Apache Ignite Best Practices: Cluster Topology Management and Data Replication

Ivan Rakov
Apache Ignite Committer
Senior Software Engineer in GridGain Systems

Too many data? Distributed Systems to the rescue!

- Horizontal scale at a lower cost

- New challenges
  - How to partition the data?
  - How to deploy cluster properly?
  - How to grow/shrink cluster topology safely?

How to achieve this and don’t lose your data?
You’ll know some best practices today.

Agenda

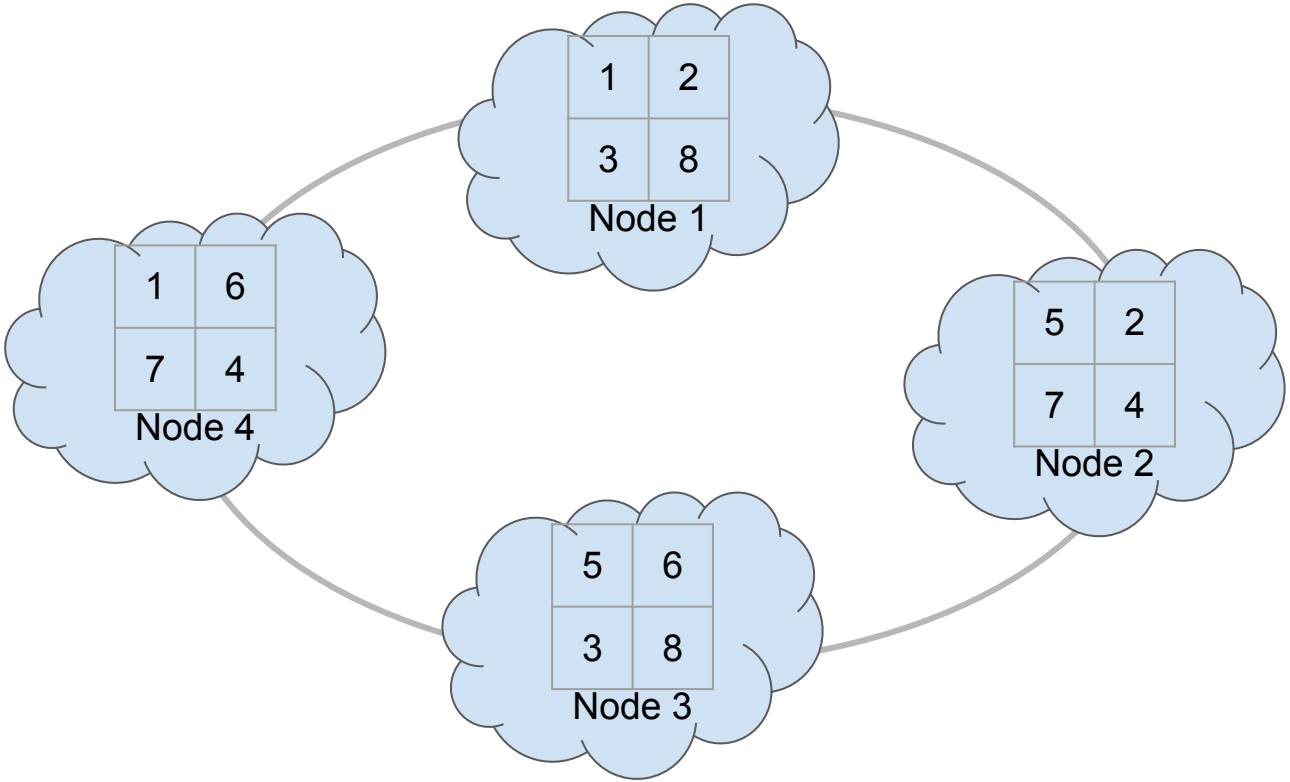
- In-memory cluster management
- Persistent cluster management
  - Baseline Topology: why?
  - Baseline Topology: and finally, what is it?
  - Baseline Topology: API
  - Use case: Managing persistent cluster with BLT
  - Split-brain scenarios: how BLT and Zookeeper may help

Agenda

- In-memory cluster management
- Persistent cluster management
  - Baseline Topology: why?
  - Baseline Topology: and finally, what is it?
  - Baseline Topology: API
  - Use case: Managing persistent cluster with BLT
  - Split-brain scenarios: how BLT and Zookeeper may help

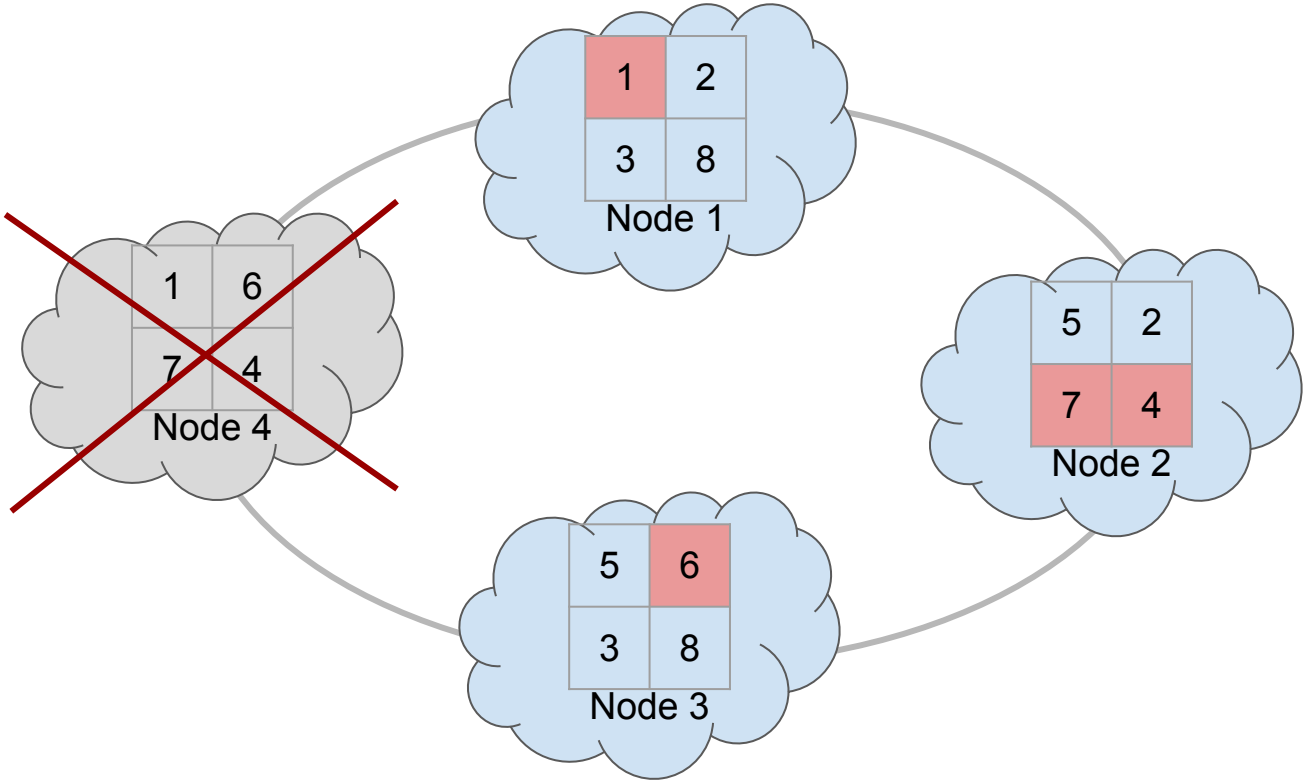
In-memory approach: replication as data loss protection

backup factor = 1 `// cacheCfg.setBackups(1);`



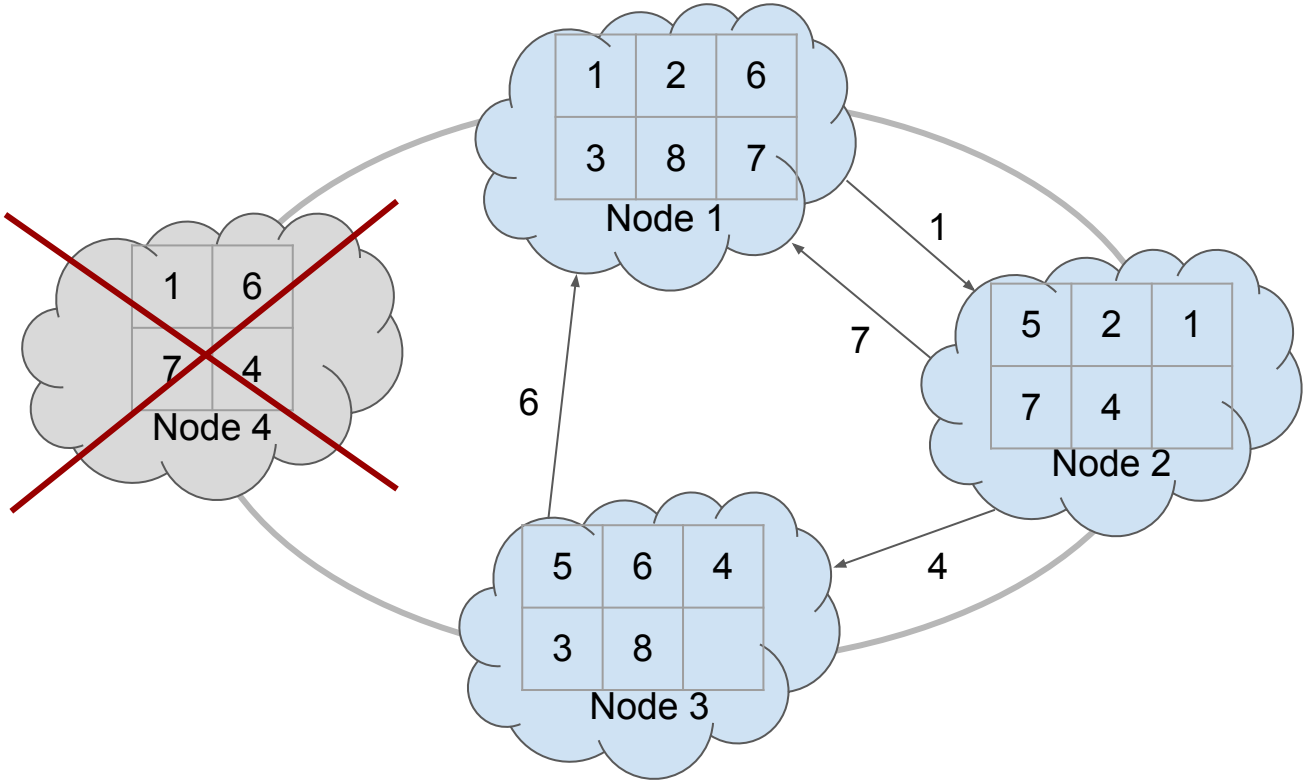
In-memory approach: replication as data loss protection

backup factor < 1, next node crash will cause data loss



In-memory approach: replication as data loss protection

topology change leads to affinity reassignment and rebalancing, backup factor = 1



Use case: in-memory mode

1. How to deploy cluster? Just start nodes.
2. How to scale and expand topology? Just start nodes.
3. How to shrink topology? Just remove nodes, but not all at once.

Your data will be automatically rebalanced over the new set of nodes.

Use case: in-memory mode

1. How to deploy cluster? Just start nodes.
2. How to scale and expand topology? Just start nodes.
3. How to shrink topology? Just remove nodes, but not all at once.

Your data will be automatically rebalanced over the new set of nodes.

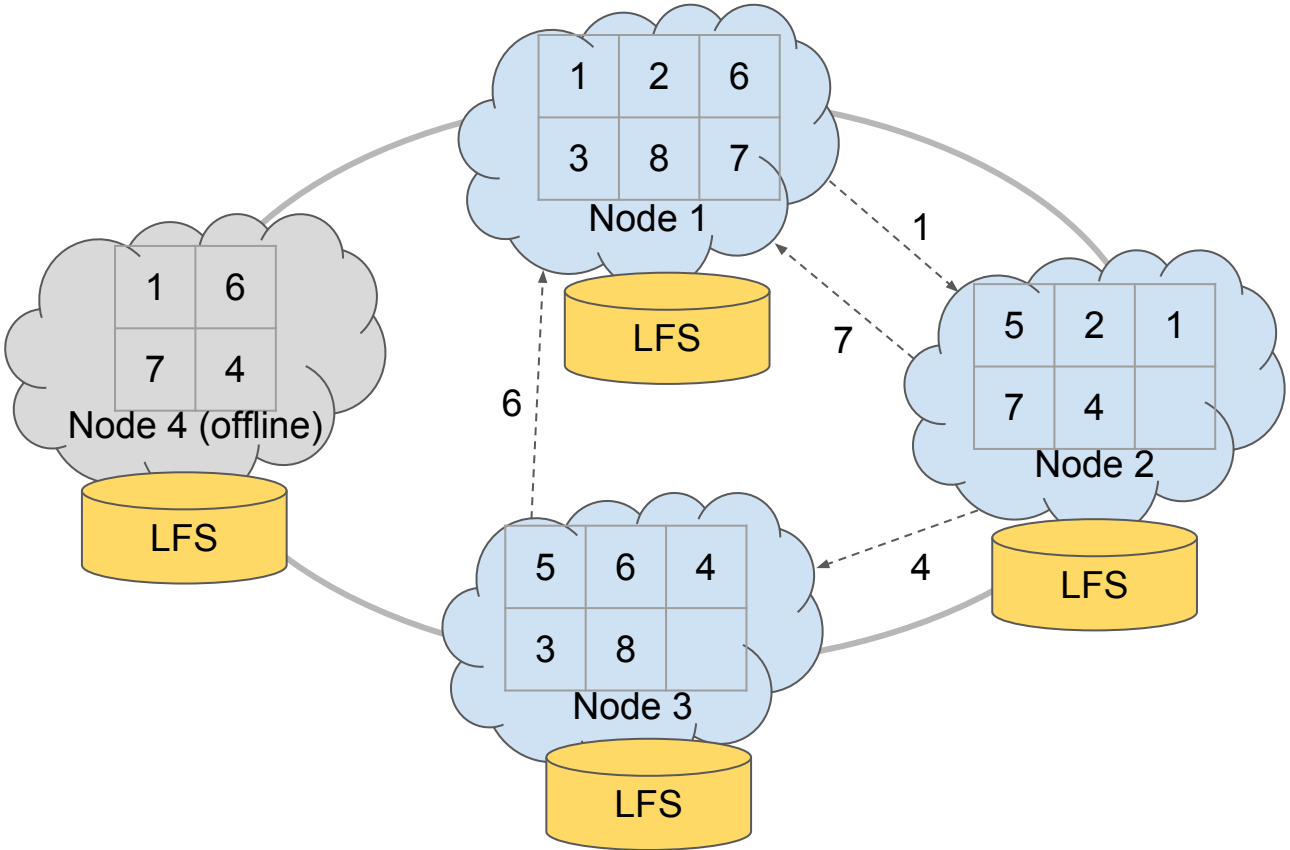
Ok, and where’s the challenge?

Agenda

- In-memory cluster management
- Persistent cluster management
  - Baseline Topology: why?
  - Baseline Topology: and finally, what is it?
  - Baseline Topology: API
  - Use case: Managing persistent cluster with BLT
  - Split-brain scenarios: how BLT and Zookeeper may help

Persistent mode: backups are still present on drives

Do we really need to panic and rebalance immediately?



Estimating rebalance time

1. In-memory mode
    - Node leaves: a lot of time.
    - Node returns: nearly the same amount.
2. Persistent mode
    - Node leaves: a lot of time, much more than in memory-only mode.*
    - Node returns after long delay or with LFS cleared: nearly the same amount.
    - Node returns after short delay: only diff is transferred.

      Delta-WAL rebalancing is awesome!

Should we panic after all? Maybe we should hold rebalancing for a while.

Agenda

- In-memory cluster management
- Persistent cluster management
  - Baseline Topology: why?
  - Baseline Topology: and finally, what is it?
  - Baseline Topology: API
  - Use case: Managing persistent cluster with BLT
  - Split-brain scenarios: how BLT and Zookeeper may help

Baseline Topology: regular set of working nodes

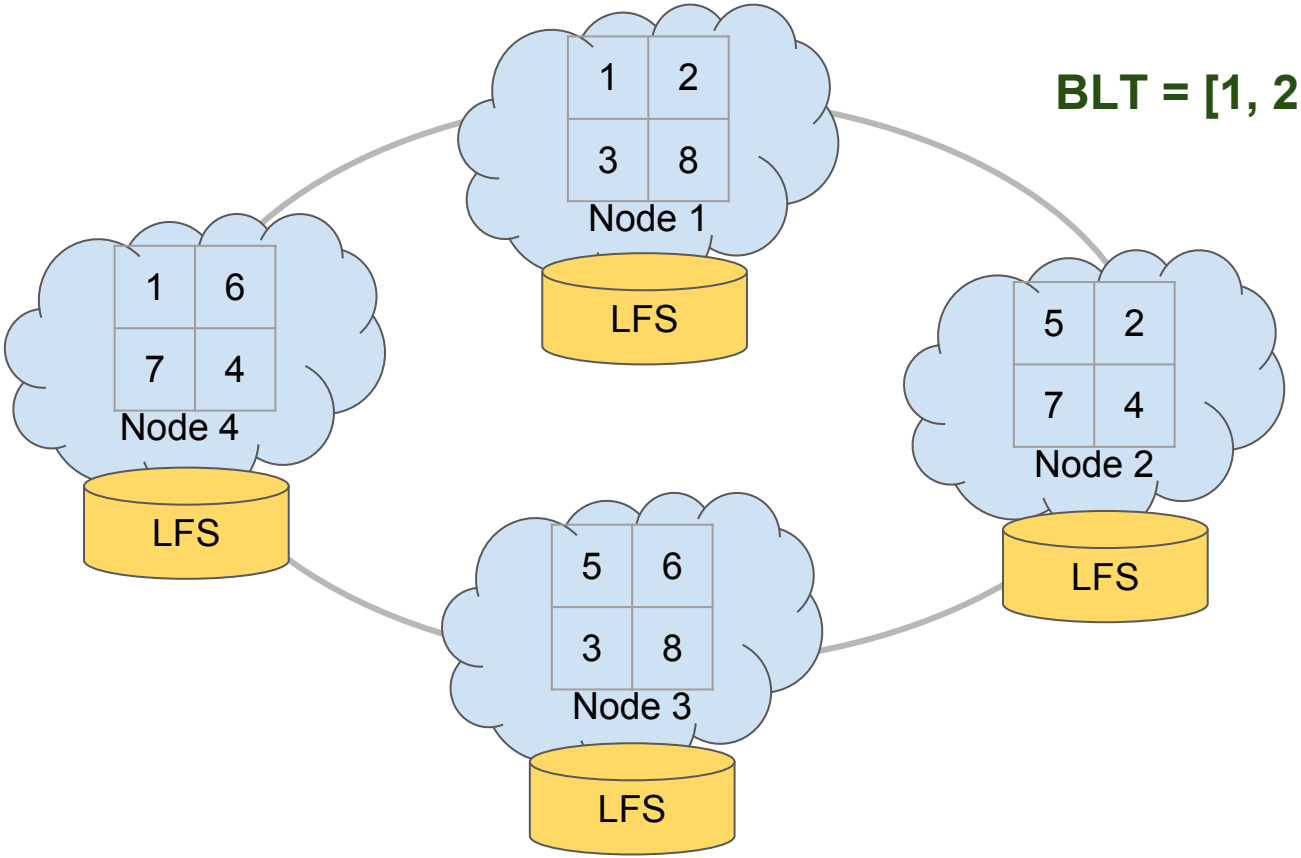
- Current online server nodes = {ONLINE}
- Temporarily unavailable server nodes = {OFFLINE}
- Set of nodes is fixed as baseline = {BASELINE}

- Without baseline topology
  - Affinity is calculated by {ONLINE} set
  - Partitions reside on {ONLINE} node set

- With baseline topology
  - Affinity is calculated by {BASELINE} set
  - Partitions reside on {BASELINE} \ {OFFLINE} node set

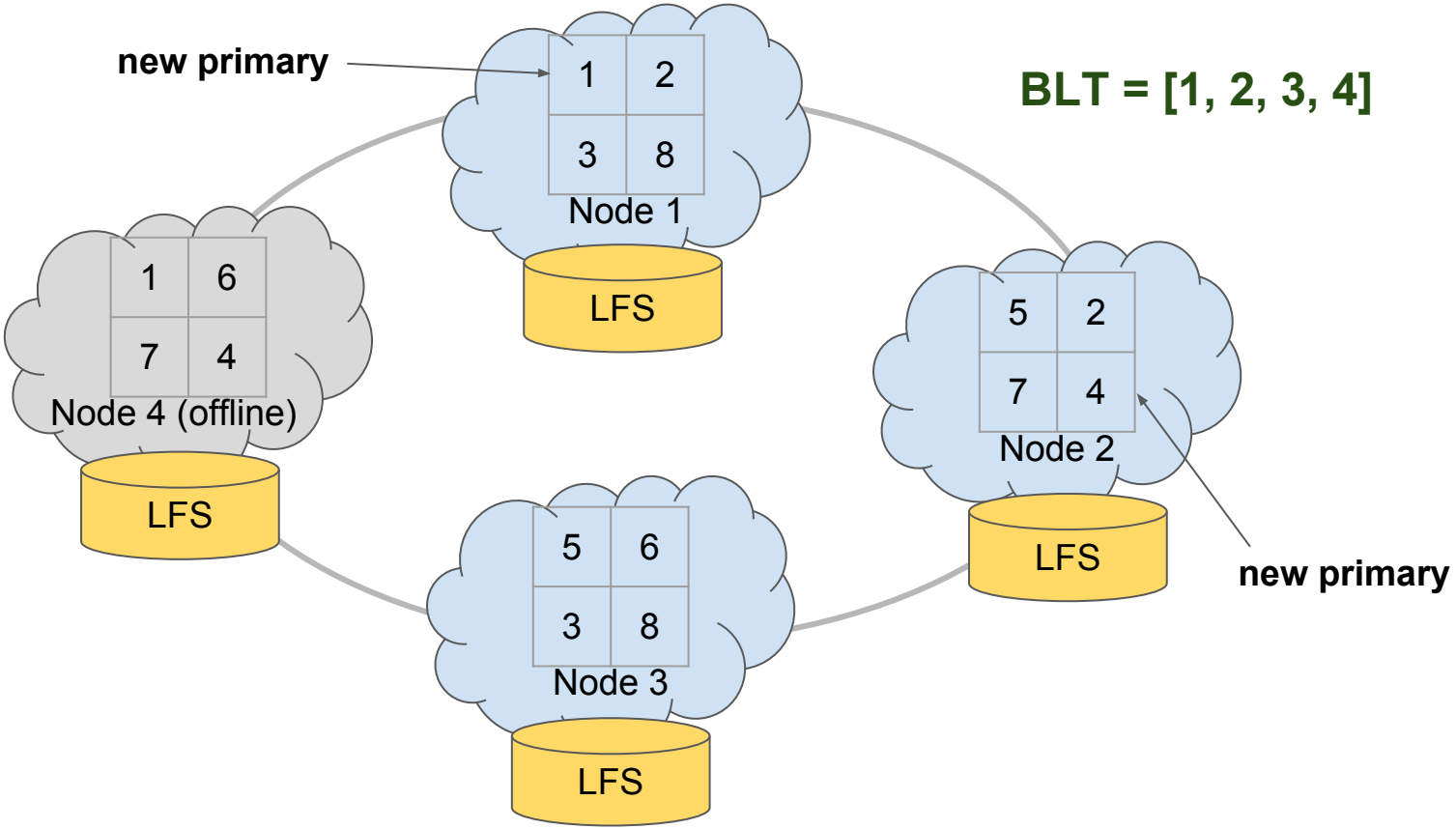
Baseline topology: regular set of working nodes

(partition → node) is calculated according to baseline instead of actual topology



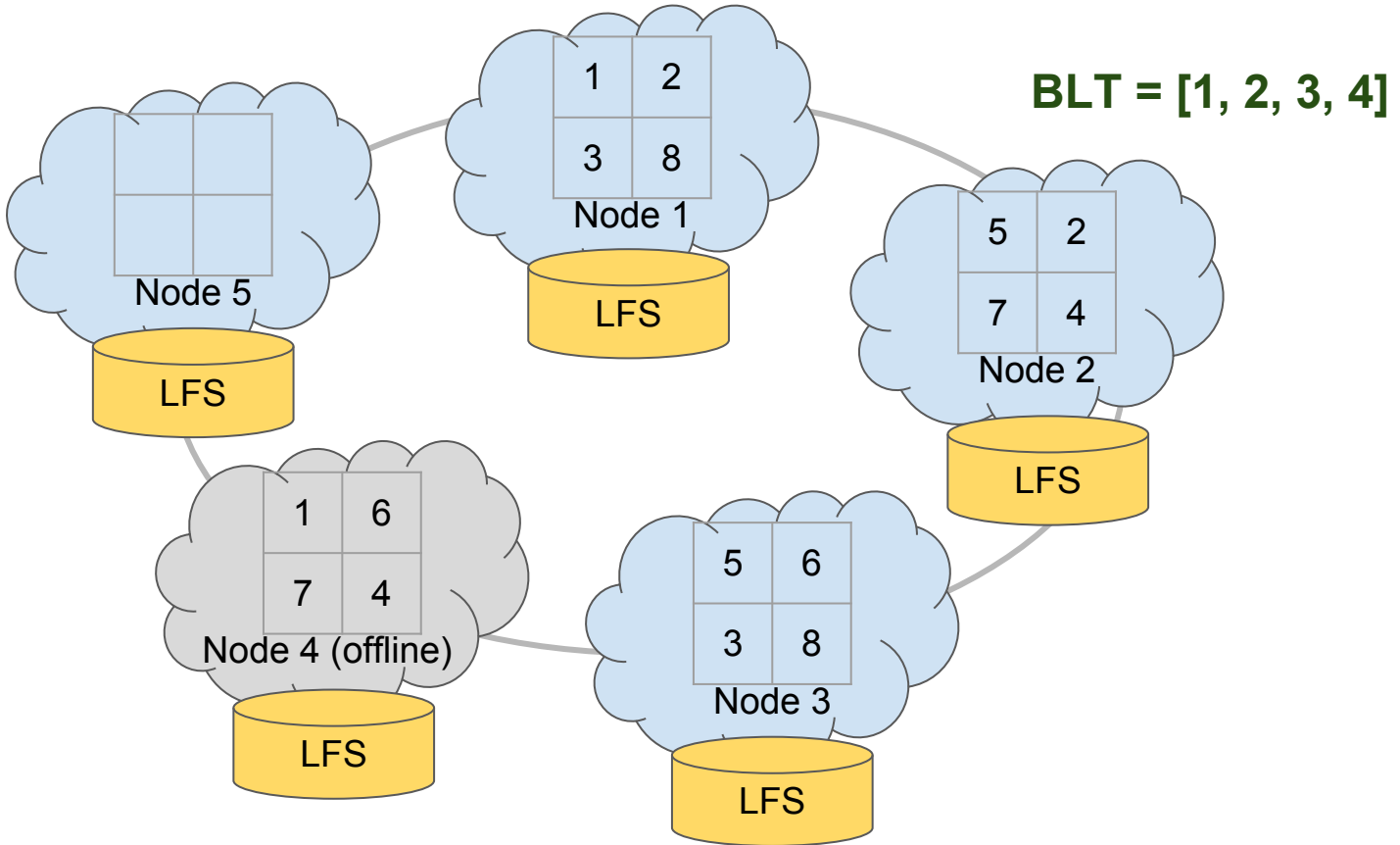
Baseline topology: regular set of working nodes

(partition → node) is calculated according to baseline instead of actual topology



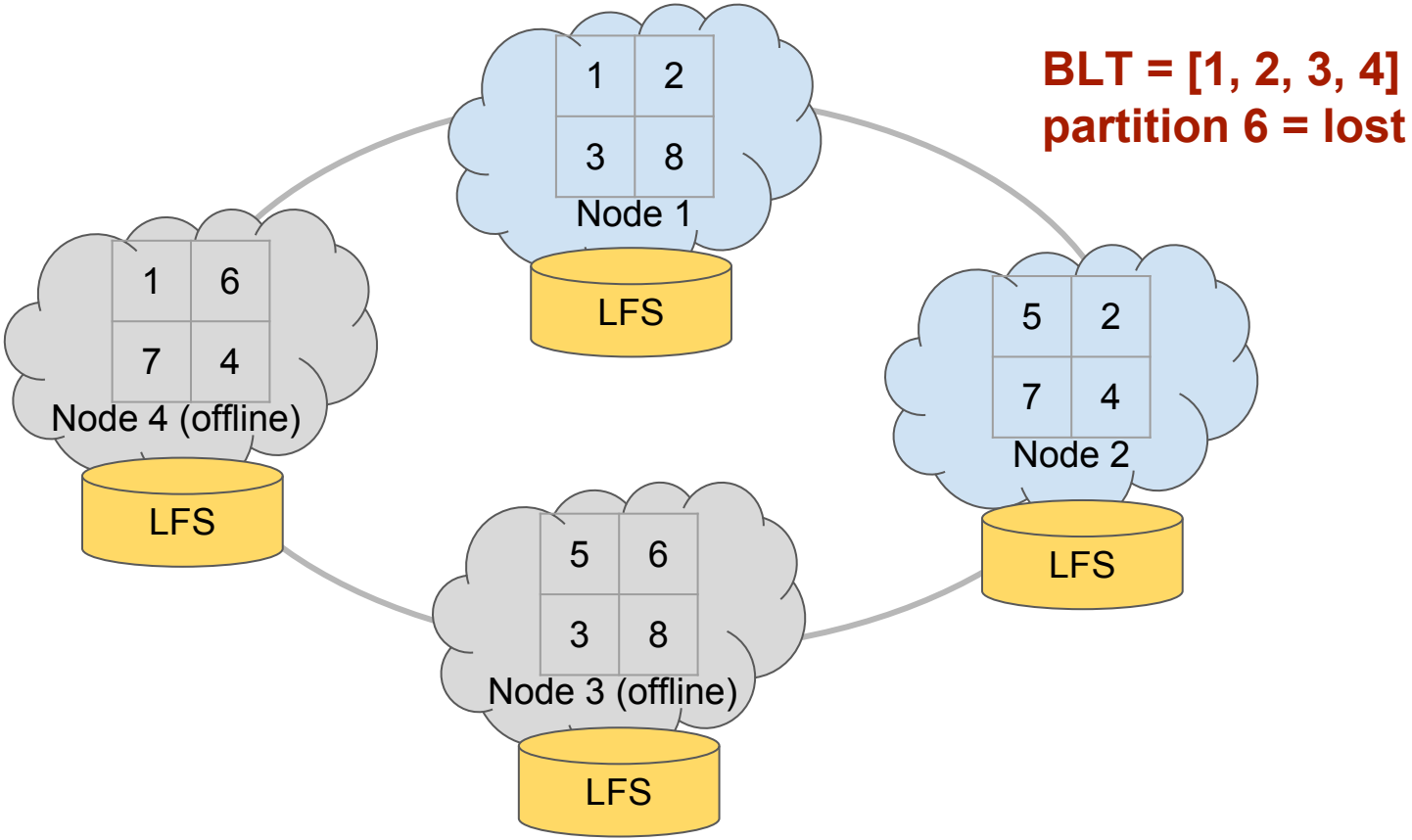
Baseline topology: regular set of working nodes

(partition → node) is calculated according to baseline instead of actual topology



Baseline topology: regular set of working nodes

Rule of thumb: node isn't expected to be back soon → change BLT



Agenda

- In-memory cluster management
- Persistent cluster management
  - Baseline Topology: why?
  - Baseline Topology: and finally, what is it?
  - Baseline Topology: API
  - Use case: Managing persistent cluster with BLT
  - Split-brain scenarios: how BLT and Zookeeper may help

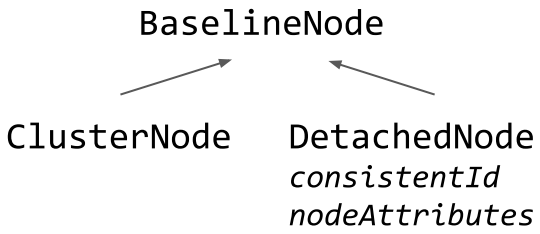
Baseline topology: API

Create:

- `cluster.active(true)`
  Unlike in-memory, persistent cluster should be activated manually
  First activation establishes baseline topology
  BLT is persisted in LFS in special “metastore” partition

- `cluster.setBaselineTopology(topVer)`
  Sets current node set as BLT, works like compareAndSet

- `cluster.setBaselineTopology(baselineNodes)`



Baseline topology bonus: automatic activation

Without baseline:

- `ignite.active(true)` on every cluster start in persistent mode

With baseline:

- First start: `ignite.cluster().setBaselineTopology(currentTopVer)`
- Next start: `<cluster activates when baseline is reached>`

Agenda

- In-memory cluster management
- Persistent cluster management
  - Baseline Topology: why?
  - Baseline Topology: and finally, what is it?
  - Baseline Topology: API
  - Use case: Managing persistent cluster with BLT
  - Split-brain scenarios: how BLT and Zookeeper may help

Use case: first cluster activation

1. Start necessary number of nodes, e.g. with `./ignite.sh`.

   At this moment:

   - Cluster is inactive and can’t process user requests
   - Baseline Topology is not defined yet

2. Activate the cluster, e.g. with `./control.sh --activate`.

   This action:

   - Will set current set of online server nodes as Baseline Topology
   - Will make cluster active and capable of responding to user requests

Use case: cluster restart

1. Deactivate cluster
2. Stop all nodes
3. Do the necessary maintenance - for example, add new .jar with actual business code
4. Start all nodes
    a. Cluster will be activated automatically once last BLT node joins the cluster
    b. If some BLT nodes are unavailable, cluster still can be activated manually with

    ```
    ./control.sh --activate.
    ```

    BLT will remain the same, therefore backup factor will be decreased.

Use case: expanding topology

1. Start new nodes
2. Add new nodes to the BLT with one of the following ways:
   a. `control.sh --baseline add <node's consistentId>`
   b. `control.sh --baseline set consId1[,consId2,....,consIdN]`
   c. `control.sh --baseline <current major topology version>`
3. Data will be rebalanced over the new set of nodes

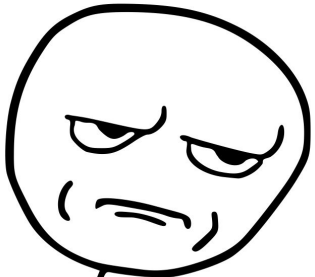
Use case: shrinking topology

1. Stop nodes that should be removed
2. Don’t stop more than `backupFactor` nodes at once.
3. Remove stopped nodes from the BLT with one of the following ways:
   a. `control.sh --baseline add <node's consistentId>`
   b. `control.sh --baseline set consId1[,consId2,....,consIdN]`
   c. `control.sh --baseline <current major topology version>`
4. Data will be rebalanced over the new set of nodes

Use case: please, don’t make me do manual actions

- Implement your own BLT change logic with Ignite Events
  https://apacheignite.readme.io/docs/baseline-topology#section-triggering-rebalancing-programmatically

- Baseline Autochange Policy: expected in AI 2.7

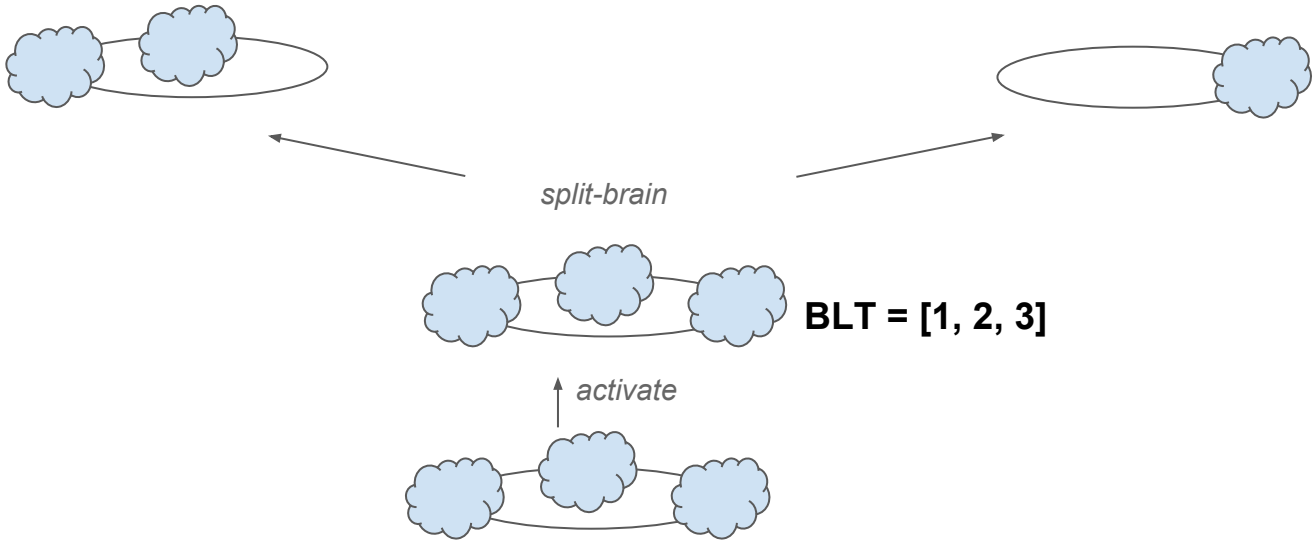


Agenda

- In-memory cluster management
- Persistent cluster management
  - Baseline Topology: why?
  - Baseline Topology: and finally, what is it?
  - Baseline Topology: API
  - Use case: Managing persistent cluster with BLT
  - Split-brain scenarios: how BLT and Zookeeper may help

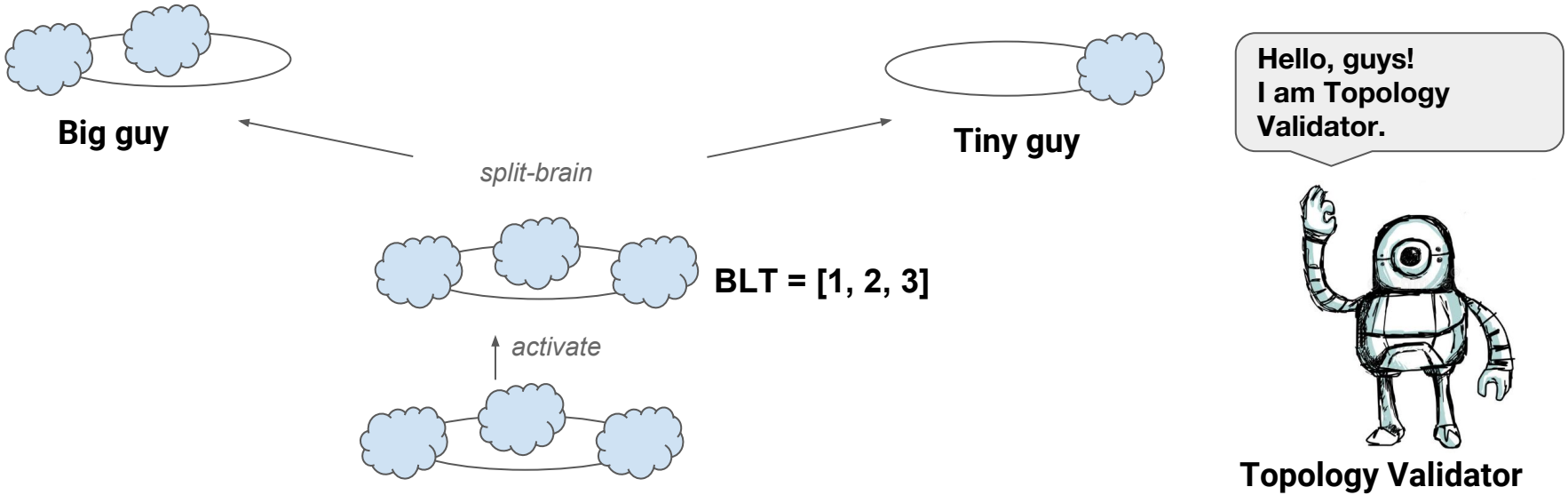
Surviving split-brain: if you want to pick CP from CAP

- The only reliable way to build CP system is shutting down all subclusters except one
- Consistency will be sacrificed during merge of conflicting updates otherwise



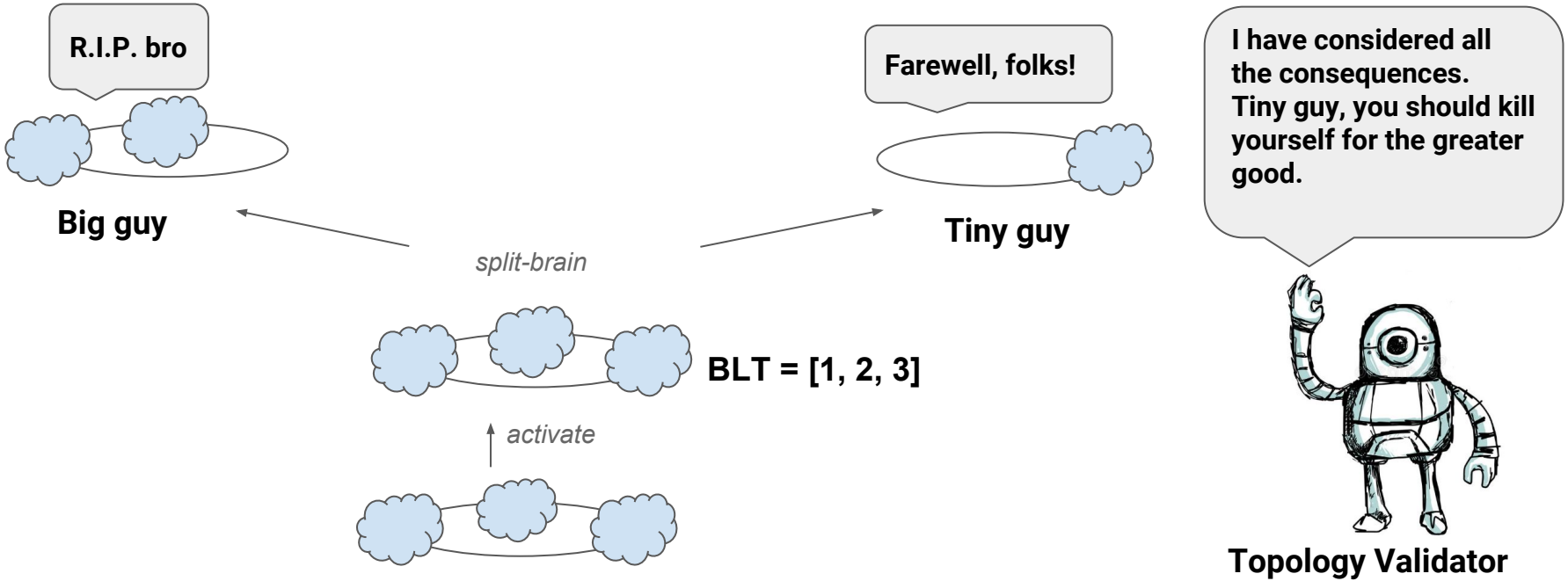
Surviving split-brain: if you want to pick CP from CAP

- Implement `TopologyValidator` to determine which part should die



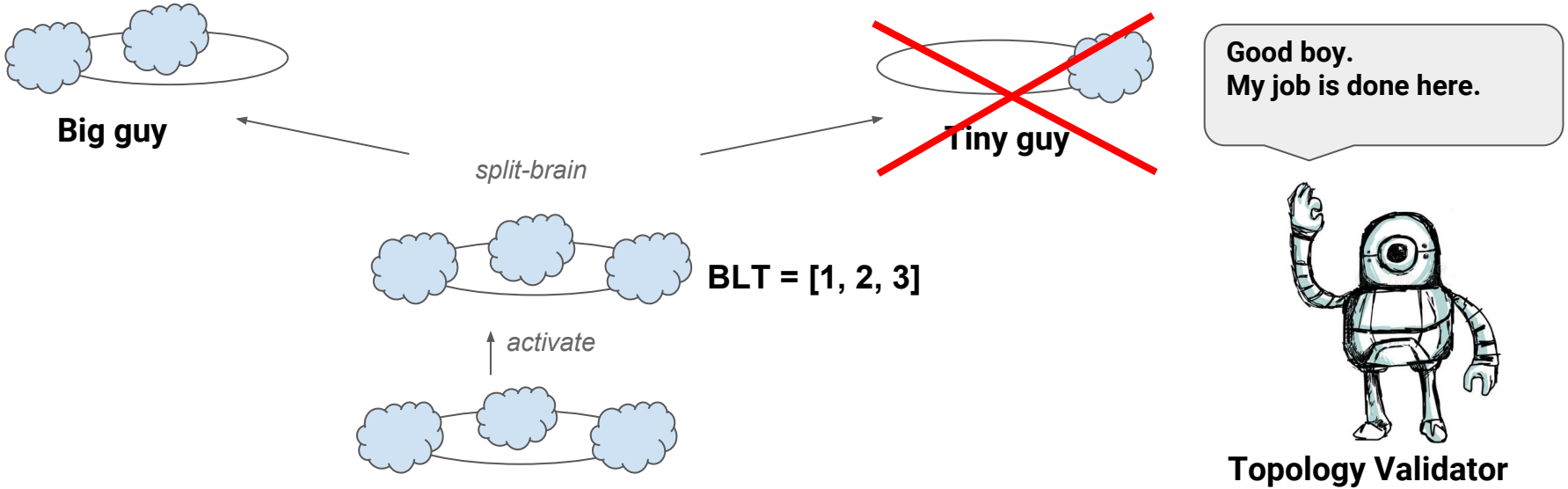
Surviving split-brain: if you want to pick CP from CAP

- `validate(nodes)` should return false on victim subcluster



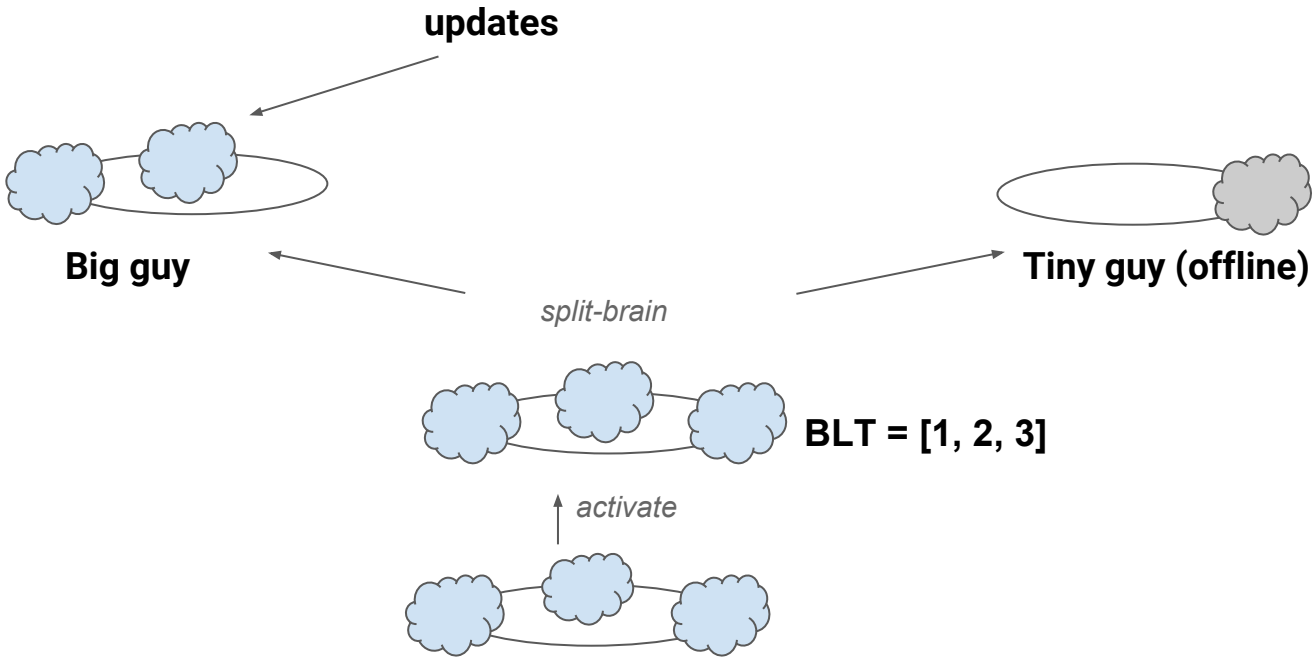
Surviving split-brain: if you want to pick CP from CAP

- If `TopologyValidator` is implemented correctly, all subclusters except one die



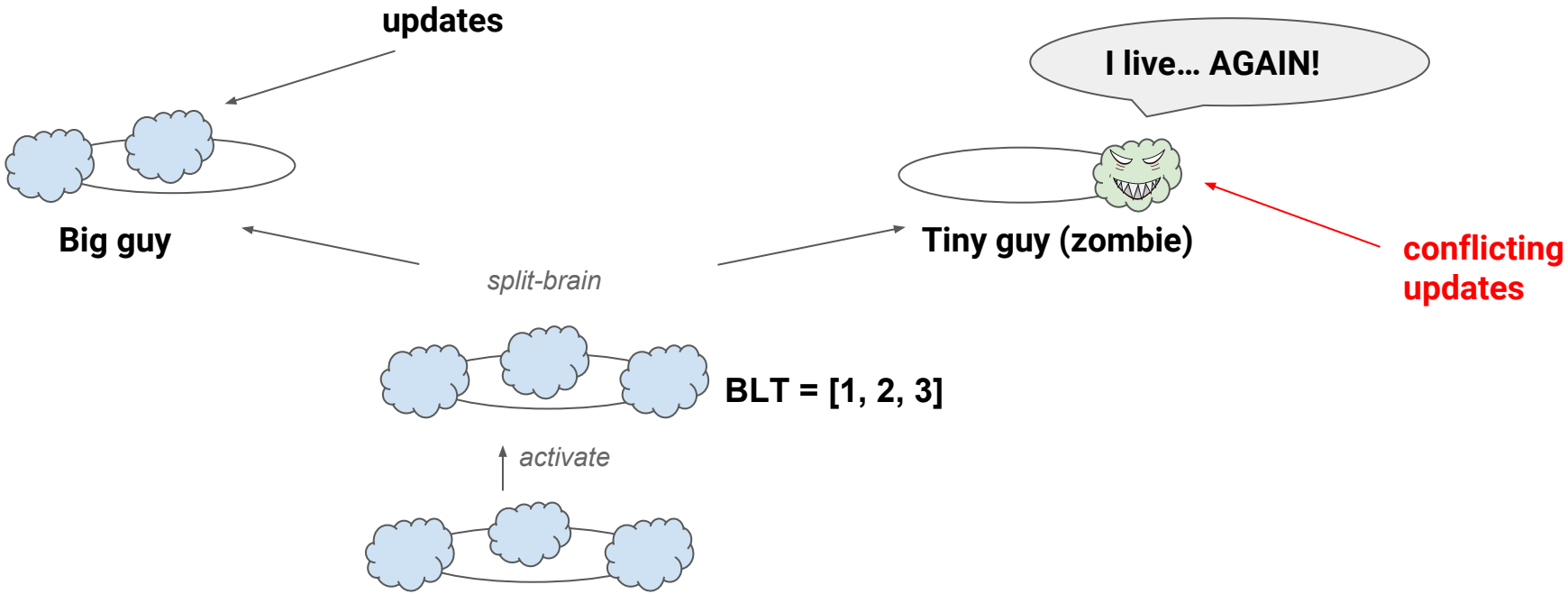
Surviving split-brain: if you want to pick CP from CAP

- All subclusters except one are offline, consistency violation is prevented

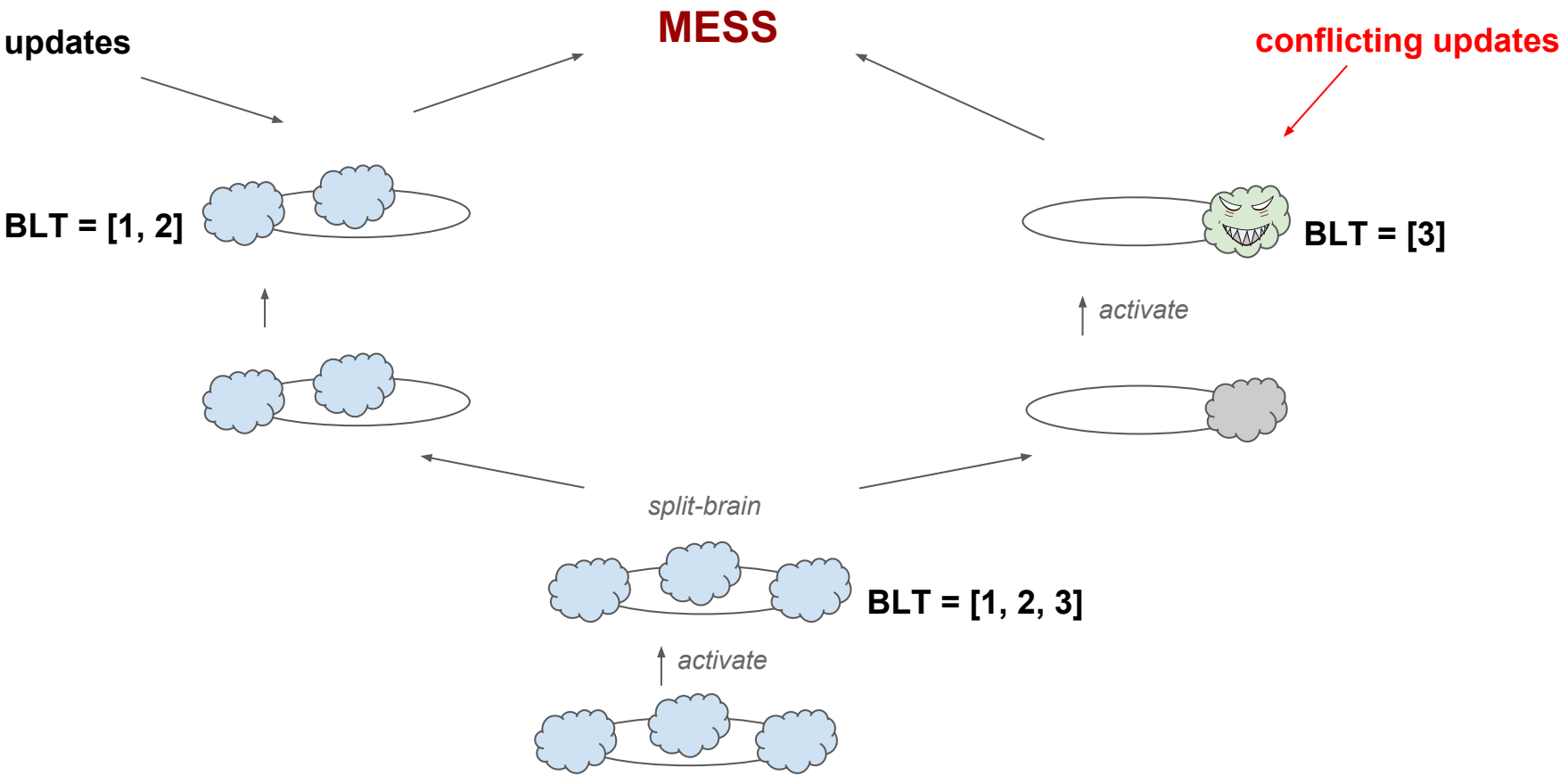


Hypothetically, you still can ruin everything

- `TopologyValidator` won't prevent killed subcluster from restart



Baseline topology bonus: post-split-brain merge protection

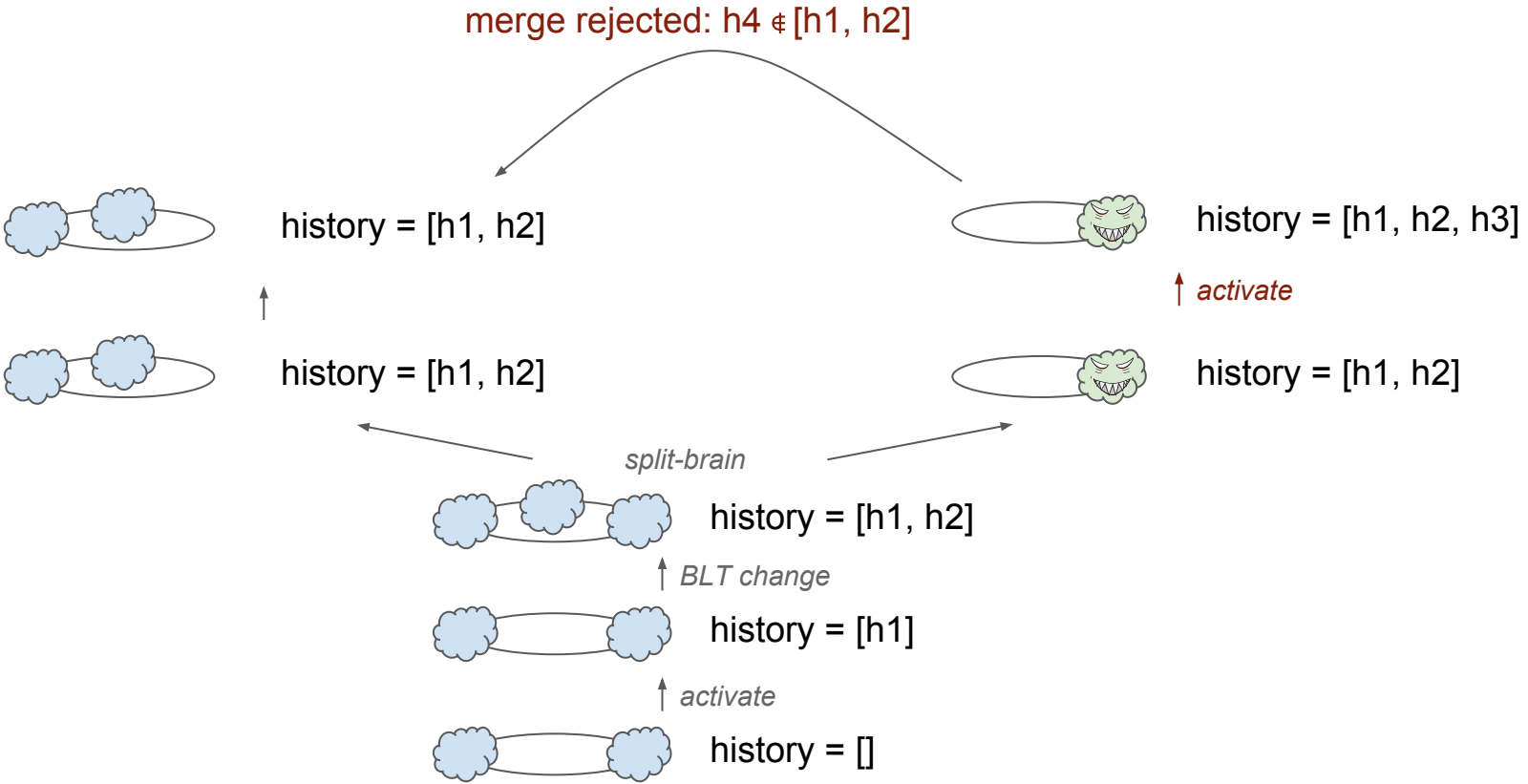


Baseline topology bonus: post-split-brain merge protection

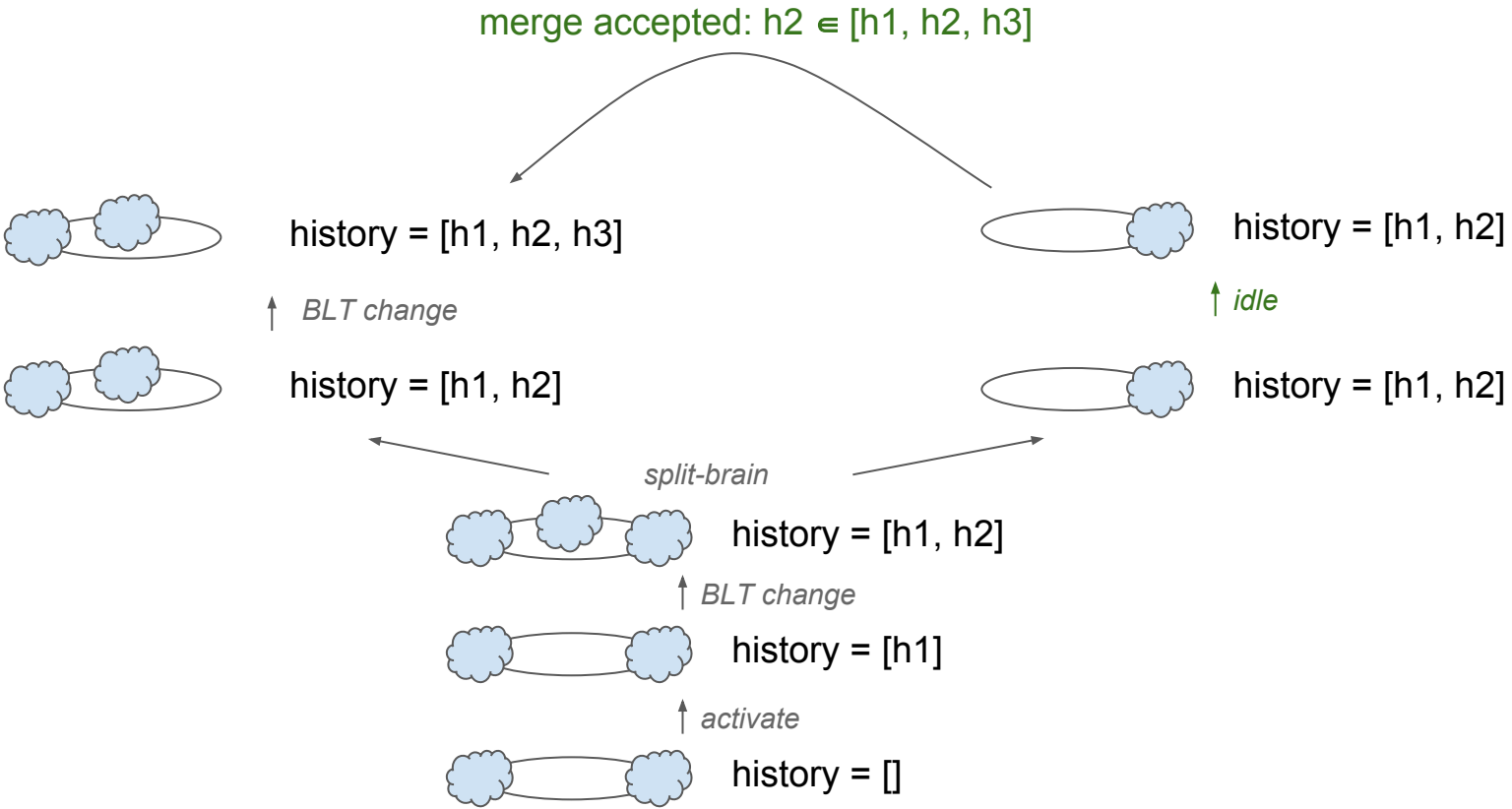
- Baseline hash chain is maintained: all historical values of `hash(baselineNodes)`
- `if (!olderBaselineHistory.contains(newerBaselineHash))`
  `<join is rejected>`
- Rule of thumb: after split, living subcluster starts first, idle subcluster joins

```
class org.apache.ignite.spi.IgniteSpiException:
BaselineTopology of joining node (C ) is not compatible with BaselineTopology in the
cluster. Branching history of cluster BlT ([198, 131]) doesn't contain branching point
hash of joining node BlT (67). Consider cleaning persistent storage of the node and
adding it to the cluster again.
```

Baseline topology bonus: post-split-brain merge protection

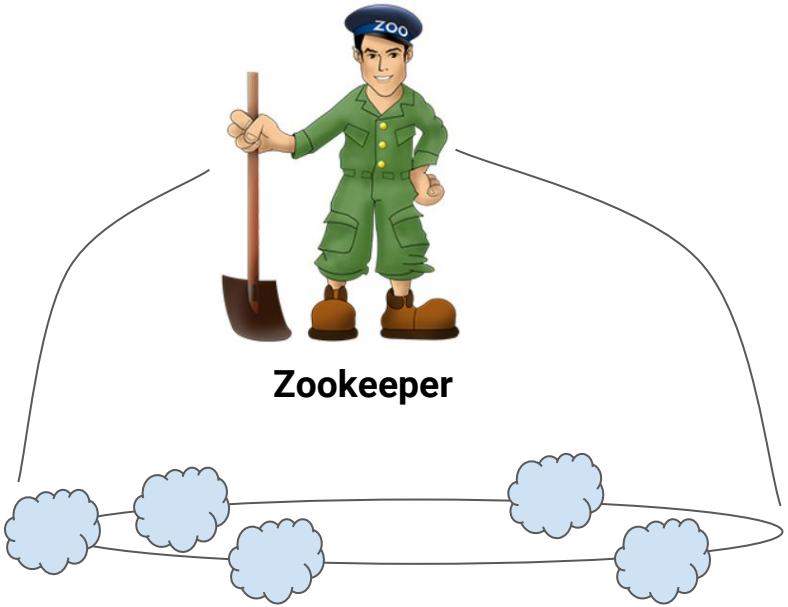


Baseline topology bonus: post-split-brain merge protection



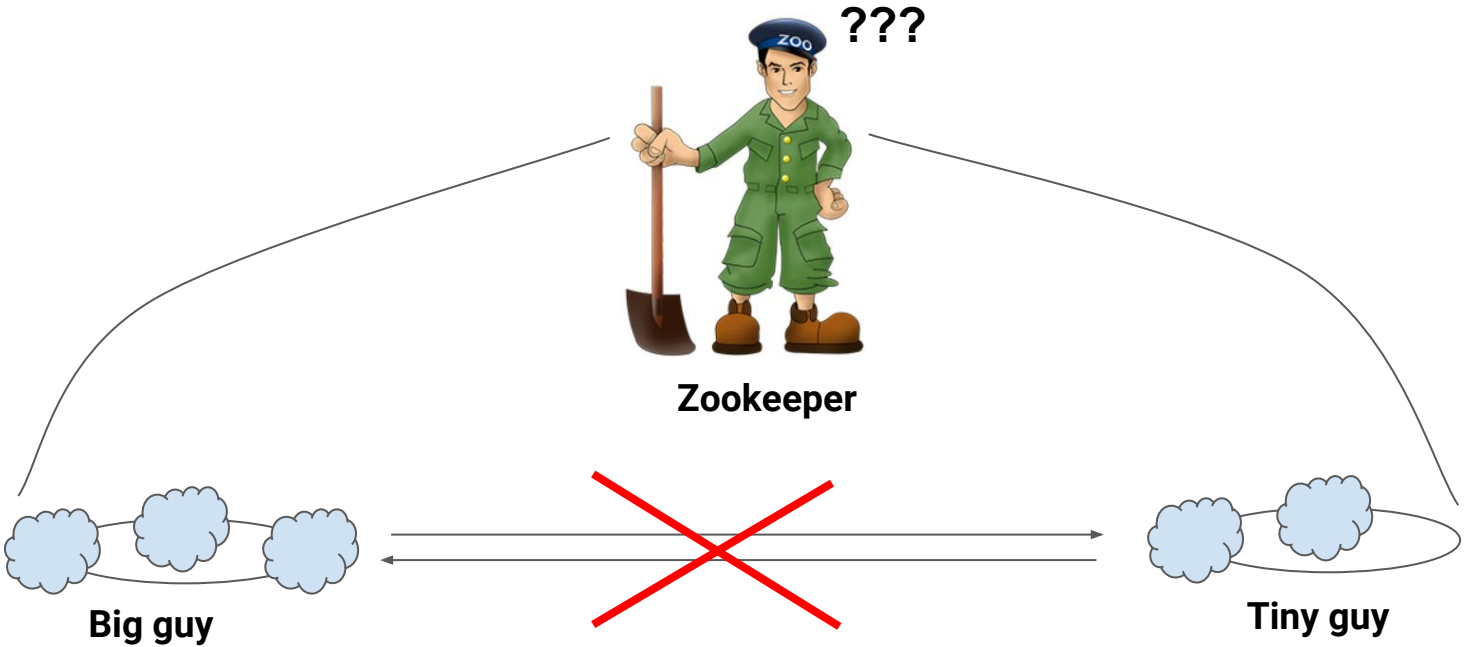
Surviving split-brain: let external Zookeeper do the job

- Use ZookeeperDiscoverySpi which will detect split-brain and kill smaller subcluster



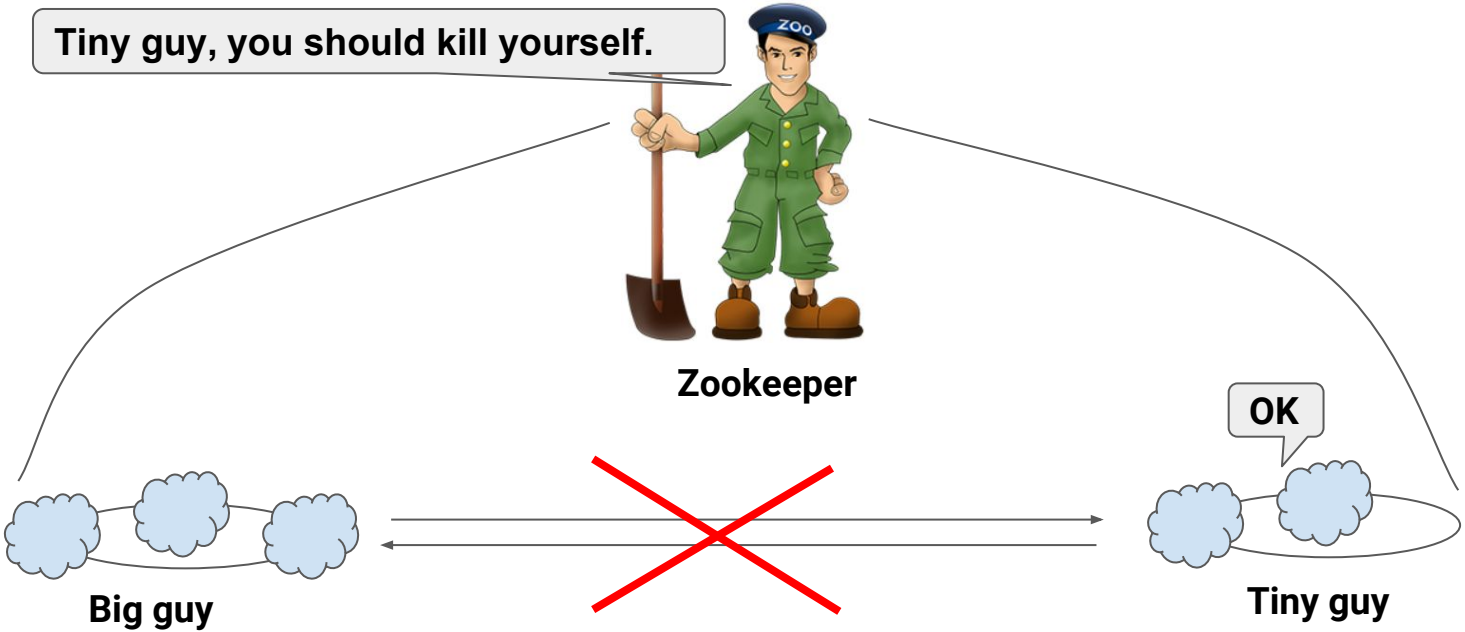
Case 1: Intracluster peer-to-peer communication loss

- Nodes will report Zookeeper about communication problems



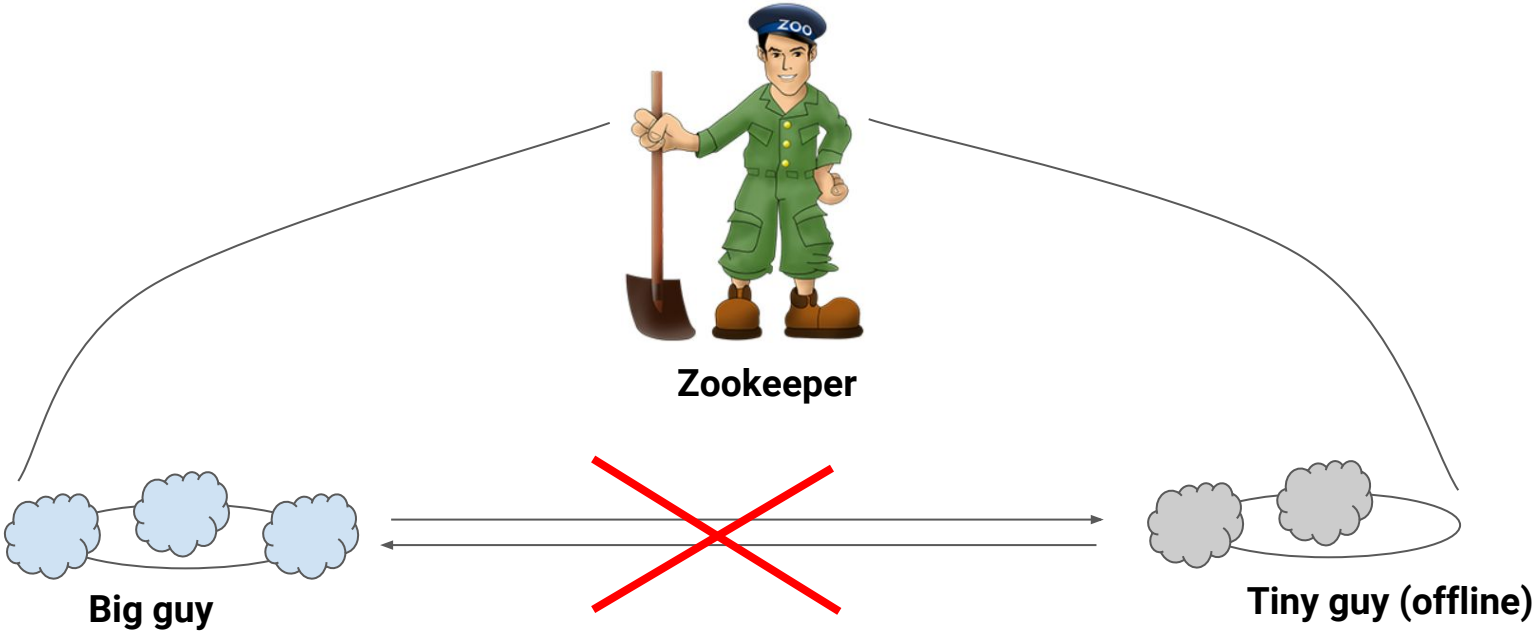
Case 1: in-cluster peer-to-peer communication loss

- Zookeeper will calculate largest fully connected subcluster and kill all other nodes



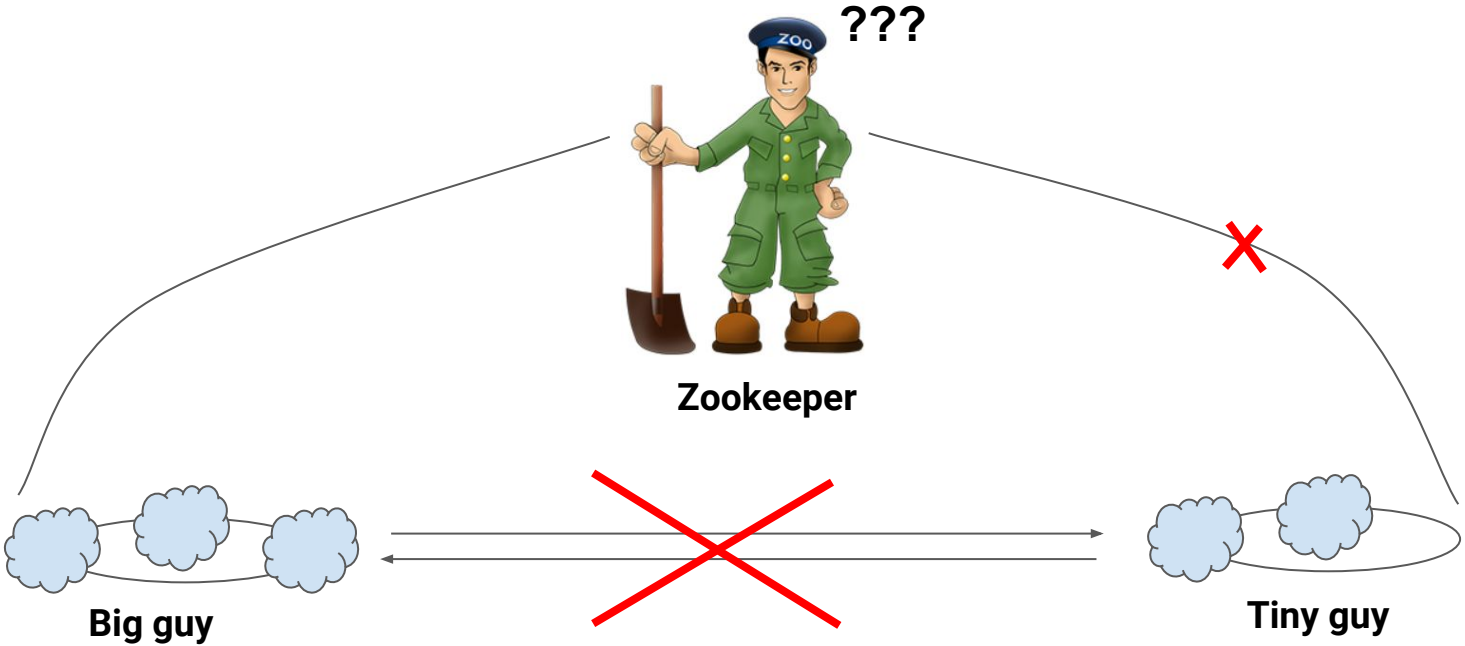
Case 1: in-cluster peer-to-peer communication loss

- All subclusters except one are offline, consistency violation is prevented



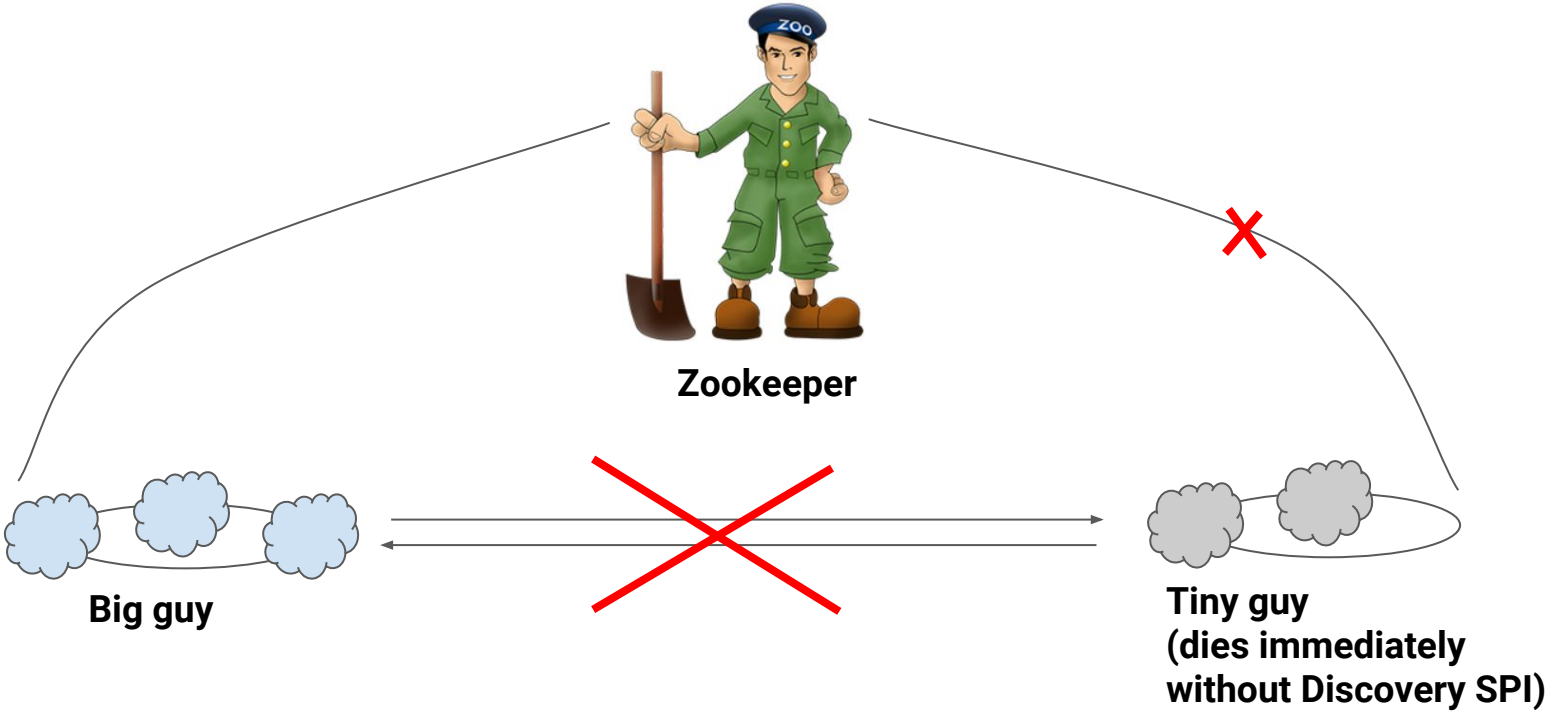
Case 2: Total subcluster isolation

- Zookeeper excludes unavailable nodes from topology



Case 2: Total subcluster isolation

- Nodes in unavailable subclusters can’t discover each other without Zookeeper



Thank you for attention!
Questions?