CS60002: Distributed Systems
Spring 2019
List of Distributed Systems Projects

1. Distributed Log Service
    a. Should support storage and retrieval of record-oriented, append-only logs
    b. Should tolerate at least one crash failure
    c. Should be scalable
    d. Systems to study: Facebook LogDevice, Apache DistributedLog (Part of Apache Bookkeeper now)

2. Distributed Publish Subscribe System
    a. Should support creation/deletion of new groups
    b. Should support users to subscribe/unsubscribe to/from one or more groups
    c. Should support two modes of users – publishers and subscribers. A user can be publisher in some groups and subscribers for others (may be both for a group)
    d. Should support both push (notification to user) and pull (periodic check by user) modes of events
    e. Should tolerate at least one crash failure, with eventual consistency for replicas
    f. Should consider both availability and performance in delivering events to subscribers
    g. System to study: Apache Kafka

3. Distributed Data Store
    a. Users should be able to upload/download/search key-value pairs in the store
    b. System should be able to tolerate one crash fault
    c. Should ensure sequential consistency if replication is used
    d. Should be able to add/delete servers from the system
    e. Should do load balancing
    f. System to study: Apache Cassandra

4. Distributed Coordination Middleware
    a. Should support setting up a cluster of nodes for an application
    b. Should support addition/deletion of nodes
    c. Should support naming of nodes
    d. Should support locking of shared data
    e. Should support atomic actions on shared/replicated data even in the presence of on crash fault
    f. Project should also develop any simple application of its choice over the middleware using the middleware APIs
    g. System to study: Apache Zookeeper

5. Distributed Directory Service

a. Should support definition/maintenance/updation of object schema (what classes of objects with what attributes).
b. Should support both must-have and may-have attributes for objects
c. Should support addition/deletion/modification of objects with schema check
d. Should support base, one-level, and subtree search for LDAP
e. Should support the directory tree to be partitioned into at least 2 parts and stored separately
f. Should be tolerant to at least one crash fault (with sequential consistency for schema change and eventual consistency for object change)
g. Should support consistent merge with timestamps
h. Systems to study: Apache Directory, OpenLDAP

6. Cluster Middleware with Central Master Node
    a. Should support addition/deletion of compute nodes (assume homogeneous)
    b. Should support job submission/withdrawal/status-check at any time
    c. Should support load balancing of jobs across compute nodes
    d. Should be tolerant to at least one crash fault (for both master and compute nodes)
    e. Should support job policies in master node
    f. Systems to study: PBS Pro, ROCKS, Google Borg

7. Cluster Middleware with No Central Master Node
    a. Should support addition/deletion of compute nodes of any types (heterogeneous)
    b. Should support job submission/withdrawal/status-check at any time at any node
    c. Should support discovery of compute nodes
    d. Should support matchmaking of jobs with compute nodes
    e. Should be tolerant to at least one crash fault
    f. Systems to study: HTCondor, Globus

8. Multimedia Content Delivery Network
    a. Should support upload/download/search of multimedia files
    b. Should support delivery of multimedia contents to users spread geographically
    c. Should support both static replicas for fault tolerance (one crash fault) and dynamic replicas based on demand for performance.
    d. Paper: to study:
    e. System to study: Akamai

9. HTTP Content Delivery
    a. Should support faster download of webpages through the services of a CDN
    b. Should support tolerance to at least one crash fault in the CDN
    c. Should try to optimize network use
    d. Should support load balancing between web servers
    e. System to study: Cloudflare

10. Distributed File System
    a. Should be able to mount remote filesystems locally to give an integrated view locally
    b. Should be able to read and write both local and remote files with the same interface (use FUSE)
    c. Should support locking of shared files
    d. Should support caching for performance (define consistency to be supported)
    e. Should tolerate at least one crash fault
    f. Systems to study: NFS

11. Distributed chat
    a. Users should be able to send and receive messages
    b. All users should see the same sequence of messages (total order)
    c. Causal relationship of messages to be maintained (causal order)
    d. No centralized server
    e. Users should be able to join and leave the chat anytime (new users should see all messages that originate after the user join)
    f. Systems to study: Orbit Web, Tox

12. Collaborative Download
    a. Should support discovery of content already present nearby (can assume a closed group)
    b. Should support entry/exit of users into the group
    c. Should support download partitioning based on load balancing/incentives
    d. Reading Material: paper-1 paper-2

13. P2P File Sharing System
    a. Users should be able to join and leave system asynchronously.
    b. Should be scalable with no single point of failure.
    c. Users should be able to search, upload and download files.
    d. Must ensure successful file download if file present in the system.
    e. Systems to study:dc++, Gnutella, bit torrent

14. Software Load Balancer
    a. Should distribute the workload between the active servers fairly.
    b. Should reduce single point of failure through redundancy and rebalance workloads when a server fails.
    c. Improve scalability by the addition of new servers dynamically.
    d. Should be resilient to single crash fault on itself.
    e. Systems to study: Seesaw, HAProxy

15. Railway Reservation System
    a. Should consider type and dynamicity of data for data replication with different consistency models
    b. Should tolerate at least one crash failure

c. Should support load balancing between backend servers

16. Travel Agency Booking System