CSCI 5105

Instructor: Abhishek Chandra

Today

- Distributed System Types
- Distributed Architectures

Types of Distributed Systems

- Distributed Computing Systems
- Distributed Information Systems
- Pervasive Systems

Distributed Computing Systems

- High-Performance Computing
  - Tightly-coupled, high-speed/capacity nodes
- Cluster Computing
  - Collection of homogeneous computers over LAN
- Grid Computing
  - Federated multi-admin heterogeneous clusters
- Cloud Computing
  - Pay-per-use elastic virtualized resources
  - IaaS, PaaS, SaaS

Distributed Information Systems

- Distributed File Systems
  - Files and users are distributed
- Distributed Databases
  - Distributed data and transactions
- World Wide Web
  - Information and users are widely distributed

Pervasive Systems

- Ubiquitous Computing Systems
  - Embedded devices, context-aware, interaction with users
- Mobile Systems
  - Mobile devices, can move with users
- Sensor networks
  - Collection of sensors collecting and processing data together

Distributed Architecture

- A distributed application runs across multiple nodes
- Software architecture: Logical organization
  - How to organize the various pieces of the application?
  - How do different pieces interact with each other?
- System architecture: Physical organization
  - Where do different pieces of the application execute?
  - Where is the control, user interface, computation, data?

Software Architecture Styles

- How to implement a distributed application
  - How are software components organized?
  - How do they communicate with each other?
- Component: Module with a well-defined interface
  - Implements some part of the application functionality
- Connector: Communication mechanism
  - Will enable components to talk and coordinate

Layered Architecture

- Components are placed in multiple layers
  - Each layer interacts with those above and below
- Common layering:
  - Application-interface
  - Processing
  - Data

Object-based Architecture

- Each component is an object
  - Encapsulates data and state
  - Exposes an interface and methods
- Distributed objects
  - Can be placed on different nodes
  - Communication via (remote) method invocations

Service-Oriented Architecture (SOA)

- Each component is a service (possibly in a different domain)
  - Use service-specific interfaces
  - Can have a complex implementation

Resource-Based Architecture

- Collection of resources managed by components
  - Can be added, deleted, modified by other applications
- REST (Representational State Transfer)
  - Single naming system
  - Common, small interface
  - Self-contained messages
  - Stateless execution

Publish-Subscribe Architecture

- Collection of autonomous processes
  - Referentially decoupled: do not directly address or communicate with each other
- Event-based coordination:
  - Events generated by some processes
  - Other processes notified of events
- Shared data space:
  - Publishers: Post events as tuples
  - Subscribers: Get tuples matching search pattern

Middleware

- A distributed layer between applications and low-level OS
  - Provides core functionality and services
  - Applications can use these for higher-level functionality
  - May rely on per-node OS/software support

System Architecture

- How is a software architecture instantiated?
  - Where are different software components placed?
- Centralized: Most functionality is in a single node
- Decentralized: Functionality is spread across symmetrical nodes
- Hybrid: Combination of the two

Centralized Architecture

- Client-server: Core functionality is in the server
- Application is vertically distributed
  - Distribution along functionality
  - Logically different component at different place
  - E.g.: UI at client, computation & data at server

Multi-tiered Architecture

- Could have variations on component distribution
  - Different amount of functionality between client-server
  - Only UI at client
  - UI+partial processing at client
  - UI+processing at client, data at server
- Multi-tiered server architecture:
  - Server functionality can be split across multiple nodes
  - E.g.: Front-end, Application server, Database

Decentralized Architecture

- Horizontal distribution of application
  - Each component is identical in functionality
  - Differ in the portion of data/state they operate on
- E.g.: File-sharing, parallel processing

Server Clusters

- Replication of functionality across nodes
  - Multiple front-ends, app servers, databases
- Client requests are distributed among the servers
  - Load balancing
  - Content-aware forwarding

Peer-to-Peer Systems

- Each component is symmetric in functionality
  - Servent: Combination of server-client
- How does a node find the other?
  - No "well-known" centralized server
- Overlay network: A logical network consisting of participant components
  - Nodes are processes/machines, links are communication channels (e.g., TCP connections)

Types of P2P Systems

- Unstructured: Built in a random manner
  - Each node can end up with any sets of neighbors, any part of application data
  - E.g.: Gnutella, Kazaa
- Structured: Built in a deterministic manner
  - Each node has well-defined set of neighbors, handles specific part of application data
  - E.g.: CAN, Chord, Pastry

Unstructured P2P Architectures

- Each node has a list of neighbors to which it is connected
  - Communication to other nodes in the network happens through neighbors
  - Neighbors are discovered in a random manner
  - Exchange information with other nodes to maintain neighbor lists
- Application data is randomly spread across the nodes
- Searching for a data item:
  - Flooding or Random Walk

Structured P2P Architectures

- Nodes and data are organized deterministically
- Distributed Hash Tables (DHT)
  - Each node has a well-defined ID
  - Each data item also has a key
  - A data item resides in the node with nearest key
- Each node has information about neighbors in the ID space
- Searching for a data item:
  - Routing through the DHT overlay network

Hierarchical Architecture

- Tree of nodes
- More scalable than a centralized architecture
  - Each node handles only part of the network
- E.g.: DNS

SuperPeers

- Special peers that maintain an index
  - Of other peers
  - Of data items and their location
- Need for superpeers:
  - Efficient search: Avoid flooding
  - Location-awareness: Find “nearest” neighbors
  - Easy Join: Node can easily find a starting peer

Hybrid Architecture

- Combination of centralized and distributed architectures
  - Some parts of the system organized as client-servers
  - Other parts organized in decentralized manner

Edge-Server Systems

- Servers on edge of the network
  - Provide localized content and compute to users
  - Decentralized set of content servers, may have P2P relationship
  - Client-Server relation to the users
  - E.g.: Content Distribution Networks (CDNs) such as Akamai

Collaborative Distributed Systems

- Work by user collaboration
  - P2P in functionality
  - Starting up is done in a client-server manner
  - E.g.: Bittorrent, Napster