Federated Architectures

4

Rich Martin
09/07/99

Overview

- What this seminar is about
- Background
- History
- Readings, Assignments, project
- Break
- Scalability
- Availability
- Managability

Seminar

- Examining an emerging spectrum of parallel/distributed machine
- Today's servers consist of many processors, interconnects, memories tied together in an ad-hoc manner
- Questions in this class:
  - Appropriate applications?
  - Why bother?
  - Programming models?
  - Unified architecture?
  - Unified OS?

How to build Scalable, Available, Managable servers?

- Technology Trend:
  - Processors are so cheap, can put in anything
  - What does this mean for severs?
- Wide range of machine classes with multiple levels of processors/memories
  - Computers inside computers
  - Raids, diagnostic systems, packet filters, etc
- Multiple levels of software
  - Programming difficult

What we'll examine in this seminar

- Range of scalable servers:
  - Symmetric Multiprocessors (SMP)
  - Massively Parallel Processors (MPP)
  - Database machines
  - Fault tolerant machines
  - Clusters
  - Federated Architecture
- System/programming interfaces
- Applications (two themes)
  - Performance
  - Fault Tolerance

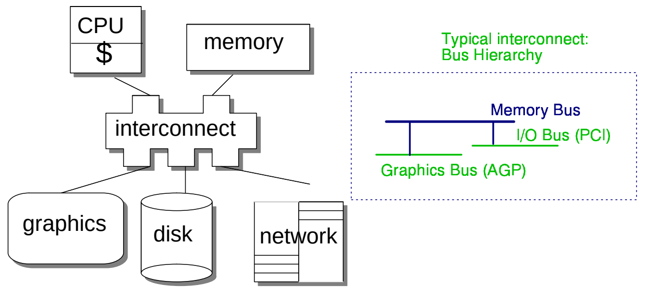
Theme for the class

- Scaling hardware is "easy" for the class of machines we'll study
- How to program a configuration is an extremely difficult task
- Q: What's the "right" hardware/software configuration for a given task?
  - Can system software make this easier?

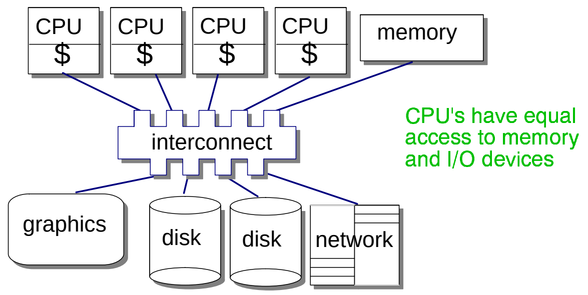
Background

- What were all those machines again?
  - Symmetric Multiprocessors (SMP)
  - Massively Parallel Processors (MPP)
  - Clusters
  - Database machines
  - Fault tolerant machines
  - Federated Architecture
- Extracting similarities/differences

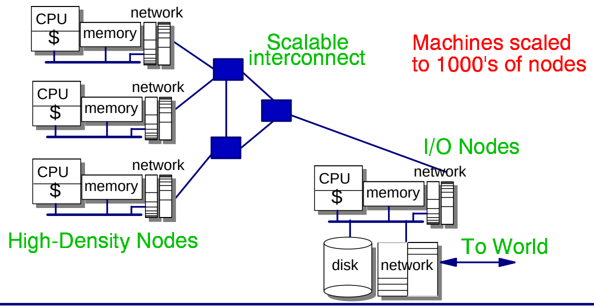
Model of a typical computer



Bigger computers: Symmetric MultiProcessor



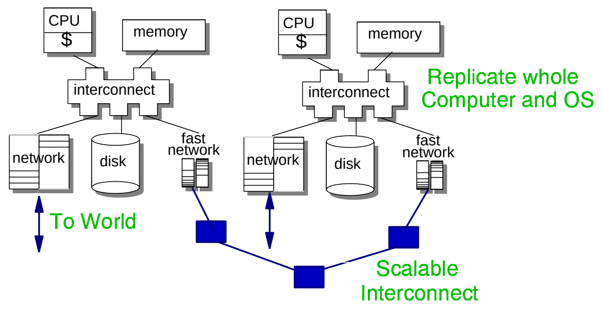
Still bigger: Massively Parallel Processor



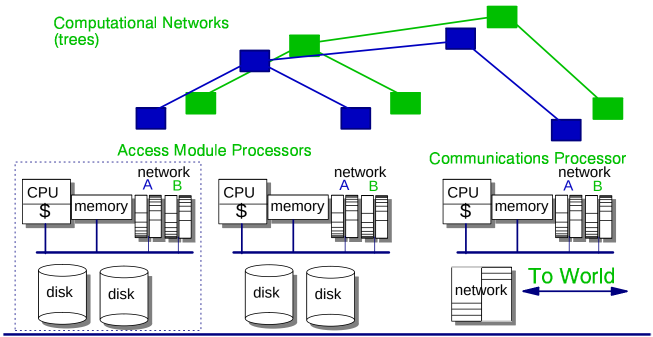
MPP Software

- Nodes often custom design
  - CPU/memory commodity
  - Network interface full-custom
- Custom μ-kernel on nodes
- "Space sharing" machine typical
  - One program on a set of nodes at once
- Some coarse timesharing (CM-5)
- I/O nodes execute kernel code only
- Message passing for communication

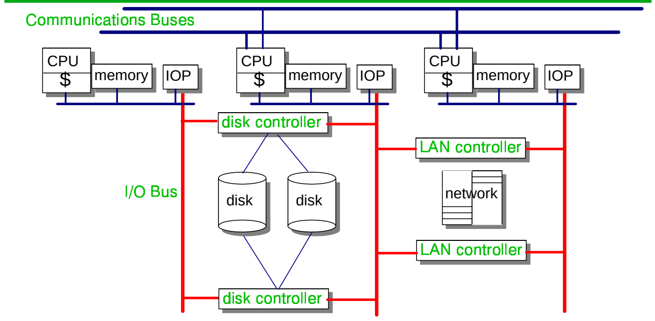
1990's way: Cluster



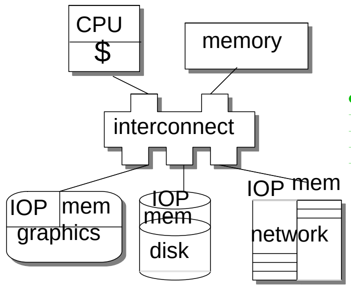
Database machine
(Teradata DBC)



Fault Tolerant Machine
(Tandem Nonstop II)



Emerging
Federated Architecture



Regular computer
on the outside

- I/O processors are small computers
  - packet filters
  - texture mapping
  - disk scheduling
  - remote management

Why it's "Federated"

- Master CPU's and OS control access and data path
- I/O processors control caching and move the data through local sub-systems
- Outside/Inside box becomes more blurred than in today's machines with unified control model

Brief History

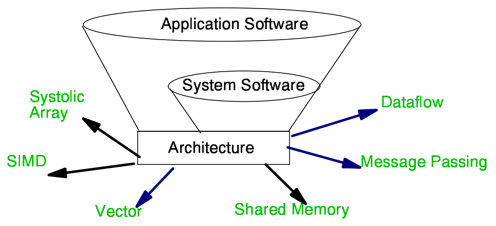
- Why study history?
  - Evaluate ideas in context of the era.
  - Why are so many ideas recycled?
  - What is the "essence" of an idea?
- Three communities deal with "simultaneous" programming (EE too!)
  - Parallel programming
  - Distributed system/OS
  - Database

Parallel Machines
Late 1980's

- Focus on computation
- Parallel prog. will/must become mainstream
- No consensus on architecture or programming model
  - Physical machine architecture mirrored programming model
- Cray-3 going to eat everyone's lunch

Parallel Machines '80 divergence

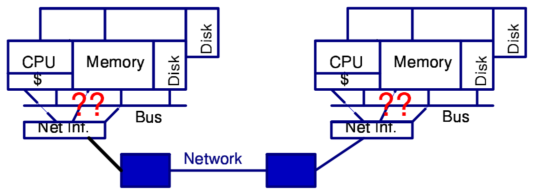
- Divergent models and machines
  - Religious wars



Parallel Machines Early 1990's

- Cray-3: Built 1. No takers.
- SIMD: Too hard to keep using special VLSI
  - CM2->CM5
- DASH: Demo'ed real shared memory scaling
- Teradata & Tandem not "parallel machines"
  - Not general, only run database apps
- MPP's: King
  - CM5, T3D had huge influence

Parallel Machine Convergence



- A set of complete computers connected via a high performance interconnect!

Parallel Machines Mid 1990's

- MPP's move to chapter 11
  - Thinking Machines, KSR bankrupt
  - Intel closes supercomputing division
  - Clusters eating their lunch
    - Similar enough hardware architecture for 32-100
- Rise of big cc-NUMA SMP's
    - scalable to 64-128
    - ccNUMA programming more like MPP than SMP
- Clusters Emerging

What happend?

- Parallel computing never was mainstream
  - Probably will always be specialized
- Why bother?
- More in some dimension (some scaling rule)
  - faster execution for constant problem size
  - bigger data set as we grow machine
  - increased throughput (transactions/hour)
  - more robust execution?
    - Add nodes --> less downtime

Parallel Computing Successes

- Backoffice business apps:
  - SMP's running BIG databases
    - Transactions/sec (Visa, Sabre, E*trade,"TPC-C")
    - Decision support (Wal-Mart, Amex, "TPC-D")
- Multimedia
  - DSP algorithms are array based
  - Vectors comming back (MMX,VIS)
- "Front end" service applications:
  - Born on clusters (Yahoo/Inktomi/Deja)

Distributed Computing

- Roots in client/server architecture world of the 1980's
  - "protocols"
    - File systems: NFS
    - Databases: ODBC
  - Example Operating Systems
    - Sprite, Amoeba, Locus, Emerald
    - Grapevine
- Fault tolerant systems
  - Tandem
  - Isis, Horus

Database Machines

- Relational Database provided a unified model
  - "everything is a table"
- SQL provided standard language
  - Small set of closed operators
  - Implicit parallelism
- Allows system desiger many ways to exploit parallelism for performance!
- Not seen as "successful" (Teradata?)

Seminar Structure

- 2-3 readings every week
- "Observations" due on reading once a week
- Project at end of the class
  - 8+ page paper on a systems project/experiment
  - Ideas due in October
  - Progress checkpoint end of November
  - Final write up due end of the semester

Observations

- 1/2-1 page write up on your thoughts on the readings for the week.
- Don't just summarize the abstract and conclusions
- Good themes:
  - why did the idea succeed/fail?
  - related ancedotes
  - alternate method to test the idea
  - weaknesses in the paper
  - cross-cutting issues
  - future work

Project

- Substantial systems related project
- Work in small groups (2-3)
- Examples:
  - Parallelize an important application on an Cluster/SMP/FCA
  - Quantify failure/recovery of certain applications
  - Separate part of OS into independent sub-system for performance/fault tolerance
  - Develop a solid model for managing clusters

What you should know before you take this class

- Senior or grad operating systems
- Strong understanding of I/O architectures and how I/O works in a real machine
- Some parallel programming
  - μ-quiz: What are the following?
    - DMA
    - Interrupt
    - Frame buffer
    - Socket
    - Barrier
    - Spin lock

Where R.U.?

- You're sitting at your desk and have a window on paul via telnet/rlogin. You type the letter "A". Describe the steps taken by all the systems involved in the process from when you hit the key until the character shows up on your screen.

- More detail is better (a page or two at most)
- Email to me by next class.

Scalability, Availability, Manageability

- What this seminar is about
- Background
- History
- Readings, Assignments, project
- Break
- Scalablility
- Availability
- Manageability

Scalability

- Newer services require fast growth (hours) while the service remains available
  - IRS web site April 14th
  - October market crash of '87
  - E*trade crashes

- Must scale down too.

- Short term SMP scalability?
  - "Fork lift" upgrade

What does scalability mean?

- Problem Constrained (PS rule)
  - Fixed problem, more resources
- Memory Constrained (MS rule)
  - Add resources -> bigger problem
- Time Constrained (TS rule)
  - E.g. How much can you sort in a minute?
- User defined (pages, ops/sec, etc)
  - E.g. TPC-C/SPEC style scaling rules
    - More ops/sec -> constant response time with larger dataset

Basic Speedup Formula

- Performance is Work/Time
- So speedup of P more units in the machine over 1 unit is:

$$\text{Speedup} = \frac{\frac{\text{Work}(P)}{\text{Time}(P)}}{\frac{\text{Work}(1)}{\text{Time}(1)}}$$

Problem Constrained Scaling

- For a fixed problem size (C) scale the machine to decrease execution time
- *Latency is the implicit metric*
  - Was seen as what parallel computing was all about
  - Very limited view of the world!
  - Example: Time to Sort 100 Million records

$$\text{Speedup}_{pc} = \frac{\text{Work}(C)}{\text{Time}(P)} X \frac{\text{Time}(1)}{\text{Work}(C)} = \frac{\text{Execution time}(1}{\text{Execution time}(P}$$

Time Constrained Scaling

- Execution time is kept fixed as we scale the machine
  - Often useful metric, as we have some maximum time to wait for an answer
  - Example: How much can it sort in a minute?

$$\text{Speedup}_{tc} = \frac{\text{Work(P)}}{\text{Time(t)}} X \frac{\text{Time(t)}}{\text{Work(1)}} = \frac{\text{Work(P}}{\text{Work(1)}}$$

Memory (resource) Constrained Scaling

- Speedup is determined by addition of resource as we scale the machine
  - User wants to see speedup as we max the machine out in some dimension
  - Example:
    - Machine holds 1 million records on 1 node
    - How fast can we sort 10 million records on 10 nodes?

$$\text{Speedup}_{mc} = \frac{\text{Work(P)}}{\text{Time(P)}} X \frac{\text{Time(1)}}{\text{Work(1)}} = \frac{\text{Increase in Wor}}{\text{Increase in Tim}}$$

User-Defined Scaling

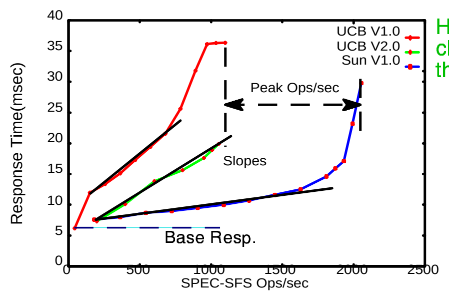
- TPC-C/ SPECweb/SPECsfs style rules

(1) Define Operations/sec
  - DB updates/sec
  - Web Requests/sec
  - NFS reads/writes/attrib checks per second

(2) Define upper limit on response time
  - SPECsfs -> 40 msec

(3) Scaling rule
  - Increase by X operations per second means:
    - increase in overall data set
    - increase in data operated on

SPECsfs Example



How will the curves change as we scale the machine?

Ideal Scaling

- More demand -> add a resource
  - Heavier network traffic->add network interfaces
  - More disk capacity -> add disks
  - More disk accesses -> add disks
  - More memory needed -> plug in memory
- Should work in the reverse direction too
- Service remains available

Advantage of Clusters

- Isolation of resources
  - Allows incremental scaling up/down
  - Increased fault isolation => increased availability?
    - Power
    - Operating system
    - Application error
- Closer to ideal scaling
  - Easier to size system to load
  - Easier to size up/down
  - Balance of resources

Cluster Disadvantage

- Lack of single image makes management a nightmare
- Many distributed systems issues
  - Consistency?
  - Naming?
  - Performance?
- Global services hard to come by!
  - Global OS?
  - Global Filesystem?
  - Global Storage system?
  - Global Networking layer?

Availabilty

- How do you operate in the presence of system faults?
- Hardware errors
- Software errors
  - Firmware
  - Device drivers
  - Scheduler
  - VM system
  - Network stacks
- Operator errors

Hardware Errors

- Model sytem with X defects per N units
- Used to be very important
  - LSI-level integration had 100's of chips per board
  - several boards per computer
    - Chance of error compounded
- Modern systems have much higher levels of integration
  - Design bugs more likely than random errors
- Can we treat them the same?

Software Errors

- Modern systems have millions of lines of code!
  - Application
  - Compiler
  - Operating System
  - Device Firmware
- Interfaces are choke points, much simpler than the system behind it
- Can we limit catastrophies by design?

Software Errors (cont)

- Defined interfaces are key in dealing with complexity
  - System Calls
  - Vnodes
  - Streams
  - Device Drivers
  - ODBC, NFS, HTTP
- Can we build software fault tolerant at the interface level?
  - fault containment
  - error recovery

Operator Errors

- Recognized as important, but not addressed well.
  - rm -rf * .tmp
    - vs rm -rf *.tmp
  - kill -9 1
    - vs kill -9 %1
  - Unplug wrong cable ....
- A systems issue? User interface issue?
- Can we build systems tolerant to operator error?

Manageability

- What does it mean to be "manageable"
  - Operator interventions per day/week/month?
  - Human time spent on machine?
  - Support $/hour uptime?
- Hard to quantify
  - Currently more an art than science
- Field is still wide open to research ideas!

Ideal Manageability

- What is the "ideal" manageability?
  - How to add resources?
    - identification
    - configuration
    - integration
  - How to remove resources?
- Maintainance?
  - What happens when something breaks?
- Q: How much effort is it to add a new disk to a system?
- Upgrade a device driver?

Availability/Manageability tie

- Systems today are brittle, easy to break
  - Can't even upgrade from 10 to 100Mb ethernet without service interruption
  - Add disk/memory/cpu?
  - Replace disk/memory/cpu?
- Software is worse
  - replace a device driver?
  - Reboot windows after a package install?

How can we get it all?

- Scalability of a cluster
- Availability of a Tandem
- Manageability of an SMP