Study Log of Distributed System

1. Introduction

* A distributed system is a collection of independent computers appears as a single machine to end user.
* Sometime, it can act as a middleware, which offers each application in separate machine with a uniform interface.
* Transparency is a special feature for distributed system, it includes following aspects:
* Access – User don’t know how a resource is accessed.
* Location – User don’t know where the resource is located.
* Migration – User don’t realize that resources may have been moved to another place.
* Relocation – User don’t realize that he/she may has been relocated to another resource while using the system.
* Replication – User don’t realize that resources may have been replicated.
* Concurrency – User don’t’ realize that the resources he/she is accessing may also be used by other users.
* Failure – User don’t realize that there is an error/exception occurs.
* Interface Definition Language(IDL), is used to define uniform interfaces for the distributed system to let others access. Generally, it at least has two arguments: one is the length of the data; another is actual data contains a string with that length. Relevant terms: Open system, IDL, Client Stub, Server Skeleton(Stub).
* Distributed Algorithm, fully decentralized algorithm:
* No complete information about the whole state
* Independent machine makes decision based on limited local knowledge
* One machine’s failure doesn’t affect to others or ruin the whole system.
* No global clock.
* False assumption when developing distributed system
* Network is reliable
* Network is secure
* Network is homogeneous
* Topology doesn’t change
* Latency is zero
* Bandwidth is infinite
* Transport cost is zero
* There is an administrator
* Transaction(Atomic, consistent, isolated, durable), Distributed Transaction
* Process, IPC(InterProcess Communication) used to communicate between processes. User shared memory and message to store data.
* User ‘Test and set’ to preserve atomic operation in local machine.
* Hard to use ‘Test and set’ for remote machine, which has different memory spaces.
* Hard to implement shared memory to make distributed system coordinate well.
* Clients and servers: a server can be client to other servers. Communicate with messages.
* Web: TCP/IP, HTTP, CGI, URL, MIME types, HTML

1. Architecture

* Architecture styles: layered, object-based, data-centered, event-based
* For event-based, publish to and subscribe from event bus, decoupled between processes.
* Application layer: user-interface, processing, data level.
* Heavy client/ heavy server: depends on requirement.
* DHT(Distributed Hash Table), add a new node to DHT, create a random id.

1. Processes

* User space/Kernel space
* Virtual machine
* UDP, one way, from sender to receiver, cheap, no setup, no guarantee, data may lost
* TCP, two way, each sends or receives data, reliable, in order, buffer required, delay
* Synchronous call: block, wait, sequential, inefficient, simple
* Asynchronous call: no hang, no wait, multi-thread, efficient, complex
* Socket/Port
* Marshalling/Unmarshalling
* XML namespace,
* Serialization/Deserialization
* Request-reply communication:
* RPC exchange protocols: R, RR, RRA

1. Communication

Protocol Layer: Application, Presentation, Session, Transport, Network, Data Link, Phisical

Steps of RPC: Client->Local OS->Remote OS->Server->Local OS->Client OS->Client

Asynchronous RPC:

Persistent communication

Berkeley Sockets

Reduce jitter: use buffer

Multicast