cloudera

Introduction to Zookeeper

Today’s speaker – Tom Hanlon

- **TomTheTrainer@cloudera.com**
- Senior Instructor at Cloudera

How I became aware of Zookeeper, and why that matters to you

- HBASE
  - Uses Zookeeper to manage HA

What’s HBASE

- Hbase is a massively scalable distributed data store, easily scales across many many nodes
- Uses Zookeeper to manage failures and to provide information about the current state of the system

The Problem

- Managing Distributed Services is a challenge
- A recurring challenge with similar repeating issues
- What issues ?
  - Who is in charge?
  - Am I in charge?
  - Are we both in charge?
  - What do I do when I start back after failing ?
  - what is the state of process X ?
  - Election of Primary out of group of peers

The Solution…

- Modify your apps to deal with these issues
- Where to start ?
  - Not trivial
  - 2 systems, Active and Standby might suffice
  - What about split brain ?

The Solution continued

- You will quickly find that you need a similar set of services
  - Distributed lock management
  - Master Election. (Slave Discovery ?)
  - Information sharing, time based
  - Some sort of hierarchy management
    - Service B is dead, do not offer service C
    - Includes startup hierarchy management

Ad-Hoc Solutions ... not good

- A collection of ad-hoc solutions is problematic in the long term
- Like services should be managed by the same tool
- A single place of reference is good

Zookeeper has what you need

Well a significant portion of what you need.

Zookeeper can provide simple services that allow client services to coordinate or synchronize activities and provide a reference point to agree on basic information about the state of the current environment

Zookeeper to the Rescue

- Zookeeper http://zookeeper.apache.org/
- Built by Yahoo as a distributed system to manage distributed services in distributed systems.
- Clean focussed Open Source tool
    - Does one thing and does it well
- A highly available distributed coordination service

Distributed Systems Need the same thing over and over again

- In order to be highly available we must deal with
  - Leader election
  - Quorum
  - temporal dependencies
  - consensus
  - shared locks
  - etc

Zookeeper has goals not unlike your own

Reliability

Availability

Simplicity

Ease of management

Clarity

Distributed Lock Server.. and more

- highly available, distributed metadata filesystem
- Runs on 3 or 5 (or more) machines
- Transparently elects Master
  - Other machines are read-only replicas

Zookeeper as a Metadata Filesystem

- Zookeepers model resembles a standard tree based filesystem
- Nodes can store up to 1MB of data
- Nodes can have child nodes
- Nodes can not be renamed
- No hard or soft links
- Guarantees
    - Ordered updates, conditional updates, and watches

A *Distributed* metadata filesystem

The distributed nature of Zookeeper is what makes these simple functions extremely useful

- Over the network creation of nodes
- Over the network updating of metadata associated with a node
- Conditional updates
- Watches *

* Watch notification similar to linux inotify

But wait.. there is more !!!

- Zookeeper can generate NodeNames in order to avoid collisions
- Can build “presence” awareness systems using Ephemeral Nodes
    - Nodes that exist as long as you are connected

Zookeeper can provide

- Naming
- Synchronization
- Leader election
- Group Management

Zookeeper details

- Presents a filesystem like abstraction with a tree of znodes
  - each node can hold data and other nodes
  - How much data? limited to 1MB
- Also stored at each node, transaction ID, creation and modified timestamps, and version number.

cloudera

Interaction with znodes

- create
- delete
- exists
- get data
- set data
- get children
- sync

More on Znodes

- All operations are:
  - Atomic
  - persistant
  - ordered
  - Access Control List per node define who can perform operations
    - ACL determines Read Write Delete Permissions

More API

- Sequential numbering
  - i.e. Create node /n/child_X, setting X to lowest unused value
- Conditional updates via the version number
  - A bit like transactional shared memory
  - Do a read, make all your changes, try and commit
  - If you fail, start again.

Watching a Node

- A watch can be set on a node allowing action to occur when a node is updated
- In fact, when any state is changed on the server, the client can be notified via a watch – including session state changes.
- Sessions are persistent across server failures – if the server you are talking to fails, you can transparently negotiate session continuation with another server.

API features

- Enough power to implement common concurrent objects
  - Can do compare-and-swap on version number (and therefore CAS on data in two round trips)
  - CAS has consensus number infinity in wait-free hierarchy, so theoretically powerful, for what that's worth
- Can build queues, semaphores and so on.
  - Examples later

Yes, but does it scale?

- Typical deployment is 5 to 7 machines
  - Always choose an odd number for maximum relative fault tolerance
- All updates go through same machine
  - So single machine is bottleneck
  - Partitioning would probably help
- Throughputs are fairly decent
  - Tens of thousands of updates per second
  - Works best in heavy read scenarios

Is it Paxos?

- No, not quite
- In standard operation, protocols look very similar
- Paxos has to cope with arbitrary message re-orderings and competing leaders

DEMO….

cloudera

(Thank you for your time)

Questions?

Appendix A: Resources

- Cloudera
  - http://www.cloudera.com/blog
  - http://wiki.cloudera.com/
  - http://www.cloudera.com/blog/2011/03/simple-moving-average-secondary-sort-and-mapreduce-part-1/
- Zookeeper
  - http://zookeeper.apache.org/

Appendix B: Demo

ZOOKEEPER DEMO

In this demo we will show some of the functionality of Zookeeper and hopefully assist you in getting started with Zookeeper.

Introduction:
Zookeeper is written in java, and we can connect to Zookeeper using Java. There is also a command line client and we will use that for our demo. Since Zookeeper presents a distriuted system coordination platform as a filesystem or directory structure there is also a tool that "mounts" that filesystem in userspace. I have no information regarding stability or production use of zkfuse, but worth checking out. There are python (http://www.cloudera.com/blog/2009/05/building-a-distributed-concurrent-queue-with-apache-zookeeper/) and other bindings so that you can start using zookeeper in your application.

Getting started:

For this demo I downloaded zookeeper and gunzipped and untarred in my home directory on my mac.

```
$ tar xvf zookeeper-3.3.3.tar.gz
```

This will create a drectory zookeeper-3.3.3

```
$ cd zookeeper-3.3.3
```

First we will configure a basic Zookeeper config file

```
$ mv conf/zoo_sample.cfg conf/zoo.cfg
```

Edit that file so that it resembles this..

```
# The number of milliseconds of each tick
tickTime=2000
# The number of ticks that the initial
# synchronization phase can take
initLimit=10
# The number of ticks that can pass between
# sending a request and getting an acknowledgement
syncLimit=5
# the directory where the snapshot is stored.
dataDir=/tmp/zoo
# the port at which the clients will connect
clientPort=2181
```

Start up Zookeeper in server mode.
*note that in real environments you would want 3 Zookeepers working together for High Availability.

```
$ bin/zkServer.sh start
```

You will get a warning

```
2011-07-27 17:56:40,231 - WARN  [main:QuorumPeerMain@105] - Either no config or no quorum defined in config, running  in standalone mode
```

This warning reflects that Zookeeper is runing in Standalone mode, a single instance.

Zookeeper is a java process so we can verify easilly that it is running by running jps as the user who launched zookeeper, or as root.

```
$ jps
14761 Jps
14743 QuorumPeerMain
```

This shows that process ID 14743 is the Zookeeper server that we launched, technically reffered to as a Quorum, but in our case a Quorum of 1.

With Zookeeper running we can explore some of the features it provides.

Administrative commands:
Zookeeper provides a set of 4-letter words, or administrative information commands.
Fore example.

```
$ echo ruok | nc localhost 2181
imok ## Zookeeper response
```

Note that there is no newline after "imok" so it is easy to lose this response visually in your terminal.
Also note that the terminal that you started Zookeeper in will still be getting messages from Zookeeper. For puposes of demonstration I like to keep that terminal output available, so launch another terminal window to run the rest of the demos, such as the 'ruok' and other 4-letter words.

If Zookeeper was not running or was not "OK" you would get no response, silence.

MORE 4 LETTER WORDS

'stat' returns statistics
In order to use the 4-letter words we will echo the word, and then pipe it through netcat to our zookeeper note that this could be a remote instance in our case we will connect locally.

STAT show statistics

```
$ echo stat l nc localhost 2181
Zookeeper version: 3.3.3-1073969, built on 02/23/2011 22:27 GMT
Clients:
 /0:0:0:0:0:0:0:1%0:51486[0](queued=0,recved=1,sent=0)

Latency min/avg/max: 0/0/0
Received: 6
Sent: 5
Outstanding: 0
Zxid: 0x0
Mode: standalone
Node count: 4
```

DUMP shows session and ephemeral nodes

```
$ echo dump | nc localhost 2181
SessionTracker dump:
Session Sets (0):
ephemeral nodes dump:
Sessions with Ephemerals (0):
```

ENVI shows the environment

```
$ echo envi l nc localhost 2181
Environment:
zookeeper.version=3.3.3-1073969, built on 02/23/2011 22:27 GMT
host.name=172.16.1.220
java.version=1.6.0_17
java.vendor=Apple Inc.
java.home=/System/Library/Frameworks/JavaVM.framework/Versions/1.6.0/Home
java.class.path=bin/../build/classes:bin/../build/lib/*.jar:bin/../
zookeeper-3.3.3.jar:bin/../lib/log4j-1.2.15.jar:bin/../lib/jline-0.9.94.jar:bin/../src/java/lib/
*.jar:bin/../conf:
java.library.path=.:/Library/Java/Extensions:/System/Library/Java/Extensions:/usr/lib/
java
java.io.tmpdir=/var/folders/w4/w4mbu0k8FlG9o1hKAIOarU+++TI/-Tmp-/
java.compiler=<NA>
os.name=Mac OS X
os.arch=x86_64
os.version=10.6.4
user.name=thanlon
user.home=/Users/thanlon
user.dir=/Users/thanlon/zookeeper/zookeeper-3.3.3
```

See the documentation for further details regarding 4 letter words.

For the rest of the demonstration you should connect to zookeeper with the Command line tool.
When you run this command you will get a bunch of INFO level information, check to verify that you do not have WARN or more severe messages.

```
$ bin/zkCli.sh
```

Run "ls /" to get information of current 'nodes'

```
[zk: localhost:2181(CONNECTED) 1] ls /
[zookeeper]
```

You see that Zookeeper uses Zookeeper to manage Zookeeper. Nepotism to be sure, but it makes sense here.

Launch another terminal and connect to Zookeeper with the CLI tool, keep both windows open.

Create a node that indicates that a service is starting and child services can begin starting.

[zk:...] create /webapp 'starting ' null

* note the space after starting I think there might be issues with the CLI,

We are creating a znode with the data of "starting" and an Access Control List of null. So anyone can view this node, anyone can create child nodes and anyone can extract the data piece from this node.

Clean up the CLI artifacts..

```
set /webapp starting  cZxid = 0x52
ctime = Wed Jul 27 20:32:18 EDT 2011
mZxid = 0x54
mtime = Wed Jul 27 20:37:29 EDT 2011
pZxid = 0x52
cversion = 0
dataVersion = 2
aclVersion = 0
....
```

Do a get to confirm..

```
get /webapp
```

Perhaps our webapp is not running unless the backend DB is up. And perhaps there is a DBmaster and a DBslave. So from the other window create those.

```
zk: localhost:2181(CONNECTED) 10] create /webapp/db_master "master " null
```

Suppose the DB servers are peers, and the first to start is instructed to be the master, and the second is instructed to be the slave.

So perhaps the second one tries to grab the master role..

```
create /webapp/db_master "master " null
Node already exists: /webapp/db_master
```

The node creation would fail and the server would then register as the slave.

```
create /webapp/dbslave "master " null
Created /webapp/dbslave
```

Once our webapp has the services it needs to run, it can register as running, or perhaps put more of a payload in the data piece and note the IP address.

```
set /webapp "192.168.1.23"
```

```
get /webapp
"192.168.1.23"
```

End of DEMO