Stable and consistent membership at scale with Rapid

Lalith Suresh, Dahlia Malkhi, Parikshit Gopalan
Ivan Porto Carreiro¹, Zeeshan Lokhandwala²

VMware Research ¹VMware ²One Concern
Which nodes are in the cluster?

Membership management and failure detection
Types of membership services

Centralized

Spark, HDFS, Zookeeper-based systems

Gossip-based

Cassandra, Akka, Redis Cluster, Dynamo
Existing solutions do not provide stability and consistency at scale.
400 node deployment of Akka Cluster
4 processes experience high packet loss
Ideal cluster size if faulty nodes removed
Unstable views

Leads to performance degradation and outages

[Cassandra-6126, Consul-916, Consul-1212, Consul-1337]
[HotOS '13, SoCC'14]
Inconsistent views
Inconsistent views are difficult to program on top of Slicer [OSDI '16], Census [ATC '08]
Rapid

Stable and consistent membership at scale
Robust against asymmetric network failures, flip-flops, packet loss etc.
Rapid

Stable and consistent membership at scale

Processes see the same sequence of membership changes
Rapid

Stable and consistent membership at scale

Bootstraps 2000 nodes 2-5x faster than Zookeeper and Memberlist
Configuration 1 ➔ [ ] ➔ Configuration 2

Monitoring Overlay ➔ Membership change proposal ➔ View change with consensus

Expander graph-based monitoring ➔ Multi-process cut detection ➔ Fast path to consensus

Stability ➔ Consistency
Rapid runs in both **centralized** and **decentralized** configurations
This Talk: decentralized design and failures
Configuration 1

Monitoring Overlay

Expander graph-based monitoring

Configuration 2

Membership change proposal

Multi-process cut detection

View change with consensus

Fast path to consensus
Expander-graph based monitoring

$K$ observers per node

Subject
Expander-graph based monitoring

Pluggable edge failure detector (directed)
Expander-graph based monitoring

$K$ observers per node

Up to $K$, edge alerts broadcasted during failures

Subject
Multi-process cut detection

Expander-based monitoring overlay
Multi-process cut detection

Subset S
Multi-process cut detection

Rest of the graph

Subset S
Multi-process cut detection
Multi-process cut detection

K

H

L
Multi-process cut detection

# Alerts

Subject
Multi-process cut detection

# Alerts

K

H

L

Candidates for removal

Subject
Multi-process cut detection

Only output detection if this range is empty

# Alerts

K

H

L

Subject
Multi-process cut detection

# Alerts

K

H

L

Subject

Stable reports

Unstable reports
Multi-process cut detection

Observer-subject failures?
Multi-process cut detection

# Alerts

K

H

L

Subject

Stable reports

Unstable reports

Implicit detections
Multi-process cut detection

Delay membership changes until churn stabilizes
Almost-everywhere agreement

All processes output the same cut

\([-\text{red}, -\text{yellow}, -\text{green}]\]

with high probability
Almost-everywhere agreement

1000 processes, 8 failures, $K=10$

$H=9$  
$0\%$ conflicts  
$L=1$

$H=9$  
$0\%$ conflicts  
$L=2$

$H=9$  
$0.06\%$ conflicts  
$L=4$
Membership change proposal

Monitoring Overlay

View change with consensus

Expander graph-based monitoring

Multi-process cut detection

Fast path to consensus
Configuration 1

Monitoring Overlay
Expander graph-based monitoring

Membership change proposal
Multi-process cut detection

View change with consensus
Fast path to consensus

Configuration 2
Almost-everywhere agreement [•, •, •] → Full agreement [•]
Almost-everywhere agreement → Full agreement

Gossip-based Counting protocol

Every node counts #votes per-proposal

Almost everywhere agreement → Full agreement
Almost-everywhere agreement  

Decide if Fast Paxos quorum ($>\frac{3n}{4}$ nodes) of identical votes

Full agreement

Almost-everywhere agreement

[Image of network diagram]

Gossip-based Counting protocol

Almost-everywhere agreement

[Image of network diagram]

Gossip-based Counting protocol
Almost-everywhere agreement → Full agreement

1000 processes, 10 node membership change
~11 KB bandwidth usage per node for 1 second
(Memberlist uses ~8 KB/s)
Evaluation

Implementation: ~2700 LOC in Java (~2600 LOC of tests)

github.com/lalithsuresh/rapid

Compared against 3-node Zookeeper cluster and Memberlist.

Experiments run on 100 VMs
(2 cores, 4GB RAM each)

Not showing Akka Cluster because it did not scale past 500 nodes.
Bootstrap times

Cluster size

Bootstrap latency (s)

- ZooKeeper
- Memberlist
- Rapid
Bootstrap times

Cluster size

ZooKeeper  Memberlist  Rapid

Bootstrap latency (s)

1000  1500  2000

2-5x improvement
1% of processes experience high packet loss
1% of processes experience high packet loss
1% of processes experience one way network partition
1% of processes experience one way network partition
Rapid

Stable and consistent membership at scale

Monitoring Overlay → Membership change proposal → View change with consensus

Expander graph-based monitoring
Multi-process cut detection
Fast path to consensus

Cluster size
Bootstrap latency (s)

ZooKeeper  Memberlist  Rapid-C  Rapid

Cluster size

1000  1500  2000

0  50  100  150  200