Cloud Computing IV

UNIT IV - COMMUNICATION AND SCHEDULING

▼ Eucalyptus

Open source software to build private/hybrid cloud environments compatible with S3 and EC2.

- **User facing services**: implements web service interfaces that handle AWS compatible APIs.
- **Cloud controller**: Provides high level resource tracking and management. Only one CLC can exist in a Eucalyptus cloud. Provides-
  a. System wide arbitration of resource control
  b. Governing of persistent user and system data
  c. Handling authentication and protocol translation
- **Cluster controller**: Manages node controllers and is responsible for deploying and managing VM instances.
- **Node controller:** Runs on the machine that hosts VMs and interacts with both OS and hypervisors to maintain the lifecycle of instances running on each of the nodes.
- **Storage controller**: Interfaces with various storage systems.
- A system resource state is maintained with data coming in from CLC and NCs of the system. When a user’s request arrives, the SRS is used to make an admission control decision for the request based on service level expectation.

▼ Failures and faults

Different types of faults - Transient, Intermittent, Permanent

Different types of failure -

1. Crash failure
2. Omission failure (send/receive)
3. Timing failure
4. Response failure (Value/State transition)
5. Arbitrary failure

Issues that lead to faults **-**

1. **Timeouts and Unbounded delays**
   (Synchronous, Timeout) Every successful request receives a response within time 2d + r
   (Asynchronous, Unbounded delay) Based off of variability, jitter and application characteristics, determine appropriate tradeoff time
2. **Network congestion and queueing**
   Virtualised multitenant resources have overheads that can add latencies towards consuming network traffic and thus increase queue lengths.
   This leads to variable delay congestion and queuing based on the workload characteristics of multiple tenants making it hard for having a consistently deterministic and optimised design.

Classes of faults -

1. Crash stop faults
2. Crash recovery faults
3. Byzantine faults

▼ **Strategies for dealing with partial failures**

1. Using asynchronous communication across internal microservices

2. Using retries with exponential backoff
3. Working around network timeouts
4. Use circuit breaker pattern
5. Provide fallbacks
6. Limit number of queued requests (Polly bulkhead isolation policy)

▼ **Failover architecture**

- **Active - Active (symmetric) failover architecture**
  Each server is configured to provide redundancy for it's peer. Databases are replicated, it mimics having only one instance of the application allowing data to stay in sync. Also called continuous availability
- **Active - Passive (Asymmetric) failover architecture**
  Applications run on a primary, master server. A dedicated redundant server is present to take over any failure but apart from that is not configured to perform any other operations.

▼ Approaches for fault tolerance

- **Fail - Safe tolerance**
  Safety state is preserved but performance might drop. No process can enter critical section for an indefinite period.
- **Graceful degradation**
  Application continues but in a degraded mode, processes enter critical section but not in timestamp order.

Approaches for building fault tolerant systems-

1. Redundancy
2. Reliability
   - MTBF = total operational time/number of failures
   - MTTF = uptime/number of tracked items

3. Repairability
    - MTTR = downtime/number of failures
4. Recoverability

> 💡 System availability = MTTF/MTTF+MTTR

▼ Consensus

An algorithm achieves consensus if it satisfies the following -

1. Agreement - all non faulty nodes decide on same output
2. Termination - all non faulty nodes decide on some output
3. Validity
4. Integrity
5. Non-triviality

The algorithm consists of three steps - Elect, Vote and Decide

Common challenges -

1. Reliable multicast - group of servers together need to receive same updates in the same order in which they were sent. Given failures leads to reliable multicast problem.
2. Membership failure detection - One of the group members fail or crash.
3. Leader election - Group of servers attempt to elect a leader from among them
4. Mutual exclusion - A group of servers together attempt to ensure mutually exclusive access to critical resources.

→ Fischer-Lynch-Paterson Impossibility - Consensus algorithms are impossible to solve in asynchronous networks.

▼ Paxos consensus algorithm

Roles in a distributed system -

1. Proposers
2. Acceptors
3. Learners

Proposers send two types of messages to acceptors - Prepare and Accept

Each proposers proposal must be a positive, monotonically increasing unique natural number.

▼ Leader election algorithms

Liveness and Safety conditions - Every node will eventually enter a state that is elected or not elected and only one node can enter the elected state and become leader.

▼ Ring election

Messages are sent clockwise. When any process pi receives an election message, it compares the attribute in the message to its own attribute and forwards it if the message attribute is greater. If not, it overwrites the message with its own attribute and forwards it. If the arrived id:attr matches that of pi, it becomes the new coordinator.

Message complexity and turnaround time = 3N - 1

▼ Modified Ring election

Processes are organised in a logical ring. Each process appends its id and passes the message to the next process. Once the message gets back to the initiator, it elects the process with the best election attribute value. It then sends a coordinator message, if coordinator’s id is in id list, election is over if not the algorithm is repeated(handles election failure).

Message complexity and turn around time = 2N

- Bully Algorithm

  When a process with the next highest ID detects leader failure, it elects itself as the new leader. It sends a coordinator message to other processes with lower identifiers.

  Else it initiates an election by sending election message only to processes with higher id than itself. It if receives no answer within timeout, it elects itself leader and send coordinator message to all lower id processes. If it receives an answer, it waits for the coordinator message.

- Cloud resource management policies (CRM policies)

1. Admission control
2. Capacity allocation
3. Optimization and load balancing
4. Energy optimization
5. QoS guarantees
6. Locale based
7. Cost

Mechanisms for implementation of CRM

1. **Control theory** - Use feedback to control allocation of resources and predict transient behaviour
2. **Utility based** - Performance model that correlates performance with cost
3. **ML** - predicts using ai-ml
4. **Market oriented** - Users provide bids and resources are provisioned as an allocation.

Components of a control system -

- Inputs - offered workload, policies for admission control, capacity allocation, load balancing
- Control system components - sensors used to estimate relevant measures of performance
- Outputs - resource allocation to individual applications
- Scheduling - Batch scheduling for throughput and min turnaround time and Real time system to meet deadlines and be predictable.

Workload partitioning rules

- OPR - earliest possible completion time
- EPR - assigns equal workload to individual nodes

▼ Zookeeper

- Open source, high performance coordination service
- Automates managements of hosts.
- Uses shared hierarchical name spaces of data registers called znodes to coordinate distributed processes.
- Exposes common services in simple interface:
  Naming
  Configuration management
  Locks and synchronisation
  Group services
- An application can create a znode, which is a file that persists in memory on the ZooKeeper servers.
- Tasks can be synchronized by updated the status in the znode.

Features -

1. Updating node status

2. Managing the cluster
3. Naming service
4. Automatic failure recovery

Architecture-

- Leader
- Follower
- Ensemble/Cluster - group of zookeeper servers
- Zookeeper web UI

Types of nodes - Persistent, Ephemeral, Sequential

Advantages -

1. Simple distributed coordination process
2. Synchronization
3. Ordered messages
4. Serialization
5. Reliability
6. Atomicity

Disadvantages -

1. No service discovery
2. Ephemeral node lifecycle is tied to TCP connection
3. Complex

Reasons to use Zookeeper -

1. Naming service

2. Distributed locks
3. Data synchronization
4. Cluster/Configuration management
5. Leader election
6. Message queue