Advanced Topics in Service-Oriented Computing and Cloud Computing, Winter 2017
http://dsg.tuwien.ac.at/teaching/courses/socloud/

# Advanced Algorithms/Techniques for Complex and Hybrid cloud systems

Hong-Linh Truong

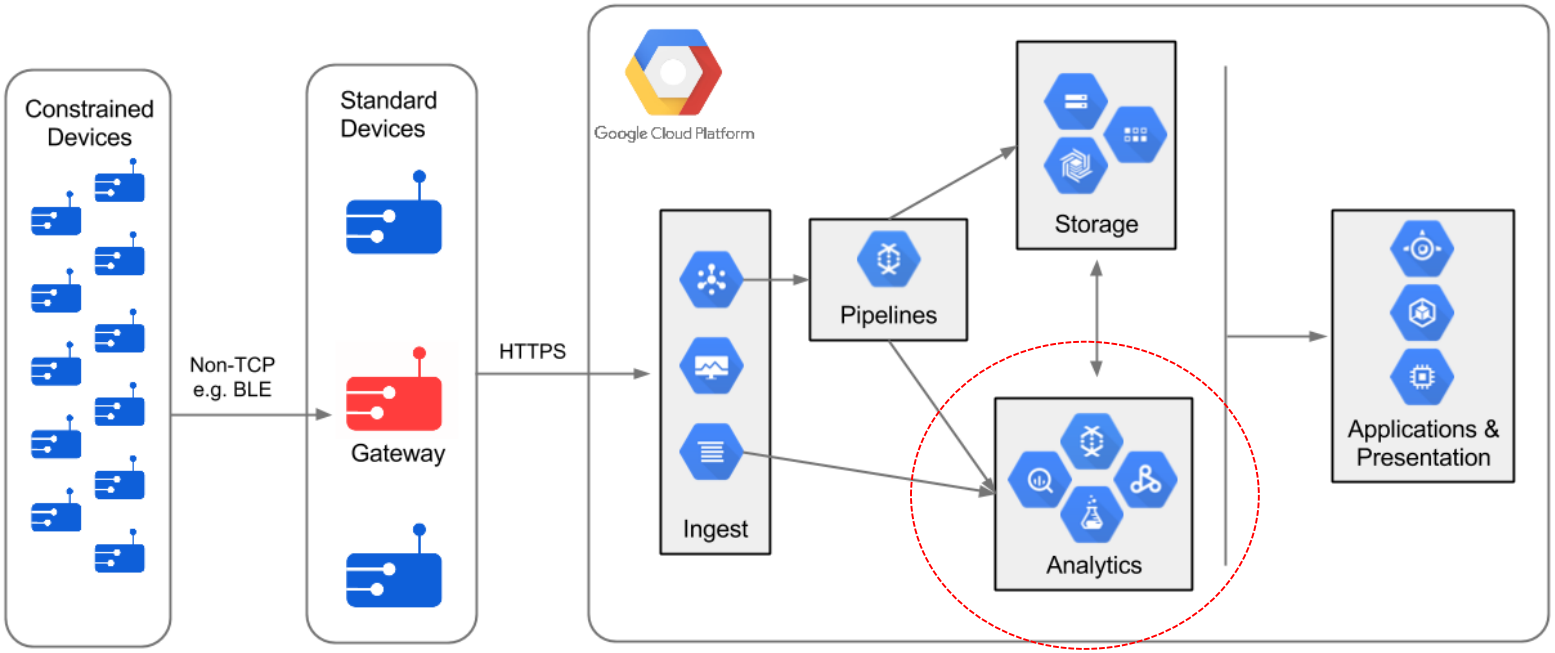
Faculty of Informatics, TU Wien

hong-linh.truong@tuwien.ac.at
http://dsg.tuwien.ac.at/staff/truong
@linhsolar

# What is this lecture about?

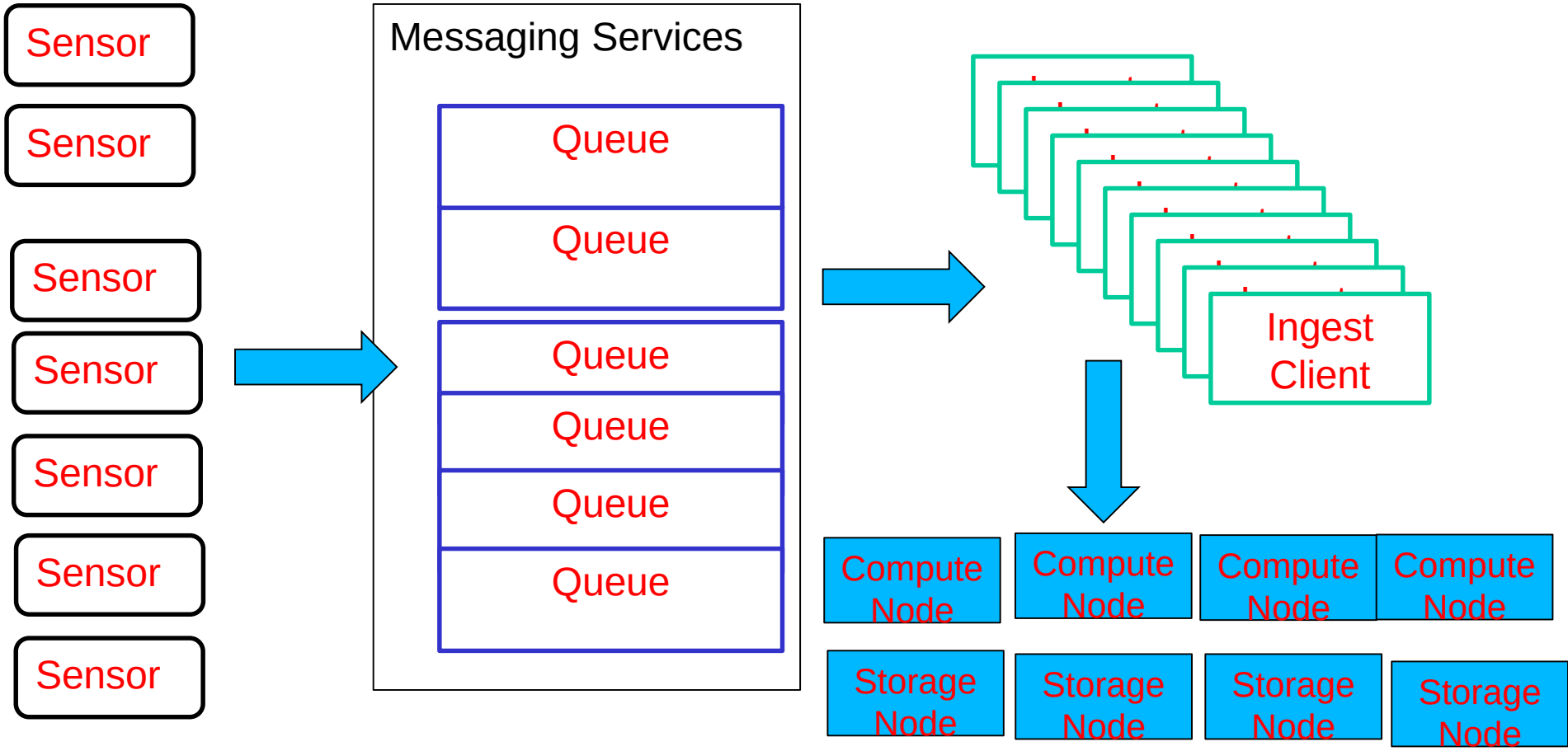
- Discuss key issues when we need multiple resources for computing, data storage and messaging in clouds
- Focus on high availability, high performance and high throughput aspects
- Examine distributed coordination with Zookeeper
- Work on various topics with real systems and concepts behind these systems

# Motivating example: IoT scenario



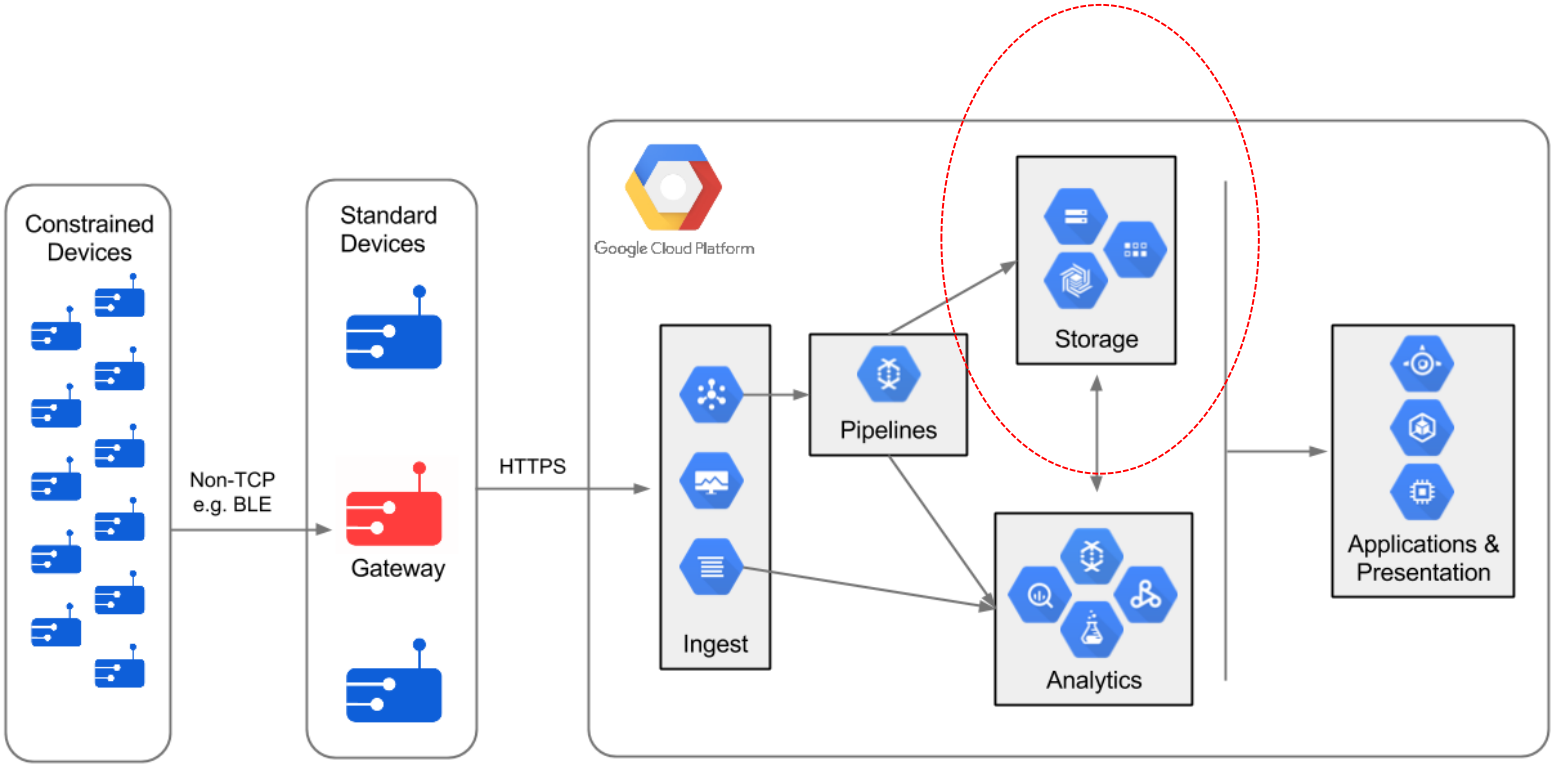

Source: https://cloud.google.com/solutions/architecture/streamprocessing

# Data, Services and Systems Management




High availability, high throughput, high performance

# Motivating example: IoT scenario

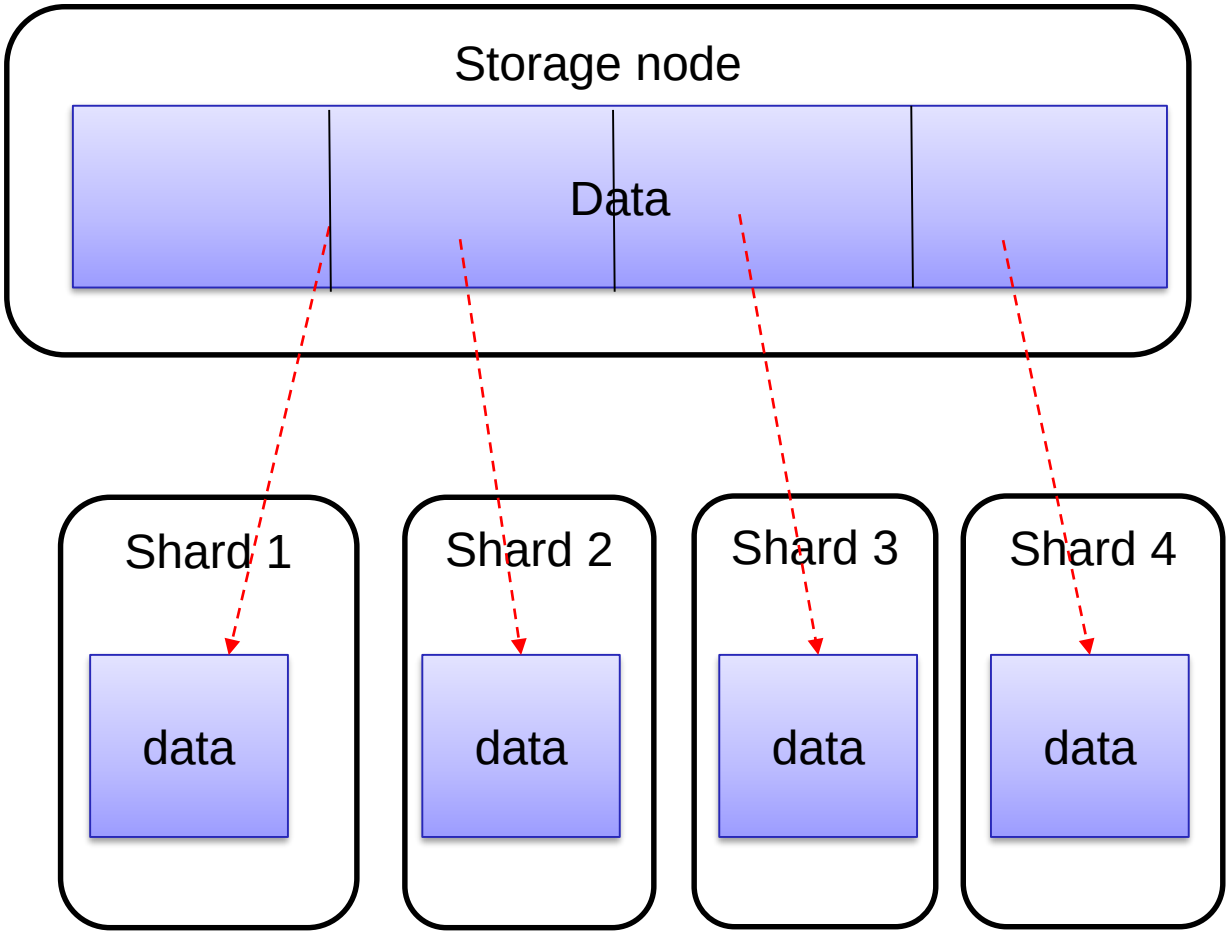



Source: https://cloud.google.com/solutions/architecture/streamprocessing

# Data Resource Management

Store a big data collection in a single node?

High availability?



# Geographic Data distribution

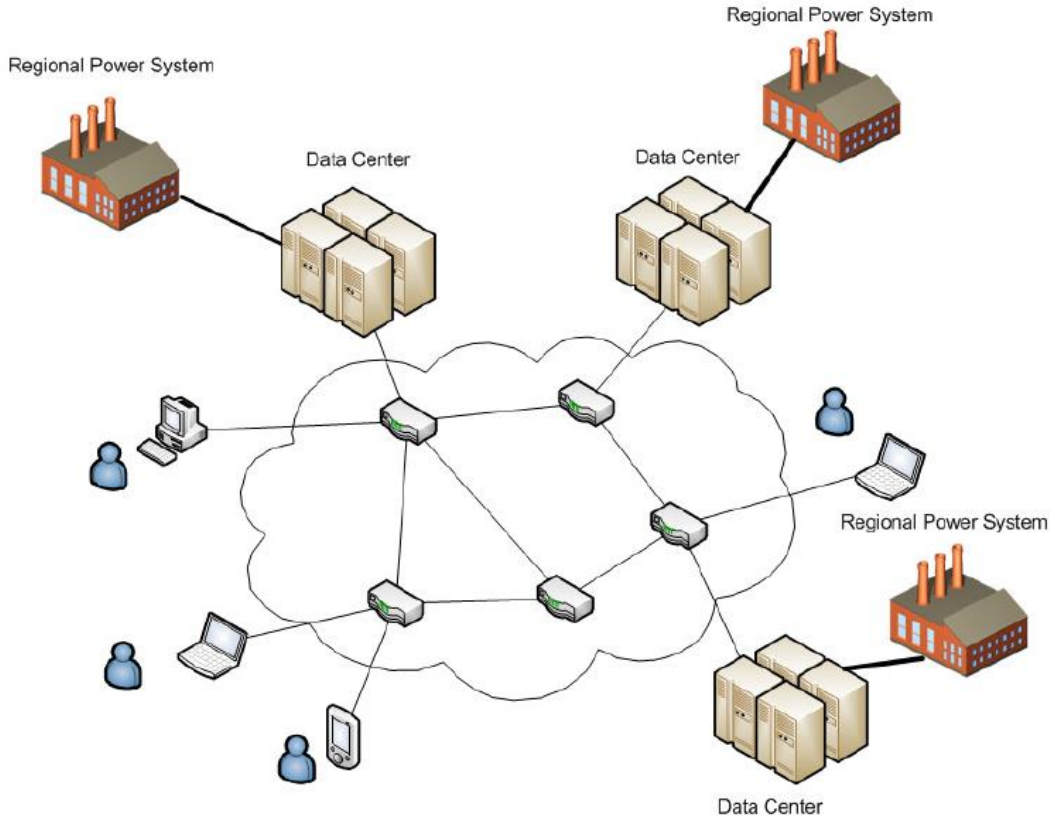
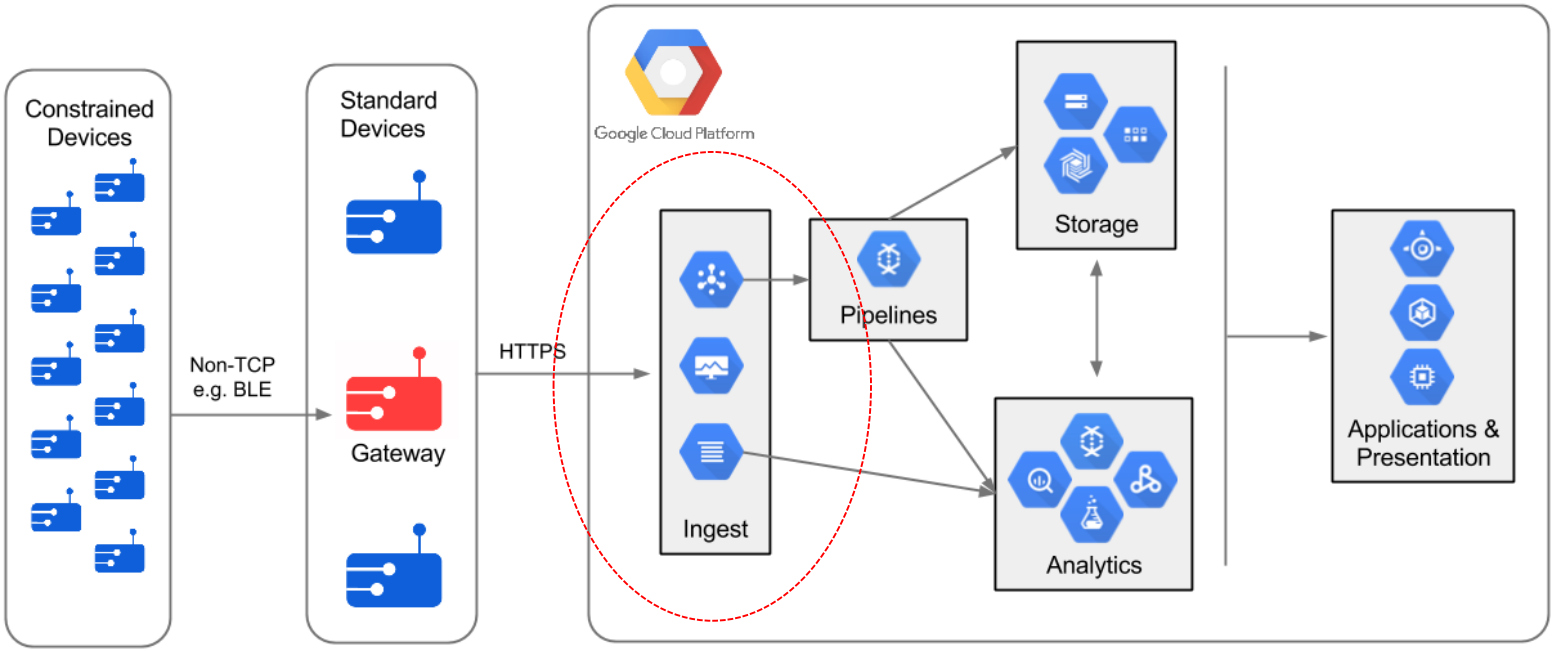



Figure 1. Model of service placement in geographically distributed data

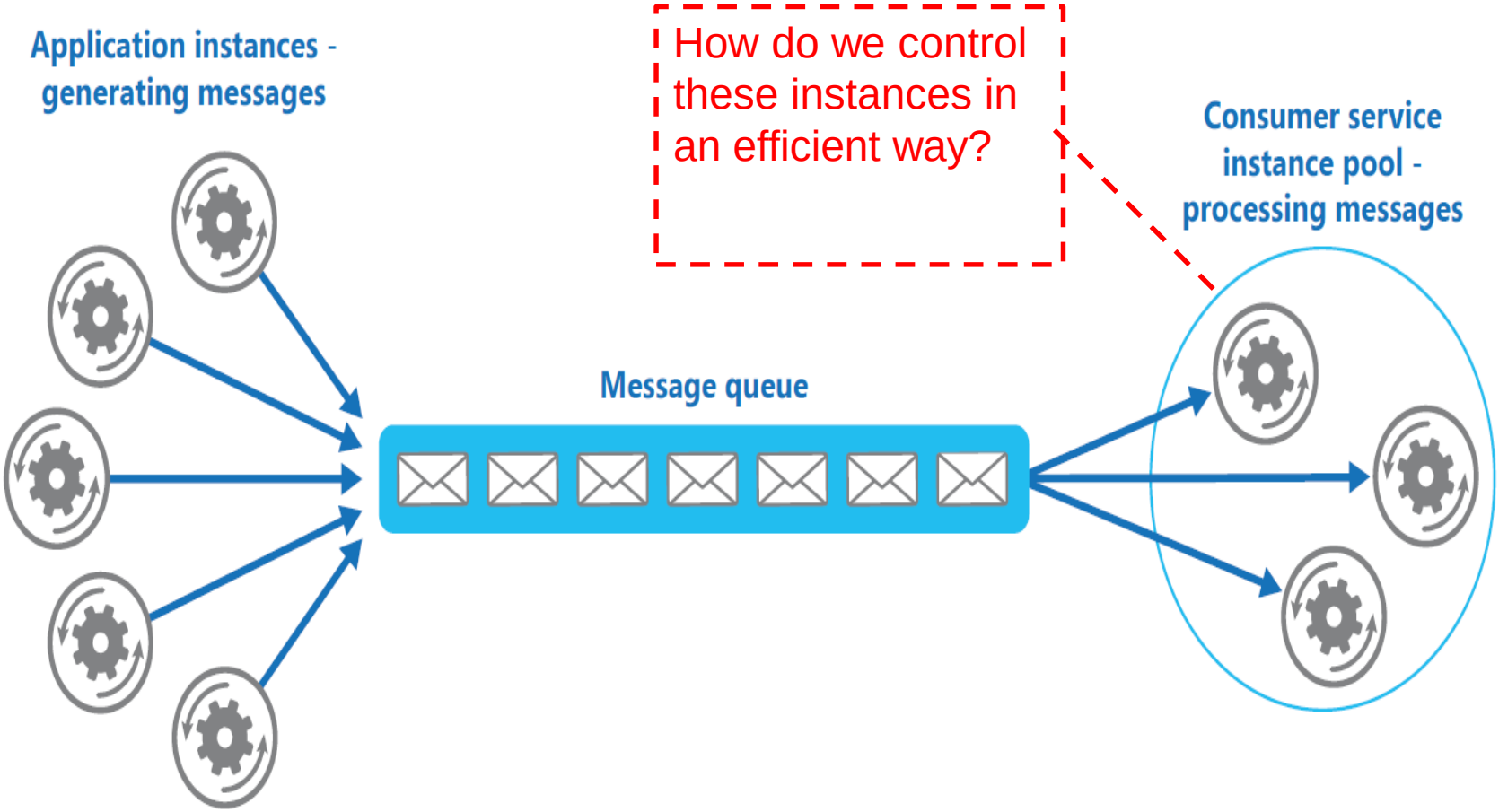
Source: Qi Zhang, Quanyan Zhu, Mohamed Faten Zhani, and Raouf Boutaba. 2012. Dynamic Service Placement in Geographically Distributed Clouds. In Proceedings of the 2012 IEEE 32nd International Conference on Distributed Computing Systems (ICDCS '12). IEEE Computer Society, Washington, DC, USA, 526-535.

# Motivating example: IoT scenario




Source: https://cloud.google.com/solutions/architecture/streamprocessing

# Queuing service management




Source: https://msdn.microsoft.com/en-us/library/dn568101.aspx

# Key problems

- How to deal with failures in complex cloud systems?
  - Master, worker, communication
- How to establish and maintain high availability and high performance?
  - Reduce latency and increase concurrent access

→ 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)

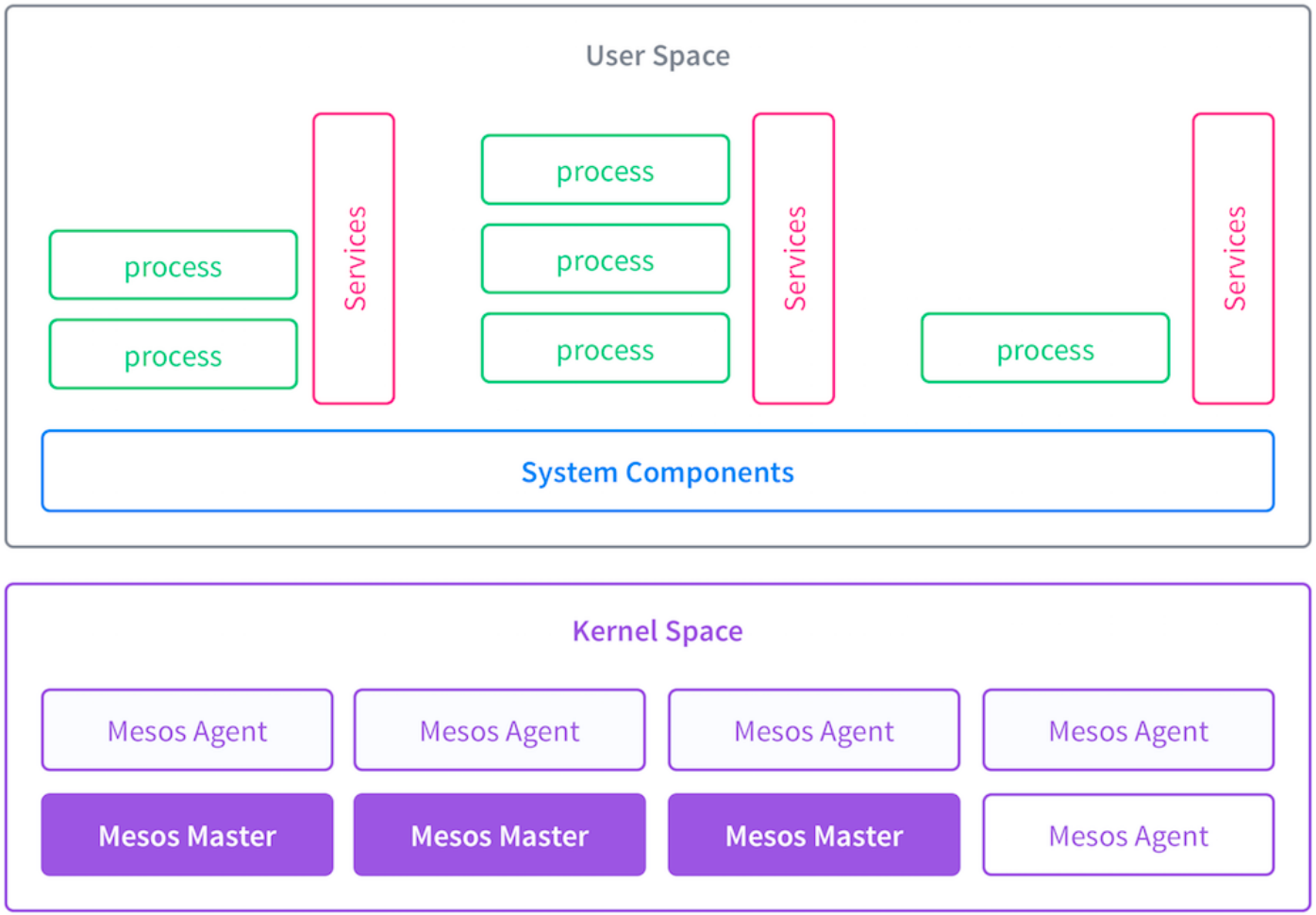
# Virtual data centers

- 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/containters + 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 (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



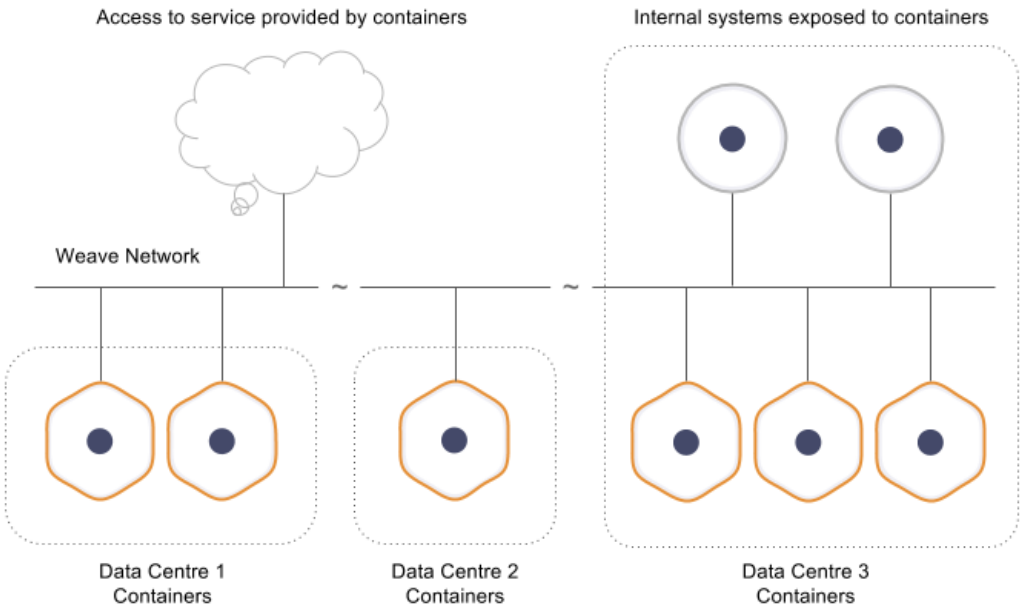

Source: https://docs.mesosphere.com/1.8/overview/architecture/

# Azure Cloud and Containers

https://docs.microsoft.com/en-us/dotnet/standard/microservices-architecture/architect-microservice-container-applications/scalable-available-multi-container-microservice-applications

# 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

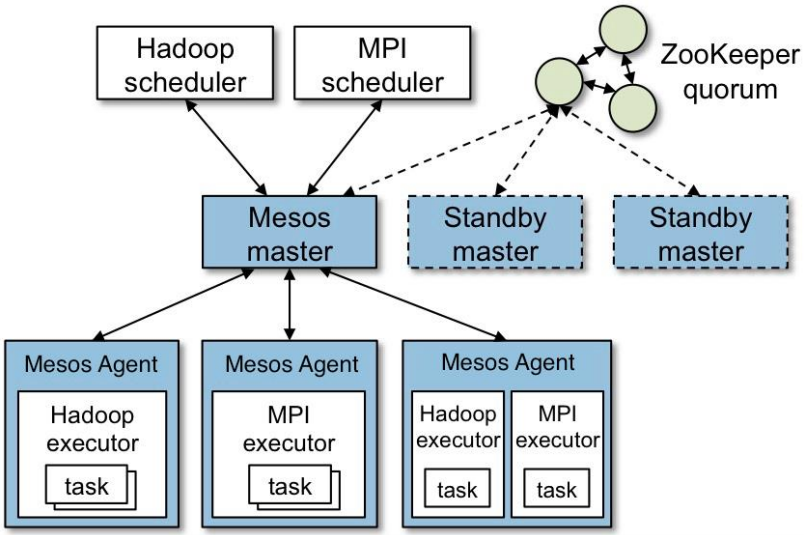



Figure source: http://mesos.apache.org/documentation/latest/architecture/

# 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

Sharding patterns/strategies: https://msdn.microsoft.com/en-us/library/dn589797.aspx

# Example

Google BigQuery support tables partition based on dates

FIGURE 5 Two Ways to Distribute 10 Years of Sensor Data for 1,000 Sites over 10 Machines

**Node 1**

| timestamp | sensor | reading |
|---|---|---|
| 19990101000000 | 1 | |
| 19990101000015 | 1 | |
| 19990101000030 | 1 | |
| 20081231235930 | 1 | |
| 20081231235945 | 1 | |
| 19990101000000 | 2 | |
| 19990101000015 | 2 | |
| 19990101000030 | 2 | |
| 20081231235930 | 2 | |
| 20081231235945 | 2 | |
| 19990101000000 | 3 | |
| 20081231235945 | 100 | |

**Node 2**

| timestamp | sensor | reading |
|---|---|---|
| 19990101000000 | 101 | |
| 19990101000015 | 101 | |
| 19990101000030 | 101 | |
| 20081231235930 | 101 | |
| 20081231235945 | 101 | |
| 19990101000000 | 102 | |
| 19990101000015 | 102 | |
| 19990101000030 | 102 | |
| 20081231235930 | 102 | |
| 20081231235945 | 102 | |
| 19990101000000 | 103 | |
| 20081231235945 | 200 | |

**Node 10**

| timestamp | sensor | reading |
|---|---|---|
| 19990101000000 | 901 | |
| 19990101000015 | 901 | |
| 19990101000030 | 901 | |
| 20081231235930 | 901 | |
| 20081231235945 | 901 | |
| 19990101000000 | 902 | |
| 19990101000015 | 902 | |
| 19990101000030 | 902 | |
| 20081231235930 | 902 | |
| 20081231235945 | 902 | |
| 19990101000000 | 103 | |
| 20081231235945 | 1000 | |

**Node 1**

| timestamp | sensor | reading |
|---|---|---|
| 19990101000000 | 1 | |
| 19990101000000 | 2 | |
| 19990101000000 | 3 | |
| 19990101000000 | 1000 | |
| 19990101000015 | 1 | |
| 19990101000015 | 2 | |
| 19990101000015 | 3 | |
| 19990101000015 | 4 | |
| 19990101000015 | 1000 | |
| 19990101000030 | 1 | |
| 19990101000030 | 2 | |
| 19991231235945 | 1000 | |

**Node 2**

| timestamp | sensor | reading |
|---|---|---|
| 20000101000000 | 1 | |
| 20000101000000 | 2 | |
| 20000101000000 | 3 | |
| 20000101000000 | 1000 | |
| 20000101000015 | 1 | |
| 20000101000015 | 2 | |
| 20000101000015 | 3 | |
| 20000101000015 | 4 | |
| 20000101000015 | 1000 | |
| 20000101000030 | 1 | |
| 20000101000030 | 2 | |
| 20001231235945 | 1000 | |

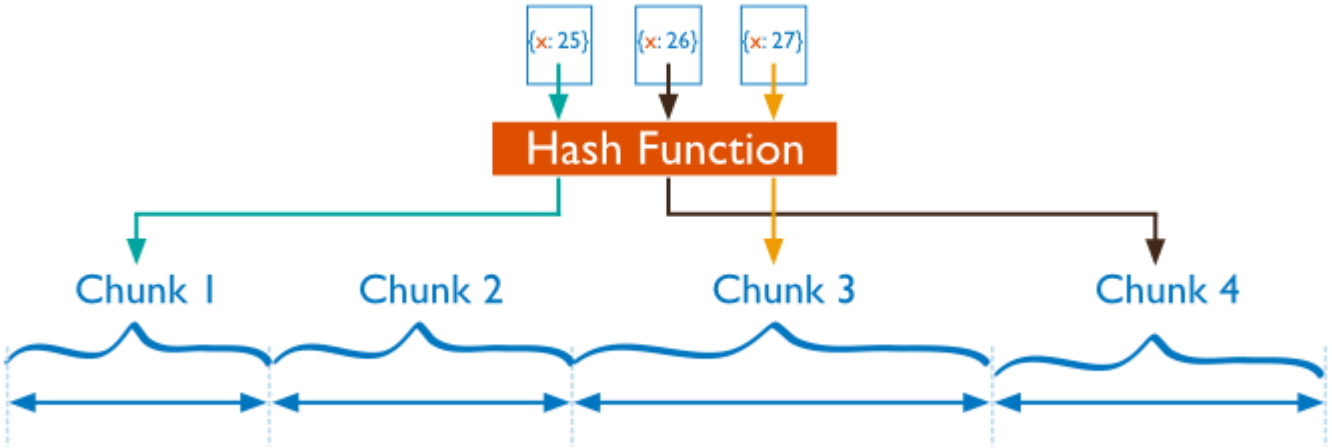
**Node 10**

| timestamp | sensor | reading |
|---|---|---|
| 20080101000000 | 1 | |
| 20080101000000 | 2 | |
| 20080101000000 | 3 | |
| 20080101000000 | 1000 | |
| 20080101000015 | 1 | |
| 20080101000015 | 2 | |
| 20080101000015 | 3 | |
| 20080101000015 | 4 | |
| 20080101000015 | 1000 | |
| 20080101000030 | 1 | |
| 20080101000030 | 2 | |
| 20081231235945 | 1000 | |

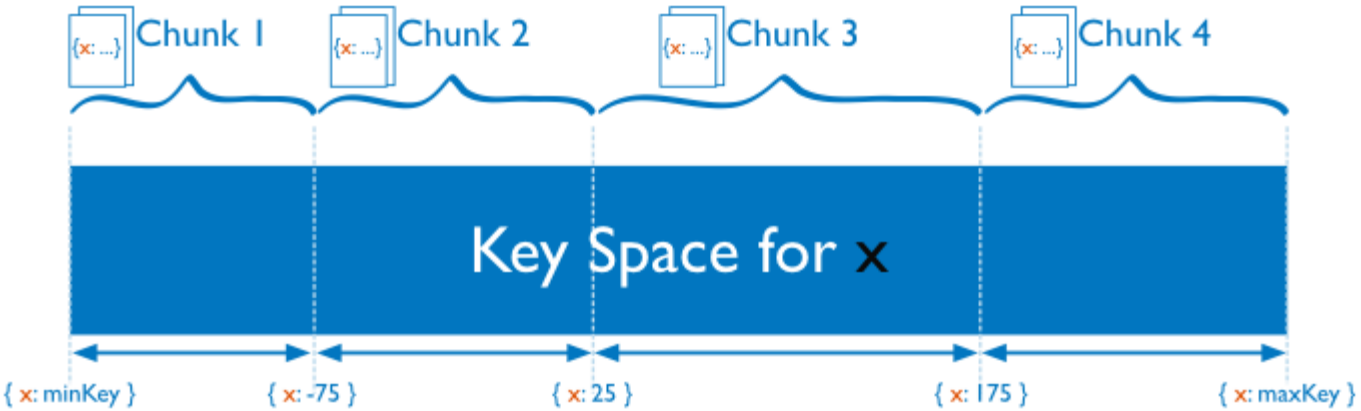
Source: http://queue.acm.org/detail.cfm?id=1563874

# Example Strategies in Mongodb

Hash

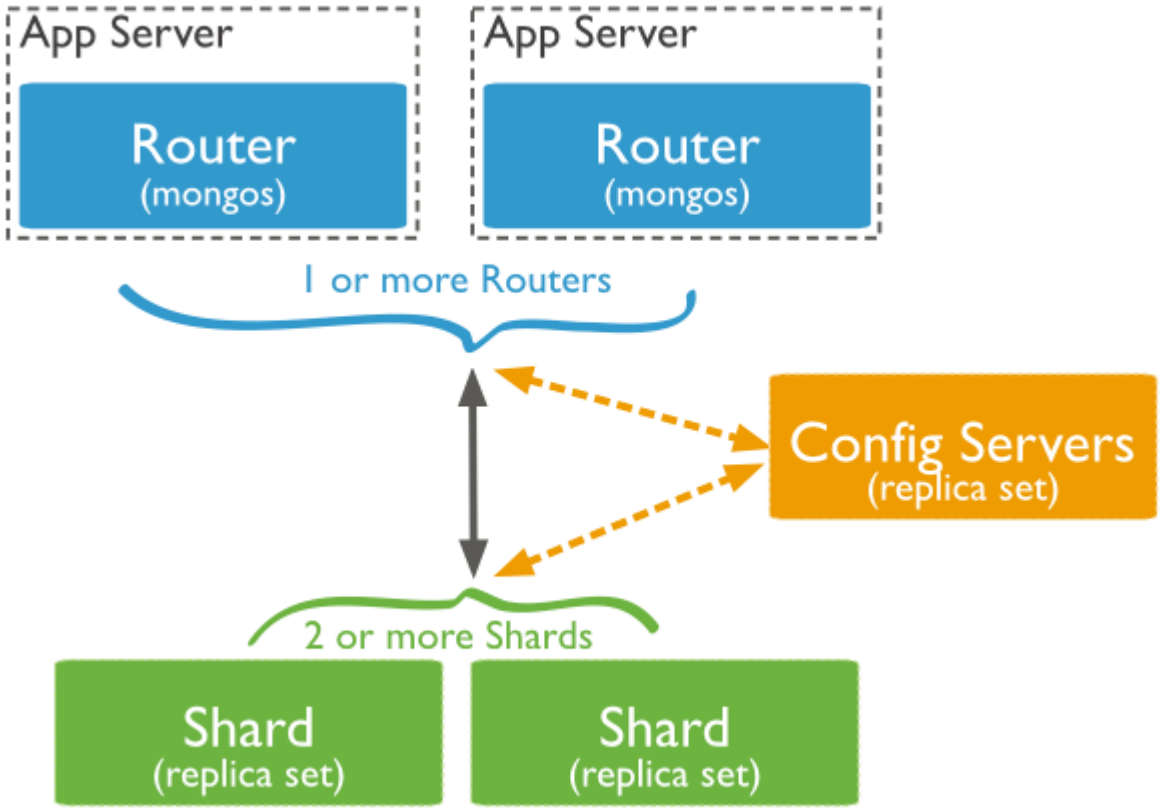



Range



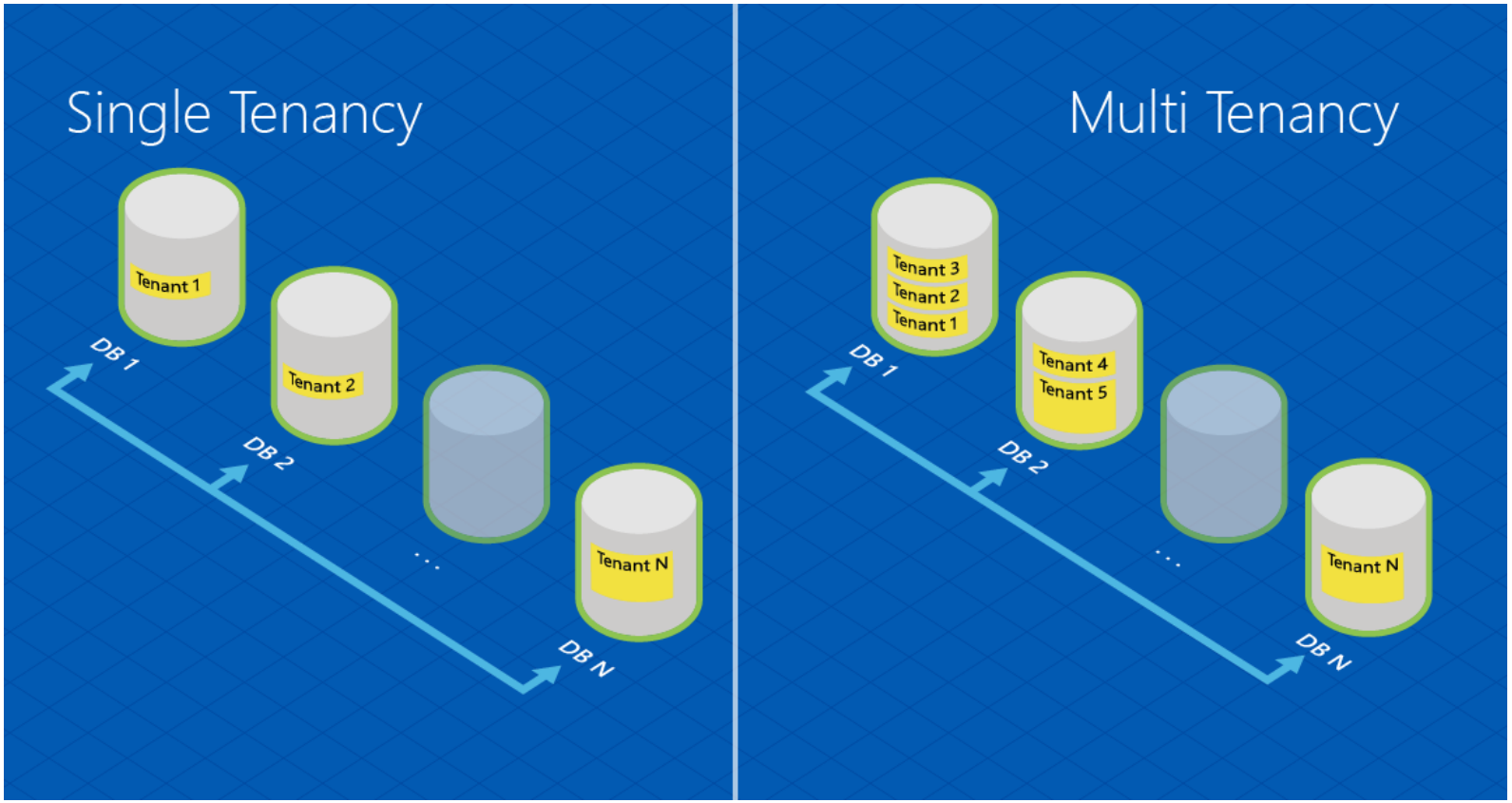

Source: https://docs.mongodb.com/v3.2/sharding/

# Shard and routing



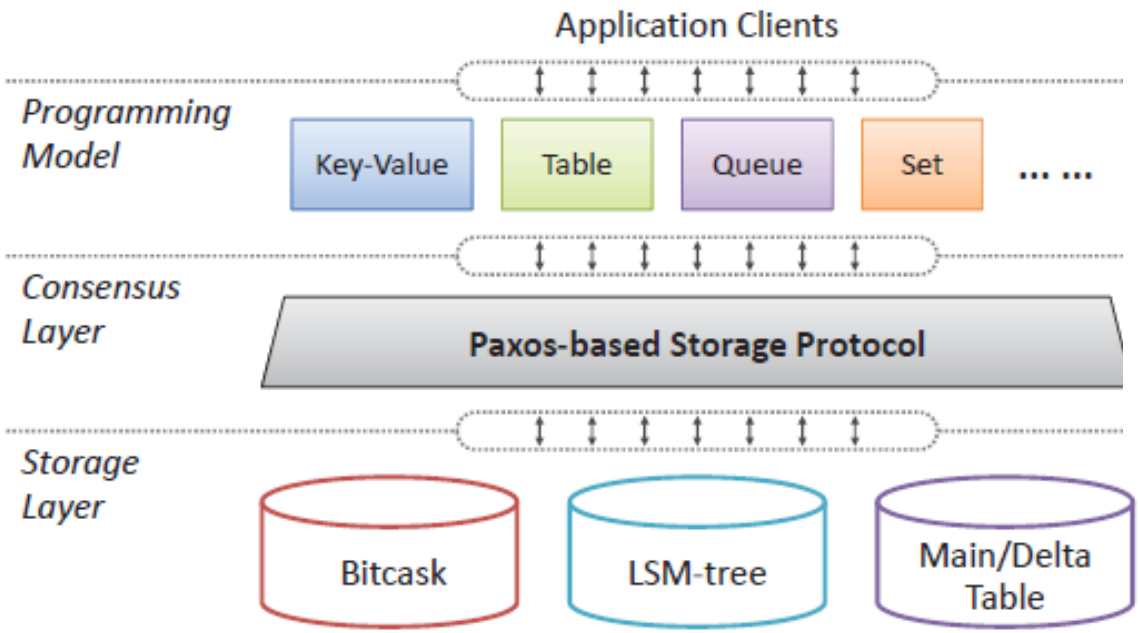

Source: https://docs.mongodb.com/v3.2/sharding/

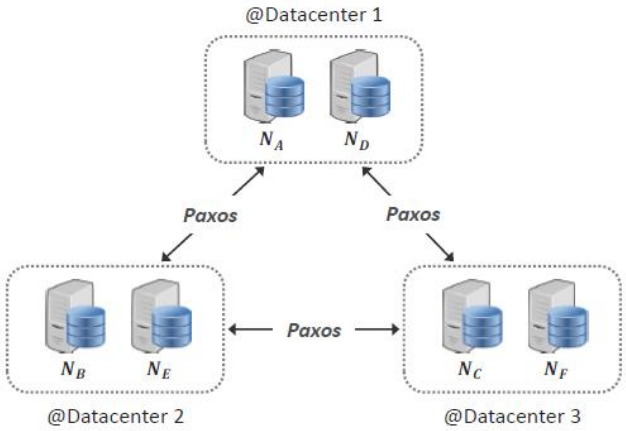
# Single or multi-tenant sharding




Source: https://azure.microsoft.com/en-us/documentation/articles/sql-database-elastic-scale-introduction/

# Resources and Consensus




Source: Jianjun Zheng, Qian Lin, Jiatao Xu, Cheng Wei, Chuwei Zeng, Pingan Yang, and Yunfan Zhang. 2017. PaxosStore: high-availability storage made practical in WeChat. Proc. VLDB Endow. 10, 12 (August 2017), 1730-1741. DOI: https://doi.org/10.14778/3137765.3137778

# 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

# MESSAGING SERVICES MANAGEMENT

# 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
- Cient/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

Source: Philippe Dobbelaere and Kyumars Sheykh Esmaili. 2017. Kafka versus RabbitMQ: A comparative study of two industry reference publish/subscribe implementations: Industry Paper. In Proceedings of the 11th ACM International Conference on Distributed and Event-based Systems (DEBS '17). ACM, New York, NY, USA, 227-238. DOI: https://doi.org/10.1145/3093742.3093908

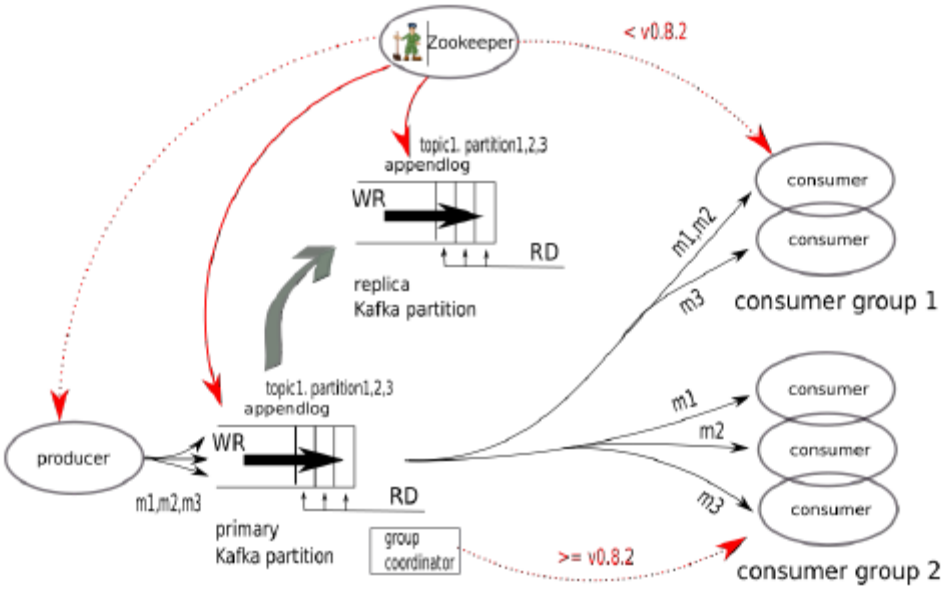



Figure 1: Kafka Architecture

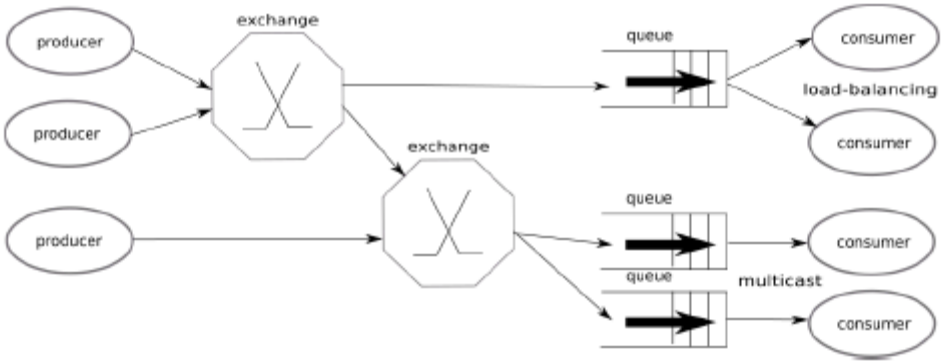
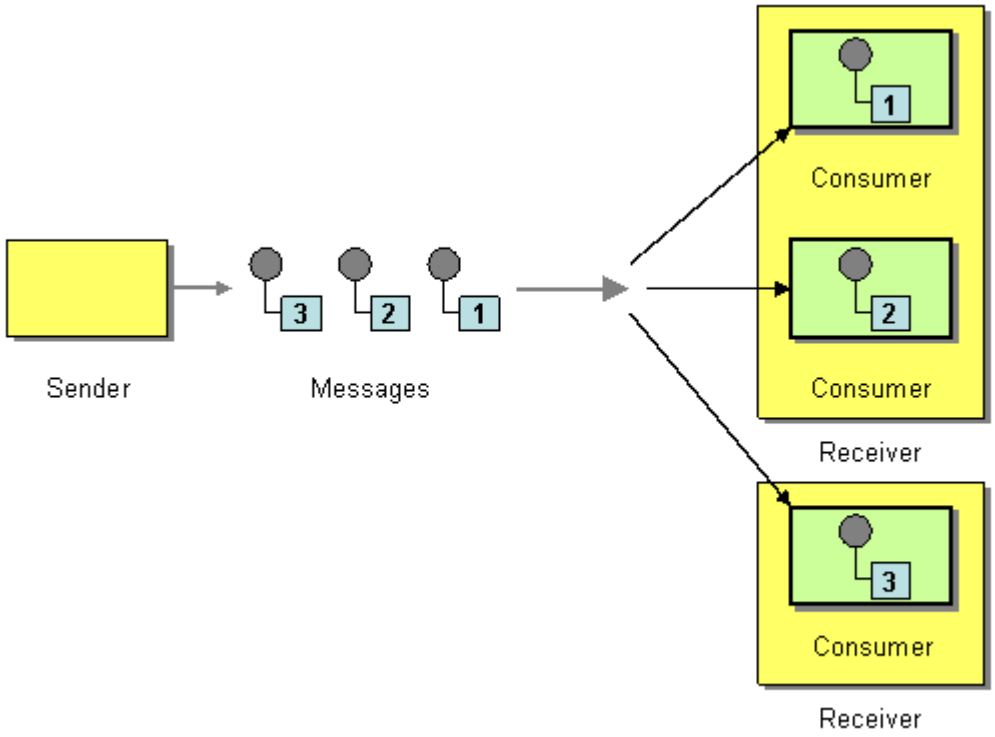



Figure 2: RabbitMQ (AMQP) Architecture

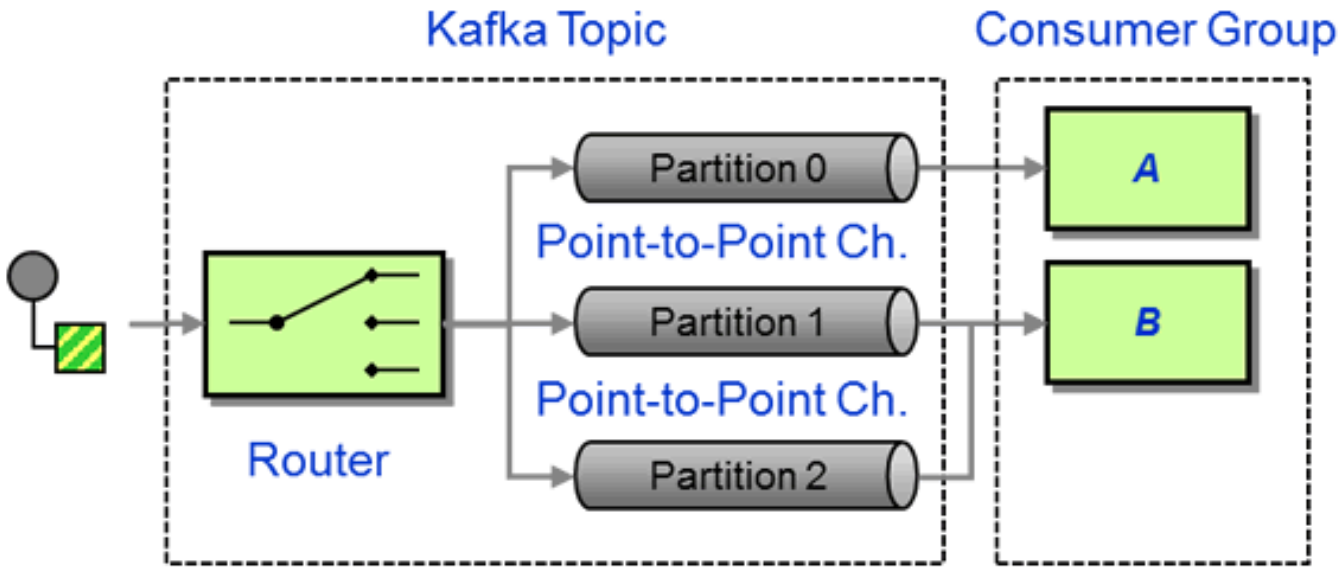
# Consumer Load balancing




What do you have to do here?

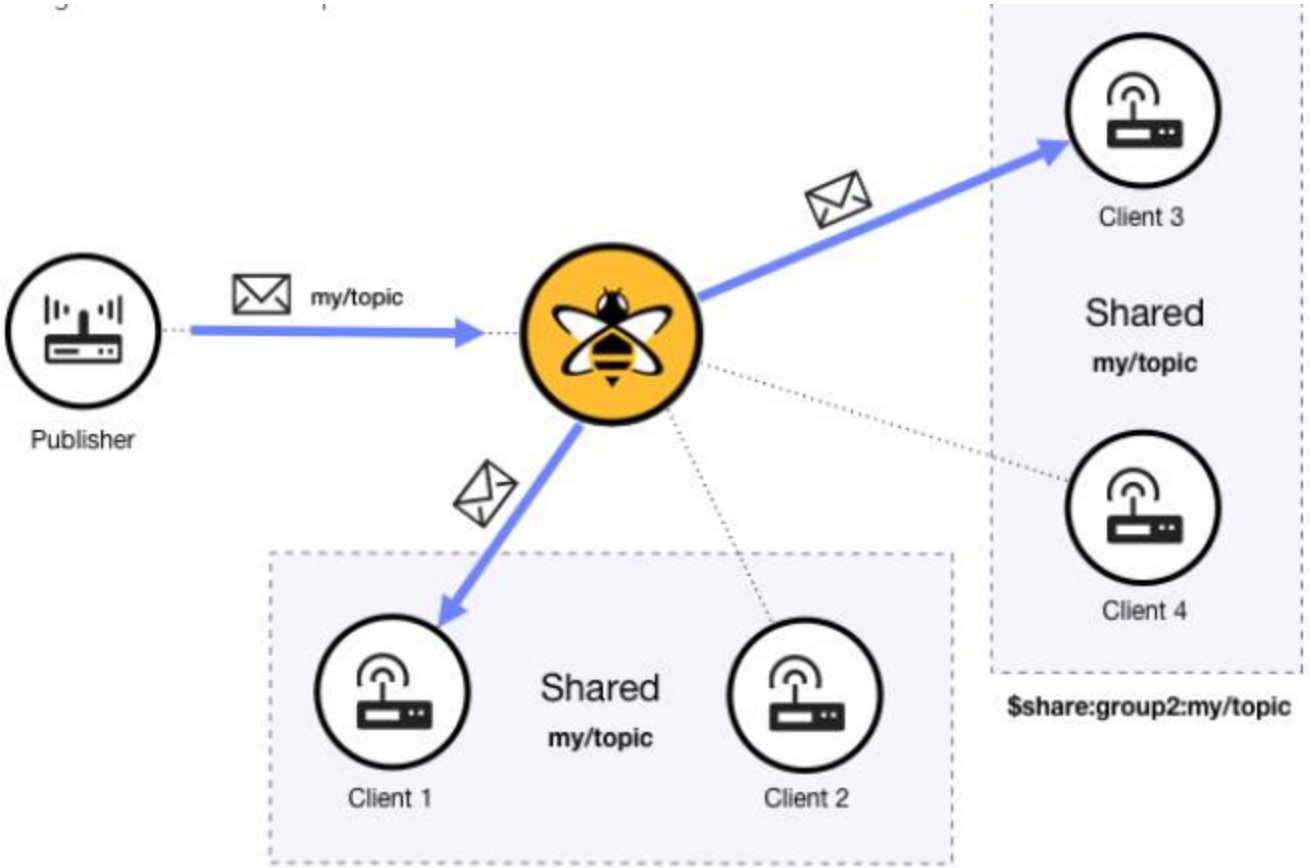
Source: http://www.enterpriseintegrationpatterns.com/patterns/messaging/CompetingConsumers.html/

# Client Load in Apache Kafka




Source: http://www.enterpriseintegrationpatterns.com/patterns/messaging/CompetingConsumers.html/

# Shared Topics with MQTT by HiveMQ

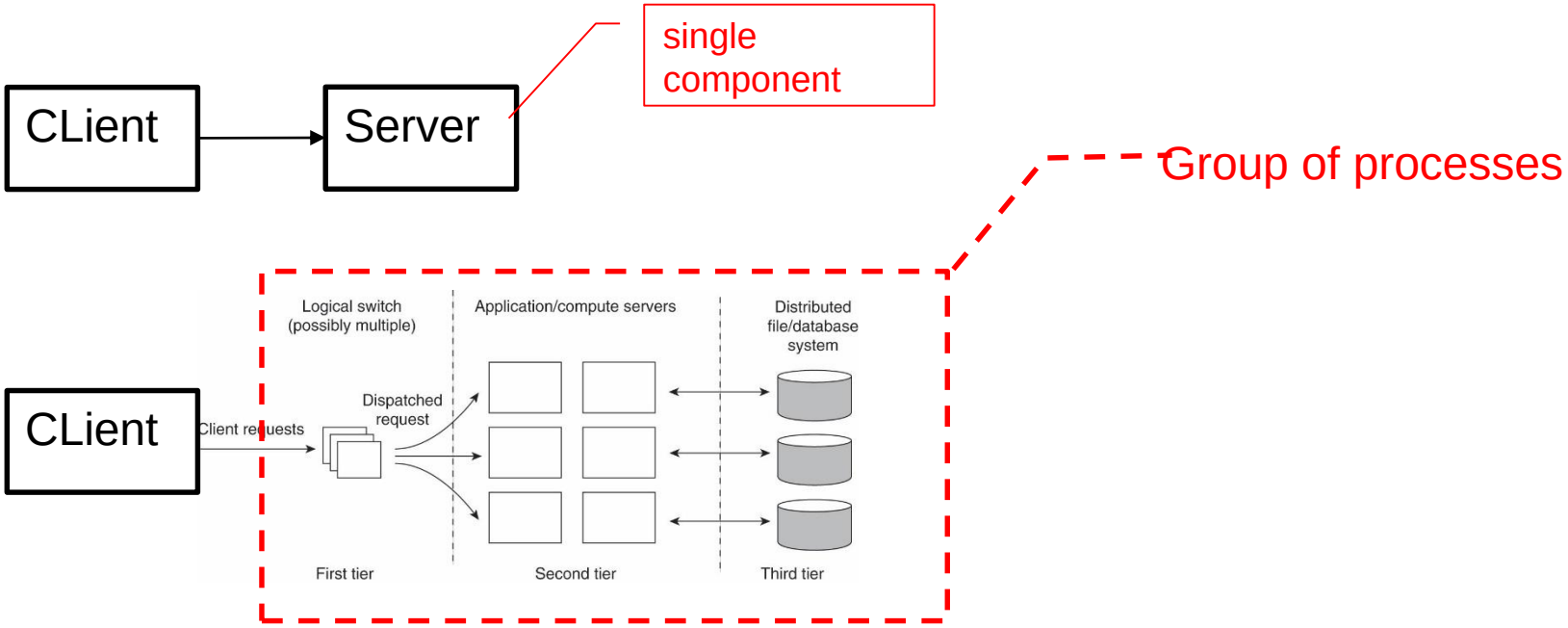



Source: https://www.hivemq.com/blog/mqtt-client-load-balancing-with-shared-subscriptions/

# DISTRIBUTED COORDINATION FOR CLOUDS

# Group redundancy architecture

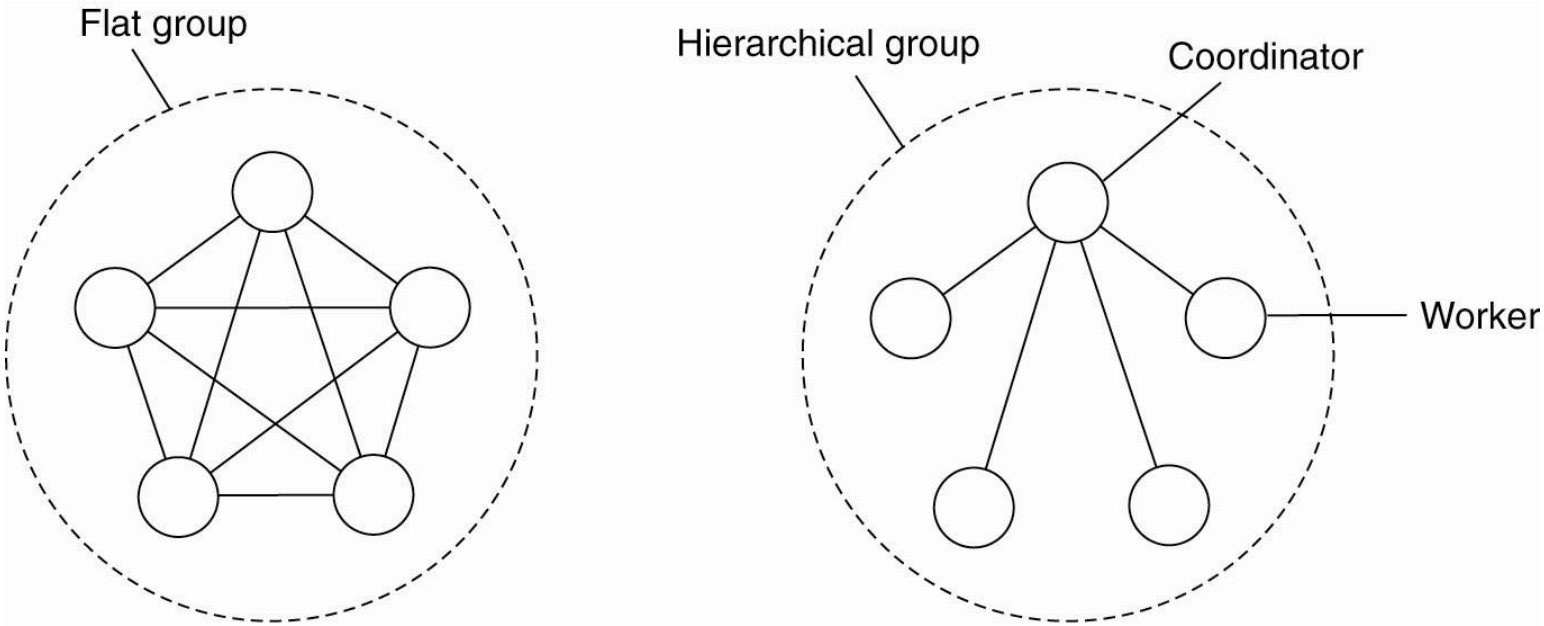
- Use group architecture for redundancy in order to support failure masking




Source: Andrew S. Tanenbaum and Maarten van Steen, Distributed Systems – Principles and Paradigms, 2nd Edition, 2007, Prentice-Hall

# Design Flat Groups versus Hierarchical Groups

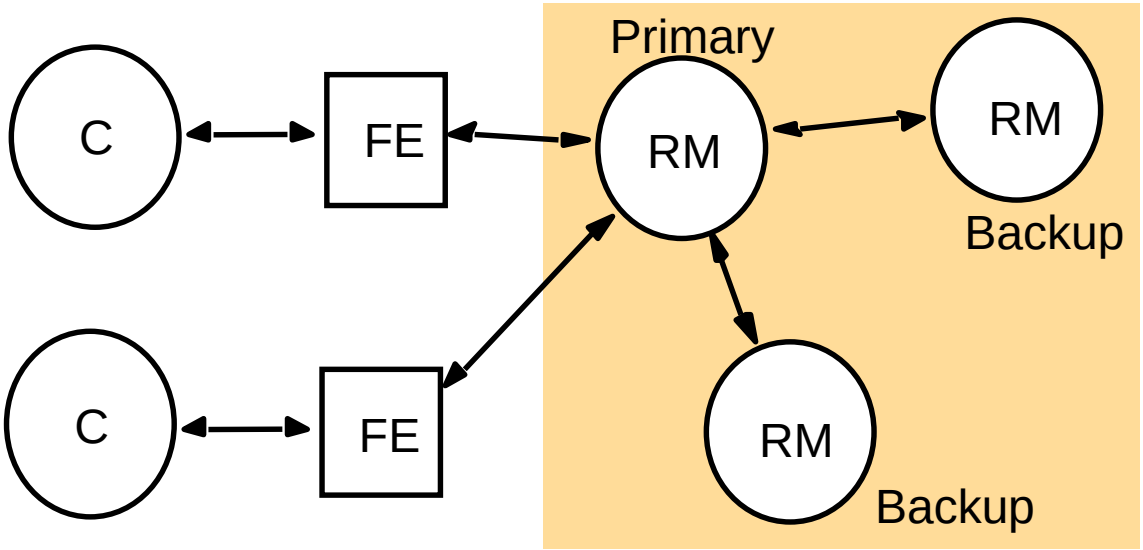
Structure a system (communication, servers, services, etc.) using a group so we can deal failures using collective capabilities



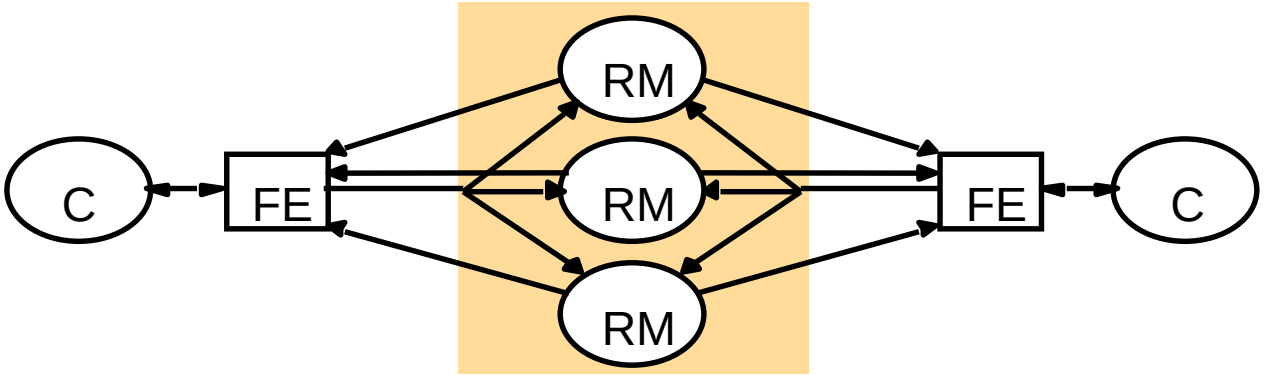

Source: Andrew S. Tanenbaum and Maarten van Steen, Distributed Systems – Principles and Paradigms, 2nd Edition, 2007, Prentice-Hall

# 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

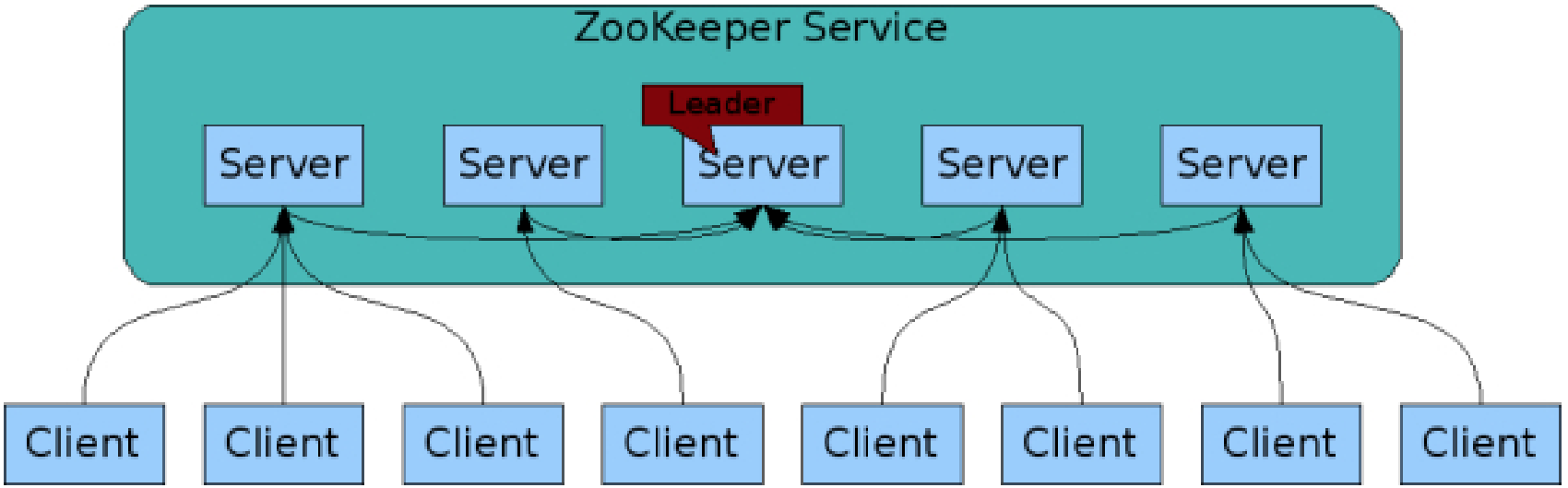
Notes from the paper: “server replication (SR), log replication (LR), synchronization service (SS), barrier orchestration (BO), service discovery (SD), group membership (GM), leader election (LE), metadata management (MM) and distributed queues (Q)”

TABLE 4. PATTERNS OF PAXOS USE IN PROJECTS

| | | Usage Patterns | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| **Project** | **Consensus System** | **SR** | **LR** | **SS** | **BO** | **SD** | **GM** | **LE** | **MM** | **Q** |
| GFS | Chubby | | | ✓ | | | | ✓ | ✓ | |
| Borg | Chubby/Paxos | ✓ | | | | ✓ | | ✓ | | |
| Kubernetes | etcd | | | | | | ✓ | | ✓ | |
| Megastore | Paxos | | ✓ | | | | | | | |
| Spanner | Paxos | ✓ | | | | | | | | |
| Bigtable | Chubby | | | | | | ✓ | ✓ | ✓ | |
| Hadoop/HDFS | ZooKeeper | ✓ | | | | | | ✓ | | |
| HBase | ZooKeeper | ✓ | | ✓ | | | ✓ | | ✓ | |
| Hive | ZooKeeper | | | ✓ | | | | | ✓ | |
| Configerator | Zeus | | | | | | | | ✓ | |
| Cassandra | ZooKeeper | | | | | ✓ | | ✓ | ✓ | |
| Accumulo | ZooKeeper | | ✓ | ✓ | | | | | ✓ | |
| BookKeeper | ZooKeeper | | | | | | ✓ | | ✓ | |
| Hedwig | ZooKeeper | | | | | | ✓ | | ✓ | |
| Kafka | ZooKeeper | | | | | | ✓ | ✓ | ✓ | |
| Solr | ZooKeeper | | | | | | | ✓ | ✓ | ✓ |
| Giraph | ZooKeeper | | ✓ | | ✓ | | | | ✓ | |
| Hama | ZooKeeper | | | | ✓ | | | | | |
| Mesos | ZooKeeper | | | | | | | ✓ | | |
| CoreOS | etcd | | | | | ✓ | | | | |
| OpenStack | ZooKeeper | | | | | ✓ | | | | |
| Neo4j | ZooKeeper | | | ✓ | | | | ✓ | | |

Ailidani Ailijiang, Aleksey Charapkoy and Murat Demirbasz
, Consensus in the Cloud: Paxos Systems Demystified, http://www.cse.buffalo.edu/tech-reports/2016-02.pdf

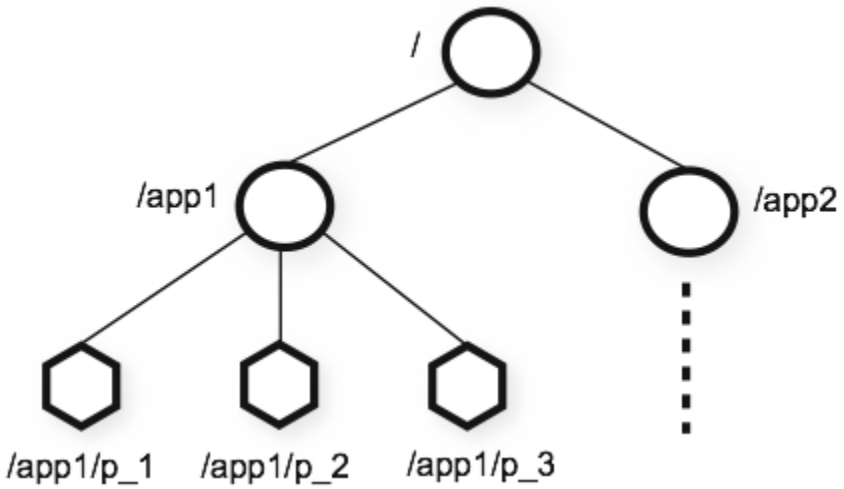
# ZooKeeper Service




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

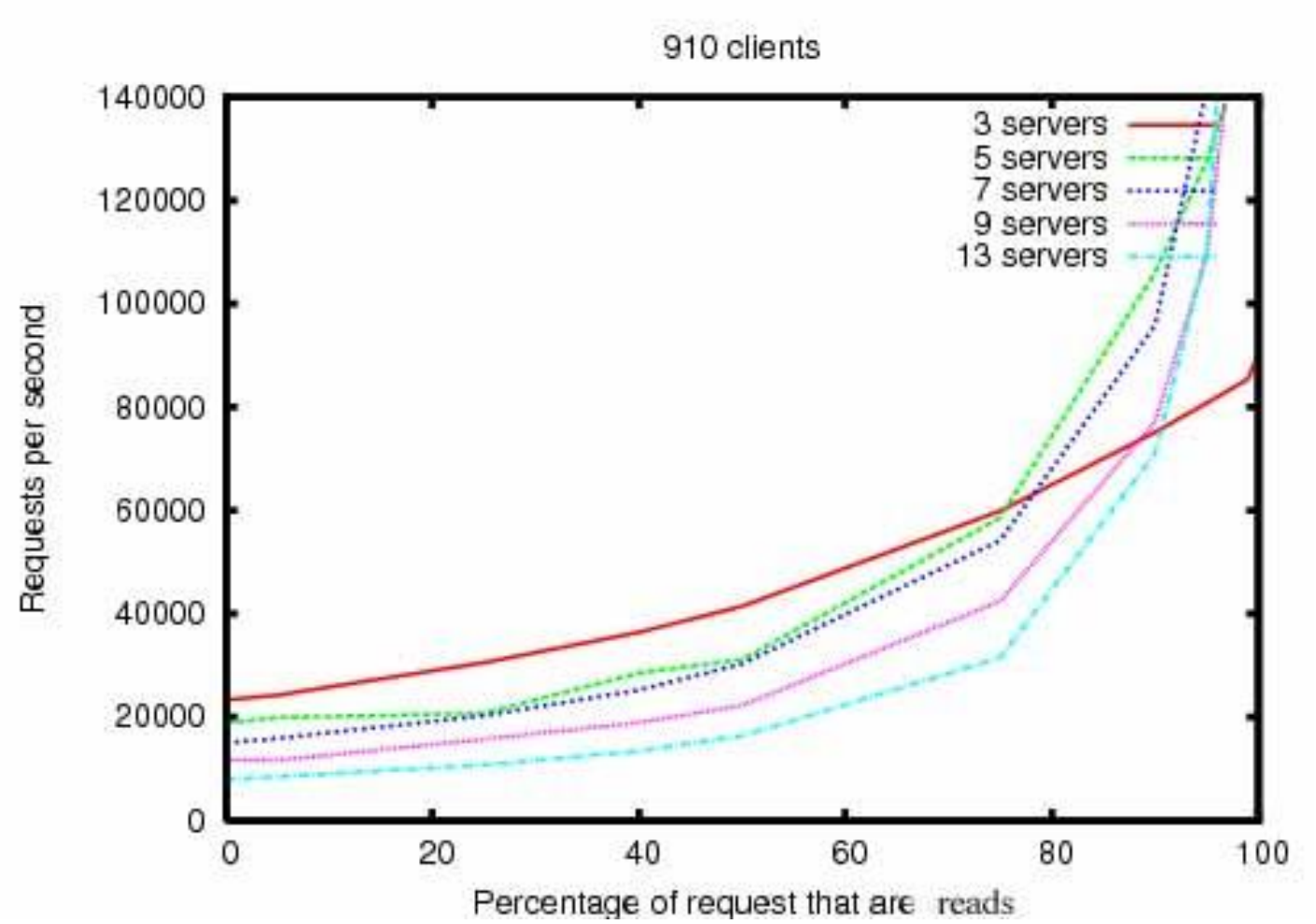
# Guarantees

- Sequential Consistency
- Atomicity - either succeed or fail
- Single System Image
- Reliability
- Timeliness

# Server configuration

- Standalone versus Quorums
- Quorums
  - Support failure recovery for ZooKeeper servers
  - Provide replications
  - Faster and more reliability

# Performance




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

# TOPICS FOR YOU

# High availability in Hadoop

- Concepts
  - https://journalofcloudcomputing.springeropen.com/articles/10.1186/s13677-016-0066-8
  - Mina Nabi, Maria Toeroe, and Ferhat Khendek. 2016. Availability in the cloud. J. Netw. Comput. Appl. 60, C (January 2016), 54-67. DOI: http://dx.doi.org/10.1016/j.jnca.2015.11.014
  - 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
  - Cuong Manh Pham, Victor Dogaru, Rohit Wagle, Chitra Venkatramani, Zbigniew Kalbarczyk, and Ravishankar Iyer. 2014. An evaluation of zookeeper for high availability in system S. In Proceedings of the 5th ACM/SPEC international conference on Performance engineering (ICPE '14). ACM, New York, NY, USA, 209-217. DOI: https://doi.org/10.1145/2568088.2576801
- Practical work with Hadoop and Zookeeper
  - https://hadoop.apache.org/docs/stable/hadoop-project-dist/hadoop-hdfs/HDFSHighAvailabilityWithNFS.html
  - https://zookeeper.apache.org/

# High availability with cluster of containers

- Concepts
  - https://researcher.watson.ibm.com/researcher/files/us-sseelam/Woc2016-KubeHA-Final.pdf
  - W. Li, A. Kanso and A. Gherbi, "Leveraging Linux Containers to Achieve High Availability for Cloud Services," 2015 IEEE International Conference on Cloud Engineering, Tempe, AZ, 2015, pp. 76-83.
  - doi: 10.1109/IC2E.2015.17
  - https://arxiv.org/pdf/1708.08399.pdf
- Practical work:
  - Kubernes: https://kubernetes.io/docs/admin/high-availability/
  - Docker: https://docs.docker.com/datacenter/ucp/1.1/high-availability/set-up-high-availability/
  - https://docs.mesosphere.com/
  - https://www.consul.io/docs/internals/consensus.html#deployment-table

# High availability for resource management systems

- Concepts – Mesos, YARN, Zookeeper
  - https://www.safaribooksonline.com/library/view/apache-mesos-essentials/9781783288762/
  - Benjamin Hindman, Andy Konwinski, Matei Zaharia, Ali Ghodsi, Anthony D. Joseph, Randy Katz, Scott Shenker, and Ion Stoica. 2011. Mesos: a platform for fine-grained resource sharing in the data center. In Proceedings of the 8th USENIX conference on Networked systems design and implementation (NSDI'11). USENIX Association, Berkeley, CA, USA, 295-308
- Practical work:
  - http://mesos.apache.org/documentation/latest/high-availability/
  - https://dcos.io/docs/1.7/overview/high-availability/
  - https://hadoop.apache.org/docs/stable/hadoop-yarn/hadoop-yarn-site/ResourceManagerHA.html

# High availability for MongoDB

- Concepts
  - https://raft.github.io/
  - In Search of an Understandable Consensus Algorithm, (Extended Version), Diego Ongaro and John Ousterhout, Stanford University
  - Wenbin Jiang, Lei Zhang, Xiaofei Liao, Hai Jin, and Yaqiong Peng. 2014. A novel clustered MongoDB-based storage system for unstructured data with high availability. Computing 96, 6 (June 2014), 455-478. DOI=http://dx.doi.org/10.1007/s00607-013-0355-8
  - Stefan Brenner, Benjamin Garbers, and Rüdiger Kapitza. 2014. Adaptive and Scalable High Availability for Infrastructure Clouds. In Proceedings of the 14th IFIP WG 6.1 International Conference on Distributed Applications and Interoperable Systems - Volume 8460, Kostas Magoutis and Peter Pietzuch (Eds.), Vol. 8460. Springer-Verlag New York, Inc., New York, NY, USA, 16-30. DOI=http://dx.doi.org/10.1007/978-3-662-43352-2_2
- Practical work:
  - https://docs.mongodb.com/manual/replication/
  - https://docs.mongodb.com/manual/core/replica-set-architecture-geographically-distributed/
  - https://www.mongodb.com/presentations/replication-election-and-consensus-algorithm-refinements-for-mongodb-3-2

# High availability for RabbitMQ or Kafka

- Concepts
  - Philippe Dobbelaere and Kyumars Sheykh Esmaili. 2017. Kafka versus RabbitMQ: A comparative study of two industry reference publish/subscribe implementations: Industry Paper. In Proceedings of the 11th ACM International Conference on Distributed and Event-based Systems (DEBS '17). ACM, New York, NY, USA, 227-238. DOI: https://doi.org/10.1145/3093742.3093908
  - Stefan Brenner, Benjamin Garbers, and Rüdiger Kapitza. 2014. Adaptive and Scalable High Availability for Infrastructure Clouds. In Proceedings of the 14th IFIP WG 6.1 International Conference on Distributed Applications and Interoperable Systems - Volume 8460, Kostas Magoutis and Peter Pietzuch (Eds.), Vol. 8460. Springer-Verlag New York, Inc., New York, NY, USA, 16-30. DOI=http://dx.doi.org/10.1007/978-3-662-43352-2_2
- Practical work:
  - https://www.rabbitmq.com/pacemaker.html
  - https://www.rabbitmq.com/ha.html
  - http://clusterlabs.org/
  - https://pubs.vmware.com/vfabric53/index.jsp?topic=/com.vmware.vfabric.rabbitmq.3.2/rabbit-web-docs/ha.html

# 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


- Hussam Abu-Libdeh, Robbert van Renesse, and Ymir Vigfusson. 2013. Leveraging sharding in the design of scalable replication protocols. In Proceedings of the 4th annual Symposium on Cloud Computing (SOCC '13). ACM, New York, NY, USA, , Article 12 , 16 pages. DOI: http://dx.doi.org/10.1145/2523616.2523623
- http://rboutaba.cs.uwaterloo.ca/Papers/Conferences/2012/ZhangICDCS12.pdf
- Flavio Junqueira & Benjamin Reed, ZooKeeper, Distributed Process Coordination, O'reilly, 2013
- Werner Vogels. 2009. Eventually consistent. Commun. ACM 52, 1 (January 2009), 40-44. DOI=http://dx.doi.org/10.1145/1435417.1435432
- Ariel Tseitlin. 2013. The antifragile organization. Commun. ACM 56, 8 (August 2013), 40-44. DOI=http://dx.doi.org/10.1145/2492007.2492022

# Thanks for your attention

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