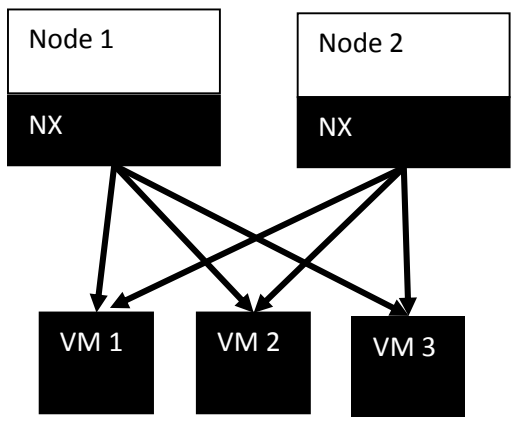
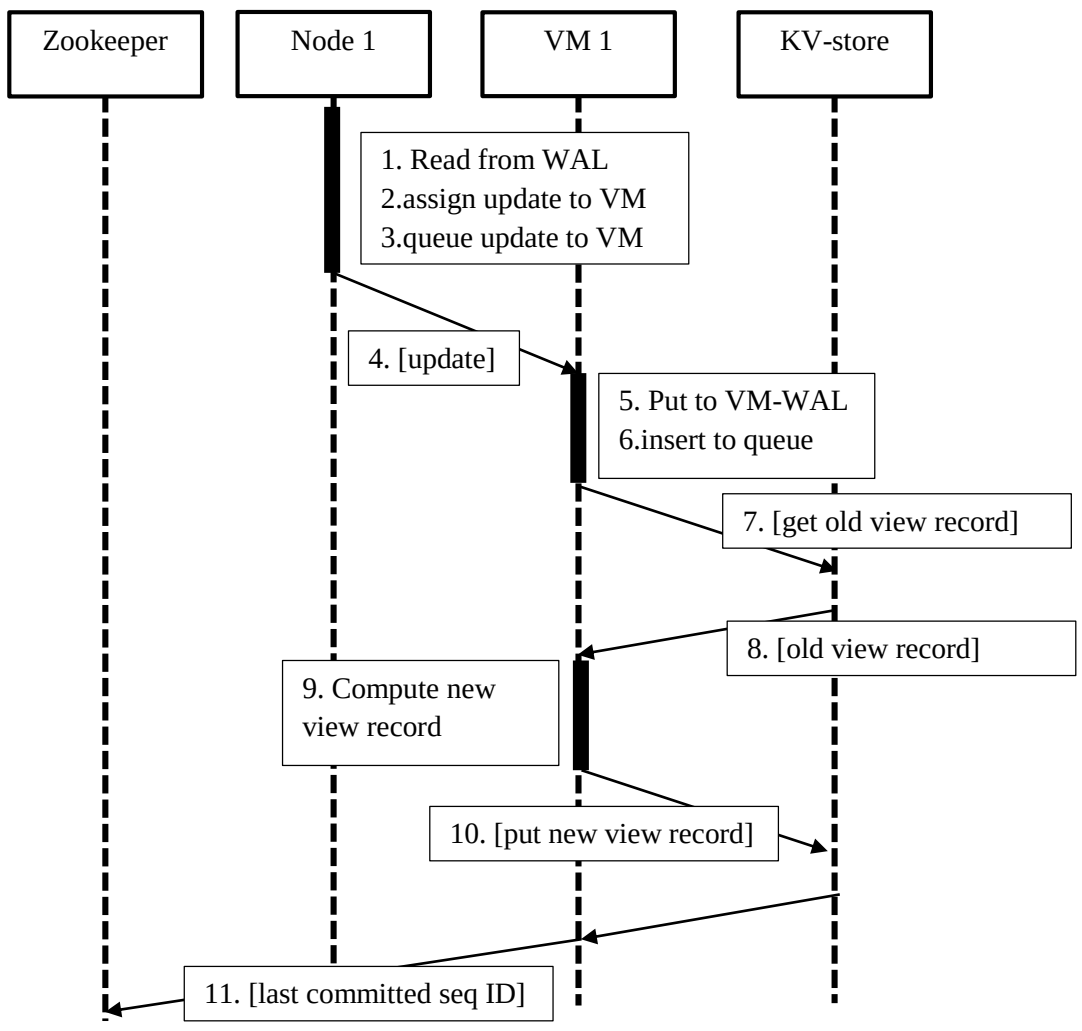
Global Hashring

Process update



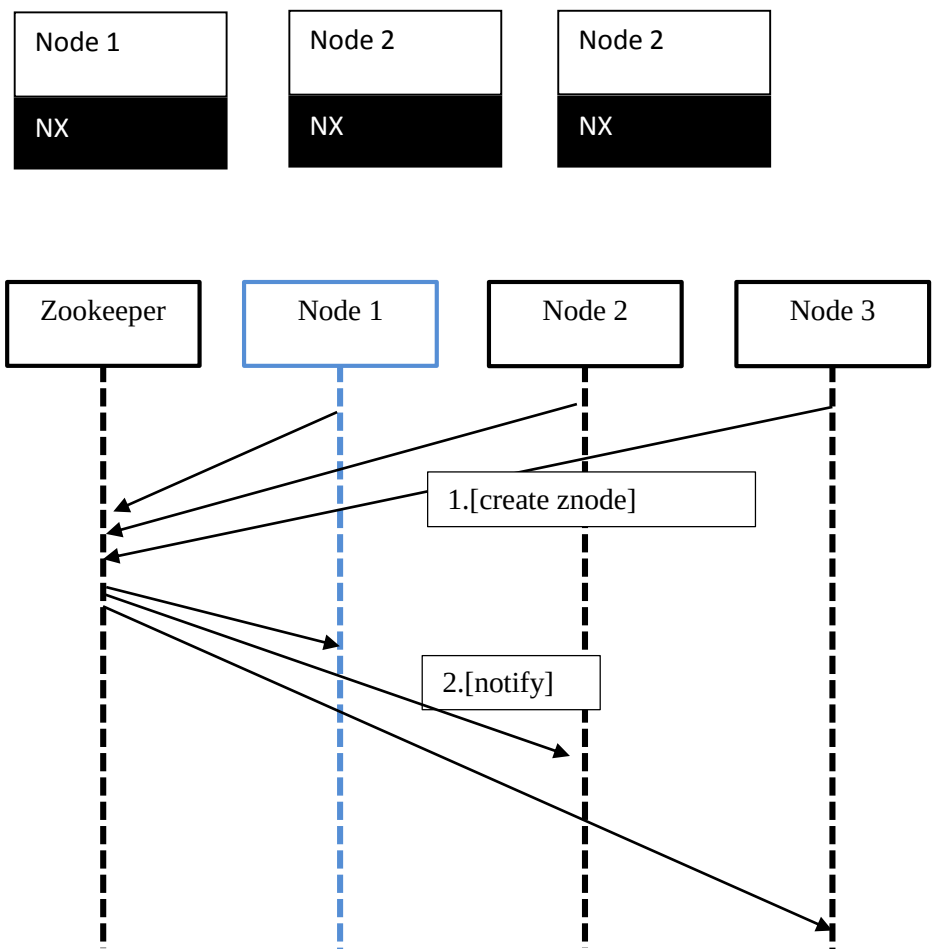
Description:
Updates are sent through a different communication channel than messages. The update stream is continuously flowing from the database nodes to the view managers. One update passes the following stages.



Algorithm:
All Nodes:[create znode] → Zookeeper
Zookeeper:[election finished] → all nodes

Global Hashring

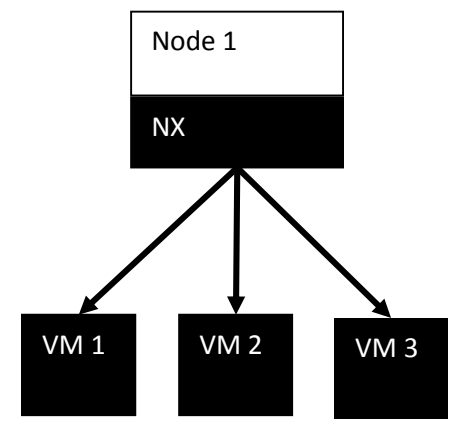
Elect Coordinator



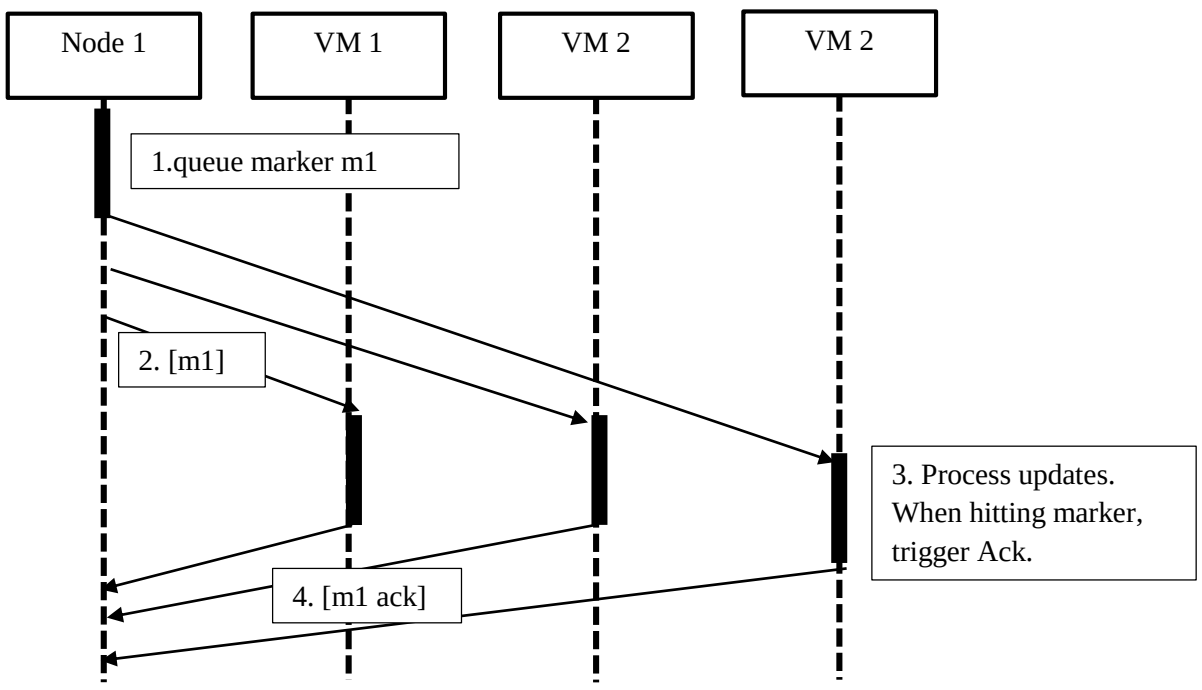
Description:
The global hash ring architecture style of the VMS does not rely on a centralized coordinator. Nevertheless, the VMS needs to perform actions to load balance or recover the system. Without a coordinator these tasks can be assigned to one of the nodes. Using a leader election algorithm the node can be determined.

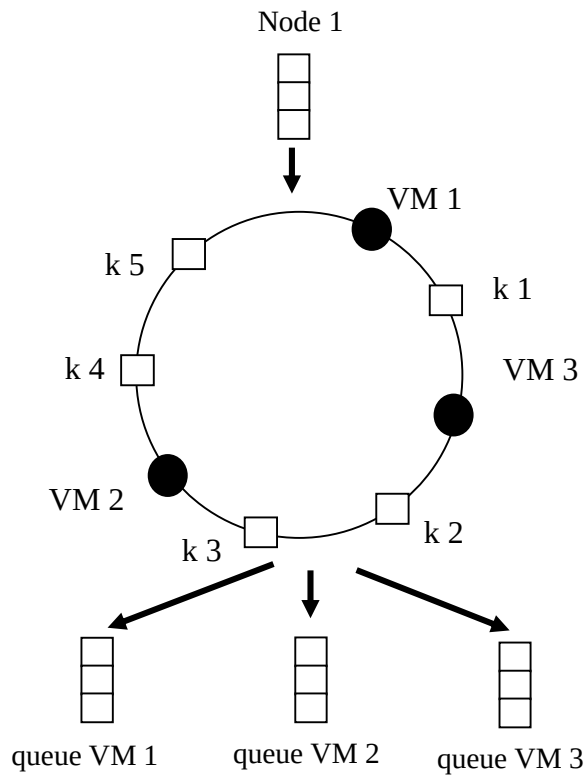
Algorithm:
1. All Nodes:[create znode] → Zookeeper
2. Zookeeper:[election finished] → all nodes

Send marker



Description:
To perform actions like the assign and withdraw view managers in a dynamic environment, the Node requires information about whether a view manager has already processed a set of updates or not. For that reason markers are placed in the view manager's queue like a normal update. From here they are sent to the view manager. At some point the VM hits the marker in the stream of updates. Then it

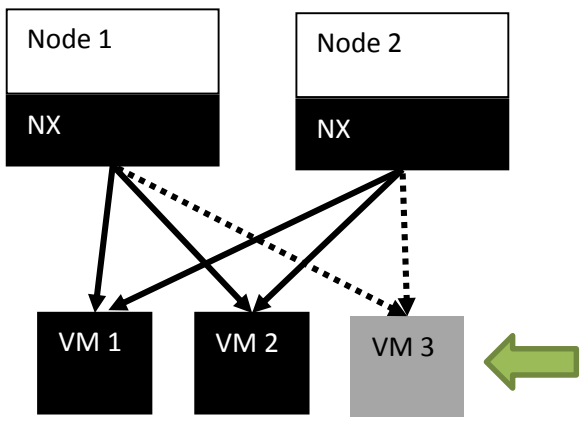




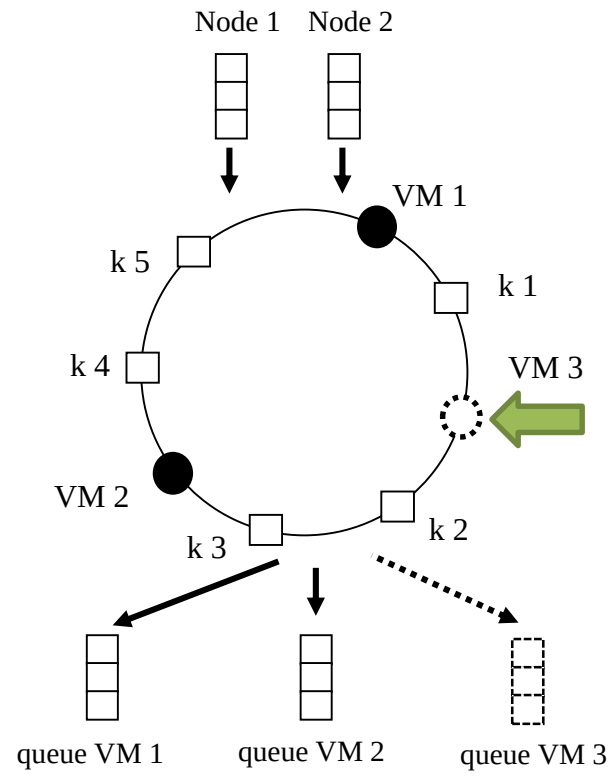
Algorithm:
All Nodes:[create znode] → Zookeeper
Zookeeper:[election finished] → all nodes

Global Hashring

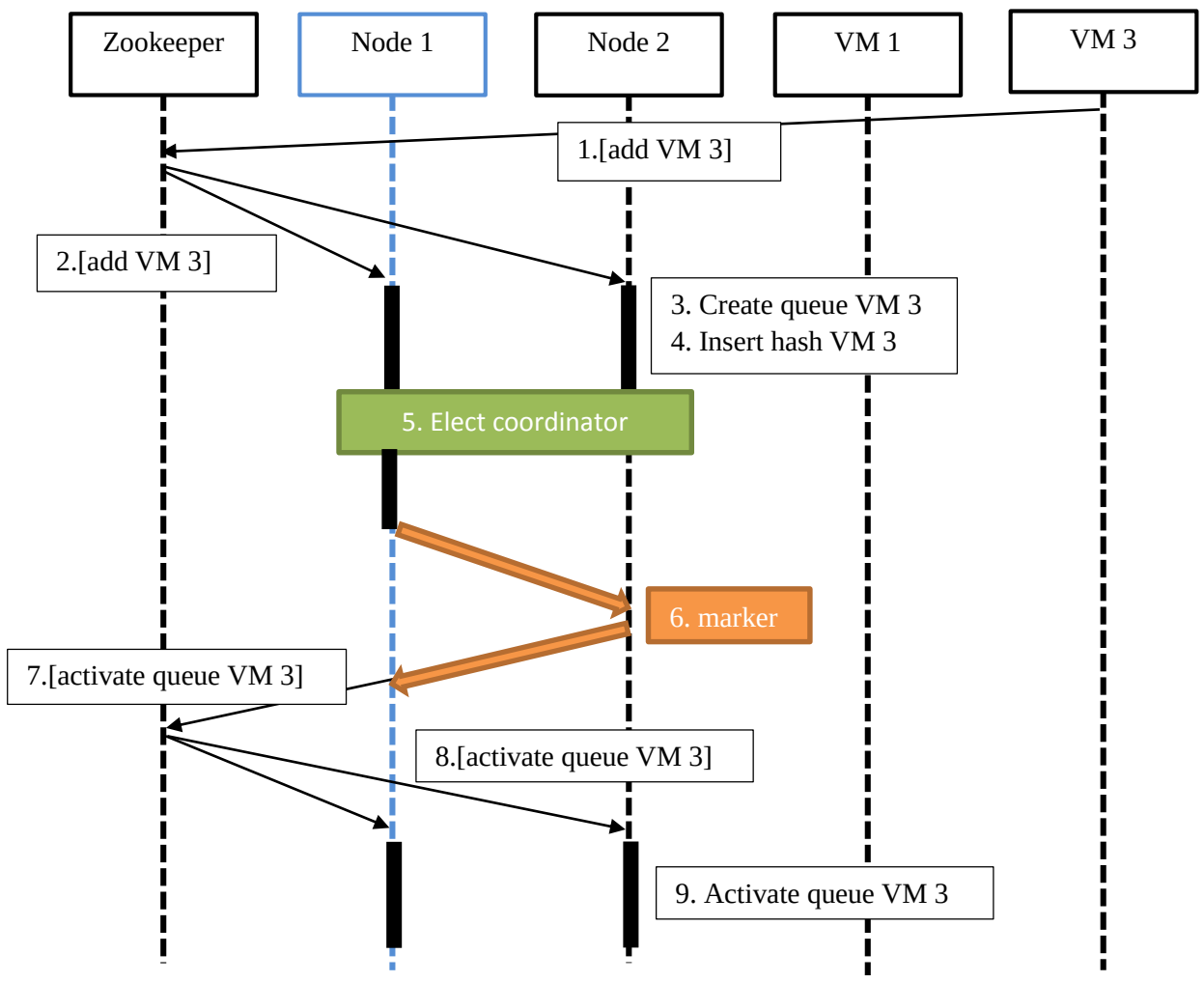
Add View Manager



A View Manager VM 3 is added to VMS. Because the implementation uses a global hash ring, the VM needs to be added to the hash rings of all nodes. To preserve the time line, the queues of all VMs that lose a key-range are deactivated. **Alternative 1**: The nodes synchronize on Zookeeper, only one node receives task of activation. The Node sends a marker to the new VM. After receiving the Ack, the node reactivates the stopped VMs and activates the new VM.
Alternative 2: All nodes send a marker to the new VM. On acknowledgement, the stopped VMs are reactivated and the new VM is activated.

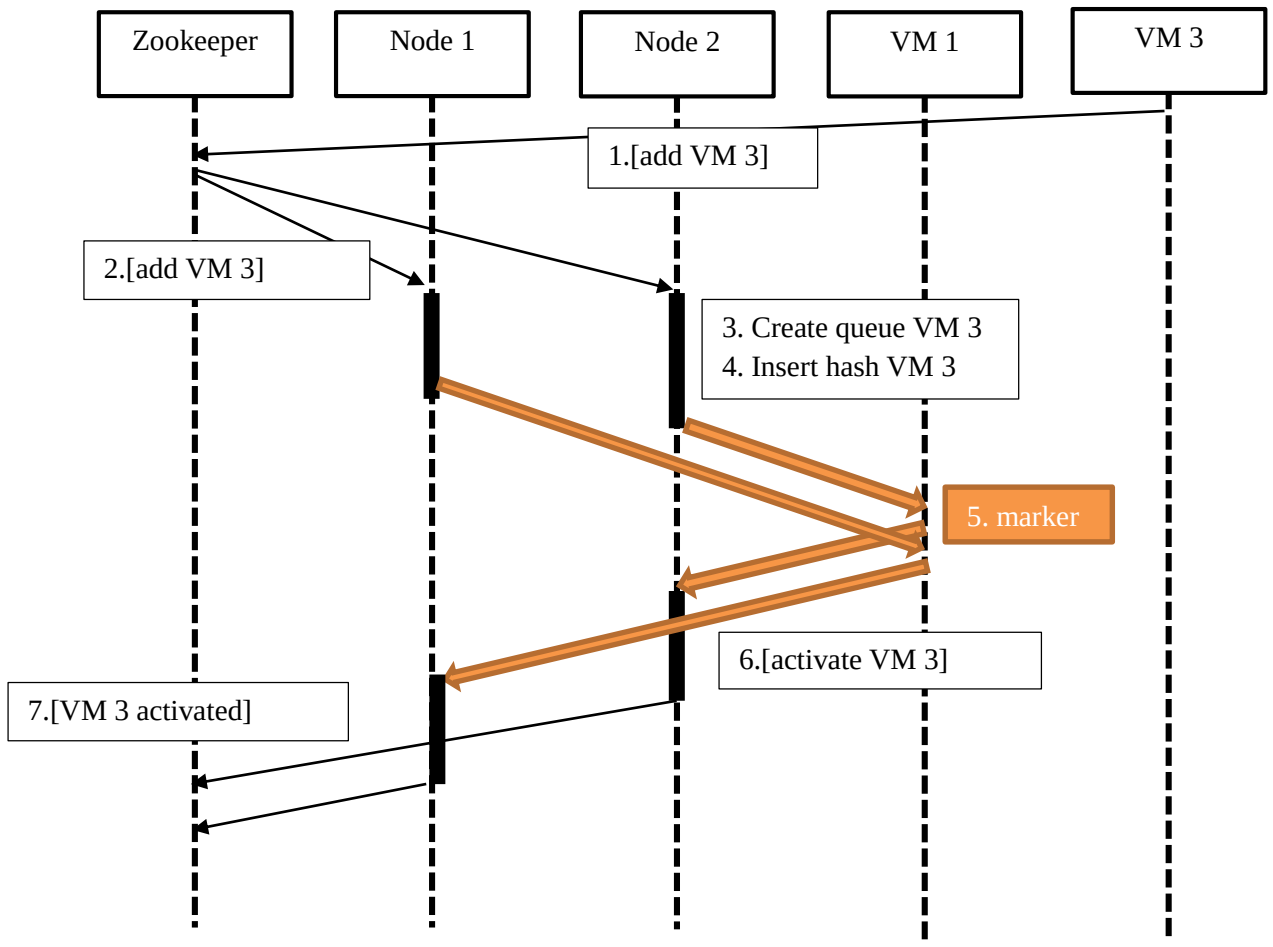


Alternative 1 (single node activate):



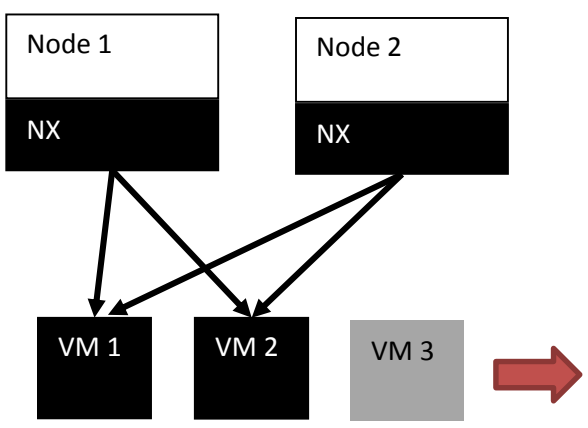
1. New VM: Send [add new VM] → Zookeeper
2. Zookeeper: [new VM added] → all nodes
3. All nodes: Create queue
4. All n odes: Insert hash of new VM
5. Coordinator: Send [marker] to all VMs that lose key range
6. VM: Send [ack] to coordinator
7. Coordinator: Send [activate new VM] to all nodes

Alternative 2 (all nodes activate):

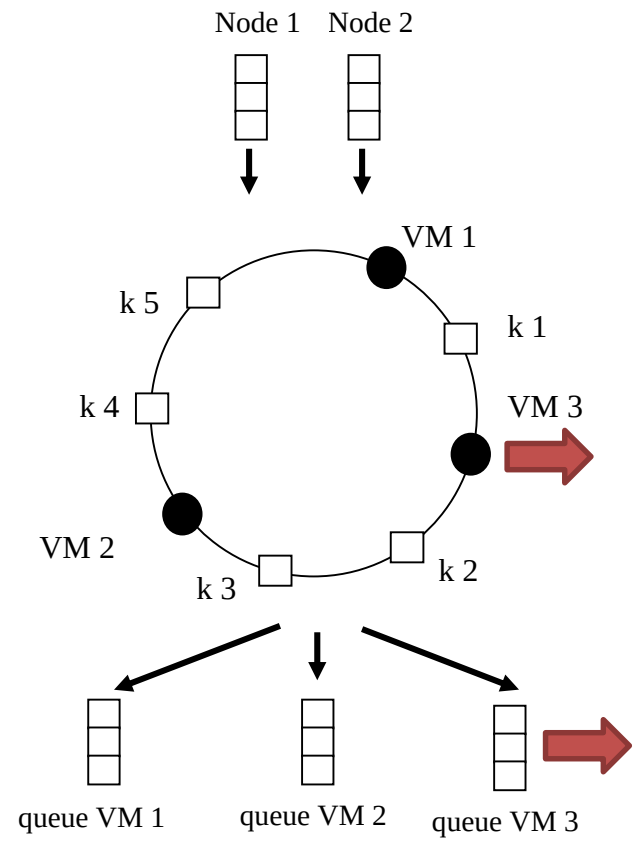


Global Hashring

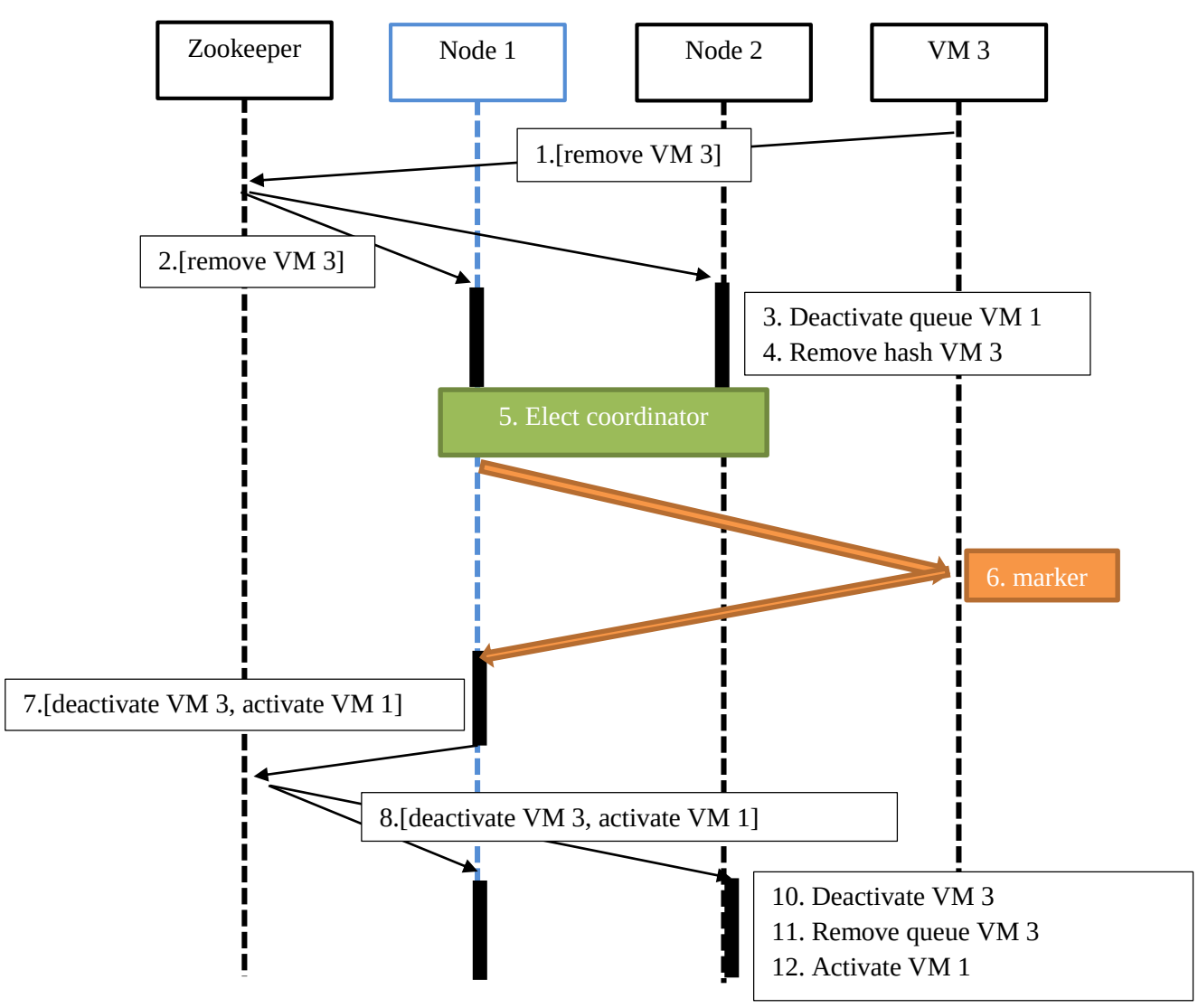
Remove View Manager



- View Manager is added to VMS
- Needs to be removed globally (from hash rings of all the nodes)
- Timeline consistency has to be preserved
- Marker is sent to the view manager that is removed

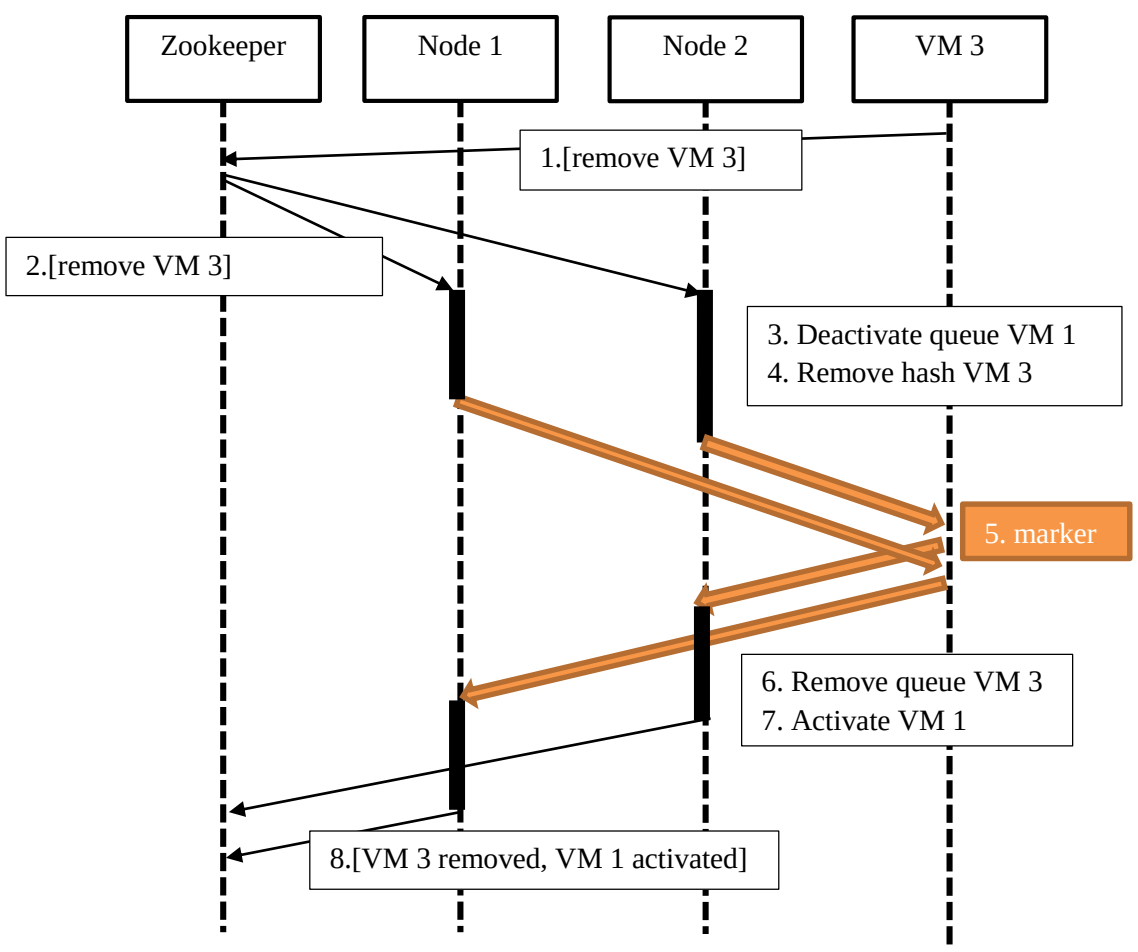


Alternative 1 (with coordinator):



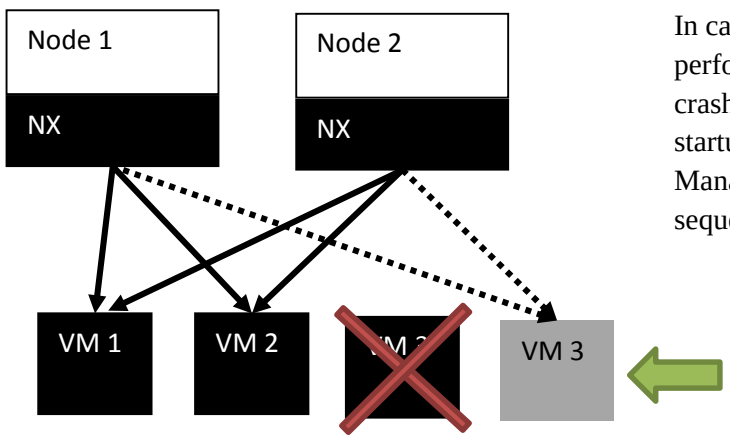
1. Coordinator: Send deactivate VM 1 + remove VM 3 from hashring to all nodes
2. All nodes: Deactivate queue VM 1
3. All nodes: Remove VM 3 from hash ring
4. All nodes: send [ack]
5. Coordinator: Queue marker to VM 3
6. Coordinator: Wait for VM 3 to ack
7. Coordinator: Send activate VM 1 to all nodes

Alternative 2 (without coordinator):

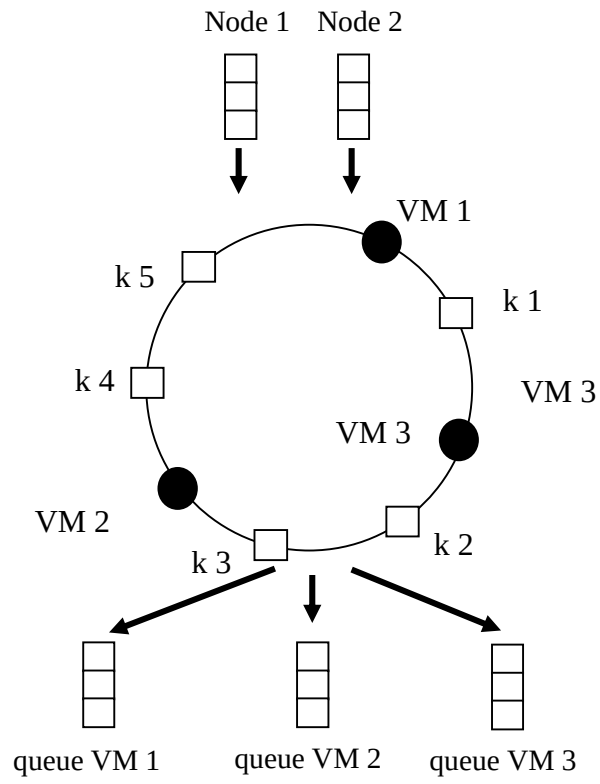


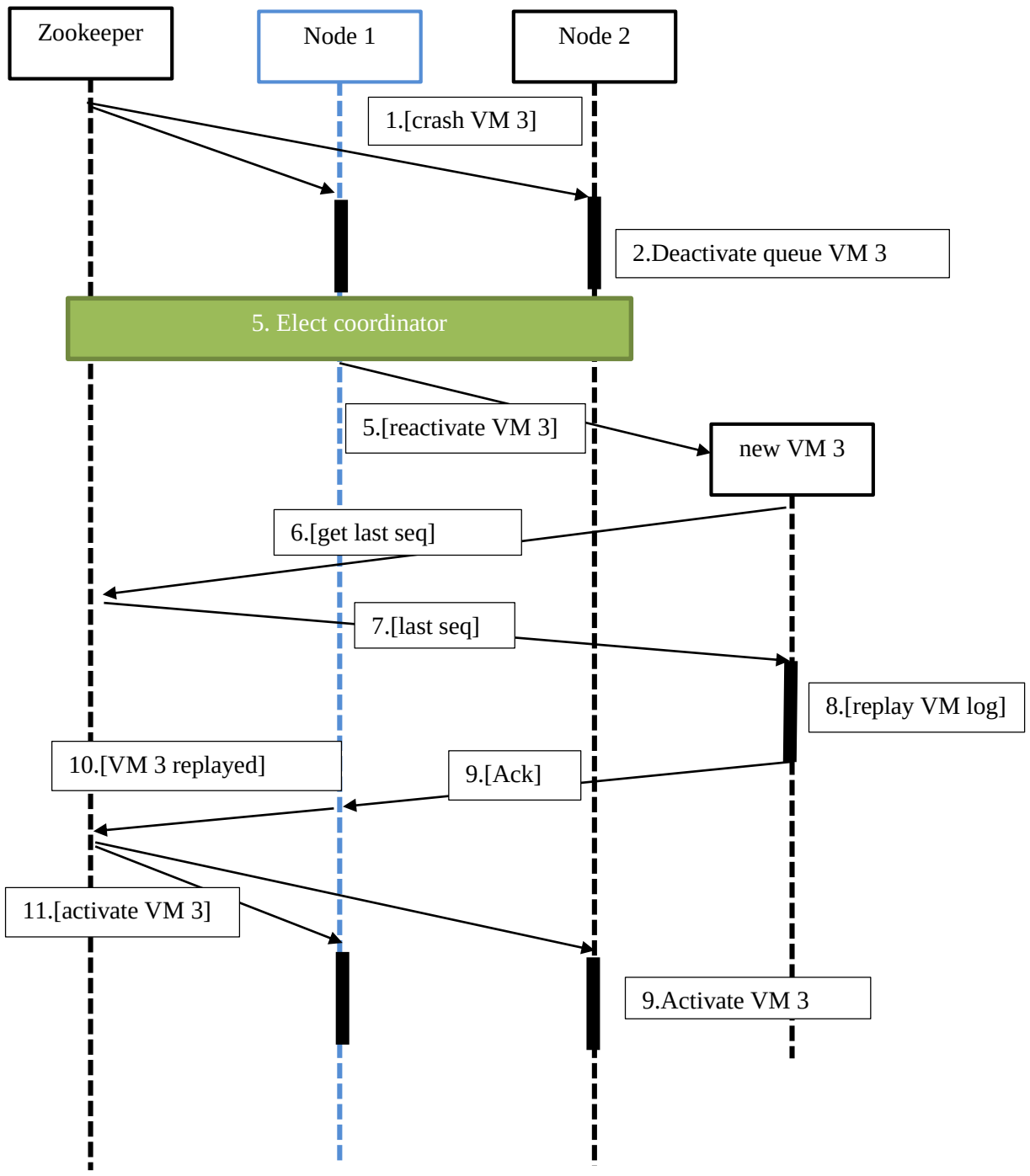
Global Hashring

Crash View Manager



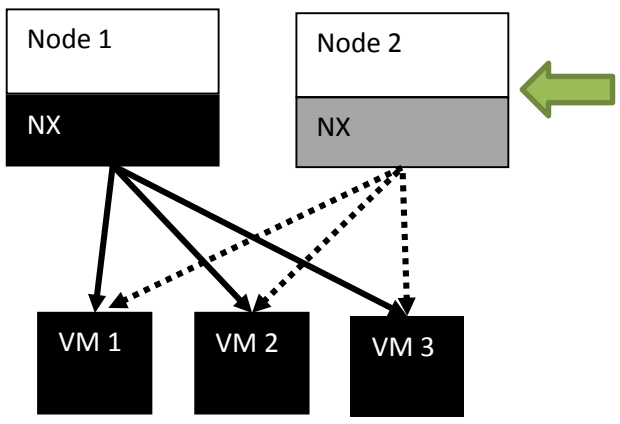
Description:
In case a view manager crashes a node is elected that performs the recovery steps. The Node restarts the crashed view manager (on a different machine). On startup, the view manager. In the first case the View Manager needs only to keep track of its *last committed* sequence ID. The WAL is replayed from this point



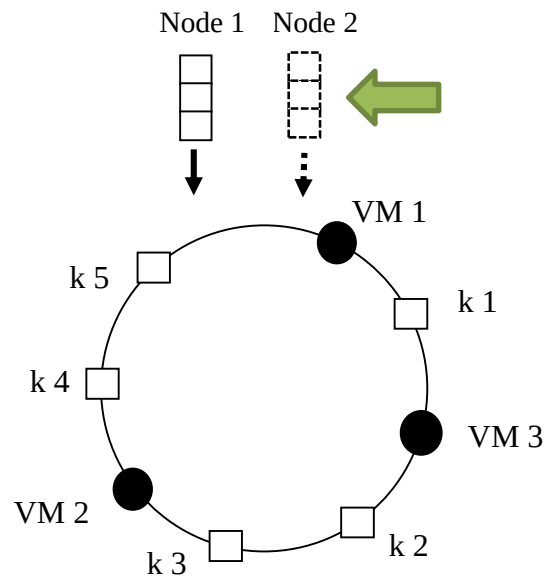


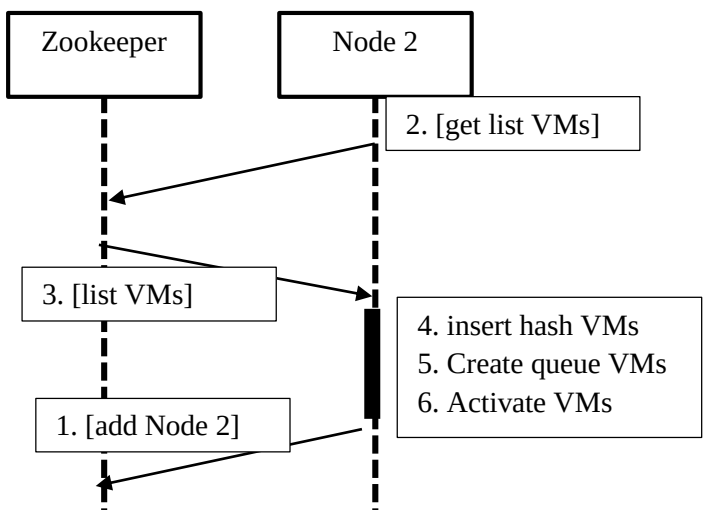
1. Coordinator: Send [deactivate queue of crashed VM] to all nodes
2. All nodes: Deactivate queue of crashed VM
3. All nodes: Send [ack]
4. Coordinator: Start new Vm
5. Coordinator: Send [replay log of crashed VM] to new VM
6. New VM: replay log of crashed VM
7. New VM: send [ack]
8. Coordinator: Send [substitute ip address] to new VM

Add Node

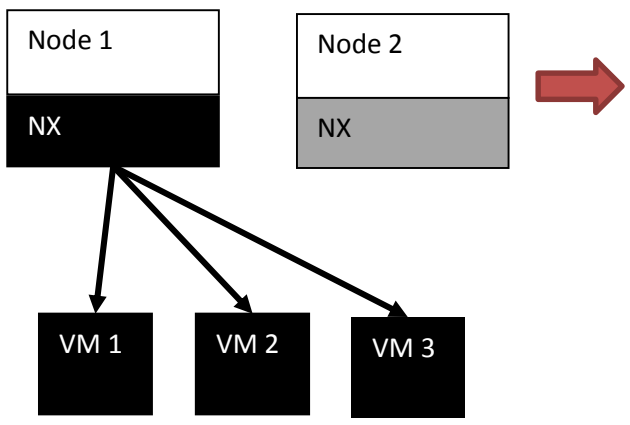


1. Node 2: Add all existing VMs to hashring
2. Node 2: Activate all queues

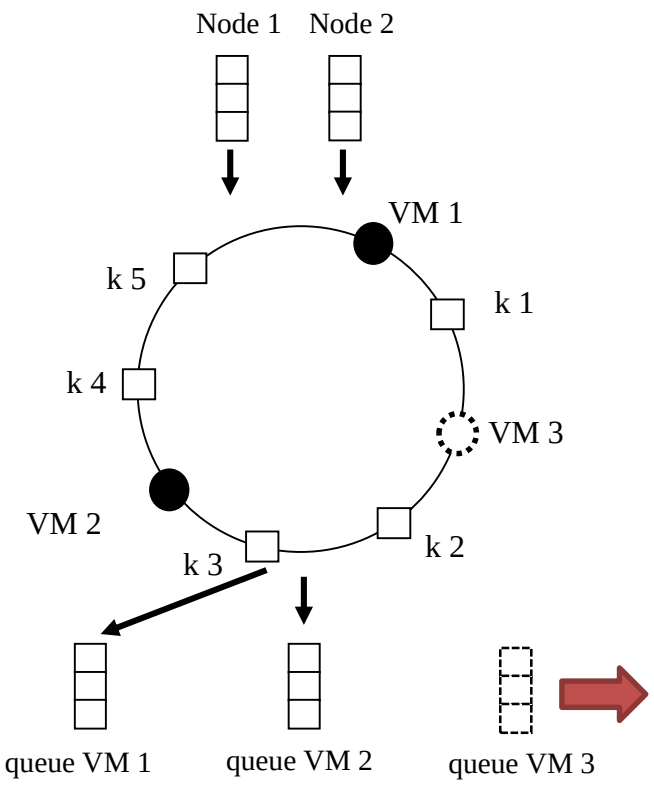


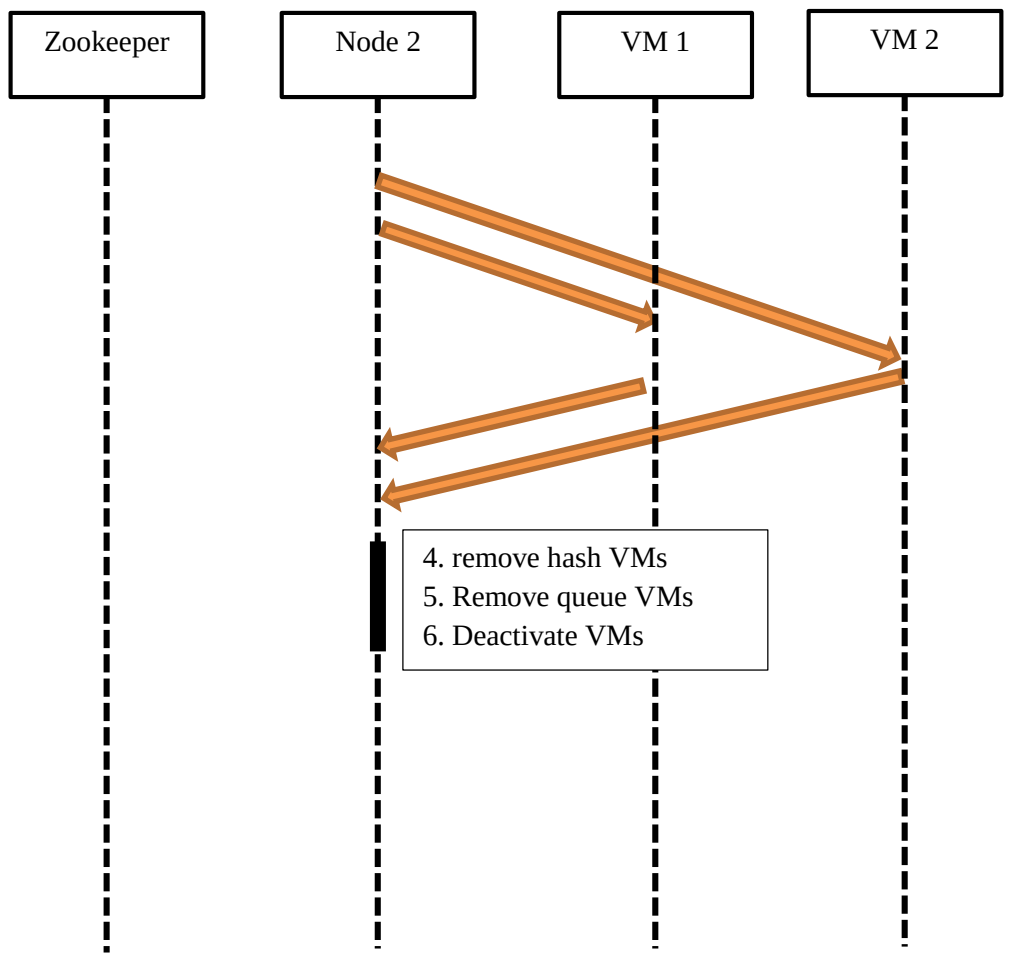


Remove Node



1. Node: Send marker to all VMs
2. Node: Deactivate/remove all queues
3. Node: Remove all VMs from hashring

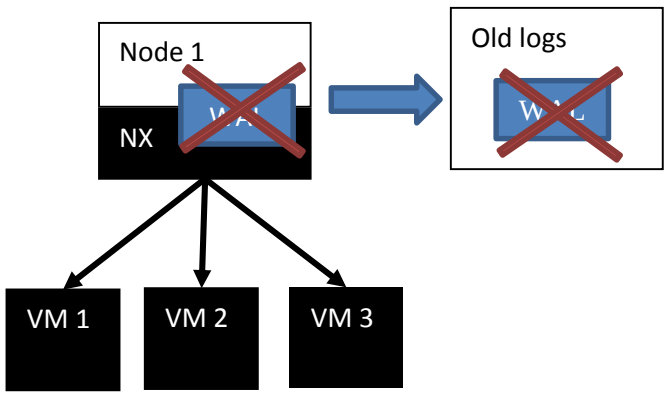




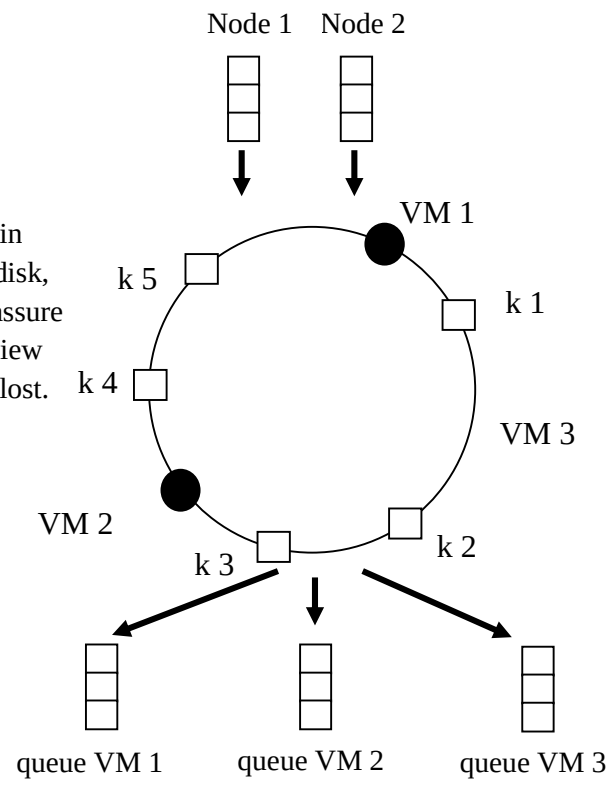
2.[Node 2 added]

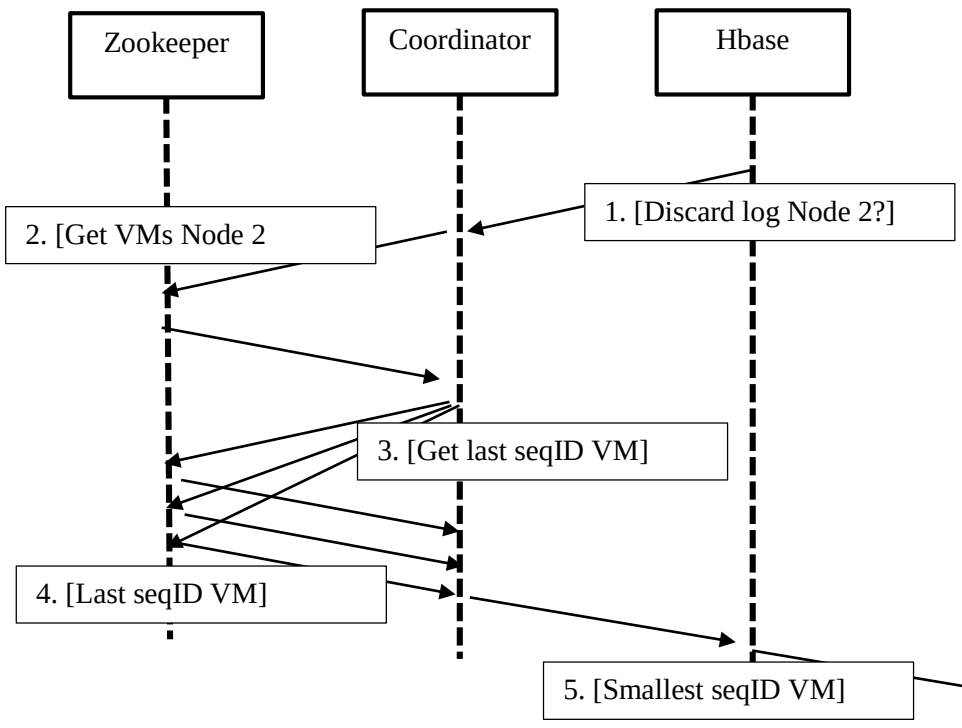
5. marker

Discard logs

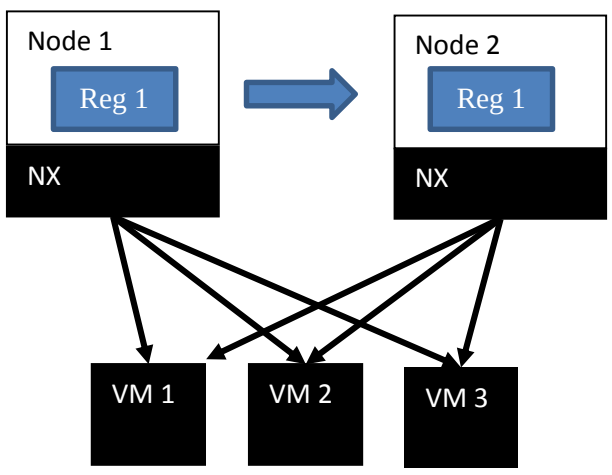


Description:
Physically, HBase writes to multiple WAL files in HDFS. As all entries of a log file are flushed to disk, the KV-store discards the log. The VMS has to assure that all entries are propagated and applied by a view manager. Otherwise, some of the entries will be lost.

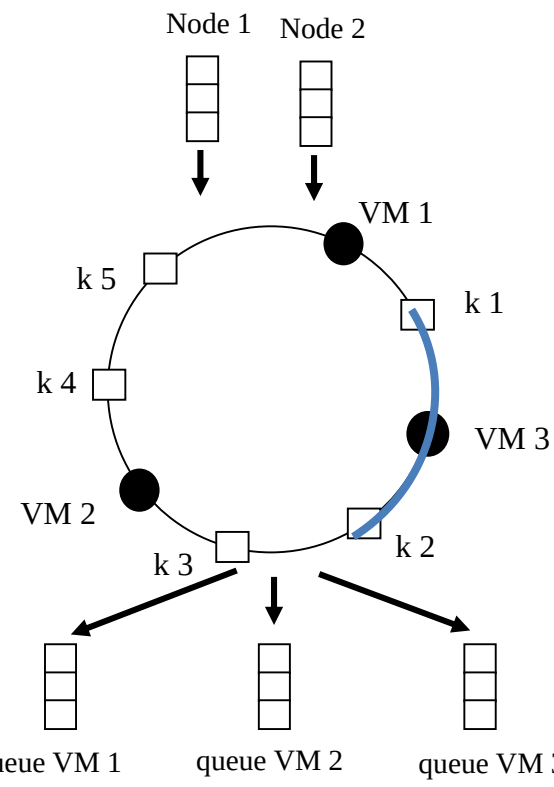




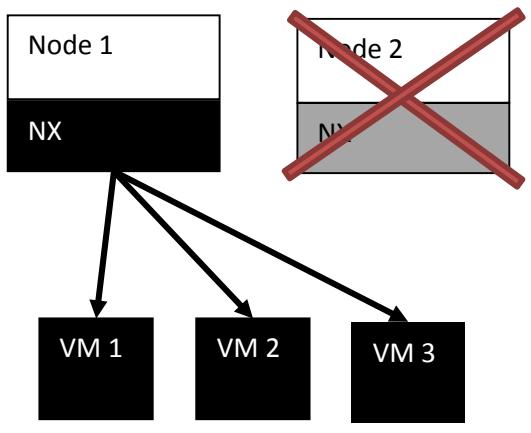
Move Region



Description:
If a node crashes in the KV-Store architecture, the abandoned WAL is recovered and replayed. During the replay the log entries are grouped by key-region. The key-regions are assigned to the remaining nodes. Using a global hash ring the VMS does not need to react to a region movement. Since all Nodes feed the same key-range and thus, the same view managers, time line of records cannot be changed.



Crash Node (move region + discard logs)



Description:
If a node crashes in the KV-Store architecture, the abandoned WAL is recovered and replayed. During the replay the log entries are grouped by key-region. The key-regions are assigned to the remaining nodes. Finally, the entries are inserted into the memstore and flushed to disk directly. The entries are not written to the WAL again. Therefore, we need to ensure that no updates get lost. The VMS needs to read all entries of the WAL before it is discarded