

Distributed Systems
1. Introduction

Paul Krzyzanowski
pkx@cs.rutgers.edu

What can we do now
that we could not do before?

Technology advances

Networking
Processors
Memory
Storage
Protocols

Networking: Ethernet – 1973, 1976

June 1976: Robert Metcalfe presents the concept of *Ethernet* at the National Computer Conference

1980: Ethernet introduced as de facto standard (DEC, Intel, Xerox)

Network architecture

LAN speeds

- Original Ethernet: 2.94 Mbps
- 1985: thick Ethernet: 10 Mbps
1 Mbps with twisted pair networking
- 1991: 10BaseT - twisted pair: 10 Mbps
Switched networking: **scalable bandwidth**
- 1995: 100 Mbps Ethernet
- 1998: 1 Gbps (Gigabit) Ethernet
- 1999: 802.11b (wireless Ethernet) standardized
- 2001: 10 Gbps introduced
- 2005-2009: 40/100 Gbps

348 – >35,000x faster

Network Connectivity

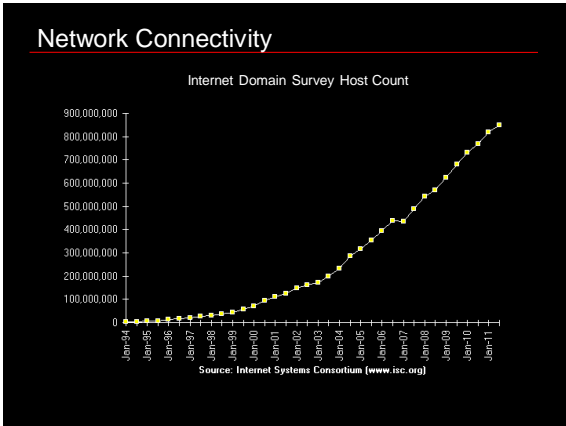
Then:

- Large companies and universities on Internet
- Gateways between other networks
- Consumers had dial-up bulletin boards
- 1985: 1,961 hosts on the Internet

Now:

- One Internet (mostly)
- 2011: 849,869,781 hosts on the Internet
- Widespread connectivity
- High-speed WAN connectivity: >50 Mbps
- Switched LANs
- Wireless networking

848 million more hosts



Metcalf's Law

The value of a telecommunications network is proportional to the square of the number of connected users of the system.

This makes networking socially interesting!

Computing Power

Computers got...

- Smaller
- Cheaper
- Power efficient
- Faster

Microprocessors became technology leaders

Computing Power

1974:	Intel 8080 2 MHz, 6K transistors
2004:	Intel P4 Prescott 3.6 GHz, 125 million transistors
2011:	Intel 10-core Xeon Westmere-EX 3.33 GHz, 2.6 billion transistors

GPUs scaled as well:
NVIDIA GF100: 3 billion transistors
16 processors each with 32 cores

We can no longer make CPUs much faster.
How do we get increased performance? *More cores.*
→Parallel system on a chip

Protocols

Faster CPU →

more time for protocol processing

- ECC, TCP checksums, parsing
- Image, audio compression feasible

Building and classifying parallel and distributed systems

Flynn's Taxonomy (1966)

Number of instruction streams and number of data streams

SISD

- traditional uniprocessor system

SIMD

- array (vector) processor
- Examples:
 - GPUs - Graphical Processing Units for video
 - APU (attached processor unit in Cell processor)
 - SSE3: Intel's Streaming SIMD Extensions
 - GPGPU (General Purpose GPU): AMD/ATI, NVIDIA

MISD

- Generally not used and doesn't make sense
- Sometimes (rarely!) applied to classifying redundant systems

MIMD

- multiple computers, each with:
 - program counter, program (instructions), data
- **parallel and distributed systems**

Subclassifying MIMD

memory

- shared memory systems: **multiprocessors**
- no shared memory: networks of computers, **multicomputers**

interconnect

- bus
- switch

delay/bandwidth

- tightly coupled systems
- loosely coupled systems

Parallel Systems: Multiprocessors

- Shared memory
- Shared clock
- All-or-nothing failure

(Negate the above to device distributed systems)

Bus-based multiprocessors

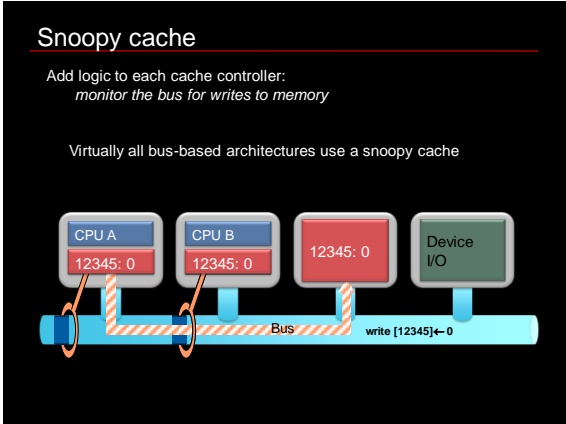
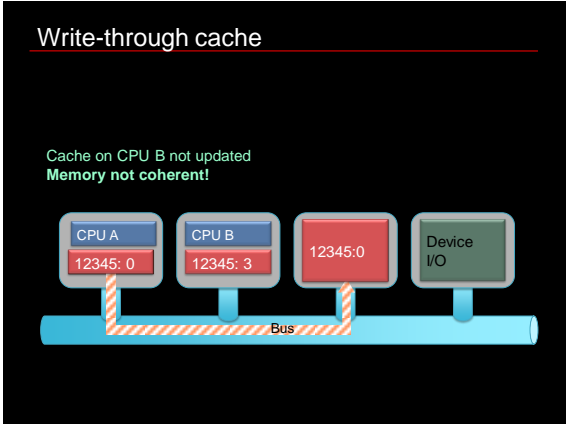
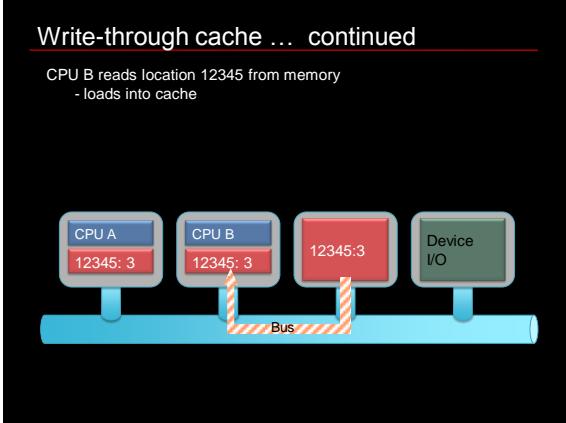
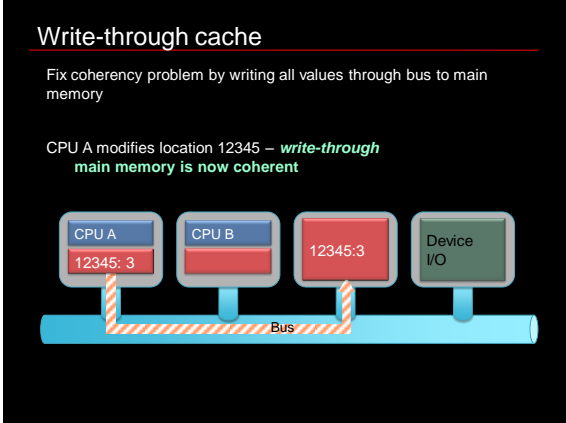
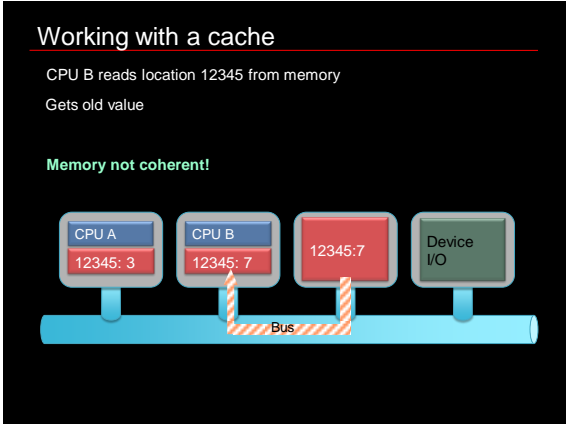
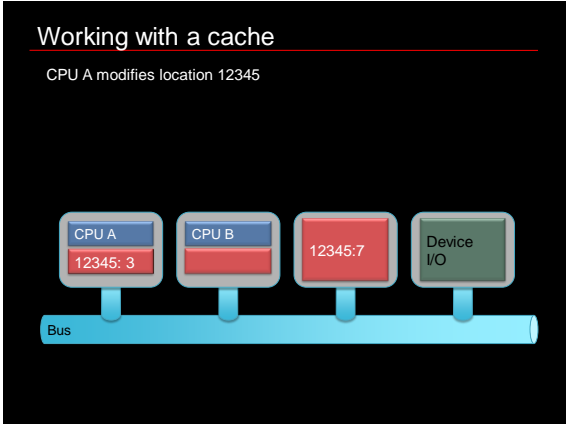
SMP: Symmetric Multi-Processing

Bus-based multiprocessors

- Great idea to deal with bus overload & memory contention
 - Add local memory
- CPU performs I/O to cache memory
 - Access main memory only on cache miss

Working with a cache

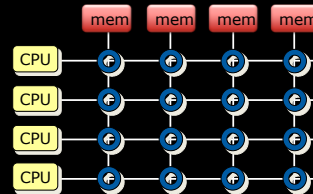
CPU A reads location 12345 from memory



Switched multiprocessors

- A bus-based architecture does not scale to a large number of CPUs (8+)

Switched multiprocessors



n^2 crosspoint switches – expensive switching fabric

NUMA

- Hierarchical Memory System
- Each CPU has local memory: fast access
- Other CPU's memory is in its own address space
 - slower access
- Placement of code and data becomes difficult
 - Operating system has to be aware of memory allocation and CPU scheduling

NUMA

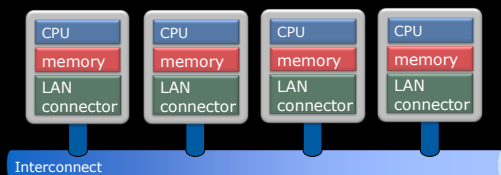
- SGI Origin's ccNUMA
- AMD64 Opteron
 - Each CPU gets a bank of DDR memory
 - Inter-processor communications are sent over a HyperTransport link
- Intel Xeon
 - Integrated Memory Controller (IMC): fast channel to local memory
- Linux >2.5 kernel
 - Multiple run queues
 - Structures for determining layout of memory and processors
- Also supported in Windows Server 2003, Windows 7, Oracle, SQL Server

Distributed Systems

- Multicomputers* or *networked computers*
- No shared clock
- No shared memory
 - Alternate communication mechanism needed
 - Use network interconnect

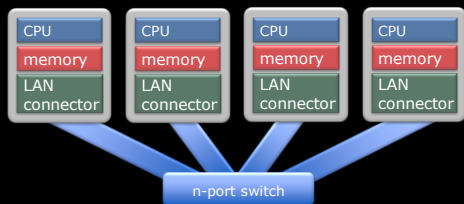
Bus-based Multicomputers

Collection of workstations on a LAN



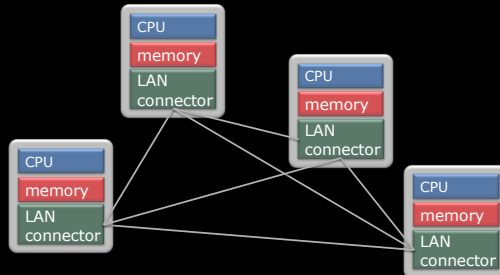
Switched Multicomputers

Collection of workstations on a LAN



Multicomputers on a Network

Logical view: any-to-any communication



What is a Distributed System?

A collection of independent, autonomous hosts connected through a communication network.

- No shared memory (must use the network)
- No shared clock

Single System Image

Collection of independent computers that appears as a single system to the user(s)

- Independent = autonomous
- Single system: user not aware of distribution

You know you have a distributed system when the crash of a computer you've never heard of stops you from getting any work done.

– Leslie Lamport

Why build distributed systems?

How can you get massive performance?

- Multiprocessor systems don't scale
- Disney's Cars 2 required 11.5 hours to render each frame (average) – some took 90 hours to render!
 - 12,500 cores on Dell render blades
- Google
 - Approximately 1 billion queries per day
 - Index >25 billion web pages
 - Hundreds of thousands of servers

Why build distributed systems?

- Performance ratio
 - Scaling multiprocessors may not be possible or cost effective
- Distributing applications may make sense
 - ATMs, graphics, remote monitoring
- Interactive communication & entertainment
 - work and play together:
 - email, gaming, telephony, instant messaging
- Remote content
 - web browsing, music & video downloads, IPTV, file servers
- Mobility
- Increased reliability
- Incremental growth

Problems

Designing distributed software can be difficult

- Operating systems handling distribution
- Programming languages?
- Efficiency?
- Reliability?
- Administration?

Network

- disconnect, loss of data, latency

Security

- want easy and convenient access

Design issues

Reliability

- **Availability:** fraction of time system is usable
 - Achieve with redundancy
- **Reliability:** data must not get lost
 - Includes security

Performance

- Communication network may be slow and/or unreliable

Scalability

- Distributable vs. centralized algorithms
- Can we take advantage of having lots of computers?

Service Models

Centralized model

- No networking
- Traditional time-sharing system
- Direct connection of user terminals to system
- One or several CPUs
- Not easily scalable
- Limiting factor: number of CPUs in system
 - Contention for same resources

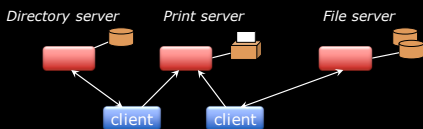
Client-server model

Environment consists of **clients** and **servers**

Service: task machine can perform

Server: machine that performs the task

Client: machine that is requesting the service



Workstation model

assume client is used by one user at a time

Peer to peer model

- Each machine on network has (mostly) equivalent capabilities
- No machines are dedicated to serving others
- E.g., collection of PCs:
 - Access other people's files
 - Send/receive email (without server)
 - Peer-to-peer file sharing
 - SETI@home computation

Processor pool model

- Collection of CPUs that can be assigned processes on demand
- Render farms

Cloud Computing

- Provide users with seamless access to:
 - Storage capacity
 - Processing
 - Network bandwidth
- Heterogeneous and geographically distributed systems
- Build a "supercomputer" on the fly via networked, loosely coupled computers

Cloud Computing

Resources are provided as a network (Internet) service

- Software as a Service (SaaS)
- Google Apps
- Salesforce.com

Multi-tier client-server architectures

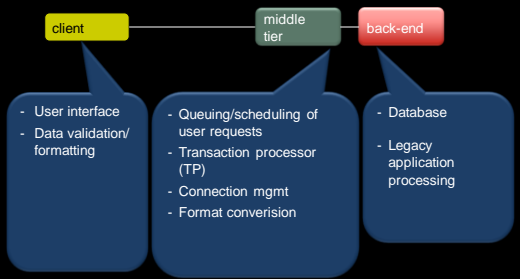
Two-tier architecture

Common from mid 1980's-early 1990's

- UI on user's desktop
- Application services on server

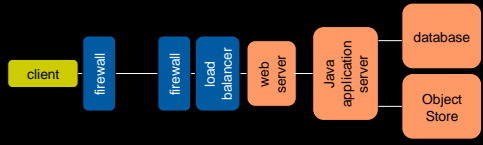


Three-tier architecture



Beyond three tiers

Most architectures are multi-tiered



The End