copy without CPU, TCP packet coalescing, low-latency interrupts, …



© 2014-2020 Paul Krzyzanowski

Example High-Speed Interconnects

Example: **InfiniBand**

- Switch-based point-to-point bidirectional serial links
- Link processors, I/O devices, and storage
- Each link has one device connected to it
- Enables data movement via remote direct memory access (RDMA)
  - No CPU involvement!
- Up to 250 Gbps/link
  - Links can be aggregated: up to 3000 Gbps with 12x links

IEEE 802.1 Data Center Bridging (DCB)

- Set of standards that extend Ethernet
- Lossless data center transport layer
  - Priority-based flow control, congestion notification, bandwidth management

© 2014-2020 Paul Krzyzanowski

Programming tools for HPC: PVM

PVM = Parallel Virtual Machine

- Software that emulates a general-purpose heterogeneous computing framework on interconnected computers
- Model: app = set of tasks
  - Functional parallelism: tasks based on function: input, solve, output
  - Data parallelism: tasks are the same but work on different data
- PVM presents library interfaces to:
  - Create tasks
  - Use global task IDs
  - Manage groups of tasks
  - Pass basic messages between tasks

© 2014-2020 Paul Krzyzanowski

Programming tools: MPI

MPI: Message Passing Interface

- API for sending/receiving messages
  - Optimizations for shared memory & NUMA
  - Group communication support
- Other features:
  - Scalable file I/O
  - Dynamic process management
  - Synchronization (barriers)
  - Combining results

© 2014-2020 Paul Krzyzanowski

HPC Cluster Example

Early example: Early (>20 years old!) effort on Linux – Beowulf

- Initially built to address problems associated with large data sets in Earth and Space Science applications
- From Center of Excellence in Space Data & Information Sciences (CESDIS)
  - Division of University Space Research Association at the Goddard Space Flight Center
- Still used!

This isn't one fixed package

- Just an example of putting tools together to create a supercomputer from commodity hardware

© 2014-2020 Paul Krzyzanowski

What makes it possible?

- Commodity off-the-shelf computers are cost effective
- Publicly available software:
  - Linux, GNU compilers & tools
  - MPI (message passing interface)
  - PVM (parallel virtual machine)
- Low cost, high speed networking
- Experience with parallel software
  - Difficult: solutions tend to be custom

© 2014-2020 Paul Krzyzanowski

What can you run?

- Programs that do not require fine-grain communication

- Basic properties
  - Nodes are dedicated to the cluster
    - Performance of nodes not subject to external factors
  - Interconnect network isolated from external network
    - Network load is determined only by application
  - Global process ID provided
    - Global signaling mechanism

© 2014-2020 Paul Krzyzanowski

HPC Cluster Example

More recent – OpenHPC (openhpc.community)

- 18 admin tools
- 3 compiler families (GNU, Intel, LLVM)
- 13 development tool packages (EasyBuild, cbuild, libtool, …)
- Lua scripting language & supporting packages
- 8 I/O libraries
  - Adios – enables defining how data is accessed
  - HDF5 – data model, library, and file format for storing and managing data
  - NetCDF – managing array-oriented scientific data
- Lustre file system
- 4 MPI packages
- 12 parallel libraries
- 14 performance tools
- Provisioning tools, resource management, runtime packages
- 6 Threaded library packages

© 2014-2020 Paul Krzyzanowski

HPC example

Rocks Cluster Distribution

- Employed on over 1,300 clusters
- Mass installation is a core part of the system
  - Mass re-installation for application-specific configurations
- Front-end central server + compute & storage nodes
- Based on CentOS Linux
- Rolls: collection of packages
  - Base roll includes: PBS (portable batch system), PVM (parallel virtual machine), MPI (message passing interface), job launchers, …

Open-source Linux cluster distribution – supported by the National Science Foundation – rocksclusters.org

© 2014-2020 Paul Krzyzanowski

Batch Processing

© 2014-2020 Paul Krzyzanowski

Batch processing

- Non-interactive processes
  - Schedule, run eventually, collect output
- Examples:
  - MapReduce, many supercomputing tasks (circuit simulation, climate simulation, weather)
  - Graphics rendering
    - Maintain a queue of frames to be rendered
    - Have a dispatcher to remotely exec process
- In many cases – minimal or no IPC needed
- Coordinator dispatches jobs

© 2014-2020 Paul Krzyzanowski

Single-queue work distribution: Render Farms

Examples:

Pixar:

- 55,000 cores running Linux and Renderman (2018)
- Custom Linux software for articulating, animating/lighting (Marionette), scheduling (Ringmaster), and rendering (RenderMan)

- Took over two years (of real time!) to render Monsters University (2013)
- Average time to render a single frame
  - Cars (2006): 8 hours
  - Cars 2 (2011): 11.5 hours
  - Disney/Pixar's Coco – Up to 100 hours to render one frame
  - Toy Story 3 (2010) – 7 hours, but some up to 39 hours

http://venturebeat.com/2013/04/24/the-making-of-pixars-latest-technological-marvel-monsters-university/2/
http://news.cnet.com/8301-13772_3-20068109-52/new-technology-revs-up-pixars-cars-2/

© 2014-2020 Paul Krzyzanowski

Batch Processing

- OpenPBS.org:
  - Portable Batch System
  - Developed by Veridian MRJ for NASA
- Commands
  - *Submit job scripts*
    - Submit interactive jobs
    - Force a job to run
  - *List jobs*
  - *Delete jobs*
  - *Hold jobs*

© 2014-2020 Paul Krzyzanowski

Load Balancing

© 2014-2020 Paul Krzyzanowski

Functions of a load balancer

- Load balancing
- Failover
- Planned outage management

© 2014-2020 Paul Krzyzanowski

Redirection

Simplest technique

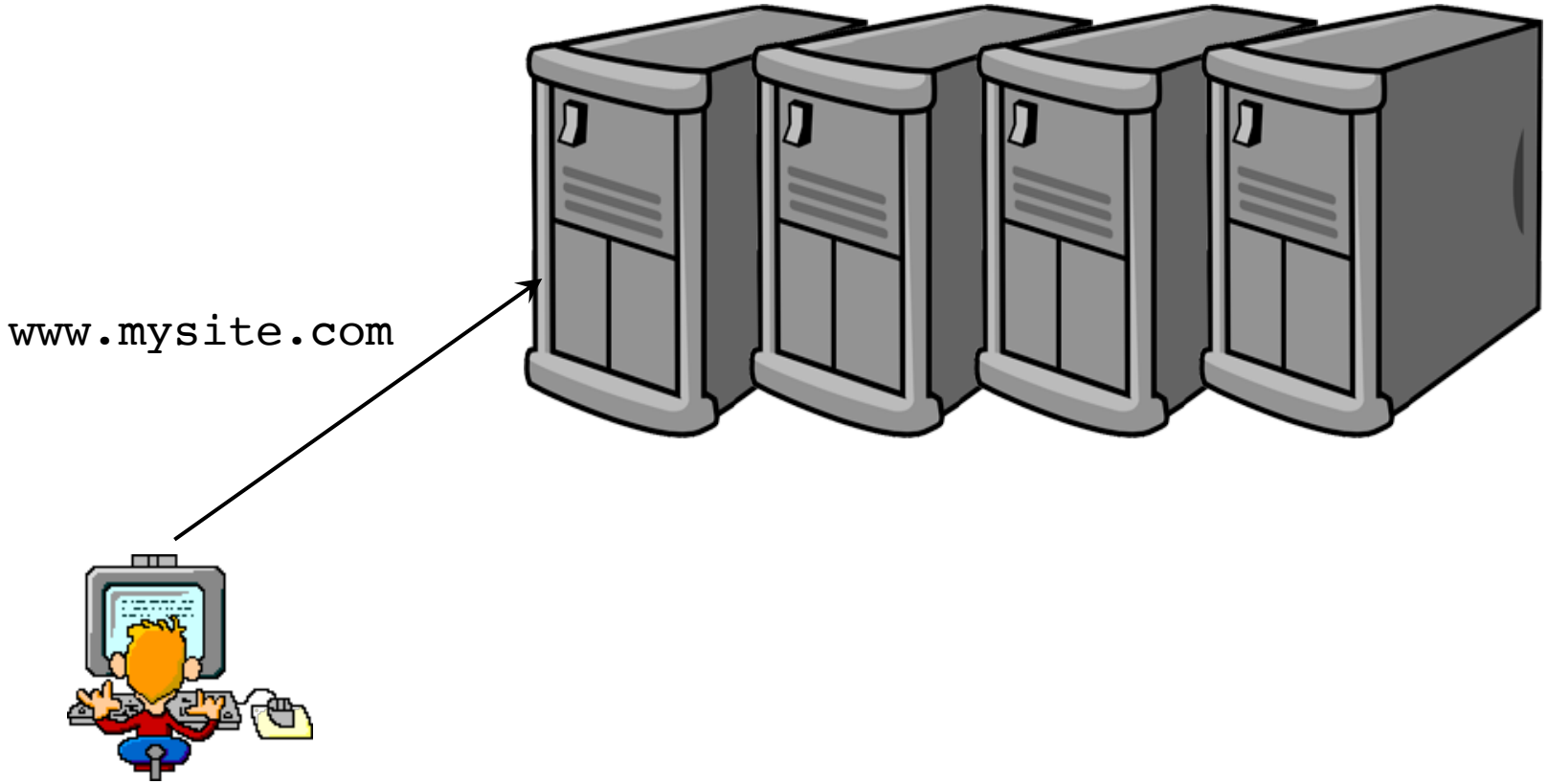
- HTTP REDIRECT error code

© 2014-2020 Paul Krzyzanowski

Redirection

Simplest technique

HTTP REDIRECT error code

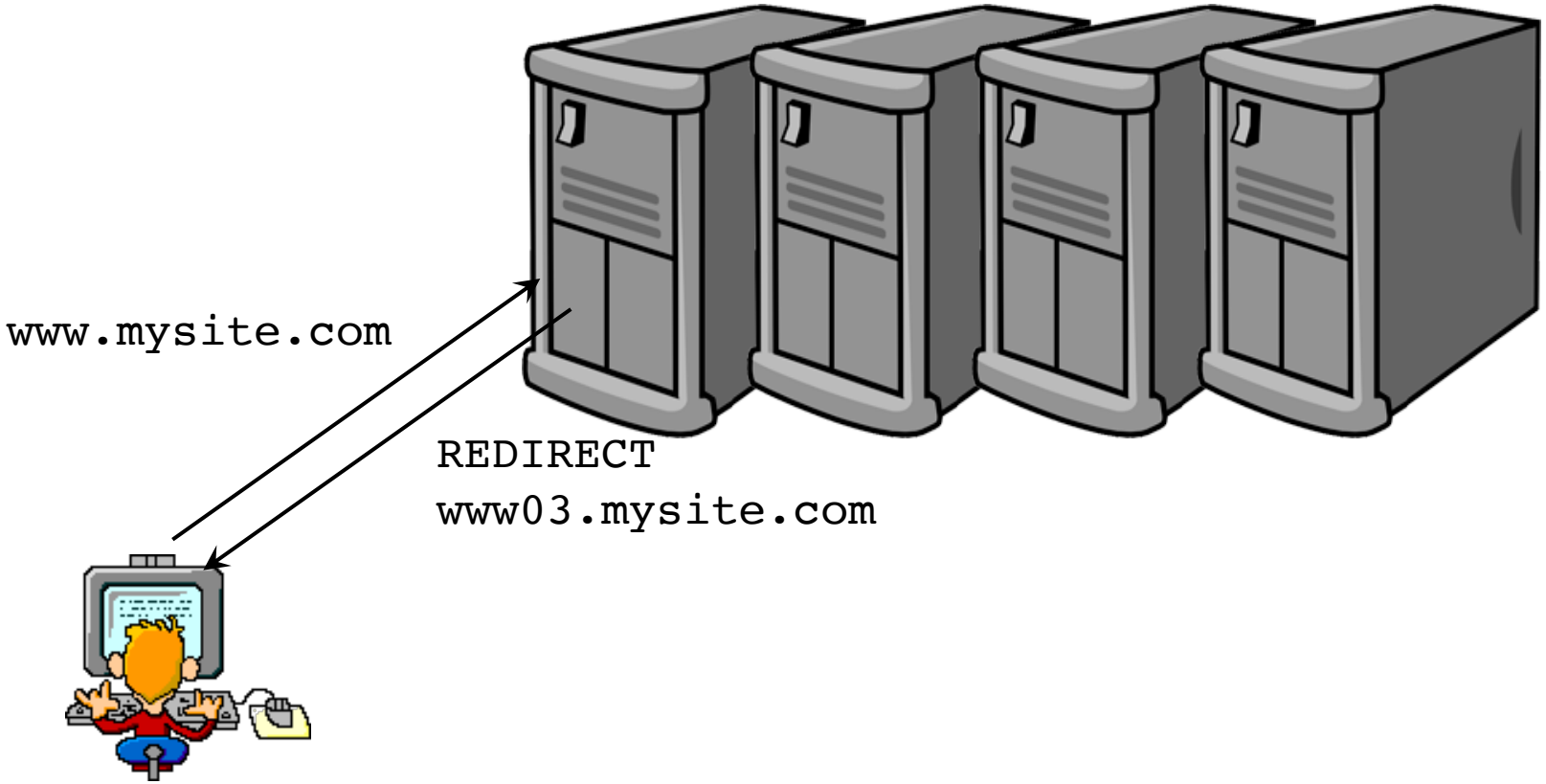


© 2014-2020 Paul Krzyzanowski

Redirection

Simplest technique

HTTP REDIRECT error code

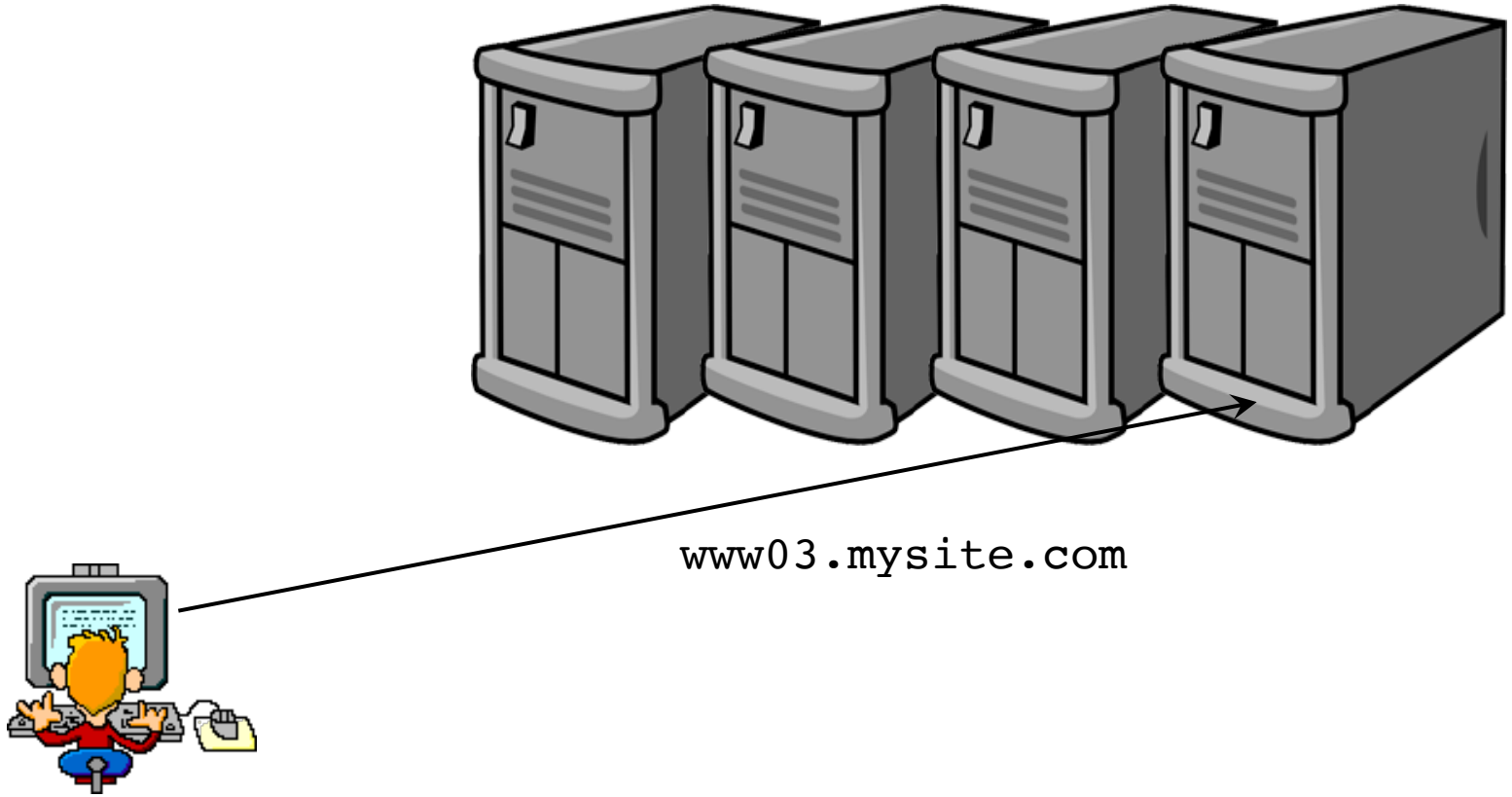


© 2014-2020 Paul Krzyzanowski

Redirection

Simplest technique

HTTP REDIRECT error code



© 2014-2020 Paul Krzyzanowski

Redirection

- Trivial to implement
- Successive requests automatically go to the same web server
  - Important for sessions
- Visible to customer
  - Don’t like the changing URL
- Bookmarks will usually tag a specific site

© 2014-2020 Paul Krzyzanowski

Load balancing router

As routers got smarter

- Not just simple packet forwarding
- Most support packet filtering
- Add load balancing to the mix

- This includes most IOS-based Cisco routers, Radware Alteon, F5 Big-IP

© 2014-2020 Paul Krzyzanowski

Load balancing router

- Assign one or more virtual addresses to physical address
  - Incoming request gets mapped to physical address
- Special assignments can be made per port
  - e.g., all FTP traffic goes to one machine
- **Balancing decisions:**
  - Pick machine with least # TCP connections
  - Factor in weights when selecting machines
  - Pick machines round-robin
  - Pick fastest connecting machine (SYN/ACK time)
- **Persistence**
  - Send all requests from one user session to the same system

© 2014-2020 Paul Krzyzanowski

DNS-based load balancing

- Round-Robin DNS
  - Respond to DNS requests with a list of addresses instead of one
  - The order of the list is permuted with each response

- Geographic-based DNS response
  - Multiple clusters distributed around the world
  - Balance requests among clusters
  - Favor geographic proximity
  - Examples:
    - BIND with GeoDNS patch
    - PowerDNS with Geo backend
    - Amazon Route 53

© 2014-2020 Paul Krzyzanowski

The end

© 2014-2020 Paul Krzyzanowski