Welcome to Distributed Systems

Today

- Welcome to distributed systems
- Distributed systems definition, goals and challenges
- Course goals and organization

What is a *distributed system*?

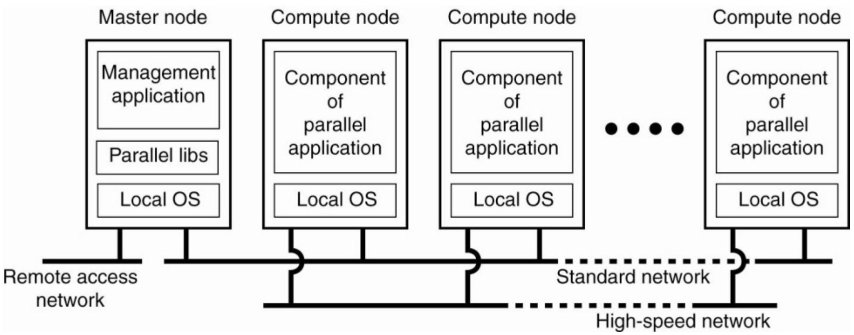
- Very broad definition
  - A collection of independent, interconnected processors that communicate and coordinate their action by exchanging messages
  - A collection of independent computers that appears to its users as a single coherent system
- Why do you want one?
  - Resource sharing – both, physical 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)

Example classes of distributed systems

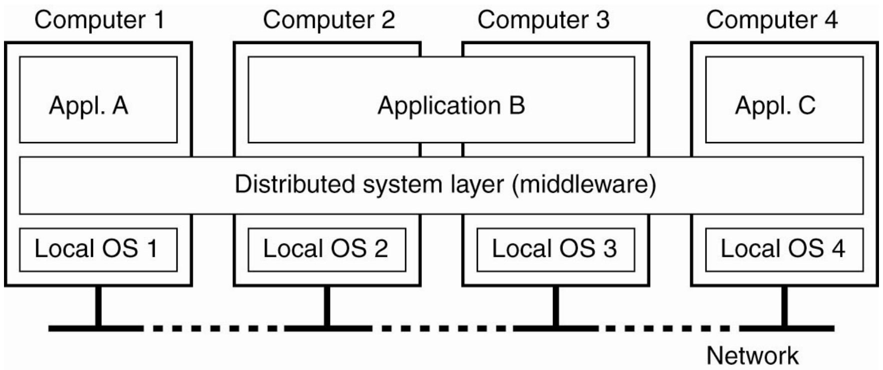
- Distributed computing systems
  - Commonly used in high-performance computing
  - Clusters
  - Grids
- Distributing information systems
  - Distributed transaction systems
  - Enterprise application integration
- Distributed pervasive systems
  - Home systems
  - Health care
  - Sensor networks

Cluster computing systems

- Collection of similar off-the-shell workstations, running their own OS, interconnected by a high-speed LAN
- Beowulf cluster configuration
  - Master node provides interface to user and handles job allocations

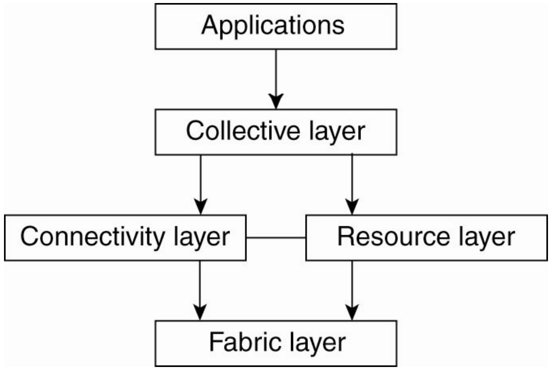


- MOSIX like-system
  - Single-system image



Grid computing systems

- Grid systems - each part potentially under a different administrative domain, hardware/software/network
- Key issue – sharing resources across organizations
  - Thus, much pain goes into standards and interfaces
- Virtual organizations – people with access rights to common resources from different organizations
- An early example architecture for grids
  - Fabric – interfaces to local resources
  - Connectivity – communication protocols for supporting transactions using multiple resources
  - Resource – management of a single resource
  - Collective – handle access to multiple resources



Distributed information systems

- Organizations have multiple networked applications – how to integrate them?
- For database-oriented application & at the lowest level
  - Transactions – set of operations with ACID properties
    - Atomic – all or nothing at all
    - Consistent – if consistent before, consistent after
    - Isolated – concurrent transactions don't interfere with each other
    - Durable – once it's done, it's permanent
  - Nested transactions for distributed systems
    - Permanence? Only for the top-level transaction
- To integrate applications independent from their databases
  - Different communication models: RPC, RMI, MOM, …

Distributed pervasive systems

- Traditional distributed systems
  - Fixed nodes with +/- permanent and good connectivity
- Pervasive systems
  - Small, battery-powered, mobile and wirelessly connected
- Some key requirements
  - Embrace contextual changes – aware of change and able to adapt
  - Encourage ad-hoc composition – different use in different contexts
  - Recognize sharing as a default
  - Be self* – we cannot assume there's a sys admin to go to

Examples of pervasive systems

- Home systems
  - More than a bunch of PCs and printers, TVs, PDAs, microwave oven, coffee maker, …
  - Need for self* is clear
  - Can you provide personal spaces with tons of storage?
- Electronic health care systems
  - Various sensors on a person, interconnected by a body-area network
  - Where to store monitoring data? How to prevent data losses? How to facilitate physicians interaction? Security and privacy?
- Sensor networks
  - 10-1000x of small nodes, each with sensing devices, wirelessly connected
  - A distributed information processing system
  - In-network processing – save communication/energy while leveraging aggregation

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
- Openness
  - Services should follow agreed-upon rules on component syntax & semantics for interoperability and portability
- Scalability
  - In numbers (users and resources), geographic span and administration complexity

Challenges – Transparency

- Types of transparency
  - Access – What's data representation? Connecting machines with different architectures and file-name conventions
  - Location – Where's the resource located? Naming is key
  - Migration – Have the resource moved?
  - Relocation – Is the resource being move?
  - Replication – Are there multiple copies?
  - Concurrency – Is anybody else accessing the resource now?
  - Failure – Has it been working all along?
- Do we **really** want transparency?
  - Impossible – remote controlling a space ship
  - A bad idea – creating false expectations
  - Against application's goals – pervasive computing and location awareness

Scalability problems

- In numbers of users and resources, geographic span and administration complexity
- Scalability in numbers - limiting features
  - Centralized services – a single server for all users
  - Centralized data – a single HOSTS file for the Internet
  - Centralized algorithms – routing with complete information
- Characteristics of decentralized algorithms
  - No machine has complete information about the system state
  - Machines make decisions based only on local information
  - Failure of one machine does not ruin the algorithm
  - There is no implicit assumption that a global clock exists

Scalability problems

- Geographic scalability – from LANs to WANs
  - Synchronous communication, where requesting client blocks until it gets a response, makes it hard to scale
  - Communication is unreliable and nearly always point-to-point – e.g. broadcast for locating a service doesn't work
  - Centralized solutions are clearly an issue as well
- Administration complexity
  - Conflicting policies, with respect to resource usage, management and security, must be handle
  - E.g. Can you trust your sys admin? Can you trust users from another domain?

Scalability techniques

Three general classes of techniques for scaling

- Hiding communication latencies
  - Key for geographic scalability
  - How? Asynchronous communication, shipping code
- Distribution
  - Break up and distribute the system – e.g. Domain Name System, World Wide Web
- Replication/Caching
  - To increase availability, balance load and avoid communication latencies
  - The drawback – consistency

Challenges & pitfalls

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*

This class – topics we will cover

- Distributed systems architectures
- Wide-area distributed systems and PlanetLab
- Communication – RPC, message oriented, …
- Processes – client, servers and migrating code
- Naming – naming and finding things
- Synchronization – getting everybody in pace
- Consistency and replication – scalability, replication and consistency
- Fault tolerance – surviving failures
- Security – ensuring privacy, …
- Common paradigms – object-based, coordination-based, …

Course outcomes

You should be able to …

- Present a conceptual model of distributed systems
- Describe key components of a distributed system and evaluate the tradeoffs of alternative architectural models
- Suggest algorithm suitable for application in distributed systems
- Build prototype implementations of distributed systems
- Demonstrate an understanding of the challenges faced by future distributed systems

Class organization

Course is organized as a series of

- Lectures – a set of lectures on the core material
  - ~50’ lecture
  - ~30’ paper discussion
- Readings – additional readings from papers and book
- Project – one single, quarter-long project (45%)
  - A simple monitoring system for distributed systems, plus
  - A visualization tool for it
  - ***Go to www.planet-lab.org and register today!***
- Homework – reading assignments + short standard homework assignments (20%)
- Exam – one take-home, final exam (25%)
- *Class participation (10%)*

Summary

- A senior course on the basic principles behind distributed systems
- Mixing lecture and seminar style models
  - Traditional lectures introducing new topics
  - Together with seminar-style discussion on the research of others
- With a practical, motivating project component (*that will not take over your life!*)