Distributed Systems

Today

- Definition
- Goals and pitfalls
- Grapevine – an early example

What is a *distributed system*?

- Very broad definition
  - A collection of independent, interconnected processors that communicate and coordinate their action by exchanging messages
  - A collection of independent computers that appears to its users as a single coherent system
- Why do you want one?
  - Resource sharing – both, physical resources and **information**
  - Computation speedup – to solve large problems, we will need many cooperating machines
  - Reliability – machines fail frequently
  - Communication – people collaborating from remote sites
  - Many applications are by their nature distributed (ATMs, airline ticket reservation, etc)

Loosely-coupled systems

- Most distributed systems are "loosely-coupled"
- Each system is a completely autonomous system, connected to others on the network
- Even today, most dist. systems are loosely-coupled
  - Each CPU runs an independent autonomous OS
  - Computers don't really trust each other
  - Some resources are shared, but most are not
  - The system may look differently from different hosts
  - Typically, communication times are long

Closely-coupled systems

- A DS becomes more “closely-coupled” as it
  - Appears more uniform in nature
  - Runs a “single” operating system
  - Has a single security domain
  - Shares all logical resources (e.g., files)
  - Shares all physical resources (CPUs, memory, disks, printers, etc.)
- In the limit, a distributed system looks to users as a centralized timesharing system, but built of a distributed set of hardware and software components

Tightly-coupled systems

- A "tightly-coupled" system usually refers to a multiprocessor
  - Runs a single copy of the OS with a single job queue
  - Has a single address space
  - Usually has a single bus or backplane to which all processors and memories are connected
  - Has very low communication latency
  - Processors communicate through shared memory

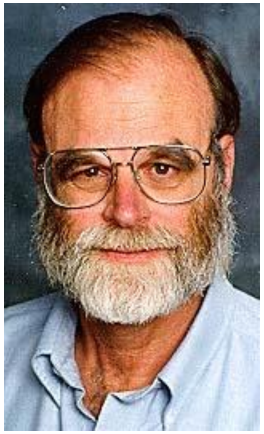
Distributed systems challenges

- Making resources available
  - The main goal of DS – making convenient to share resources
- Security
  - Sharing, as always, introduces security issues
- Providing transparency
  - Hide the fact that the system **is** distributed
  - Types of transparency
    - Access – What's data representation?
    - Location – Where's the resource located?
    - Migration – Have the resource moved?
    - Relocation – Is the resource being move?
    - Replication – Are there multiple copies?
    - Concurrency – Is there anybody else accessing the resource now?
    - Failure – Has it been working all along?
  - Do we **really** want transparency?

Distributed systems challenges

- Openness
  - Services should follow agreed-upon rules on component syntax & semantics
- Scalability
  - In numbers (users and resources), geographic span and administration complexity
  - Some useful techniques
    - Asynchronous communication
    - Distribution
    - Caching/replication
- Adding to the challenges, common false assumptions
  - The network is reliable / secure / homogenous
  - The topology does not change
  - Latency is zero / Bandwidth is infinite /Transport cost is zero
  - There is one administrator

Distributed computing economics*



- From a 2003 article by Jim Gray
  - American computer scientist (1944-2007)
  - First Ph.D. from Berkeley CS
  - Touring award winner 1999
  - Do you know what *transactions* are?
- Computing is free
  - SETI@Home is a 54 teraflop machine (2003), beating the top 4 of the top500 supercomputers
  - Google supports trillion of searches per day
  - Hotmail carries trillion of emails per year
  - Amazon offers free book searches
  - …
- Well, not really free; advertising pays for most of it

* J. Gray, Distributed Computing Economics, ACM Queue, May/June 2008

Distributed computing economics

- Computing is not free, actually it costs millions
  - Hundreds of billions of dollars/year in hardware
  - More than 1 trillion/year in TCO (operation >> capital)
- Megaservices have low operation staff costs
  - In 2002, Google had a operational staff of 25 managing its 2 petabyte database and 10k distributed servers
  - But not all can benefit from megaservices economies of scale
    - Companies report needing 1 admin per TB/100 servers/gigabit bandwidth
  - And outsourcing doesn't work for most things
    - Except high-tech, low-touch businesses …
    - that is ~identical across most companies (email, payroll, web hosting …)
  - SETI@Home is an exception as it sidesteps operation costs and is not funded by advertising
    - Harvests wasted cycles and pays with screen saver & feel-good

Distributed computing economics

- Web services
  - MS, IBM, Amazon and others argue much of the traffic will be computer/computer interaction
  - Web service – a software system designed to support inter-operable machine/machine interaction over the network
    - Its interface is described in a machine-processable format in WSDL (Web Service Description Language)
    - A new computing model – Internet-scale distributed computing
    - HTTP Internet is for people/computer interaction
- Web services can reduce the cost of publishing & receiving information
  - Other services interact with them through XML-encoded SOAP messages sent over HTTP
  - Cheaper programming and management with an information-structuring model

Distributed computing economics

- Grid and computing-on demand
  - Enable mobile apps that can be provisioned on demand
  - Most computational tasks can be made mobile if written in a portable language using portable interfaces – hard to achieve
- Assuming we solved this, what about economics?
  - Computation task has four demands
    - Networking - delivering questions/answers
    - Computation – transforming information to produce new information
    - Database access – access to reference information needed by the computation
    - Database storage – storing information away for later use
  - Ratios among these and their relative costs are pivotal
  - The ideal mobile task is stateless, has tiny network i/o and huge computational demand
    - **YES** SETI@home
    - **NO** Simulation of crack propagation in mechanical objects
  - Put computation near the data (filter data early)

Extracting guarantees from chaos*

- The peer-to-peer vision – leverage the resources of thousands to billions of participants to provide
  - Durability
  - Anonymity
  - Scalability
  - Security
  - …
- How realistic is this? Let's look at a typical application – a distributed file service
  - Idea – replace the local hard disk with a pool of storage spread throughout the network
  - A few examples – FreeNet, Gnutella, FreeHaven, Oceanstore, …

* Based on J. Kubiatowicz, Extracting guarantees from chaos, CACM 46(2), Feb. 2003.

Goals and challenge

- Desired properties for a distributed file service
  – Availability – you can get your data 24x7
  – Durability – And if you put it there it will stay there forever
  – Access control – Information is protected, from both unauthorized reads and writes
  – Authenticity – adversaries cannot substitute a forge document for a requested one
  – Denial of Service resilience – it's difficult for an adversary to compromise availability

Goals and challenge

- Other, more general goals
  - Massive scalability – Thousands to millions of users
  - Anonymity – Difficult for an outsider to ascertain who has produced a document and who has examined it
  - Deniability – Users can deny knowledge of data stored on their machines
  - Resistance to censorship – No one can censor information once it is written to the system
- The problem – an unreliable and untrusted infrastructure without knowledge of the underlying platform
  - Most of the systems are not professionally managed
  - Participants could be adversarial
  - Running over a large, shared black-box

Taming the chaos …

- Fault tolerance through replication
  – Plain replication can be expensive, erasure coding may help
- Location-independent routing
  – Assign a unique identifier to objects and nodes, and route cooperatively
- Cryptography – Protecting the authenticity (source) and integrity (correctness) of information
  – Encryption for privacy
  – Authenticity and integrity through one-way hash functions and signatures
- Byzantine agreement – How to allow a set of peers to agree on an action even in the presence of adversaries?

Taming the chaos

- Correlated failure analysis – Failure independence does not hold (same network, same OS, …)
  - Clustering based on pair-wise correlations?
- Leveraging the differences – Some peers are "more equals" than others, try to leverage that
  - Superpeers for routing
  - Actively manage nodes for Byzantine agreement, …
- Learn from others
  - Reduce stress on the underlying infrastructure by leveraging other systems perspectives
- Going for probabilistic guarantees
  - Stability through statistics – SETI@home asks multiple peers to perform identical computation and excludes bad results through voting

More? Take EECS 345

- Introductory course on distributed systems covering topics such as
  - Building blocks for distributed systems
  - Naming
  - Synchronization
  - Replication and consistency
  - Fault tolerance
  - Security
- Mix of lecture-/seminar- based course
  - I'll introduce a topic on Tuesdays
  - You'll present and we'll all discuss a research paper on Thursdays
  - You'll work on one large-project throughout the quarter