ECE/CS 757: Advanced Computer Architecture II Clusters

Instructor:Mikko H Lipasti

Spring 2017

University of Wisconsin-Madison

Lecture notes based on slides created by John Shen, Mark Hill, David Wood, Guri Sohi, Jim Smith, Natalie Enright Jerger, Michel Dubois, Murali Annavaram, Per Stenström and probably others

Lecture Outline

- Introduction & Examples
- Case studies
  - VAX Cluster: N. Kronenberg et al., VAXclusters: A Closely-Coupled Distributed System, ACM Transactions on Computer Systems, May 1986, pp. 130-146.
  - Google Cluster: Luiz Andre Barroso, Jeffrey Dean, Urs Holzle, Web Search For a Planet: The Google Cluster Architecture, IEEE Micro, 23(2):22-28, March-April 2003.
  - IBM Blade Center: D. Desai, et al., BladeCenter System Overview, IBM Journal of R and D, Nov. 2005, pp. 809- 821.
  - Microsoft Catapult, Hot Chips 2014
- **Reading:** L. Barroso, J. Clidaras, U. Hölzle, Chapters 1-3, Chapter 5, "The Datacenter as a Computer: An Introduction to the Design of Warehouse-Scale Machines, Second edition," Synthesis Lectures on Computer Architecture, http://www.morganclaypool.com/doi/abs/10.2200/S00516ED2V01Y201306CAC024

Definition etc.

- A bunch of computers connected with a network

  that can be viewed by the user as a single system

  – Each computer has its own address space

  – Programming commonly via message passing

  – Single system image (common login, filesystem, OS, software)

- Advantages

  – Easy to design

  – Relatively inexpensive

    Commodity off-the-shelf parts

  – Can serve dual duty

    Desktop system + Network of Workstations (NOW)

Example Clusters (via Google)

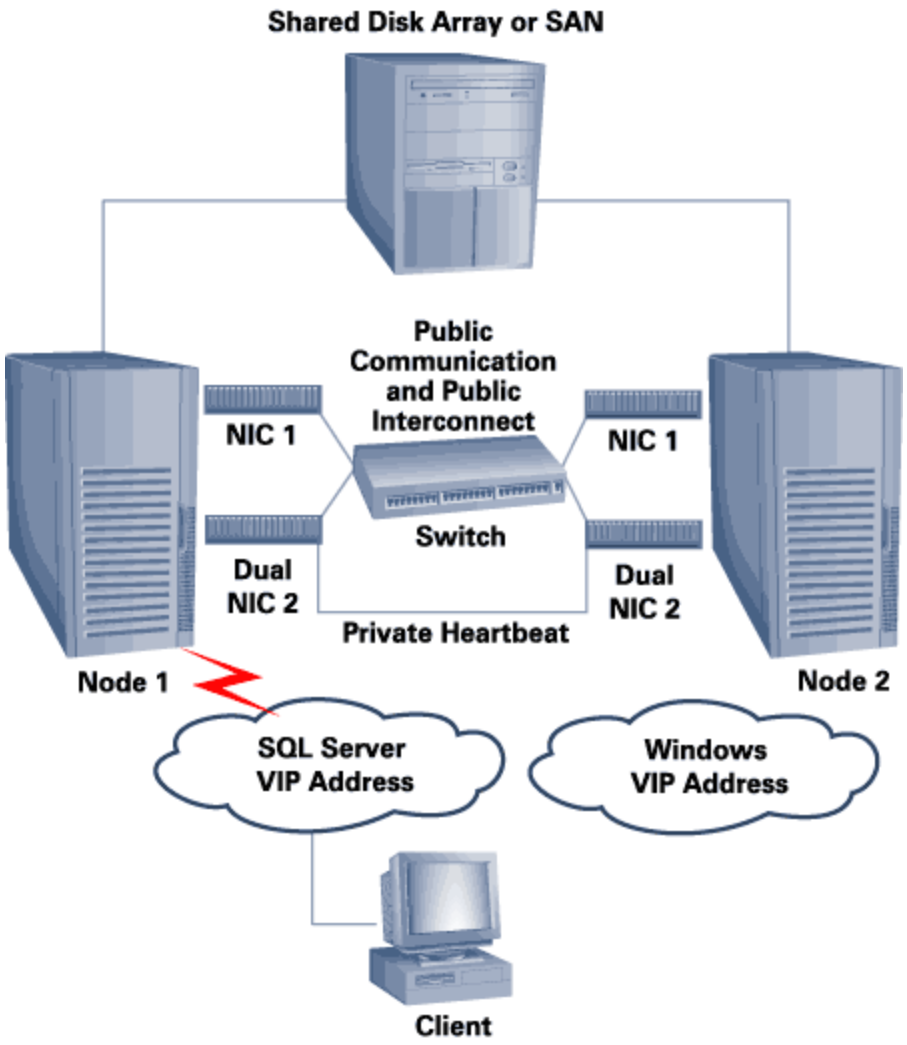
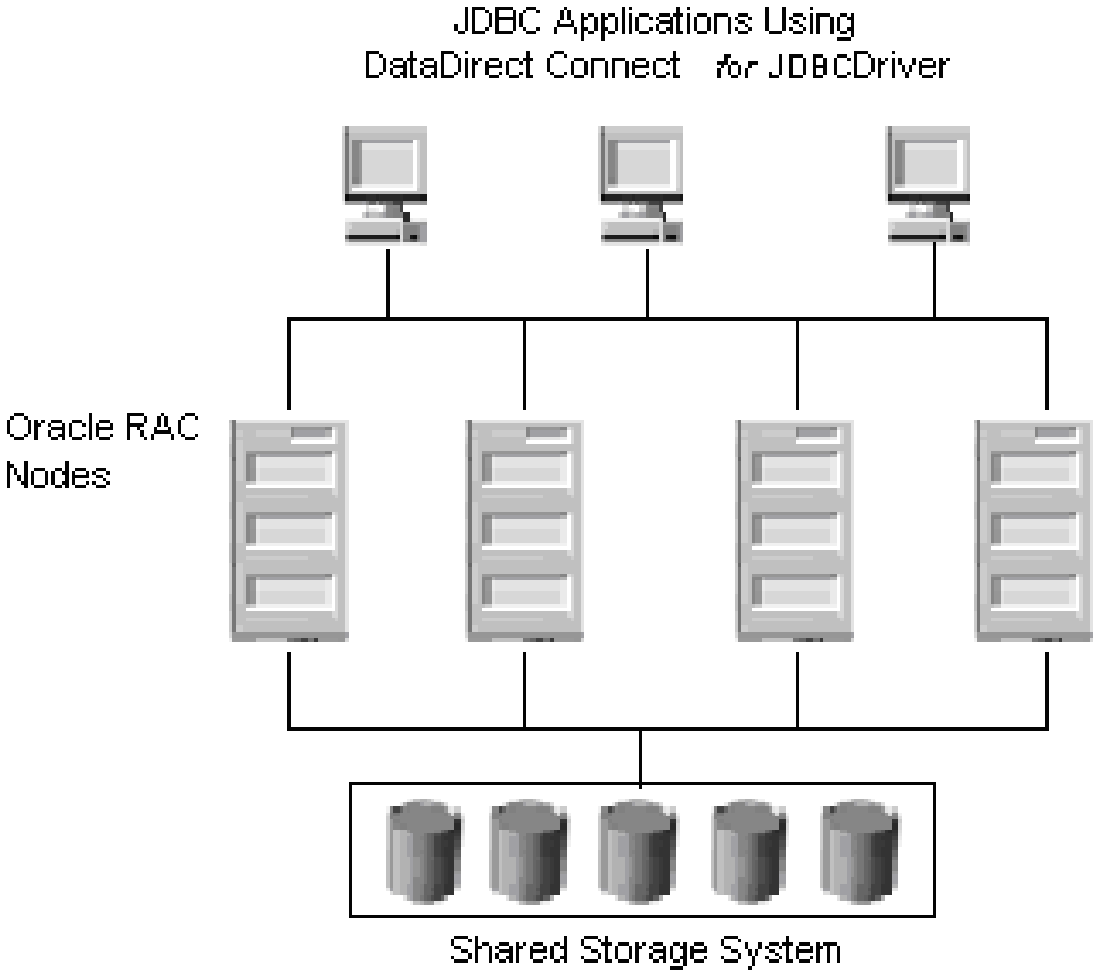
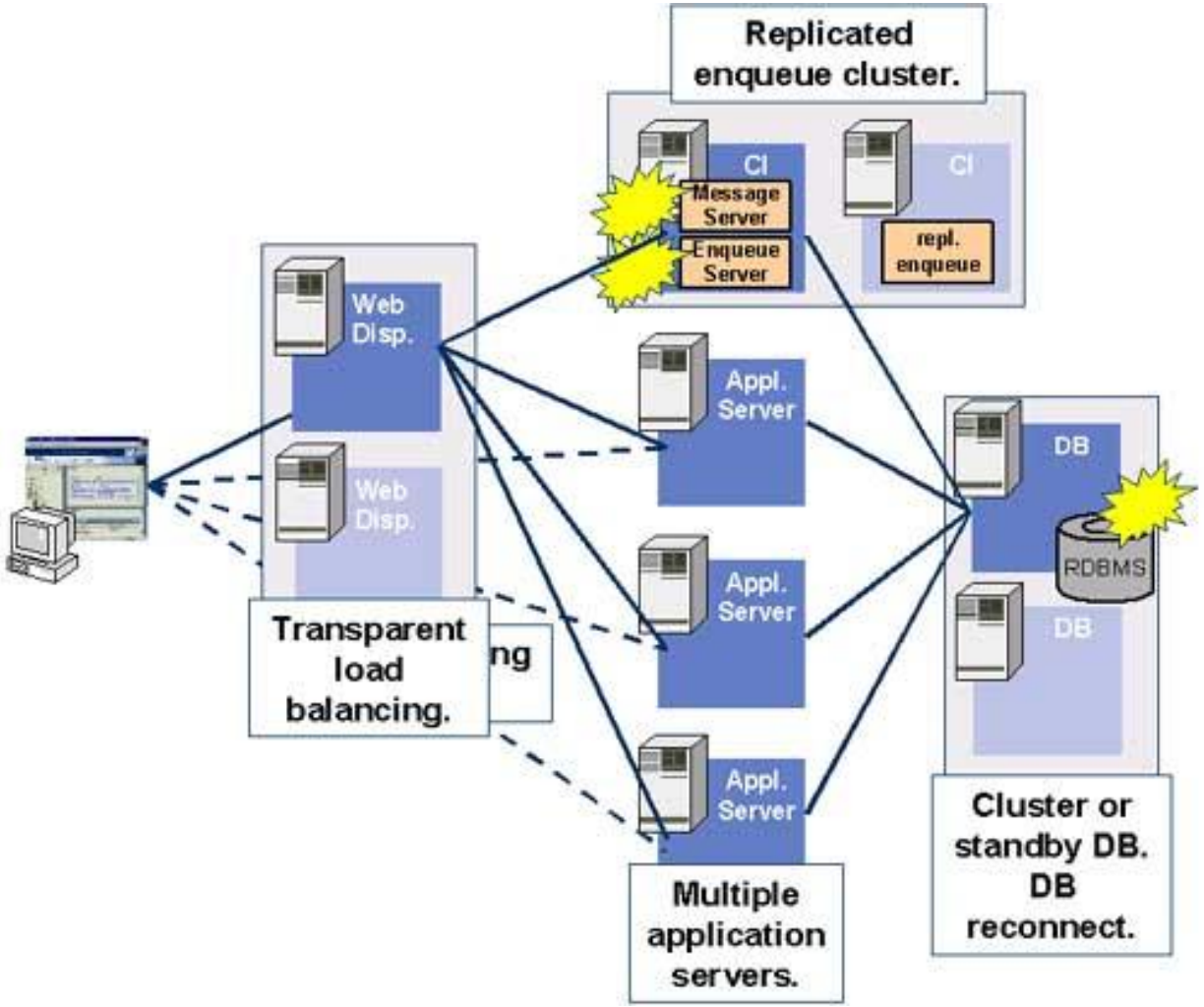


Figure 1: Basic cluster architecture

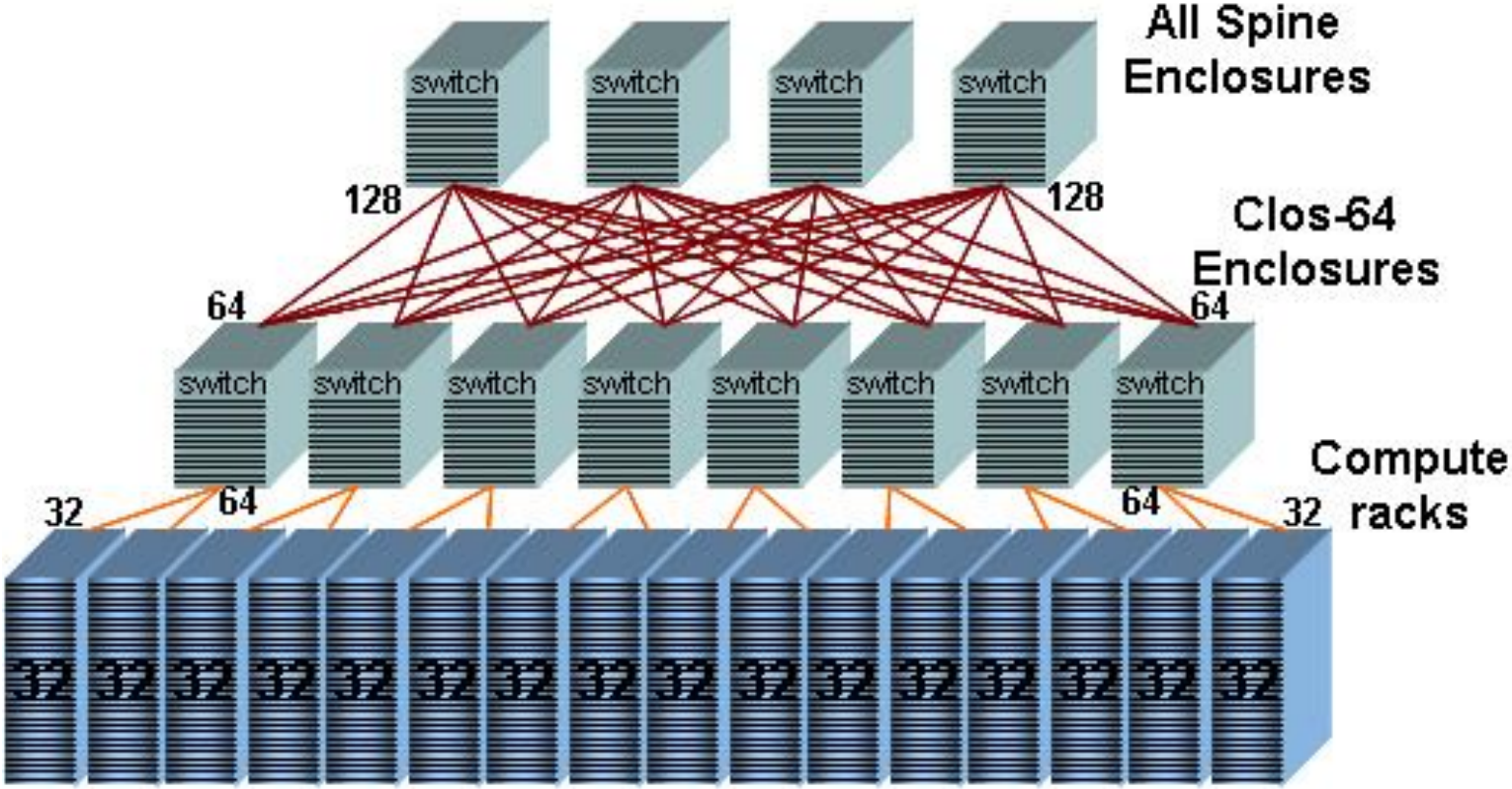
Example Clusters



Example Clusters

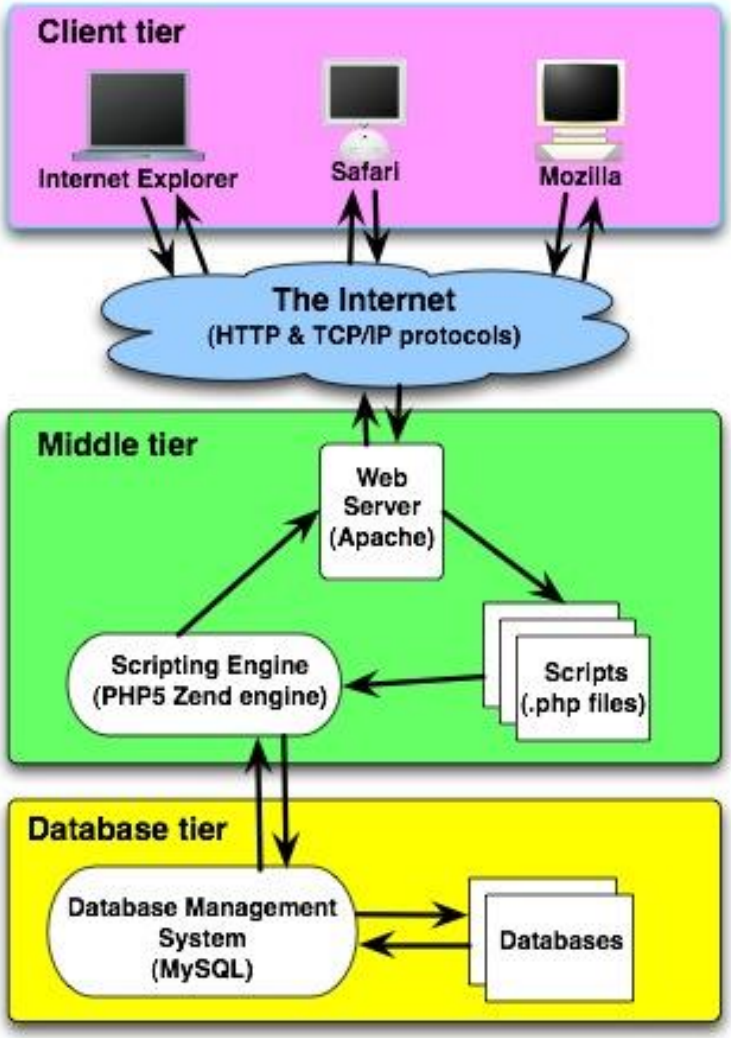


Example Clusters



Lonestar: 512 Compute Nodes

Example Clusters

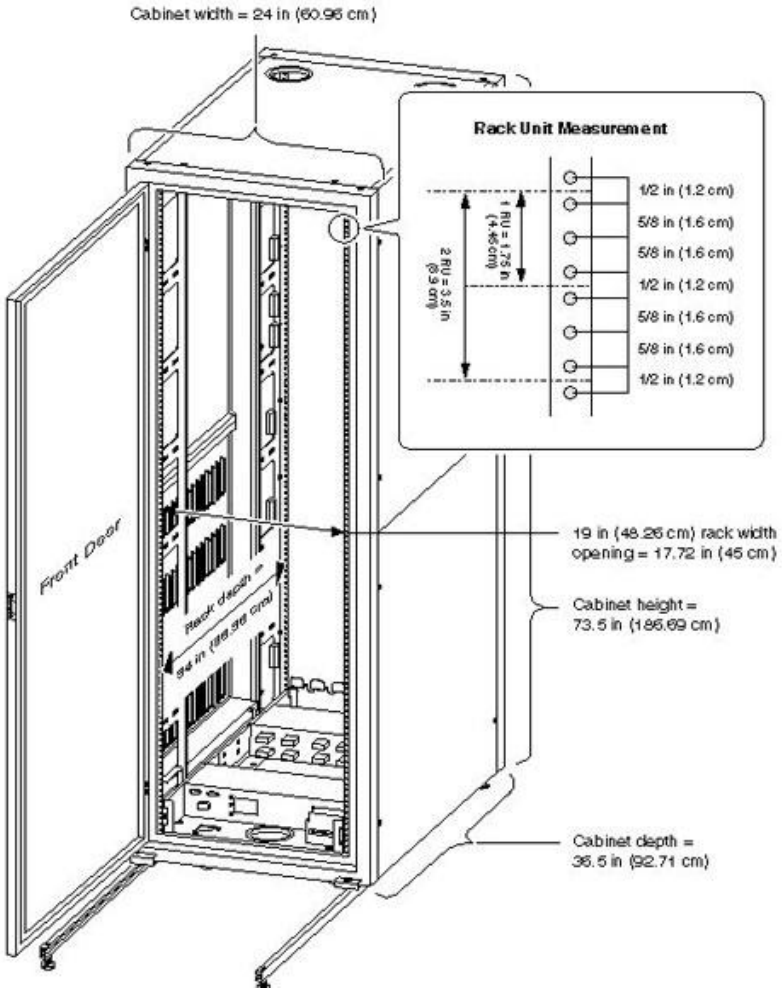


Packaging/Physical Design

- Workstations (PCs) on a LAN

Packaging/Physical Design

- Rack Mounted PC Boards



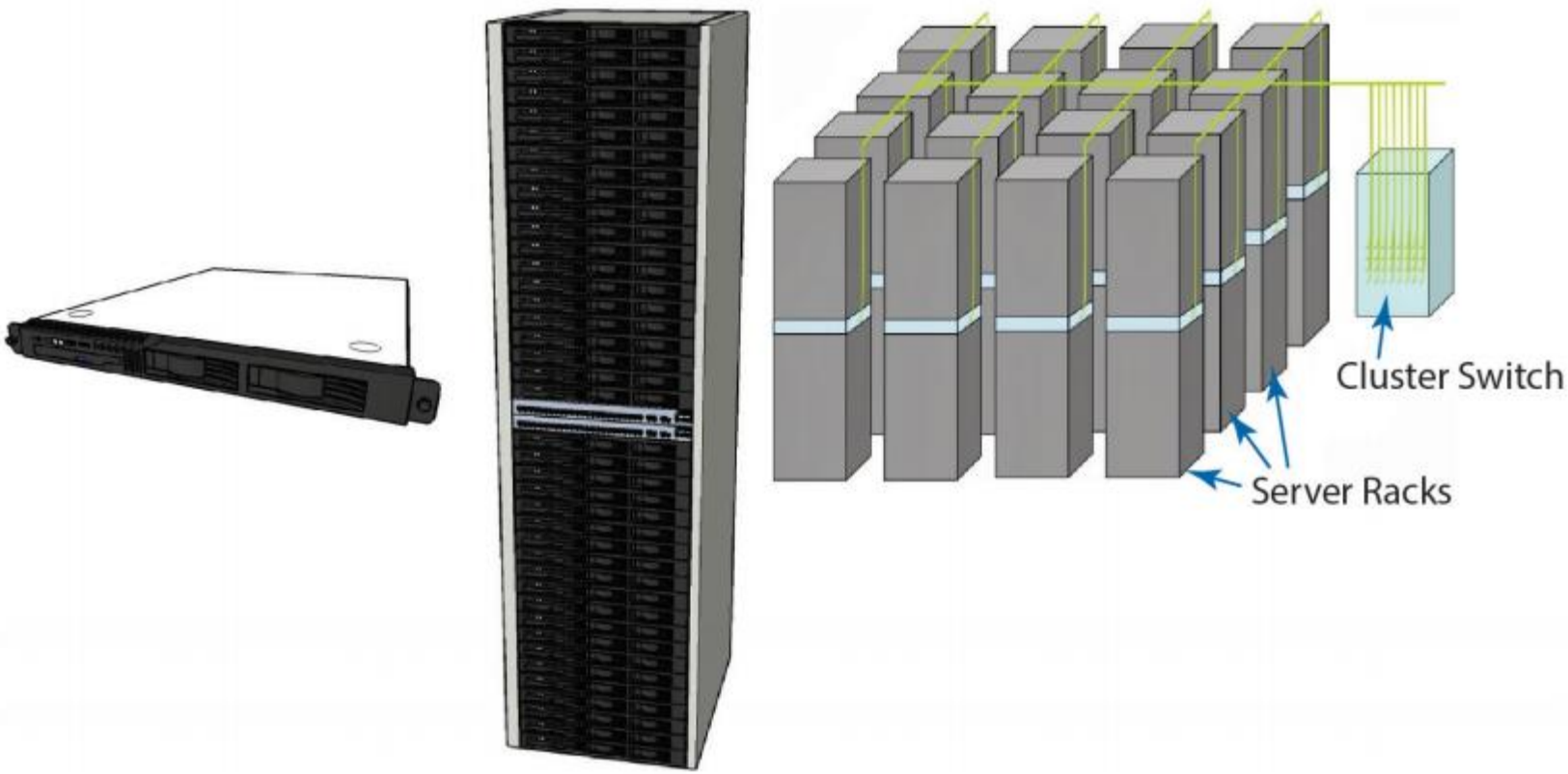
Packaging/Physical Design

- Blades
  - Shared power supply, cooling, etc.

Packaging/Physical Design

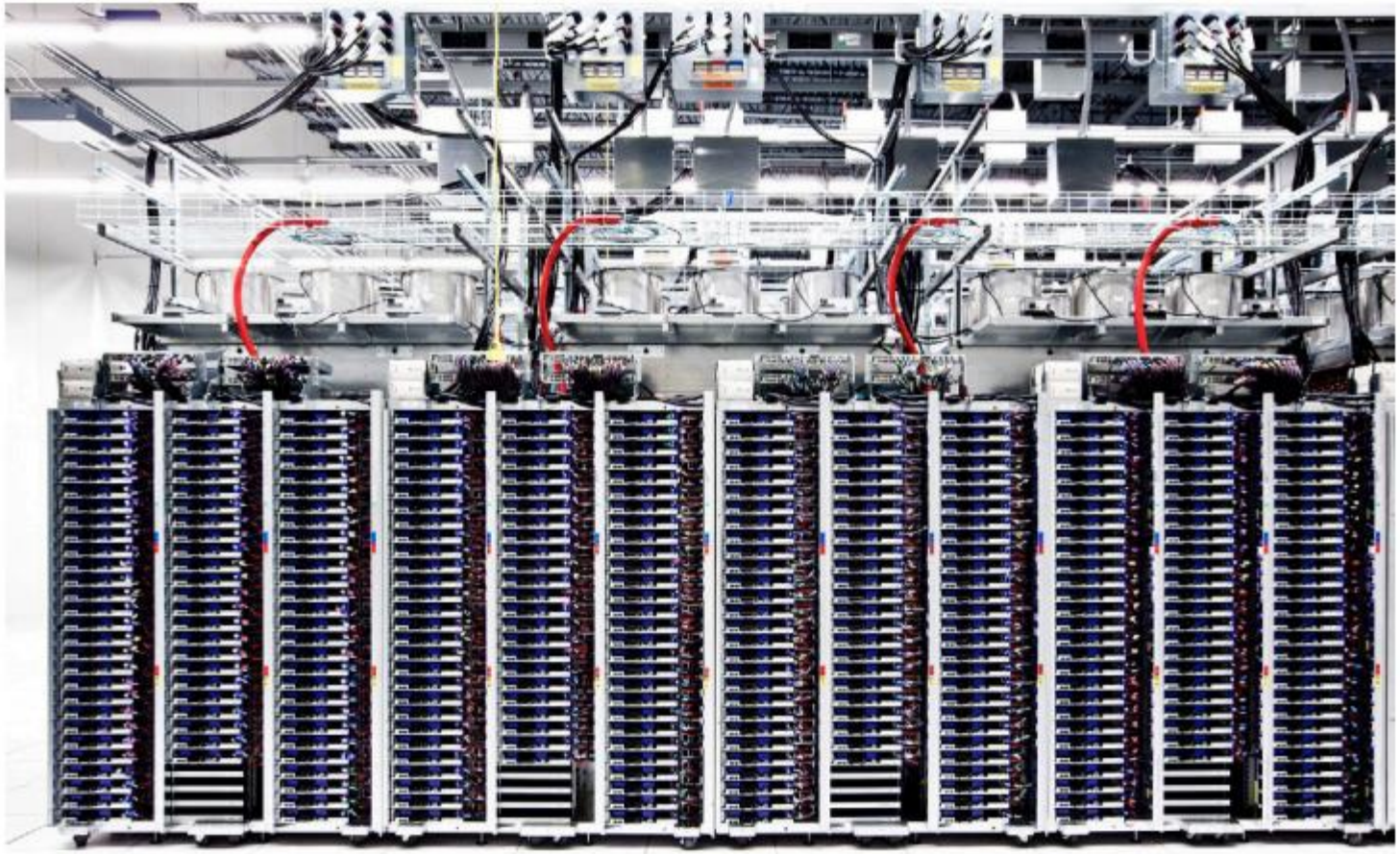
- Sun's Project Black Box (Modular Datacenter)
  - "Portable computing" – up to 17 tons, 280 RU
  - BYOPS, BYOIU
  - No field-replaceable parts – why?

Packaging/Physical Design



- Warehouse scale (from synthesis lecture)
- 1U form factor nodes

Packaging/Physical Design



- Single aisle in Google data center (circa 2012)

Applications

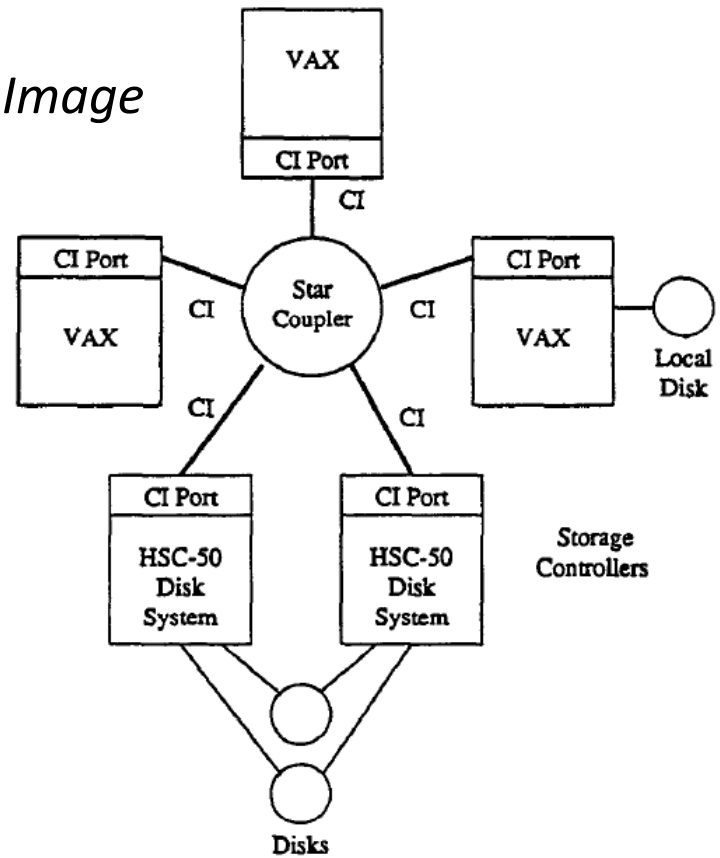
- Commercial
  - Large shared databases, web indices, email archives, …
  - Largely independent threads
  - “Architecture” often means software architecture
  - May use higher performance storage area network (SAN)
- Scientific
  - Inexpensive high performance computing
  - Primarily based on message passing
  - May use higher performance node-to-node network
  - Where HPC clusters end and MPPs begin isn’t always clear

Software Considerations

- Throughput Parallelism
  - As in many commercial servers
  - Distributed OS message passing
  - VAXcluster early example
- True Parallelism
  - As in many scientific/engineering applications
  - Use programming model and user-level API
  - Often augmented with GPGPUs at each node
- Programming Models
  - Message-Passing
    - Commonly used
  - Shared memory
    - Virtual Shared Memory, Software Distributed Shared Memory
    - PGAS – software abstraction, runtime invokes remote DMA
- Of course, a real system can do both throughput and true parallelism

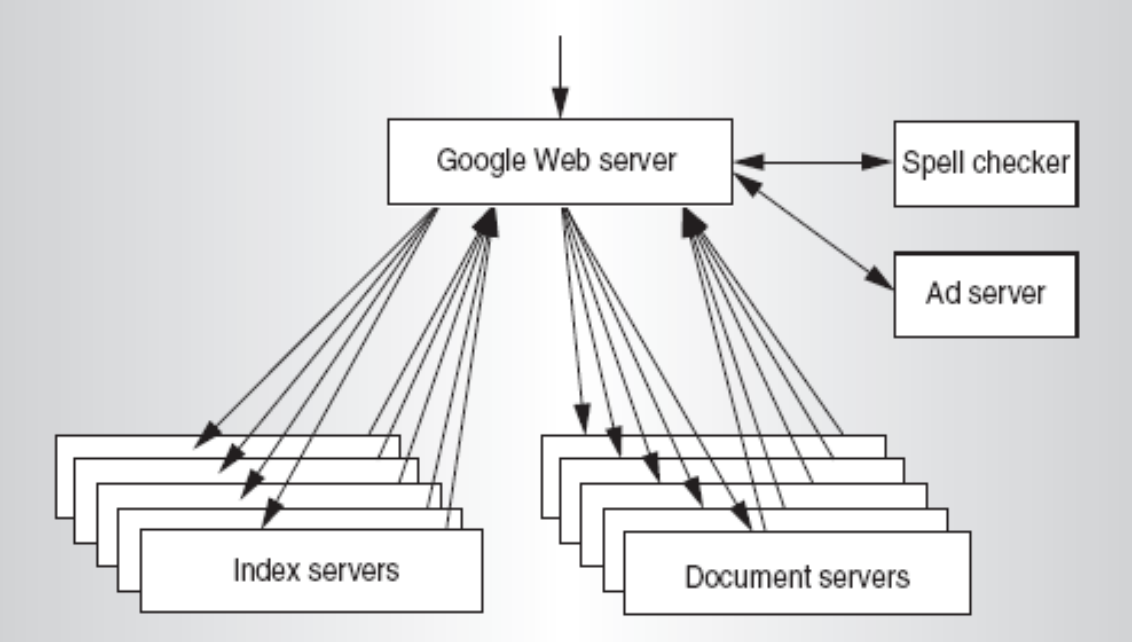
An Early System: VAXcluster

- The first(?) cluster system, circa 1983
- Off-the-shelf computers (VAXes)
  - Plus proprietary network
  - Plus communications interfaces (CI)
- From the user perspective – *Single System Image*
  - User sees same system regardless of processor node being used
  - Single file system
  - Shared devices, print services, etc.
  - Throughput parallelism
- Network
  - Logically a bus, physically a star
  - Uses CSMA (ethernet-like) but different arbitration scheme



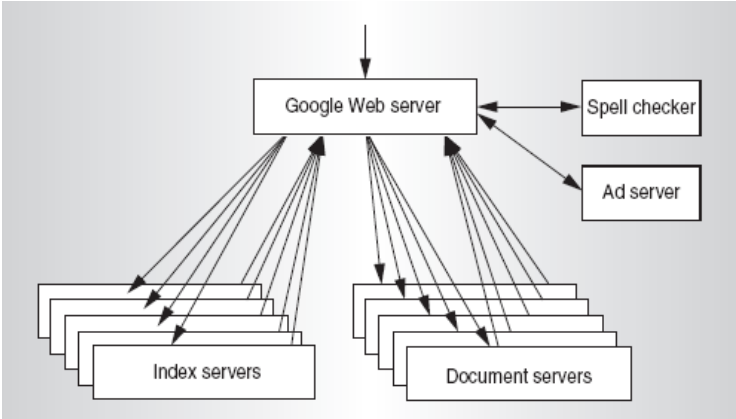
Case Study: Google Server

- Architecture Overview
  - Provide reliability in software
    - Use cluster of PCs
    - Not high-end servers
  - Design system to maximize aggregate throughput
    - Not server response time
    - (Can parallelize individual requests)



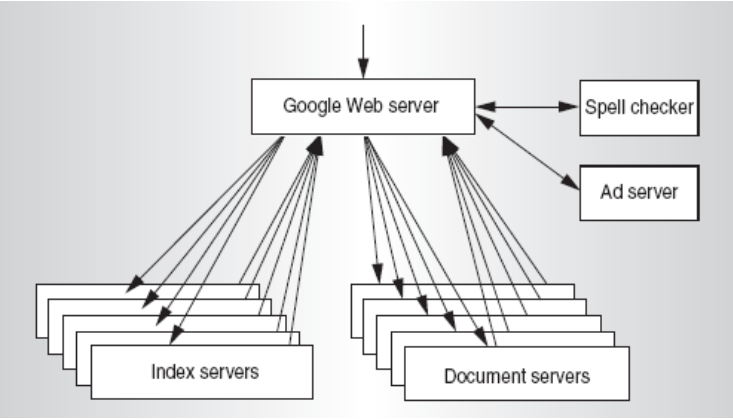
Application Summary

- **Geographically distributed clusters**
  - Each with a few thousand machines
- **First perform Domain Name System (DNS) lookup**
  - Maps request to nearby cluster
- **Send HTTP request to selected cluster**
  - Request serviced locally w/in that cluster
- **Clusters consist of Google Web Servers (GWSes)**
  - Hardware-based load balancer distributes load among GWSes



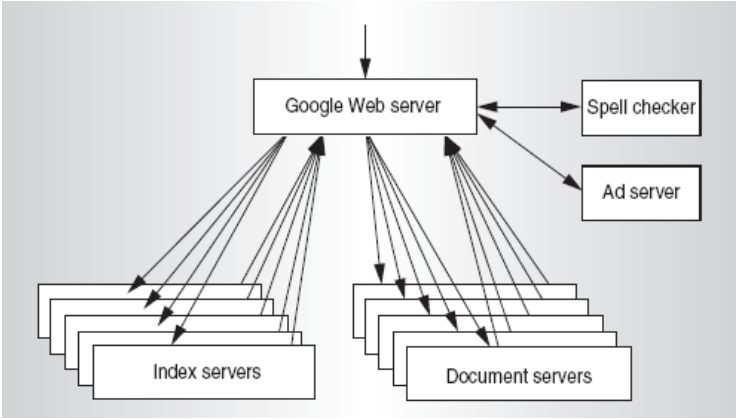
Query Execution

- **Phase 1:**
  - Index servers consult inverted index that maps query word to list of documents
  - Intersect hit lists and compute relevance index
  - Results in ordered list of document ids (*docids*)
- **Both documents and inverted index consume terabytes of data**
- **Index partitioned into "shards", shards are replicated**
  - Index search becomes highly parallel; multiple servers per shard
    load balanced requests
  - Replicated shards add to parallelism and fault tolerance



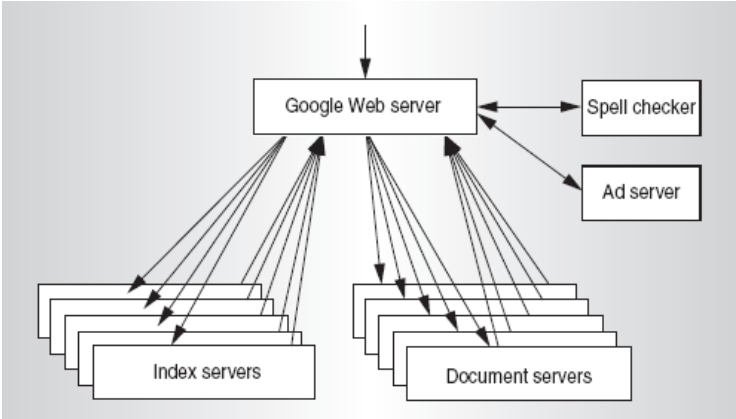
Query Execution

- **Phase 2:**
  - Start w/ docids and determine title, resource locator, document summary
- **Done by document servers**
  - Partition documents into shards
  - Replicate on multiple servers
  - Route requests through load balancers



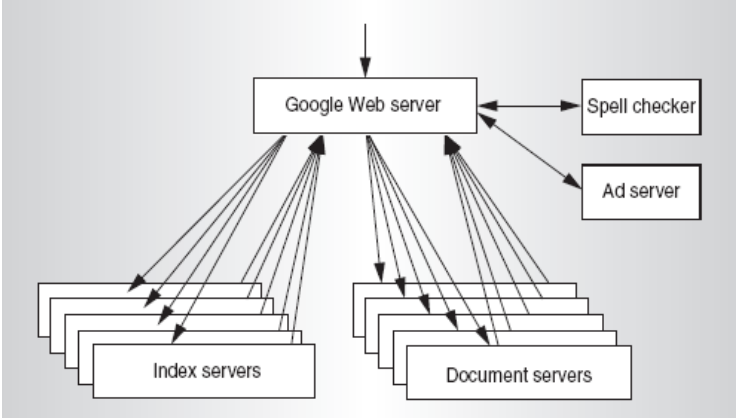
Parallelism/Fault Tolerance

- **High levels of "natural" parallelism**
  - Both inter-query and intra-query
  - Near-linear speedups
- **Replication helps both parallelism and fault tolerance**
  - Permits software-based fault tolerance
  - (The requirements for fault tolerance aren't very high, though)
- **Most accesses are read-only**
  - Combined w/ replication, simplifies updates
  - Queries are diverted during update

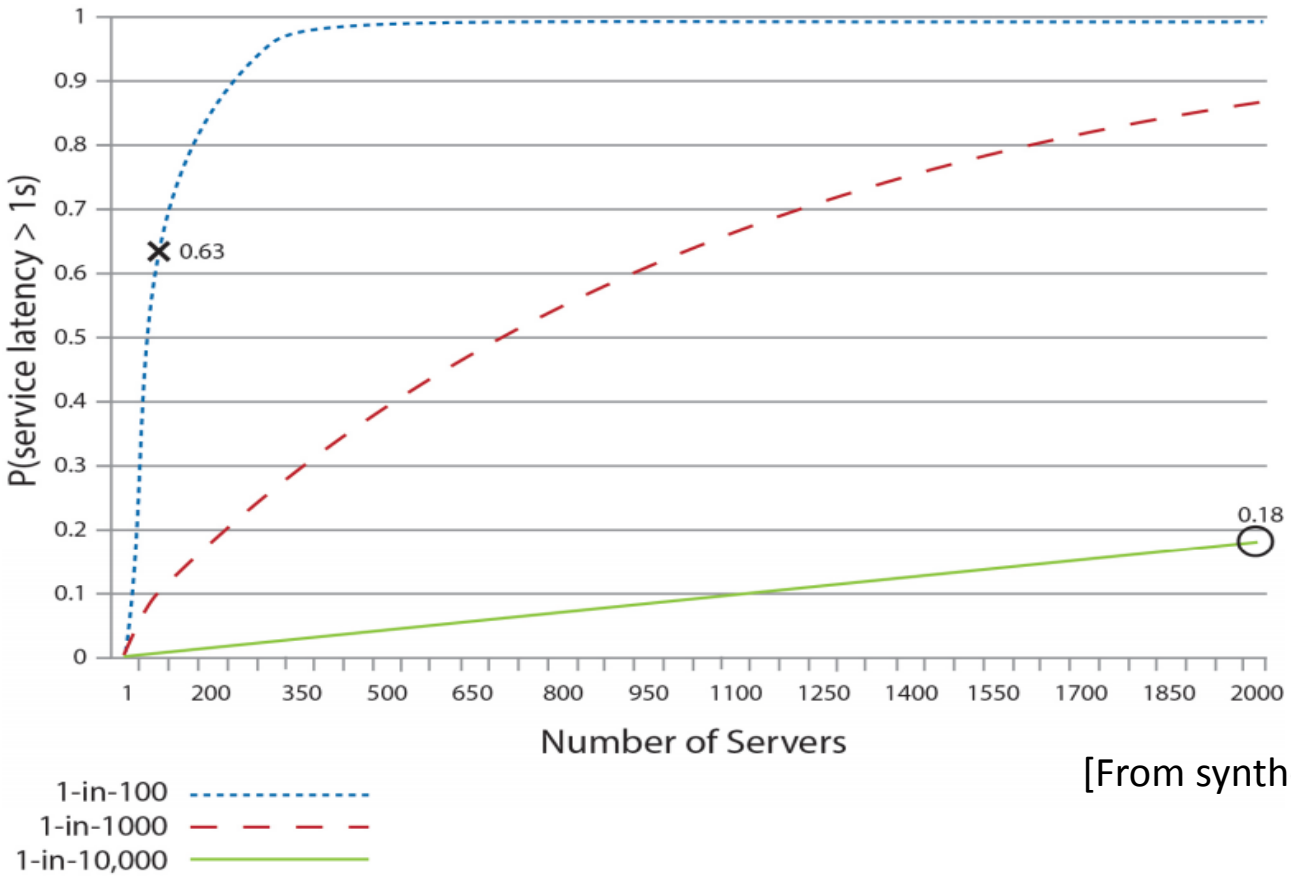


Hardware

- **Performance/Price beats pure Performance**
  - Use dual CPU servers rather than quad
  - Use IDE rather than SCSI (now SATA, PCIe)
- **Use commodity PCs**
  - Essentially mid-range PCs except for disks
  - No redundancy as in many high-end servers
- **Use rack mounted clusters**
  - Connect via ethernet
- **Result is order of magnitude better performance/price than using high-end server components**



Tail Latencies

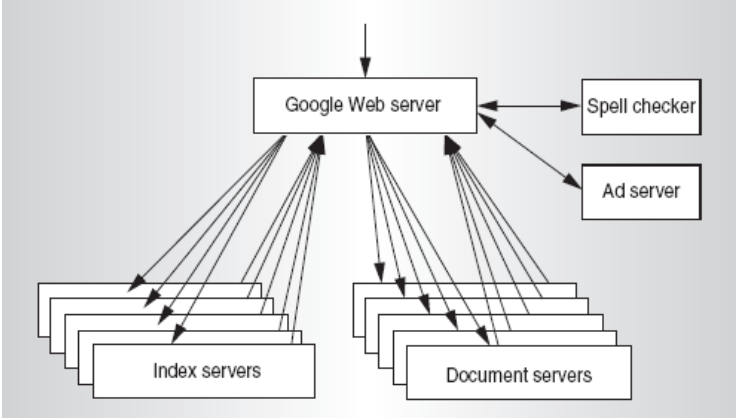


[From synthesis lecture]

- Tail latency: slowest response to parallel query
  - Redundancy (in time: retry, in space: multiple instances)
  - Unpredictable, careful performance engineering needed to avoid

Power

- **Power is a very big issue**
  - Google servers 400 watts/ft^2
  - High end servers 700 watts/ft^2
  - Typical commercial data center – 70-150 watts/ft^2
    $\Rightarrow$ special cooling or additional space, anyway
    using high-end servers would make matters worse



Energy-Proportional Computing

Luiz André Barroso and Urs Hölzle, "The Case for Energy-Proportional Computing," IEEE Computer, Dec 2007.

- Cluster nodes not easy to power up/down
  - Response time due to load variation
  - Persistent state stored at nodes (db shard)
- Usually run under partial/low load
  - Energy use should be proportional to load
  - CPUs pretty good at this (power management)
  - System level is worse (DRAM)
  - I/O not great (disk spin management)
  - Networking gear *was* terrible (all static power)

Application Characteristics

- **For 1GHz Pentium III**
  - Index server
- **Data is percent per instruction retired**
  - Br mispredict – 50 per 1K insts.
  - L1 I miss – 4 per KI
  - L1 D miss 7 per KI
  - etc
- **CPI actually isn't bad**
  - Branch mispredict rate is not very good
  - Data organized for optimal searches will likely have this property
- **Argues for CMP or SMT or both**
  - ILP is limited (1 mispredict per 20 insts)
  - Pentium 4 has twice the CPI and similar branch prediction perf.
  - Observe 30% speedup w/ hyperthreading (at high end of Intel projections)

Table 1. Instruction-level measurements on the index server.

Characteristic	Value
Cycles per instruction	1.1
Ratios (percentage)	
Branch mispredict	5.0
Level 1 instruction miss*	0.4
Level 1 data miss*	0.7
Level 2 miss*	0.3
Instruction TLB miss*	0.04
Data TLB miss*	0.7

* Cache and TLB ratios are per instructions retired.

Memory System

- **For 1GHz Pentium III**
- **Good temporal locality for instructions**
  - Loops doing searches
- **Data blocks**
  - Very low temporal locality
  - Good spatial locality within index data block
    - So data cache performance is relatively good.
- **Memory bandwidth not a bottleneck**
  - Memory bus about 20% utilized
  - Relatively high computation per data item
    - (or maybe just high mispredicts per data item)
  - Modest L2, longer (128 byte) cache lines seem to be a good design point

Table 1. Instruction-level measurements on the index server.

Characteristic	Value
Cycles per instruction	1.1
Ratios (percentage)	
Branch mispredict	5.0
Level 1 instruction miss*	0.4
Level 1 data miss*	0.7
Level 2 miss*	0.3
Instruction TLB miss*	0.04
Data TLB miss*	0.7

* Cache and TLB ratios are per instructions retired.

Architecture Bottom Line

- **Large clusters of small-SMP nodes**
  - Not large scale shared memory MPs
- **Commodity PC components**
- **Characteristics of Google application**
  - Focus on Performance/Price
  - No private state
- **Many web servers (although smaller than Google) have similar characteristics**
- **A lot more details and up-to-date information in Synthesis Lecture from same (Google) authors**

IBM BladeCenter

- **Motivation –**
  - Circa 1999 -- Rack mounted servers becoming difficult
    - 42 1u servers require a lot of cables in a very small space
    - No redundant power supplies/cooling make servicing difficult
- **Solution : Use server blades**
  - Similar to technology used in telecommunications/network switches and hubs
  - Chassis w/ backplane for interconnect (network and power)
  - Shared (and redundant) power supplies/cooling

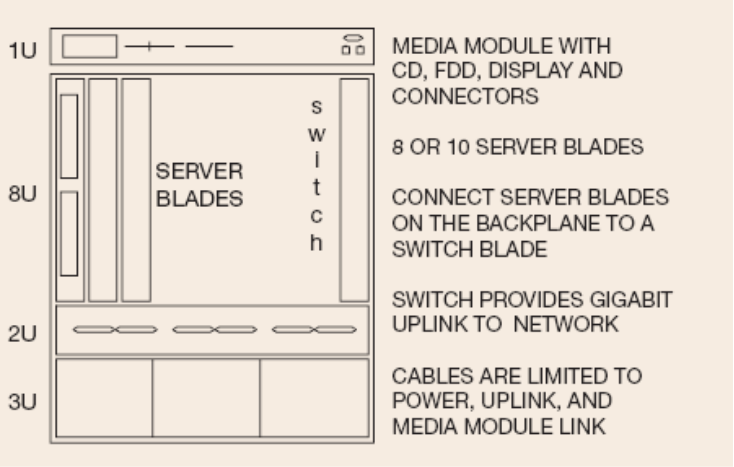
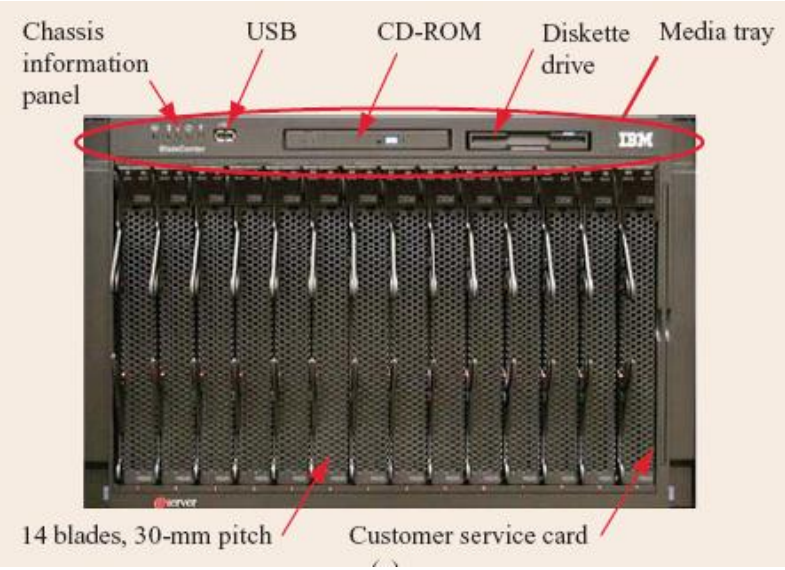


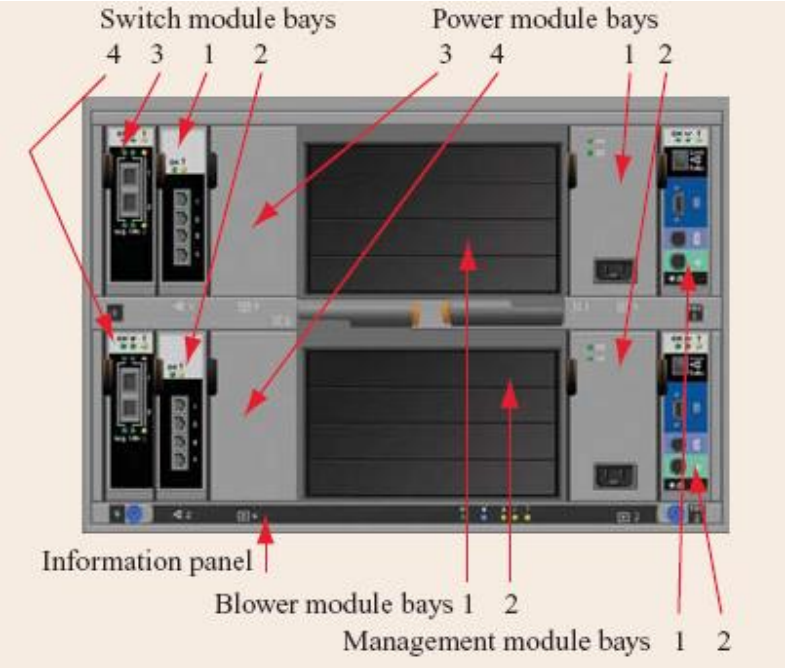
Figure 1

The earliest concept drawing of the BladeCenter architecture, created on October 20, 1999.

Hardware

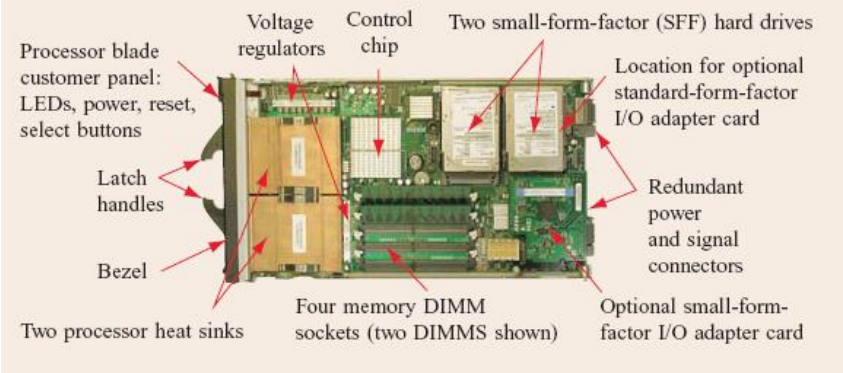
- Midplane
- 14 server blades in front
  - Plus shared media stuff
  - Double density of 1U server
- Switch, Power supply modules, etc. in back
- Airflow front to rear
- Redundant power supplies



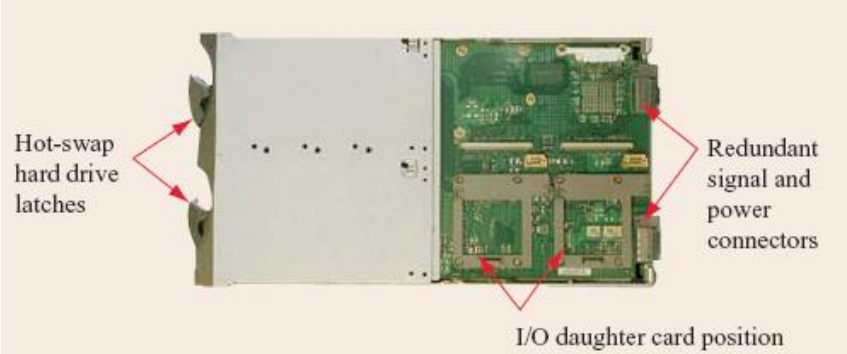


Hardware, contd.

- Processor Blade



- Storage Expansion Blade

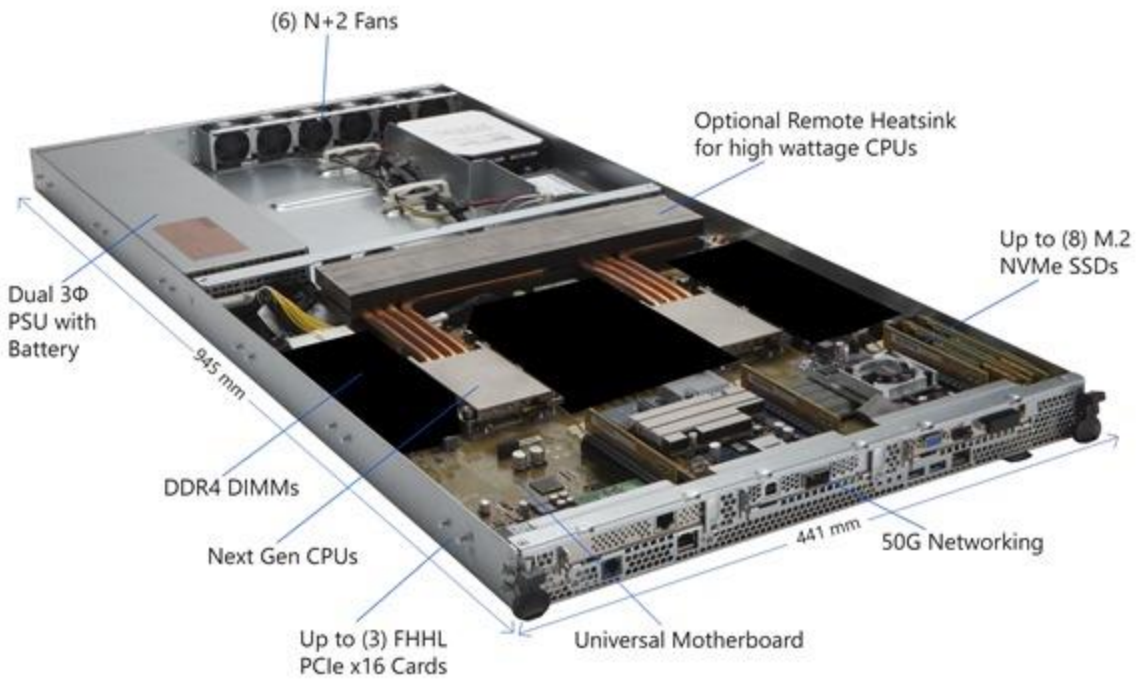


- Switch Module

Open Compute Initiative

- Google succeeded in large part due to very good (proprietary) engineering
- Companies that arrived later struggled to compete: Facebook, Microsoft Bing, …
- Facebook launched Open Compute Initiative
  - Designing everything together leads to many efficiencies
  - Open-source (platform) hardware
    - Compute, network, storage, rack, power
  - Also SW: HipHop, HipHopVM for PHP
- Since embraced by Rackspace, Microsoft, Intel, AMD, etc., even Google participates

Open Compute Initiative



- Microsoft Project Olympus 1U Node
- Chassis, packaging, cooling, etc. all open source
  - Components like CPUs still proprietary

Microsoft Catapult

- Integrate FPGA into each datacenter node
  - Migrate functionality from SW to HW
  - Why? For energy efficiency
- Slides:

  https://www.microsoft.com/en-us/research/wp-content/uploads/2014/06/HC26.12.520-Recon-Fabric-Pulnam-Microsoft-Catapult.pdf

Clusters - Summary

- Reasons for clusters
  - Performance – horizontal scaling
  - Cost: commodity h/w, commodity s/w
  - Redundancy/fault tolerance
- Hardware Challenges
  - Cost, commodity components
  - Power, cooling, energy proportionality
  - Reliability, usability, maintainability
- Software Challenges
  - Programming model
  - Applications with suitable characteristics
  - Load balancing
  - Tail latency