Apache ZooKeeper

Patrick Hunt (@phunt)
Cloudera/ZooKeeper PMC

Agenda

- The fallacies of distributed computing
- A quick history
- What is ZooKeeper?
- Use cases
- Guidelines for success
- Common problems, limitations

Let me know if you have questions

Fallacies of Distributed Computing

1. The network is reliable
2. Latency is zero
3. Bandwidth is infinite
4. The network is secure
5. Topology doesn't change
6. There is one administrator
7. Transport cost is zero
8. The network is homogeneous

History

- December 2006 - first commit (cvs!)
- November 2007 - 0.0.1 on Sourceforge
- June 2008 - moved to Apache
- October 2008 - 3.0.0 on Apache
- June 2010 - Usenix "Best Paper" award
    - "ZooKeeper: Wait-free Coordination for Internet-scale Systems"
- November 2010 - ZooKeeper moves to TLP
- November 2011 - ZooKeeper 3.4 released
- Current: 3.4.9 & 3.5.2-alpha are out

What is ZooKeeper?

ZooKeeper is much more than a distributed lock server!

A highly available, scalable, distributed, configuration, consensus, group membership, leader election, naming, and coordination service

Why use ZooKeeper?

- Difficulty of implementing these kinds of services reliably
  - brittle in the presence of change
  - difficult to manage
  - different implementations lead to management complexity when the applications are deployed

What is ZooKeeper again?

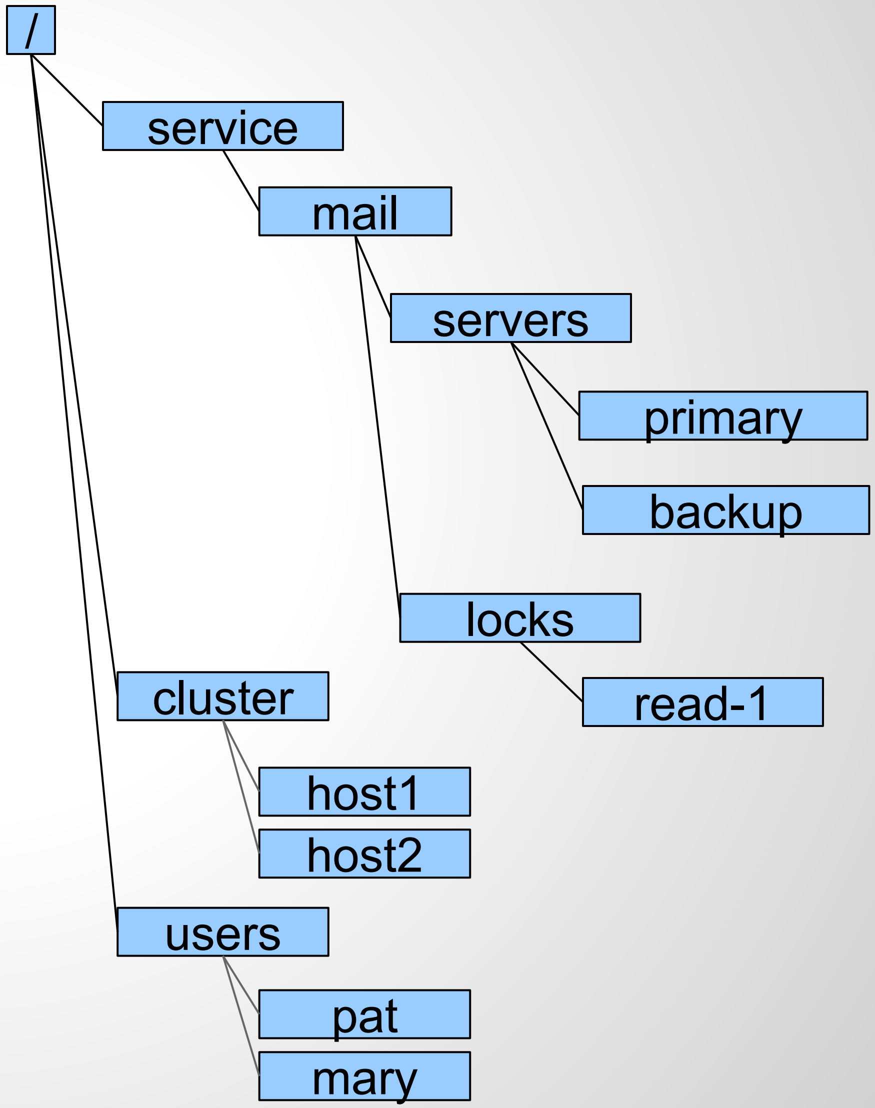
- File API without partial reads/writes
- No renames
- Ordered updates and strong persistence guarantees
- Conditional updates (version)
- Batch updates (multi)
- Watches for data changes
- Ephemeral nodes
- Generated file names

Any Guarantees?

1. Clients will never detect old data.
2. Clients will get notified of a change to data they are watching within a bounded period of time.
3. All requests from a client will be processed in order.
4. All results received by a client will be consistent with results received by all other clients.

Data Model

- Hierarchical namespace
- Each znode has data and children
- data is read and written in its entirety



ZooKeeper API

String create(path, data, acl, flags)

void delete(path, expectedVersion)

Stat setData(path, data, expectedVersion)

(data, Stat) getData(path, watch)

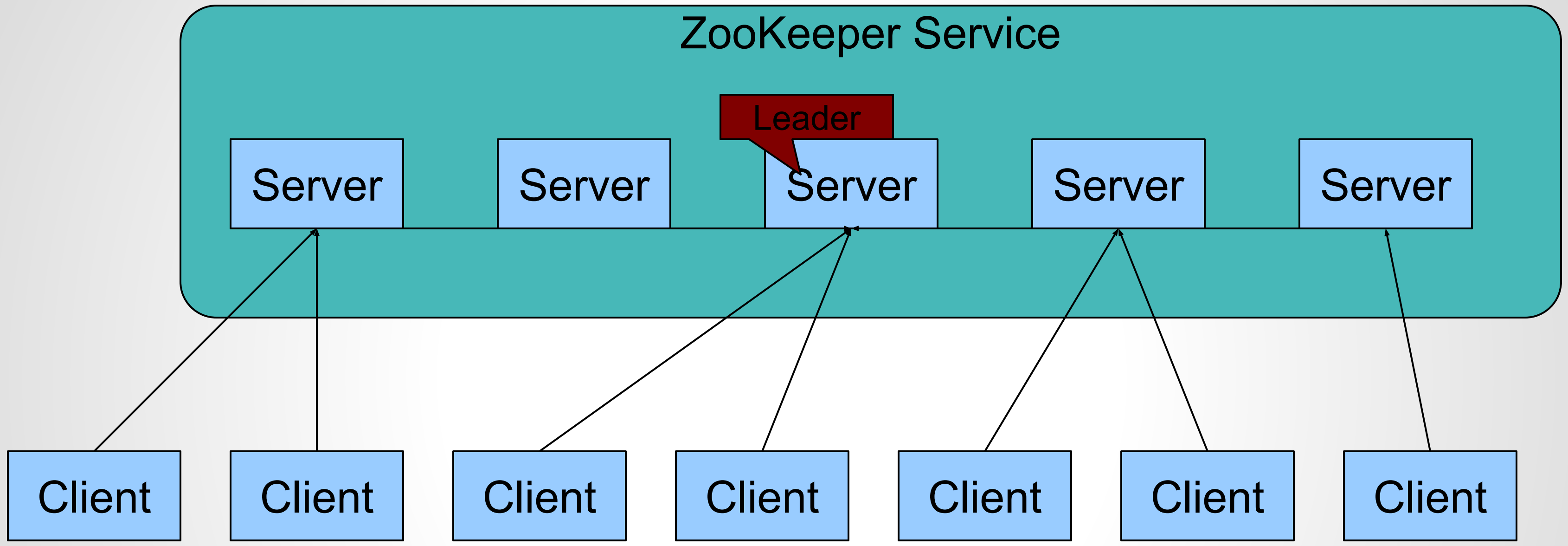
Stat exists(path, watch)

String[] getChildren(path, watch)

void sync(path)

List<OpResult> multi(ops)

ZooKeeper Service



- All servers store a copy of the data (in memory)
- A leader is elected at startup
- Followers service clients, all updates go through leader
- Update responses are sent when a majority of servers have persisted the change

A sampling of use cases

- Configuration Management
- Leader Election
- Group Membership
- Work Queues
- Cluster Management
- Load Balancing
- Sharding

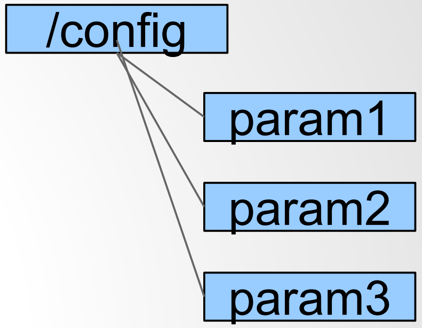
Configuration Management

Administrator

1. setData(“/config/param1”, "value", -1)

Consumer

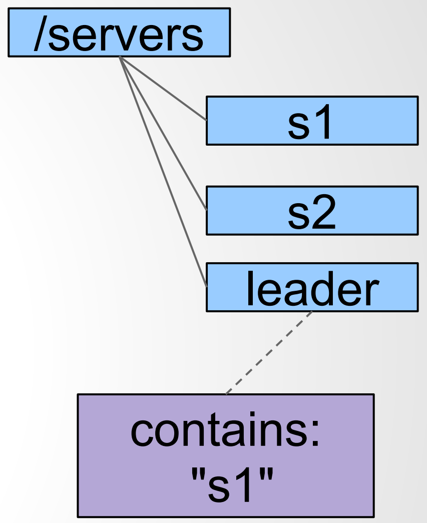
1. getData("/config/param1", true)



Leader Election

1. getdata(“/servers/leader”, true)
2. if successful follow the leader described in the data

   and exit
3. create(“/servers/leader”, hostname, EPHEMERAL)
4. if successful lead and exit
5. goto step 1



** Don't confuse this with ZooKeeper Ensemble Leader Election

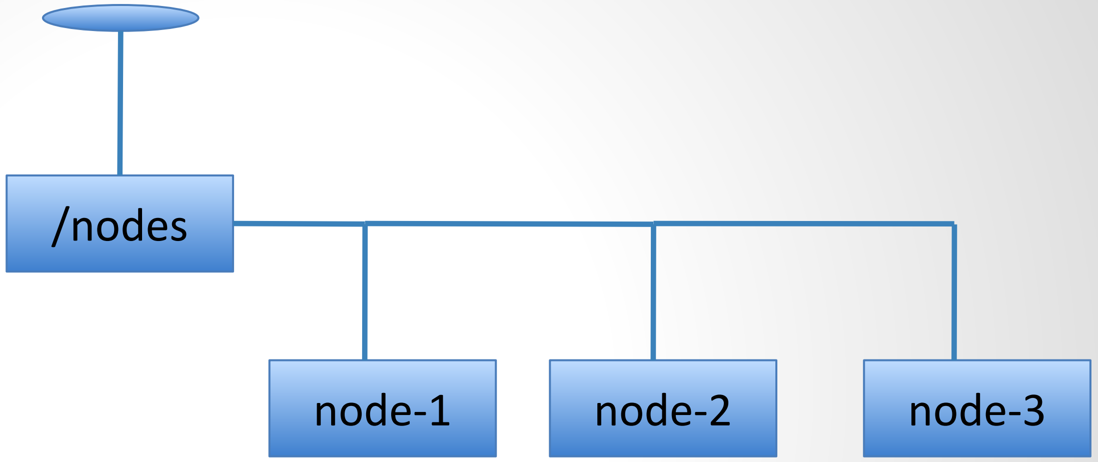
Leader Election in Python

```
handle = zookeeper.init("localhost:2181", my_connection_watcher, 10000, 0)
(data, stat) = zookeeper.get(handle, “/app/leader”, True);
if (stat == None)
        path = zookeeper.create(handle, “/app/leader”,  hostname:info,
    [ZOO_OPEN_ACL_UNSAFE], zookeeper.EPHEMERAL)
        if (path == None)
            (data, stat) = zookeeper.get(handle, “/app/leader”, True)
            #someone  else is the leader
            # parse the string path that contains the leader address
        else
            # we are the leader continue leading
else
    #someone else is the leader
  #parse the string path that contains the leader address
```

Cluster Management

Monitoring process:

1. Watch on /nodes
2. On watch trigger do *getChildren(/nodes, true)*
3. Track which nodes have gone away



Each Node:

1. Create /nodes/node-${i} as ephemeral nodes
2. Keep updating /nodes/node-${i} periodically for node status changes (status updates could be load/iostat/cpu/others)

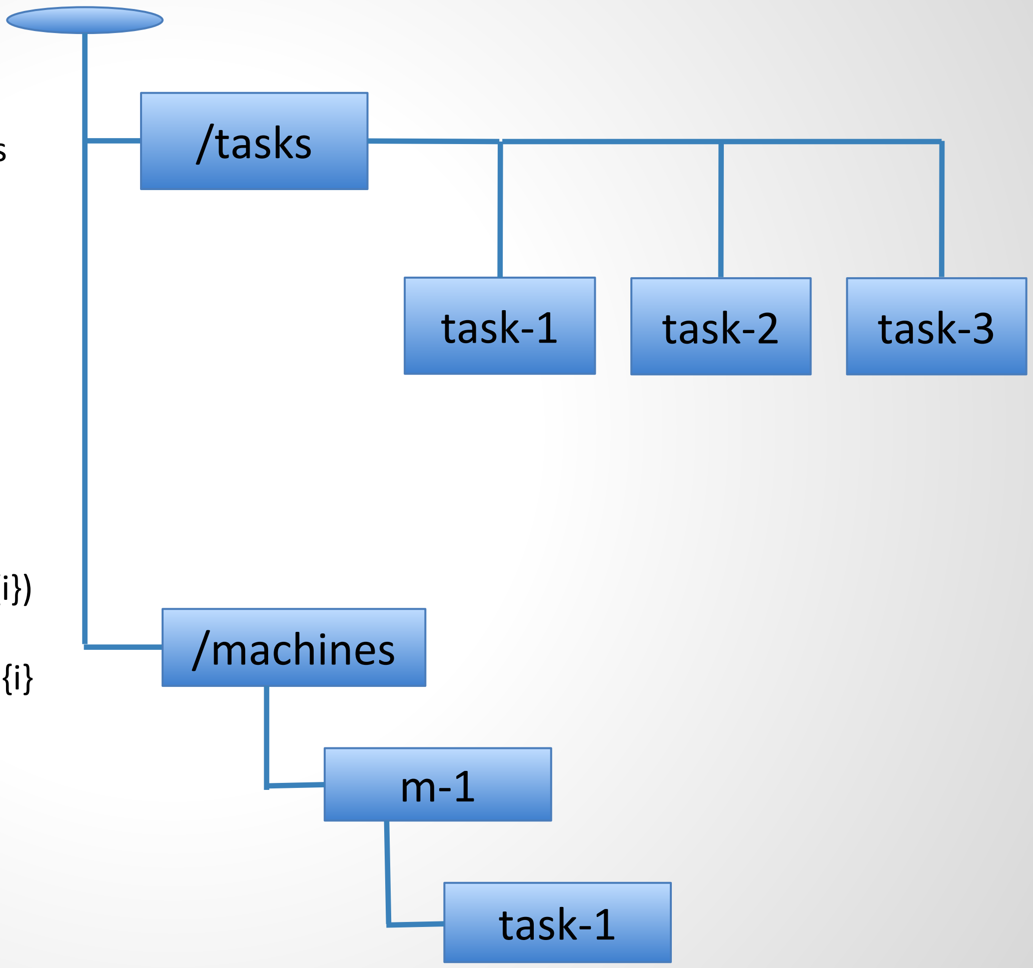
Work Queues

Assigner process:

1. Watch /tasks for published tasks
2. Pick tasks on watch trigger from /tasks
3. assign it to a machine specific queue by creating create(/machines/m-${i}/task-${j})
4. Watch for deletion of tasks (task completed)

Machine process:

1. Machines watch for /(/machines/m-${i}) for any creation of tasks
2. After executing task-${i} delete task-${i} from /tasks and /m-${i}



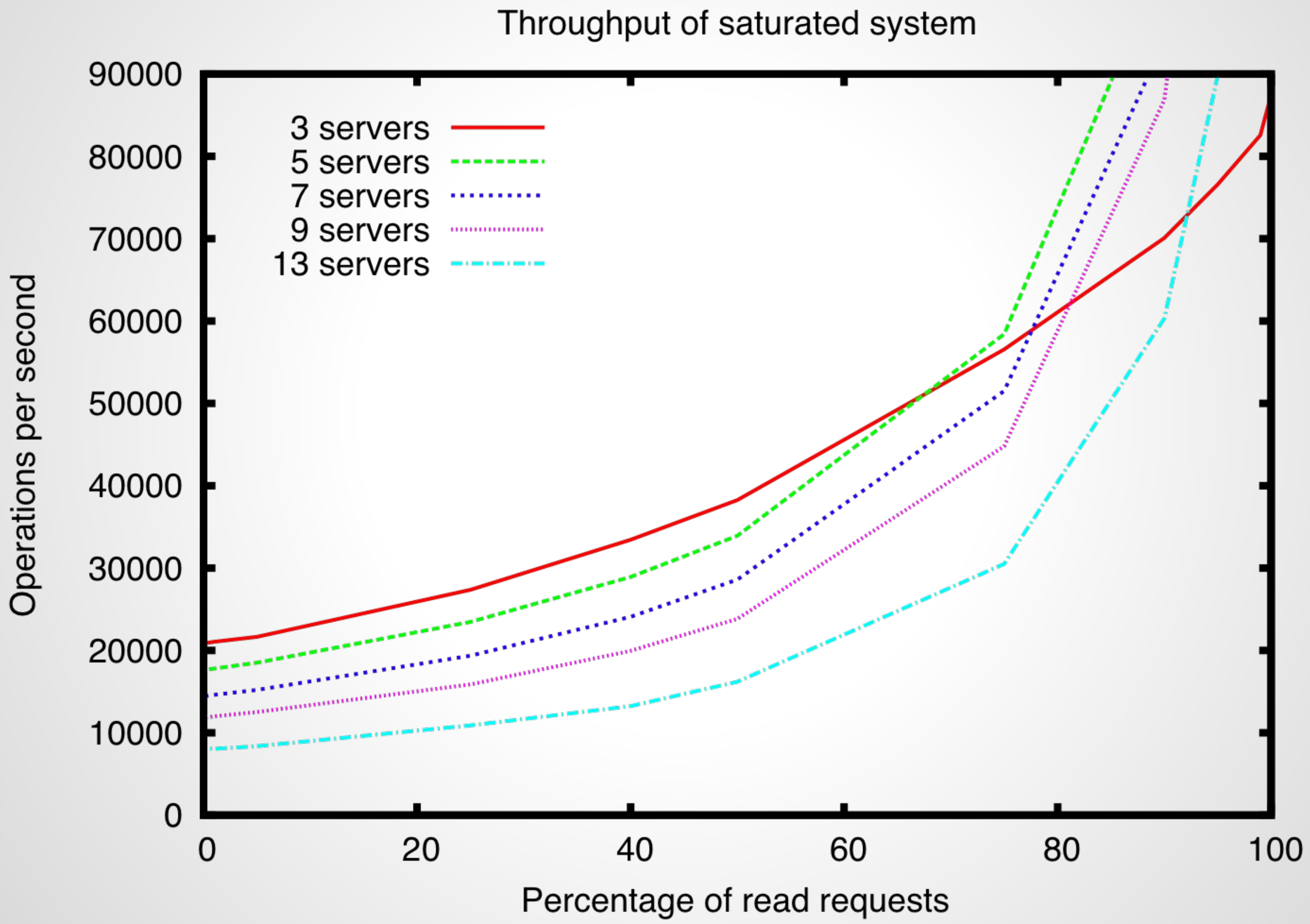
Ensemble Size?

What's the right size for the ensemble?

- Majority rule voting
  - Don't use an even number of servers

- 1 - standalone, no reliability
- 3 - allows for one failure
- 5 - optimal for online production serving

Performance Numbers.



Maintenance

- Minimal
  - Ensure that you clean the datadir (autopurge added in 3.4)
- The rest is automatic
  - e.g. servers bootstrap from the Leader

Monitoring Tools

- Command port (four letter words - 4lw)
- JMX
- slf4j/log4j logging

HTTP/JSON:

- Jetty is in 3.5.x - ZOOKEEPER-1346

Where are we?

- Multi Tenant
- Observers
- Recipes
  - Reusable code libraries
- Bindings

  Java, C, Perl, Python, REST, Ruby?

- Third party code - Apache Curator!

Who is using ZooKeeper?

- **Mesosphere**!
- Many Apache projects including;
  - HBase, Hadoop, Solr, Kafka, Blur, Helix, Pig, Hive...
- Yahoo!
- Twitter
- LinkedIn
- Netflix
- Youtube
- Facebook
- Pinterest
- Airbnb
- Many more (see the "powered by" wiki page)

What do we do next?

- ZooKeeper ensemble dynamic reconfig
  - Added in 3.5.0, stabilizing
- More security work
  - transport & auth - e.g. ZOOKEEPER-1045
- Usability – timeouts from zookeeper clients are a headache – ZOOKEEPER-22
- Scaling
- More/better multi-tenancy

Common Problems

- Sessions timing out frequently
    - Client side GC or swapping?
    - Heartbeating - session timeout vs expiration
- High latency on client operations
    - Dedicated spindle?
    - Monitoring - low cost, high ROI
- Remember - there is no magic
    - Network/disk/cpu/memory

Limitations

- Not a db/filesystem/K-V/etc...
- ZooKeeper is not horizontally scalable
  - Max session count
  - Max operations per second (see graph above)
- 1mb max data size (configurable)

Q&A

- Questions?

- Links:

http://zookeeper.apache.org

https://github.com/phunt