Distributed Systems

Introduction

Paul Krzyzanowski
pxk@cs.rutgers.edu
ds@pk.org

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**: 100 Gbps (over optical link)

Network Connectivity

439 million more hosts

Then:

- large companies and universities on Internet
- gateways between other networks
- dial-up bulletin boards
- 1985: 1,961 hosts on the Internet

Now:

- One Internet (mostly)
- 2006: 439,286,364 hosts on the Internet
- widespread connectivity
  High-speed WAN connectivity: 1- >50 Mbps
- Switched LANs
- wireless networking

Computing power

Computers got…

- Smaller
- Cheaper
- Power efficient
- Faster

Microprocessors became technology leaders

9,000x cheaper
4,000x more capacity

year	$/MB	typical
1977	$32,000	16K
1987	$250	640K-2MB
1997	$2	64MB-256MB
2007	$0.06	512MB-2GB+

Storage: disk

131,000x cheaper in 20 years
30,000x more capacity

Recording density increased over 60,000,000 times over 50 years

1977: 310 KB floppy drive – $1480
1987: 40 MB drive for – $679
2008: 750 GB drive for – $99 ($0.13 / GB)

Music Collection

4,207 Billboard hits
- 18 GB
- Average song size: 4.4 MB

Today
- Download time per song @12.9 Mbps: 3.5 sec
- Storage cost: $5.00

20 years ago (1987)
- Download time per song, V90 modem @44 Kbps: 15 minutes
- Storage cost: $76,560

Protocols

Faster CPU →
more time for protocol processing
- ECC, checksums, parsing (e.g. XML)
- Image, audio compression feasible

Faster network →
→ bigger (and bloated) protocols
- e.g., SOAP/XML, H.323

Why do we want to network?

- 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

"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."

Coupling

Tightly versus loosely coupled software

Tightly versus loosely coupled hardware

Design issues: Transparency

High level: hide distribution from users
Low level: hide distribution from software

- **Location transparency**:
  users don't care where resources are
- **Migration transparency**:
  resources move at will
- **Replication transparency**:
  users cannot tell whether there are copies of resources
- **Concurrency transparency**:
  users share resources transparently
- **Parallelism transparency**:
  operations take place in parallel without user's knowledge

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?

Martin Fowler on Distribution Strategies

- First Rule of Distributed Object Design : Don't distribute your objects!
- Transparency between components allows for multiple distribution strategies - usually at the cost of performance.
- Minimize the amount of inter-process collaborations.
- Fine-grained interfaces internally
- Coarse-grained interfaces at the distribution boundaries.

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

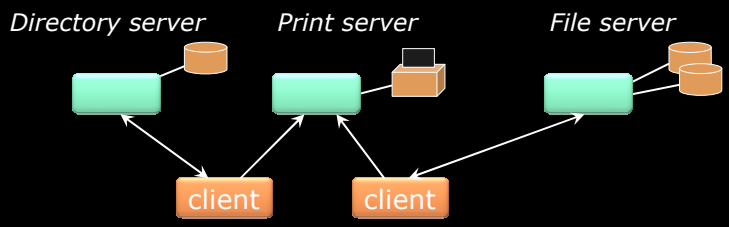
Client-server model

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)
  - Gnutella-style content sharing
  - SETI@home computation

Processor pool model

What about idle workstations (computing resources)?

- Let them sit idle
- Run jobs on them

Alternatively…

- Collection of CPUs that can be assigned processes on demand
- Users won't need heavy duty workstations
  - GUI on local machine
- Computation model of Plan 9

Grid computing

Provide users with seamless access to:

- Storage capacity
- Processing
- Network bandwidth

Heterogeneous and geographically distributed systems

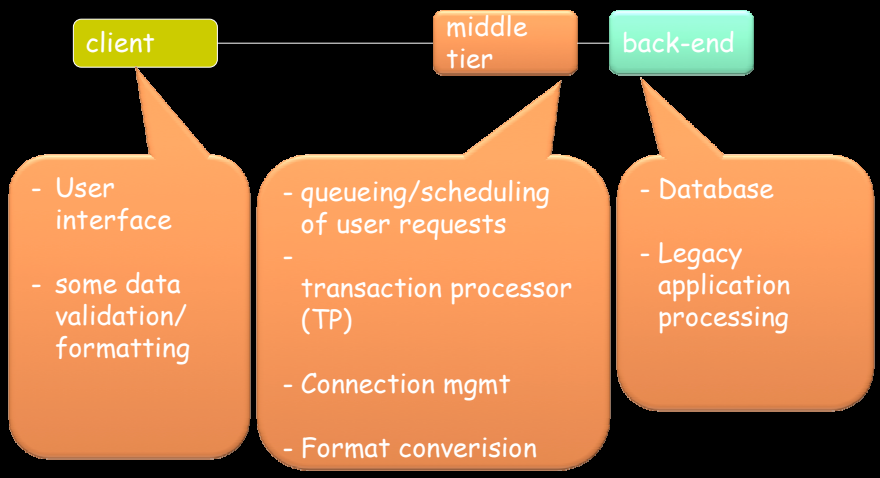
Multi-tier client-server architectures

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