CS418 — Operating Systems

Lecture 15

Distributed Processing

Textbook: Operating Systems
by William Stallings

1. Client/Server Computing

- Client/Server Terminology
  - 1. **API** (Applications Programming Interface): A set of function and call programs that allow clients and servers to intercommunicate.
  - 2. **Client**: A networked information requester.
  - 3. **Middleware**: A set of drivers, APIs, or other software that improves connectivity between a client and a server.
  - 4. **Server**: A computer, usually a workstation or a minicomputer, that stores information for manipulation by network clients.
  - 5. **Relation Database**: A database in which information access is limited to the selection of rows that satisfy all search criteria.
  - 6. **Structured Query Language (SQL)**: A language developed by IBM and standardized by ANSI for creating, updating or querying relational databases.

2. The difference between client/server system and ordinary distributed processing

- What is the difference between client/server system and ordinary distributed processing?
  - 1. User-friendly.
  - 2. More emphasis on centralizing corporate databases and many network management and utility functions.
  - 3. User has greater choice in selecting products from a number of vendors.
  - 4. Networking is fundamental to the operation.

3. Client/Server Applications

- Think of TCP/IP as an example.

4. Database Applications

- Database applications fit client/server system, because
  - **1**. Practical database is usually of huge size. Such capacity and power is not needed and is too expensive for a single-user PC/workstation.
  - **2**. It would be too expensive to move the entire file (e.g. 1-million-record file) to the client for searching.

5. Classes of Client/Server Applications

- **Host-based processing**: Not really a client/server system.
- **Server-based processing.** Think of the following example: the client provides a graphical user interface and server does all the processing.
- **Client-based processing.** Most of the processing are done by the client, except for some complex data validation routines and other database logic functions, which are best performed at the server. This is the most common client/server approach.

  Example?

- **Cooperative processing:** Application processing is optimized. More difficult one, but probably will be more popular in the future.

6. Three-Tier Client/Server Architecture

- Application software is distributed among three types of machines: a user machine, a middle-tier server, and a backend server. The middle-tier server acts as both a client and a server.

7. File Cache Consistency

- Individual systems can use file caches to store recently accessed file records.

  Example?

- When caches always contain exact copies of remote data, we say that the caches are **consistent**.
- Now can you see the **cache consistency** problem?
- Solution?

8. Some Famous Distributed Algorithms

- Leadership Election:
  - **1**. Each process has a unique ID known to all members.
  - **2**. The process with the highest ID is the leader.
  - **3**. Any process may fail at any time.
- Algorithm 1—**The BULLY election algorithm**:
  all process do the following
  - **1**. P notices there is no reply from the coordinator.
  - **2**. P sends an **elect** message to all processes with higher IDs.
  - **3**. If there is any reply then P exits.
  - **4**. If there is no reply then P wins, obtains any state needed to function as a leader, then sends a **coordinator** message to all processes.
  - **5**. On receipt of an **elect** message a process must both reply to the sender and start an election if it is not already holding one.

- Algorithm 2—**A ring-based election algorithm**:
  all process do the following
  - **1**. P notices the coordinator is not functioning.
  - **2**. P sends an **elect** message containing its own ID to the next process in the ring.
  - **3**. On receipt of an **elect** message
    a without the receiver's ID — add this ID and pass on the message.
    b with the receiver's ID (the message has been round the ring)—send a message (**coordinator**, highest ID in the message) around the ring.