CSCI 251: Concepts of Parallel and Distributed Systems

Alvin Lin

October 18th, 2017

Distributed Systems

Topics:

- OS and distributed OS concepts
- Interprocess communications
- Clocks and global states
- Coordination
- Distributed Transactions
- Blockchain
- RMI and RPC
- Request-Reply protocol
- Marshalling and data representation

Distributed Computing

- Concurrent components
  - Independent
  - Communicate and coordinate through message passing or shared memory

- Lack of a global clock
  - Each component has its own clock
  - Clock synchonization is a huge issue
- Independent failure of components
  - Components are isolated from one another for fault tolerance
  - Redundancy

Operation System Layer

- Middleware is on top of OS layers
- OS on a particular node provides abstraction of local resources, such as processing, storage, and communications.
- Middleware is across all notes to utilize combinations of local and remote resources. It controls the interactions among objects/processes.

OS Mechanisms

- Encapsulation: OS kernel and server processes provide server interfaces to clients to meet their needs but hide the system details.
- Protection: protection from illegitimate accesses, protect kernel addressses and system registers.
- Concurrent processing: sharing of resources among processes using communication and scheduling for efficient utilization.

Core OS Components

- Process manager: handles creation and operations, resource management for address space and threads.
- Thread manager: handles creation, synchronization, and scheduling. Threads are attached to processes.
- Communication manager: handles communication between threads on the same processor and threads on remote processors.

- Memory manager: handles physical memory, virtual memory, and cache.
- Supervisor: handles interrupts, system calls, and exceptions.

The time it takes to fetch data from the disk t_d, memory t_m, and cache t_c have the following relation:

$$t_d > t_m > t_c$$

Protection

The operation system is expected to protect from illegitimate accesses. It maintains access privileges between users and supervisors and separates the address spaces between the user processes. During exceptions, control is transferred from the user to the kernel, but switching takes time and memory.

Levels of Communication

Five Levels:

- Message Passing: communication primitives, send and receive, direct, consists of one direct logical communication path from a sending process to a receiving process.
- Request/Reply
- Transactions
- Transport Connection
- Packet Switching

Message Synchronization

- Nonblocking send: source is released after copying message to buffer.
- Blocking send: source is released after transmitting message to network.
- Reliable blocking send: source is released after the message is received by the receiver's buffer.
- Explicit blocking send: source is released after the message has been received by the receiving process.

- Request and reply: the sender is released after the message is received by the received, a service is executed at the received, and a response is returned to the sender.

Buffer spaces at both kernels allow for asynchronous operations to reduce blocking and minimize deadlocks.

Pipe

Pipes are APIs in Windows and Unix systems that allow for the transfer of uninterrupted byte sequences. Named pipes can be shared among disjoint processes.

Sockets and Ports

A socket is a communication end point managed by transport services. But TCP and UDP use socket abstraction, messages are sent to an IP address and port number. The socket of a receiving process must be bound to a port and the IP address of the computer. A process can use multiple ports to receive messages.

Connection-oriented Client/Server Socket Communication

1. Server and client issue socket calls separately.
2. Server issues a bind call.
3. Listen socket call: the server is willing to accept a connection.
4. Client issues a connect call to the server.
5. The server will reply with an accept call.
6. The server reads the client's request and responds.

Data Representation and Marshalling

- Data in "data structures" must be converted to a sequence of bytes.
- Data representation is different in different computers. This requires the use of an agreed upon format for transmission, since conversion will happen at the sender as well as the receiver.
- The sender's format should provide details to the recipient.

- Marshalling is the process of assembling data items for transmission.

Remote Procedure Call (RPC)

- RPC is the most popular request/reply communication process.
- RPCs are very similar to procedure calls and provide access transparency and communication transparency. It can be considered as an API for the transport service.
- RPC introduces vulnerabilities, so security features have to be built on top of secure RPC. This usually involves an authenticated protocol for RPC for mutual authentication, message integrity, confidentiality, and originality.

Reminders

Professor Mohan Kumar:
`mjkvcs@rit.edu`
`https://cs.rit.edu/~mjk`

Rahul Dashora (TA):
`rd5476@mail.rit.edu`

You can find all my notes at `http://omgimanerd.tech/notes`. If you have any questions, comments, or concerns, please contact me at alvin@omgimanerd.tech