CSE 552

Distributed and ~~Parallel~~ Systems

Intro

- http://www.cs.washington.edu/homes/bershad/cs552
  - online course notes
- cs551 is a prereq
  - will assume the readings, although some review
- readings must be done before class
  - substantial fraction of grade depends on submitted course summaries
- participation is critical

Topics

- Introduction to distributed systems
- Messaging and communication
- Distributed programming
- Reliability
- Some network services
- Mobile code
- Disconnected operation and mobile computing

Today's Discussion

- Background on distributed systems
  - more information comes from first two readings
- Distributed vs. parallel
  - and why we're only going to really be looking at one of these

History

- 1950: programmers get their own machine
- 1960: batch programming, glass walls
- 1970: time sharing, ARPAnet
- 1980: personal computing, INTERNET
- 1990: distributed systems, WWW
- 2000: ubiquitous internetworking
  - other speculations?

5

What is a distributed system?

A distributed system is a collection of independent computers that appear to the users or user as a single computer

- Multiple processing elements
- Interconnection hardware
- Independent failure mode
- Shared state

we'll exploit this

- what about these examples?
  - shared memory multprocessor?
  - Ethernet with packet filtering bridge?
  - NFS?
  - the WWW?

6

Failure containment vs. failure prevention

- Easier to contain failures for a few than to mask from all
  - "You may not be able to get your work done, but I can!"
- Less stringent definition
  - multiple processing elements
  - interconnection hardware
  - failure containment

Distributed Systems Enabled by Two Technologies

- Cheap processors
  - improvements in processor technology and speed
    - th 10 dollar rolls royce
  - implication is you can buy more
- Cheap networks
  - fast serial lines
  - local area nets
  - wide area nets
  - implication is you can hook up more

Distributed Systems Delayed by One Technology

- Software
  - or rather, the lack of it
- A distributed system is more complicated than a centralized system
- So, why do it?
  - economics
  - this means that someone believes it is cheaper to distribute functionality across many processors than to centralize

The Benefits

- Performance
  - but, remember LOCUS
- Scalability (incremental growth)
  - heterogeneity over space and time
  - scalability involves naming, algorithms, mechanisms (implementation)
- Availability
  - can you get to the service regardless of the path
- Reliability
  - any answer vs the wrong answer
  - sometimes no answer is the right answer
- Inherent distribution
  - walmart, cscw, the web
  - still, an economic argument

The Costs

- Here is what you may have to pay
  - increased software complexity
  - poorer performance in the easy cases
  - more people time
  - network costs
  - security
  - weird programming models
  - bizarre application semantics

Why does a distributed system “hurt”

- interconnection of independent components
  - if A works, and B works, A and B may not work
    - example: sendmail
- interference between previously isolated components
  - If A works, and B works, then A or B may not work
    - loaded ethernet, swamped receiver with UDP
- effects of scale
  - centralized component (mail server)
  - centralized tables (a single name server)
  - centralized algorithms -- require complete information
    - optimal load balancer, sequential consistency
- partial failure
  - A can talk to B can talk to C, but A can’t talk to C
  - A trust B and C trust A but B is corrupt

Economics make distributed programming hard?

- Example: train tracks and narrow valleys

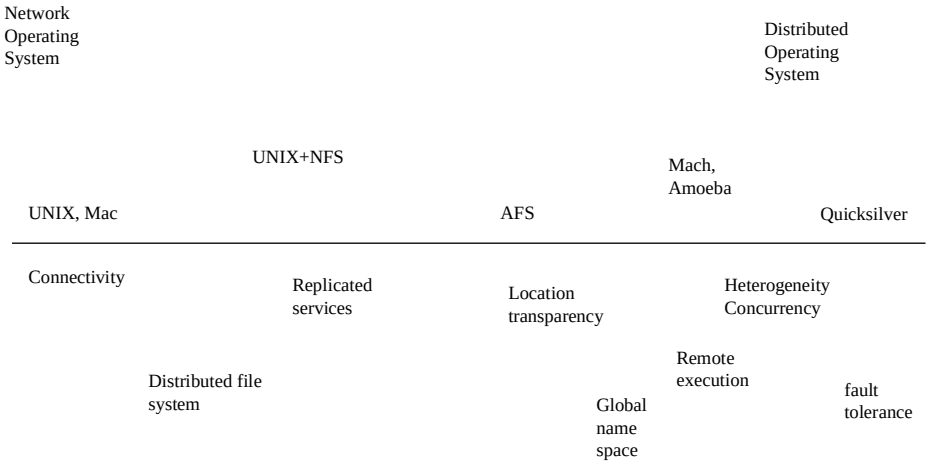
13

Useful techniques

- Global name space
- Replicate to increase availability and reliability
  - read replication is a big win
- Trade availability for consistency
- Use hints wherever possible
  - ARP… usually in the context of naming. can afford to get it wrong
- Stashing for autonomous operation
  - use local first, remote as a last resort
- Exploit locality with caches
- Timeouts for revocation
  - reduce network and server load
- Use a standard remote invocation mechanism
- Encryption for authentication and security
- Leverage existing infrastructure

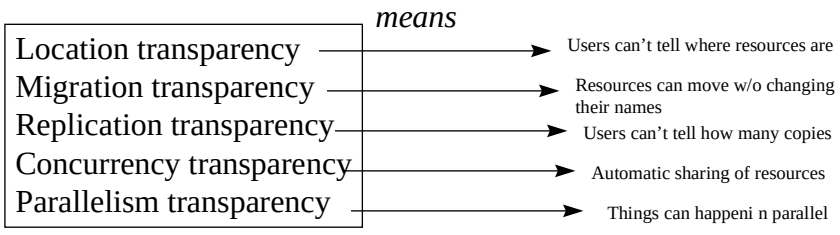
14

A Software Taxonomy

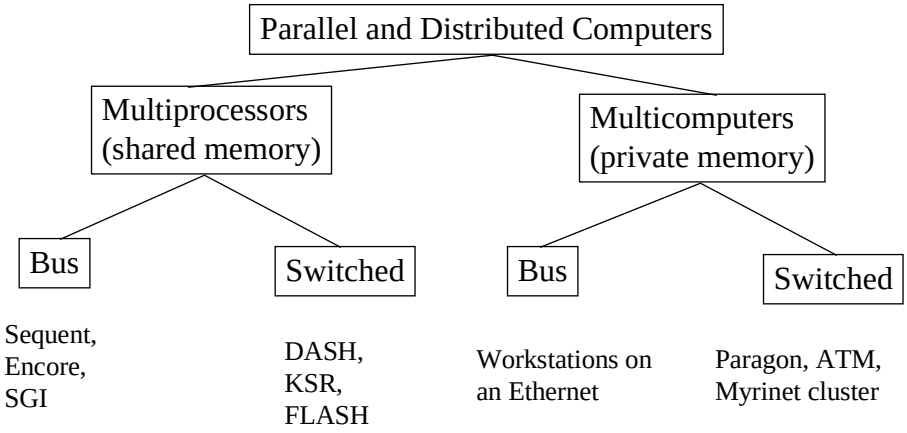


Transparency is a key concern

- "appears…"
- There are different kinds of transparency



A Hardware Taxonomy



LATENCY IS THE KEY DETERMINANT

D vs. P

- This is going otbe a course in distributed systems
- If you make the latency small enough, it becomes a course in parallel systems
  - except…
    - Some issues are latency independent
      - eg, programming models, threads
    - but others are not
      - reliability

Readings for Thursday

- Two intro papers (discussed today)
- The grapevine paper
  - available outside my office
  - early distributed system
    - shows why even easy things can be hard