ZOOKEEPER

Coordinating the cluster

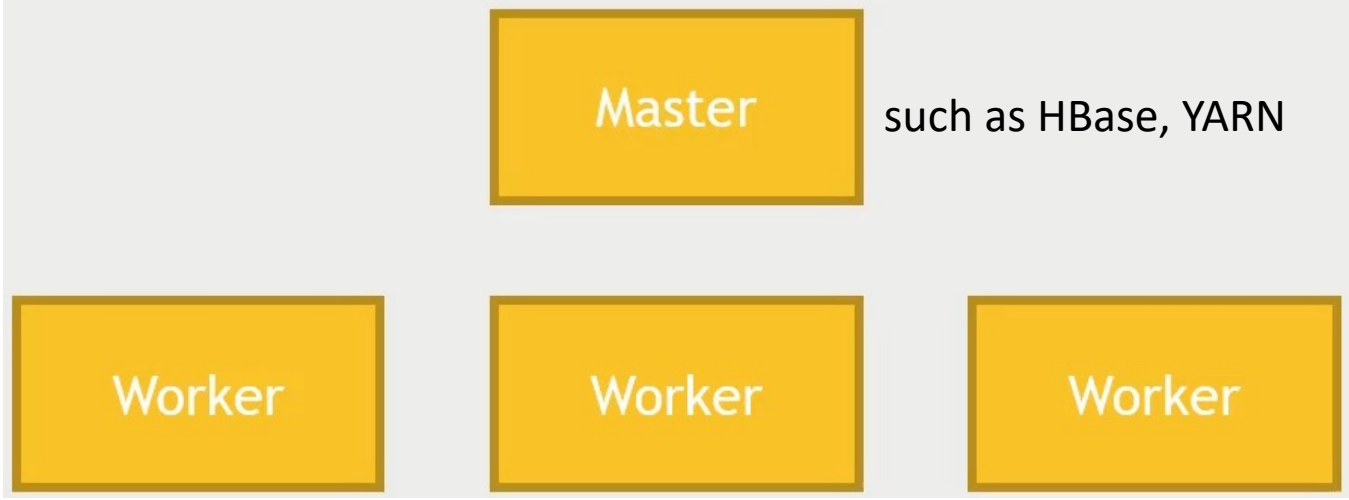
Bernard Lee Kok Bang

What is Zookeeper?

- Keeps track of information that must be synchronized across the cluster [a system that sits off site to maintain consistent state of our Hadoop cluster]
  - *which node is the master?*
  - *what tasks are assigned to which workers?*
  - *which workers are currently available?*
- a tool that applications [such as HBase] can use to recover from partial failures in the cluster
- An integral part of HBase, MapReduce, Drill, Storm…

Failure modes

- Master crashes, need to fail over to a backup master node [but only allowed one node to become master node]
- Worker crashes - its work needs to be re-distributed
- Network trouble - part of the cluster can’t see the rest of it



Ephemeral data

- ***data that only exists for a short period of time*** and is often referred to as “self-destructing” or “disappearing” messages.
- ***only one version is available at a time***
- *vs persistent data structure*

“Primitive” operations in a distributed system

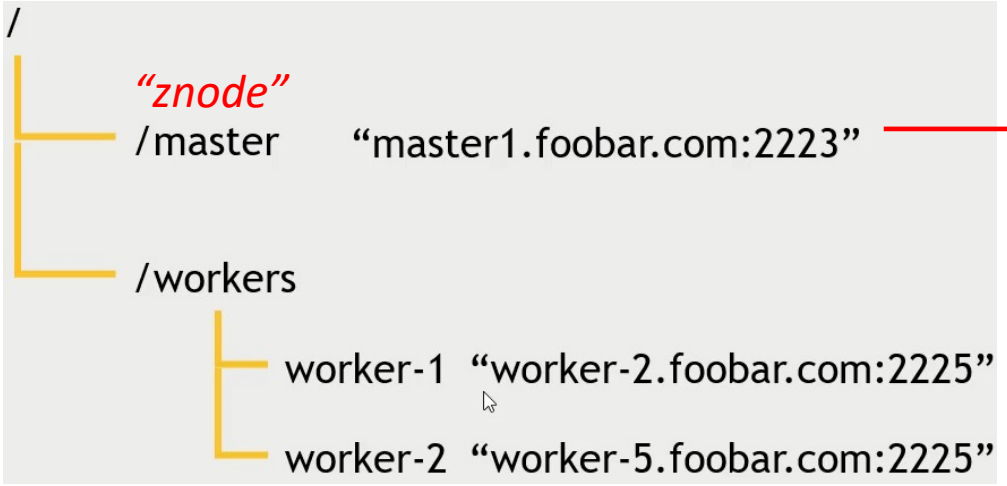
- Master election
  - *One node registers itself as a master, holds a “lock” on that data*
  - *Other nodes cannot become master until that lock is released*
  - *Only one node allowed to hold the lock at a time*
- Crash detection
  - *“Ephemeral” data on a node’s availability automatically goes away if the node disconnects, or fails to refresh itself after some time-out period*
- Group management → **keeping track of what workers are available**
- Metadata
  - *List of outstanding tasks, task assignments [new nodes know where to pick up from]*

But…

- ZooKeeper’s API is not about these “primitive” measure
- Instead, they built a more general-purpose system that makes easy for applications to implement them

ZooKeeper's API

- A little *distributed file system* [any application can write and read from]
  - *With strong consistency guarantees*
  - *Replace the concept of "file" with "znode"*
- Here's the ZooKeeper API:
  - *Create, delete, exists, setData, getData, getChildren*



only one client at a time has ownership as the master znode

Notifications

- A client can register for notification on a znode
  - *Avoid continuous polling*
  - *Example: register for notification on "/" master - if it goes down, try to take over as the new master* ["ephemeral" nature of the znode]

Persistent and ephemeral znodes

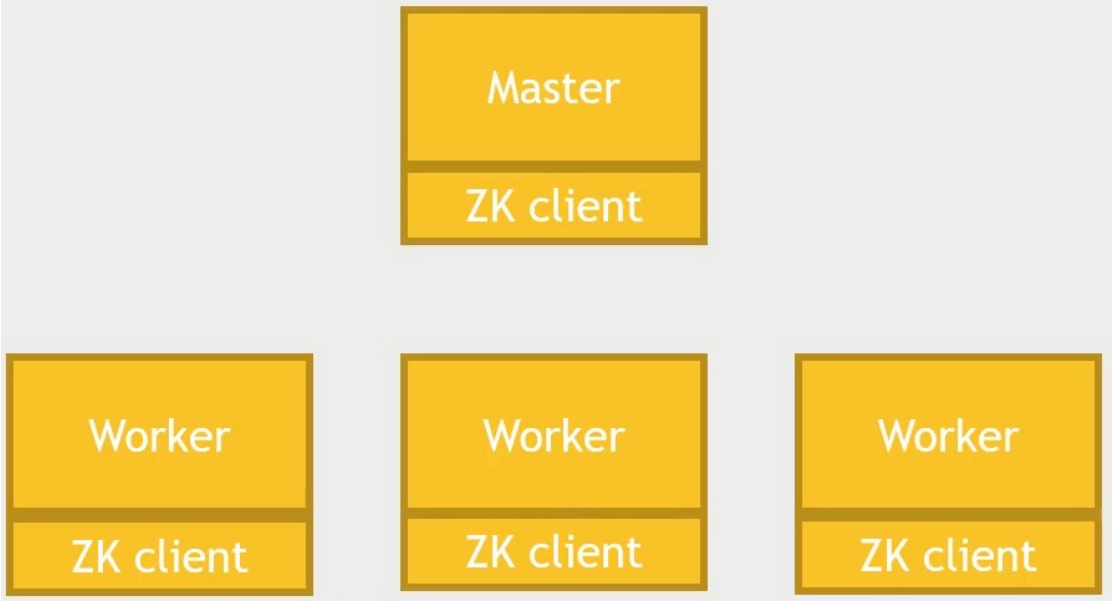
- Persistent znodes remain stored until explicitly deleted

  i.e., assignment of tasks to workers must persist even if master crashes [we still want the information of the outstanding works that need to be done to live on]

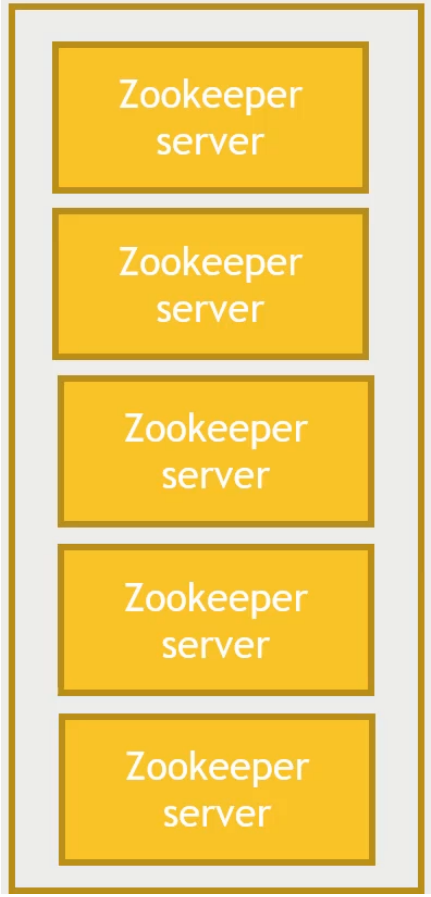
- Ephemeral znodes go away if the client that created it crashes or loses its connection to ZooKeeper

  i.e., if the master crashes, it should release its lock on the znode that indicates which node is the master!

ZooKeeper Architecture



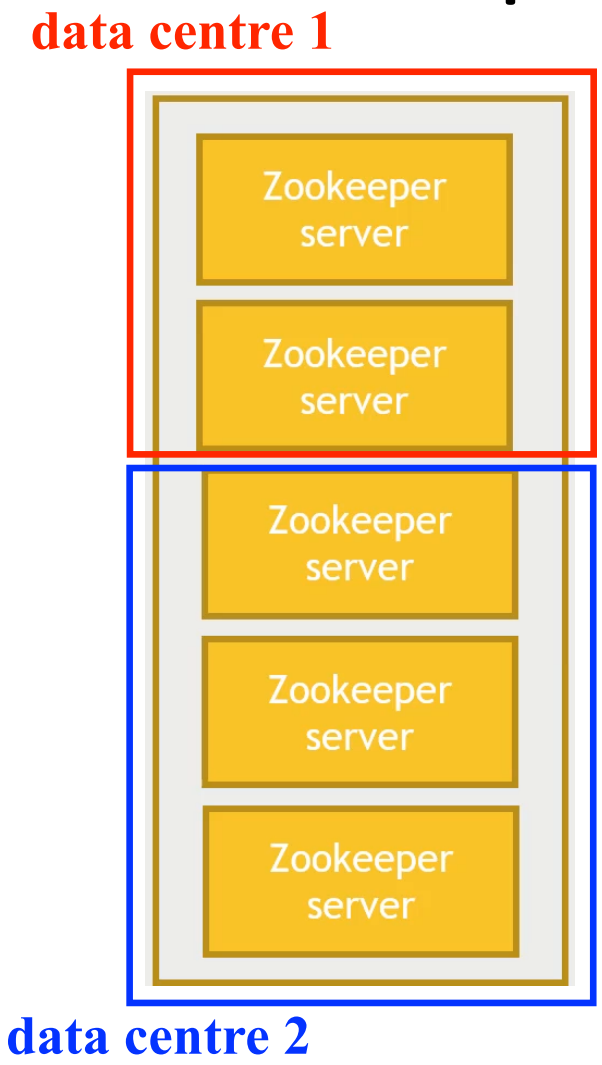
Clients have a list of ZooKeeper servers to connect to



ZooKeeper ensemble

- more than one ZooKeeper servers to avoid single point of failure!!!
- replicating data amongst itself

ZooKeeper quorums



ZooKeeper Ensemble

- we can specify minimum number of servers that need to agree on something before we accept the answer from a client
- Need at least five servers with *a quorum of three* to get reliable ZooKeeper performance
- ideally should spread across different data centers
- *Quorum of two vs quorum of three*

Quorum of one

We need five ZooKeeper servers & three quorums

ZooKeeper Server 1

Not an ideal situation – would have single point of failure

ZooKeeper Server 2

Still not an ideal situation – if there is a crash happens, then which server has a better say?
Decision is 50% vs 50% - in situation where master node is down

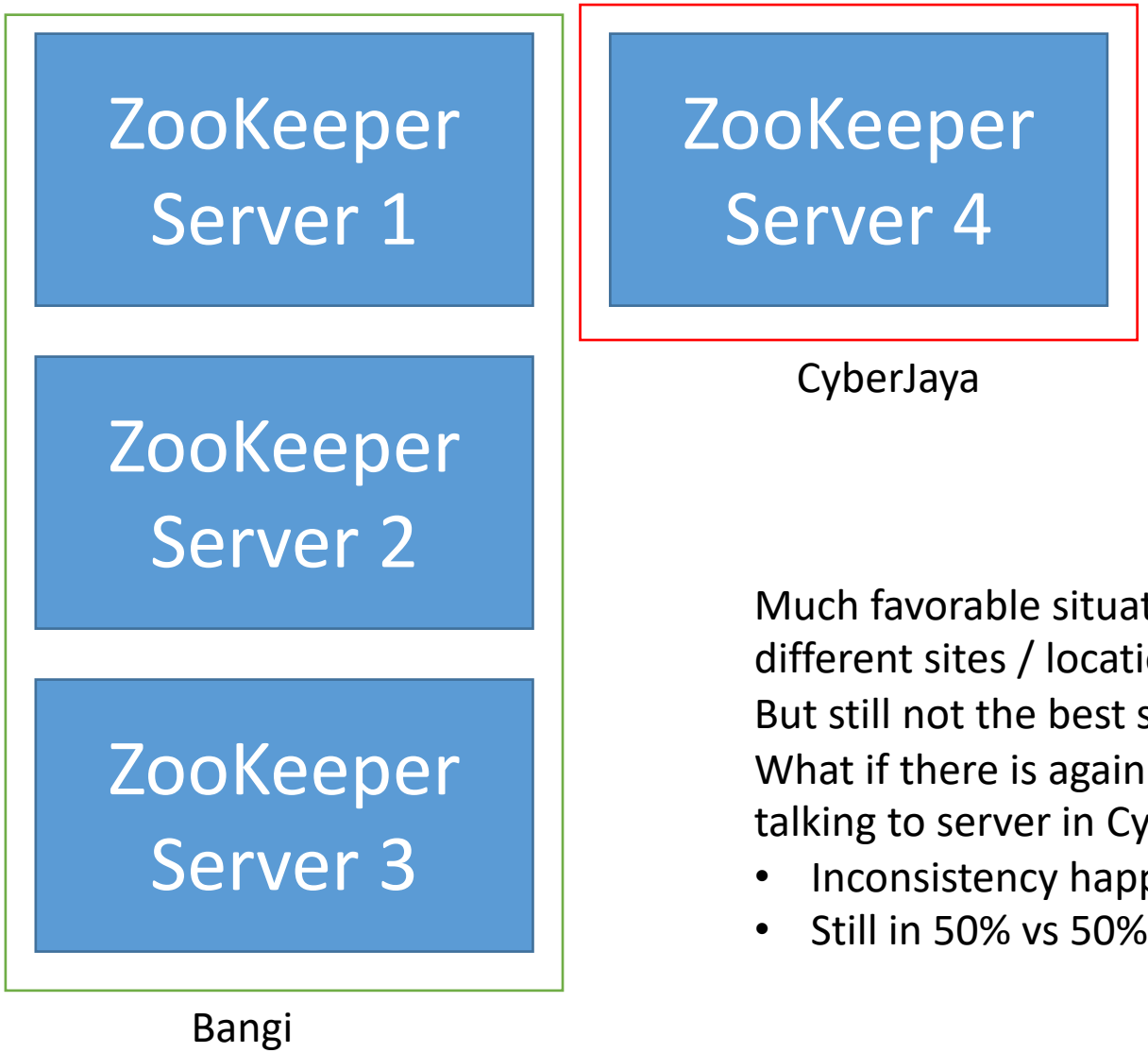
ZooKeeper Server 3

Could be a favorable situation, if there is a crash happened, so long as two of the servers agree on each other, then we can set up the new node as the new master node: 2/3 servers
But still not the best situation… why?
What happened if there is a complete network failure in Bangi?

Bangi

Quorum of two

We need five ZooKeeper servers & three quorums



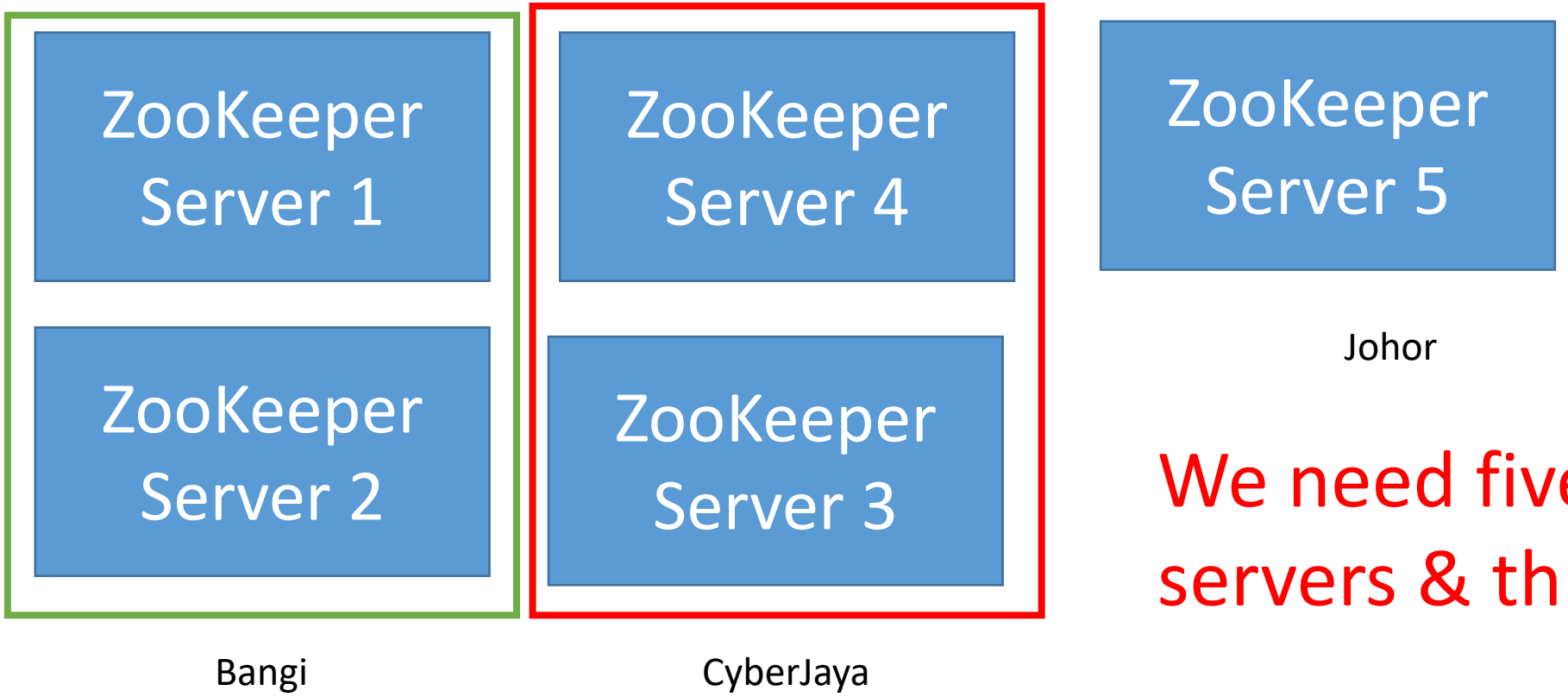
Much favorable situation, whereby we have ZooKeeper servers at two different sites / locations

But still not the best situation…

What if there is again a network problem in Bangi? Servers in Bangi are not talking to server in Cyberjaya

- Inconsistency happens; CyberJaya is online, but Bangi is offline;
- Still in 50% vs 50% situation

Quorum of three



We need five ZooKeeper servers & three quorums

- Five servers and three quorums would be the best architecture design

Let’s play with the ZooKeeper

- Login to HDP ssh session as root
  - *su root*
- cd to ZooKeeper sub-directory folder
  - *cd /usr/hdp/current/zookeeper-client/*
- cd into bin folder to check for zkCli.sh script
  - *cd bin*
- execute ZooKeeper command line interface
  - *./zkCli.sh*

Basic ZooKeeper Operation

- Check what tools are associated with ZooKeeper on the root level
  - *ls /* → **zookeeper looks like a file system**
- Create an “ephemeral” znode master
  - *create -e /testmaster “127.0.0.1:2223”*
  - *ls /* #testmaster master znode would be created
  - *get /testmaster* #check who is the znode master
  - *quit* #as if the znode master [ephemeral state] has just died
  - *./zkCli.sh* → **log back in to zookeeper**
  - *ls /* #testmaster master znode does not exists