CMPSCI 677 Operating Systems **Spring 2016**

Lecture 2: January 25

Lecturer: Prashant Shenoy *Scribe: Huaizu Jiang*

2.1 Overview

- Announcement
- Architectures for distributed systems

2.2 Announcement

- There will be materials on Moodle about docker and web application programming using such as Python, PHP, etc.
- There will be programming assignments using tools like GitHub.
- Class policy: no laptop, no device, no cell phone.

2.3 Architectural Styles

There are four important architecture styles of distributed systems:

2.3.1 Layered Design

In this approach the distributed application is partitioned into layers. Each layer can communicate with the immediate layer above it and the immediate layer below it. Layer i is able to communicate only with layer $i+1$ and layer $i-1$ via some interfaces. This approach is used in network protocal stack, for example, TCP/IP stack. For distributed systems, the layered design is used for many real-world scenarios, like multi-tier web applications.

2.3.2 Object-based

In this approach, the system is partitioned into objects (modules). It generalizes the layered design, where objects communicate with others (in contrast to two adjacent layers in the layered design) using methods that each object exposed (remote procedure calls). In a distributed system, all objects resided on different machines. This approach is popular in client-server applications.

2.3.3 Event-based Architecture

In this approach, components uses events for communications. In such a system, there exists a component, event bus, which is responsible for receiving and delivering events to other components. It is also referred as publish-subscribe paradigm, where some components publish events and push them into the event bus and some components subscribe to certain types of events.

2.3.4 Shared Data-Space

This approach is also referred as bulletin-board model, where components are decoupled in both space and time. Similar to the even-based architecture, component publish items in the shared data space. However, instead of sending published items to particular components as in the even-based model, components directly query and pick up items that they are interested in from the data space.

2.4 System Architectures

2.4.1 Client-Server

The most common architecture to implement a distributed system is the client-server model. There are two entities in the server: client(s) and server(s). A client makes a request to the server. The server processes the request and replies back to the client. For example, a web browser, as a client, sends HTTP requests to the web server. The web server processes the requests and replies to the browser. There are three levels for the client-server application: user interface level, (request) processing level and data level.

Search engine is a simple example of the client-server application. In the user interface level, users type keywords and make queries. Keyword queries would be converted to database queries via the query generator that is in the processing level. The database residing in the data level, which stores crawled webpages, process queries and return results (including titles and meta-information of interested webpages) to the processing level. The ranking part first ranks returned webpages based on relevances to the keyword queries. A result page is constructed and sent back to the user interface by the HTML generator.

The boundary between the client and the server is dependent on applications.

- For the search engine such as Google, the user interface is on the client side and all others resided on the server side.
- For applications where the user interface interacts with servers, such as Google forms and Gmail, part of the user interface is on the client side and all others on the server side.
- For applications on smart devices like tablet and cell phone, the entire user interface and part of the application tier reside on the client and all others on the server. The user interface is part of the application that do some computations locally.
- For desktop applications, most of user interface and application is local. Only the database resides on servers.
- For efficiency purposes, some portion of the database could be cached locally. However, the database may be too large to fit in a local machine. Therefor, part of the database is on the client and rest on the server.

For the web application, the server needs to act as a client (the interaction between application and database), referred as three-tiered architecture. The three common tiers for web applications could be HTTP, J2EE and database.

2.4.2 Decentralized Architectures

In the client-server, it is assumed that a server has more resources than a client. So servers could provide services to clients. In a peer-to-peer (P2P) system, there is no distinction between machines, where entities (peers) in the system are equal. This is why it is called a decentralized architecture. In the P2P system, each node can both process and create requests. The most common application for the P2P system is to store and retrieve data using a key-value pair. Each data item is associated with a unique key. The entire data is across all peers, where each peer stores part of the data that can be retrieved by the key values. Decentralized architectures can be structured or unstructured.

2.4.2.1 Structured P2P Systems

Chord Each peer is given a node id and a set of key values are generated with a distributed hash table. Key values for each node starts from largest one of the previous node and no greater than its id. For a key k, the smallest node with $id \geq k$ has the given data item. When a node is added, it randomly picks a id and the data will be re-partitioned. It is a structured P2P system.

CAN A Content Addressable Network (CAN) is also a structured P2P system. Each data item in the system has multiple keys (corresponding to different attributes such as file names and file types). CAN is a d-dimensional coordinate system and partitions the data into multiple d-dimensional regions. Each data item maps to a point in the d-dimensional coordinate system and each node is responsible for data items within a particular region. When a new node joins, it randomly picks a and split with node for that point. When a node leaves, its corresponding data items would be merged to other nodes. It is more difficult to deal with node leaving than joining since the data merge may not give symmetric partitions.

2.4.2.2 Unstructured P2P Systems

Unlike the structure (the node id in Chord and coordinate system in CAN), in an unstructured P2P system, each node randomly connects to a set of other nodes to become their neighbors. Since no certain network topology, to deal with queries, each node would "flood" the system, which means that each node propagates in the network inquiring its neighbors if containing the queried data item. The result would be sent back to the original node through the network backwards. There will be heavy traffic in the system since there are a lot of queries propagating in the network.

SuperPeers SuperPeers are introduced to reduced the amount of traffic. Some of nodes form a group and elect a leader, called super peer, within the group. The super peer is responsible for answering queries on behalf of the cluster. The queries would then only propagate only in the super peer network. The most common example of super peer is Skype. The super peer takes care of which user is online and offline. The super peer in each group is the most reliable user but may also change due of unavailability of the previous one.

2.4.3 Hybrid Architectures

Edge-server systems The edge-server system extends the topology of client-server to client-proxy-server. Proxies are useful for caching web servers. The basic assumption is that clients are closer to proxies than to servers. Clients send requests to proxies instead of original servers, where proxies work on behalf of servers. The request would propagate locally between clients and proxies and therefore get responses immediately. On some cases, proxies can not process queries and forward them to servers to get replies. Proxies are also referred to as edge servers since they are placed at the edge of servers instead the network.

Collaborative Distributed Systems BitTorrent is a collaborative P2P download system. There are two key ideas. The first one is that the data is divided into several chunks. Chunks of a file would be downloaded in parallel from multiple peers and reassembled after downloading. In this manner, the file could be downloaded more quickly. The other key idea is the force altruism, which encourage users to participate the network instead of shutting down the computer once download is finished. If you download more than you share, you will be punished such as reducing your downloading speed from others. To download a file, users have to download the corresponding .torrent file from the server first. The .torrent file contains a list of trackers, where each tracker keeps track of active peers in the system. The user would be connected to some active peers and the download starts.