Advanced Algorithms/Techniques for Complex and Hybrid cloud systems

Hong-Linh Truong

Faculty of Informatics, TU Wien

hong-linh.truong@tuwien.ac.at
http://www.infosys.tuwien.ac.at/staff/truong
@linhsolar
What is this lecture about?

- Discuss key issues when we need multiple resources for computing, data storage and messaging in edge, fog, and cloud computing
- Focus on high availability, high performance and high throughput aspects
- Examine distributed coordination, running example with Zookeeper
- Select and discuss a topic with real systems and concepts behind these systems
Motivating example: a software system for IoT scenarios

Real example: 5M sensor/monitoring points with ~1.4B events/day ~ 72GB/day
Data, Services and Systems Management

Sensor
Sensor
Sensor
Sensor
Sensor

Messaging Services

Queue
Queue
Queue
Queue
Queue

Ingest Client
Compute Node
Compute Node
Storage Node
Storage Node

High availability, high throughput, high performance
Motivating example: a software system for IoT scenarios

Figure source: https://cloud.google.com/solutions/architecture/streamprocessing
Data resource management

Store a big data collection in a single place?

High availability?
Geographic data distribution

Motivating example: a software system for IoT scenarios

Figure source: https://cloud.google.com/solutions/architecture/streamprocessing
Queuing service management

How do we control these instances in an efficient way?

Key problems

- How to deal with failures in complex edge, fog and cloud systems?
  - Master/producer, worker/consumer, communication, etc
- How to establish and maintain high availability and high performance?
  - Reduce latency and increase concurrent access/processing

→ Algorithms and techniques for on-demand data centers, redundancy, replication and recovery
Techniques

- Resource provisioning and management
  - Also related to elasticity (lecture 3)
- Routing based on load and metadata
- Recovery from failures
  - Distributed process coordination for cloud systems
- Data sharding & replication
  - Within individual data centers
  - Among geo-distributed data centers
COMPUTING RESOURCES AND MANAGEMENT
Virtual data centers

- On-demand virtual data centers
  - Compute nodes, storage, communication, etc.
  - We focus on establishing virtual data centers working like a single distributed system (e.g., a cluster)
Virtual data centers

- **Challenges**
  - Provision resources/nodes (using VMs or containers)
  - Configure networks within virtual data centers
  - Configure networks between virtual data centers and the outside systems
  - Deploy software into the virtual data centers
  - Maintain the virtual data centers
Generic on-demand data center

- Set of VMs/containers + storage/filesystems that can be used for different purposes
  - Think about a cluster of VMs or containers, instead of a traditional cluster of physical machines

- Steps
  - Create VMs/containers and orchestration management system (e.g. using Mesos/ Kubernetes)
  - Configure VMs/containers to create a virtual network
    - Create/configure virtual networks
    - VMs/containers discovery
    - Examples: Weave (https://www.weave.works/install-weave-net/)
Example -- DC/OS

Source: https://docs.mesosphere.com/1.8/overview/architecture/
Azure Cloud and Containers

Example - Weave Net and docker

- Work with Kubernetes & Mesos as well
- Key idea: using network plug-in for containers + P2P overlay of routers in the host

Source: https://www.weave.works/docs/net/latest/introducing-weave/
Application-specific virtual data centers

- Specific virtual data centers for specific purposes
  - E.g., Data-center of nodes for Hadoop or Spark
- First, create generic data centers but customized for specific software stack
- Second, deploy specific software frameworks

Figure source: http://mesos.apache.org/documentation/latest/architecture/
Edge/Fog and Cloud continuum

- Several issues in real world:
  - Single software stack (e.g., OpenStack) versus multiple software stacks
  - Containers versus VMs
  - Coordination across edge, fog and cloud infrastructures

Key focuses on high availability and high performance

- **Layers:**
  - VM/container layer
  - Network layer
  - Resource management layer

- **Important techniques**
  - Redundancy
  - Monitoring
  - Elasticity
  - Distributed coordination for resources
DATA MANAGEMENT
Data sharding

- Limited storage space, computing capabilities and network
- High latency due to geographical communication
- Sharding
  - Distributed large-amount of data (of the same app-structure) onto distributed nodes
- Replication can be also applied
Sharding strategies

- Different strategies
  - Lookup: query to find a shard
  - Range: a range of keys is used to determine a shard
  - Hash: determined shard based on the hash of a key

Example

Source: http://queue.acm.org/detail.cfm?id=1563874
Example strategies in MongoDB

Hash

Range

Source: https://docs.mongodb.com/v3.2/sharding/
Shard and routing

Source: https://docs.mongodb.com/v3.2/sharding/
Single or multi-tenant sharding

Resources and consensus

High Availability and High Performance for Cloud data

- Resources and resource management
  - High availability of data storage
  - Load balancing

- Data management
  - Data distribution
  - Replication
  - Encoding/Integrity
High Availability and High Performance in queuing systems

- Moving messages fast!
- Brokering service: load balancing and high availability
  - Clustering of several broker nodes
  - Resource management
- Client/consumer: load balancing and high availability
  - Using queues, sharing topics, and consumer groups
  - Resource management for consumers
    - This is at the consumer side, not in the queuing system, but techniques are quite similar, as discussed in resources and resource management
Clustering Brokers


Figure 1: Kafka Architecture

Figure 2: RabbitMQ (AMQP) Architecture
Consumer Load balancing

What do you have to do here?

Source: http://www.enterpriseintegrationpatterns.com/patterns/messaging/CompetingConsumers.html/
Client Load in Apache Kafka

Source: http://www.enterpriseintegrationpatterns.com/patterns/messaging/CompetingConsumers.html/
Shared Topics with MQTT by HiveMQ

Source: https://www.hivemq.com/blog/mqtt-client-load-balancing-with-shared-subscriptions/
Group redundancy architecture

- Use group architecture for redundancy in order to support failure masking

Design flat groups versus hierarchical groups

Structure a system (communication, servers, services, etc.) using a group so we can deal failures using collective capabilities

Replication architecture

Passive (Primary backup) model

Active Replication

Source: Coulouris, Dollimore, Kindberg and Blair, Distributed Systems: Concepts and Design  Edn. 5
Distributed coordination

- A lot of algorithms, etc.
  - Paxos family
- Well-known in the cloud
  - Zookeeper

Notes from the paper: “server replication (SR), log replication (LR), synchronization service (SS), barrier orchestration (BO), service discovery (SD), group membership (GM), leader election (LE), metadata management (MM) and distributed queues (Q)”

ZooKeeper Service

Source: https://zookeeper.apache.org/doc/r3.4.10/zookeeperOver.html
ZooKeeper data -- znodes

- Data nodes called znodes
- Missing data in a znode → Problems with the entity that the znode represents
- No partial read or write for a znode
- Persistent znode
  - /path deleted only through a delete call
- Ephemeral znode
  - The client created it crash
  - Session expired

Source: https://zookeeper.apache.org/doc/r3.4.10/zookeeperOver.html
ZooKeeper API

- Simple APIs
- Client call server
  - Create, delete, exists, getData, setData, getChildren
- Watch/notification
  - Clients setting a watch in order to receive notifications about znodes
Guarantees

- Sequential Consistency
- Atomicity - either succeed or fail
- Single System Image
- Reliability
- Timeliness
How can we coordinate resources in edge/fog/cloud continuum

- Which protocols/techniques can we use?
- Do the known protocols like Zookeeper, etcd, consul, etc., fit, if yes where?

Figure 1: The mF2C Architecture.

TOPICS FOR YOU
High availability in Hadoop

- **Concepts**
  - Feng Wang, Jie Qiu, Jie Yang, Bo Dong, Xinhui Li, and Ying Li. 2009. Hadoop high availability through metadata replication. In Proceedings of the first international workshop on Cloud data management (CloudDB ’09). ACM, New York, NY, USA, 37-44. DOI=http://dx.doi.org/10.1145/1651263.1651271

- **Practical work with Hadoop and Zookeeper**
  - https://zookeeper.apache.org/
High availability with cluster of containers

- **Concepts**
  - Large-scale cluster management at Google with Borg: [https://ai.google/research/pubs/pub43438](https://ai.google/research/pubs/pub43438)

- **Practical work:**
  - Kubernetes: [https://kubernetes.io/docs/admin/high-availability/](https://kubernetes.io/docs/admin/high-availability/)
  - Docker: [https://docs.docker.com/datacenter/ucp/1.1/high-availability/set-up-high-availability/](https://docs.docker.com/datacenter/ucp/1.1/high-availability/set-up-high-availability/)
  - [https://docs.mesosphere.com/](https://docs.mesosphere.com/)
  - [https://www.consul.io/docs/internals/consensus.html#deployment-table](https://www.consul.io/docs/internals/consensus.html#deployment-table)
High availability for resource management systems

- Concepts – Mesos, YARN, Zookeeper
  - Designing Cluster Schedulers for Internet-Scale Services: [https://queue.acm.org/detail.cfm?id=3199609](https://queue.acm.org/detail.cfm?id=3199609)
  - Other papers in coordination in fog/edge systems

- Practical work:
  - [https://dcos.io/docs/1.7/overview/high-availability/](https://dcos.io/docs/1.7/overview/high-availability/)
High availability for MongoDB

- **Concepts**
  - [https://raft.github.io/](https://raft.github.io/)
  - In Search of an Understandable Consensus Algorithm, (Extended Version), Diego Ongaro and John Ousterhout, Stanford University

- **Practical work:**
  - [https://docs.mongodb.com/manual/replication/](https://docs.mongodb.com/manual/replication/)
  - [https://docs.mongodb.com/manual/core/replica-set-architecture-geographically-distributed/](https://docs.mongodb.com/manual/core/replica-set-architecture-geographically-distributed/)
High availability for RabbitMQ or Kafka or other messaging protocols

- **Concepts**

- **Practical work:**
  - https://www.rabbitmq.com/pacemaker.html
  - https://www.rabbitmq.com/ha.html
  - http://clusterlabs.org/
Summary

- It is important to learn some key techniques to enable big, dynamic could systems
  - On-demand data centers:
    - Allow us to obtain compute resources and storage resources for dealing with dynamic workload
    - Resources “as a data center” (rather than isolated)
  - Data sharding + resource management
    - Fundamental requirement for big data
  - Distributed coordination
    - Allow us to manage failures and support high availability
- They are highly interdependent topics that should be studied together
- We also need to look for application-specific algorithms and learn them
Some further readings

- Flavio Junqueira & Benjamin Reed, ZooKeeper, Distributed Process Coordination, O’reililly, 2013
Thanks for your attention

Hong-Linh Truong
Faculty of Informatics
TU Wien
hong-linh.truong@tuwien.ac.at
http://www.infosys.tuwien.ac.at/staff/truong