ZooKeeper

the unsung hero

Thomas Koch

June 6, 2011

me

Thomas Koch
`http://www.koch.ro`
finished music, physics
5 years software developer: PHP (RIP!), Java, Hadoop, Search, Crawling, ERP, Groupware
currently: finishing computer science bachelor, ETA: Q1/2012
tags: quality, ATTAC, FSFE, FIfF, social responsibility, Romania, Switzerland

Outline

what does it look like?

- file system of zNodes
- zNode is a file (max. 1MB)
- zNode if a folder
- watches / notifications
- atomic operations
- optimistic locking (changes provide last seen version number)

zNode flags

ephemeral: removed when client disappears
sequential: append incremental number

zNode flags

ephemeral: removed when client disappears
sequential: append incremental number

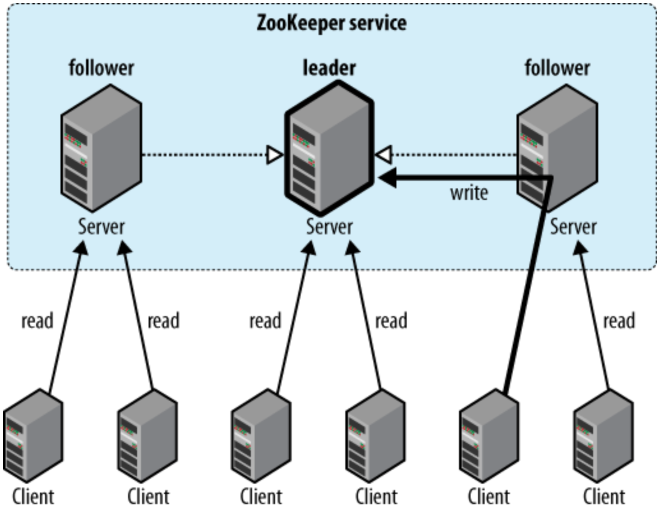
create("/my-znode-",SEQUENTIAL) $\rightarrow$ /my-znode-00000001

zNode flags

ephemeral: removed when client disappears
sequential: append incremental number

create("/my-znode-",SEQUENTIAL) → /my-znode-00000001
create("/my-znode-",SEQUENTIAL) → /my-znode-00000002

ZooKeeper architecture



ZooKeeper guaranties

- Sequential Consistency
- Atomicity (A from ACID)
- Single System Image
- Reliability (Durability from ACID)
- Timeliness
- No Split Brain

why?

- simple, powerful building blocks for distributed algorithms
- **one** place for coordination (instead of a jungle of cron jobs / daemons / lock files)

standard algorithms

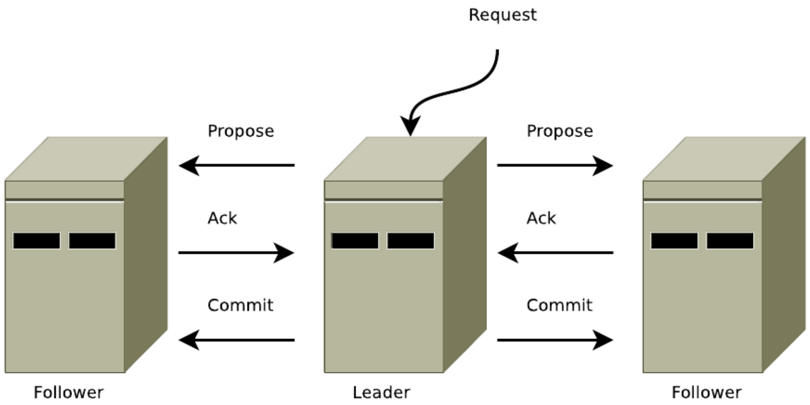
- distributed lock (with wait queue)
- leader election
- configuration store
- rendezvous, group membership
- (low scale) producer consumer queue

Outline

ZAB - ZooKeeper Atomic Broadcast

two modes: recovery / broadcast

ZAB: Broadcast mode



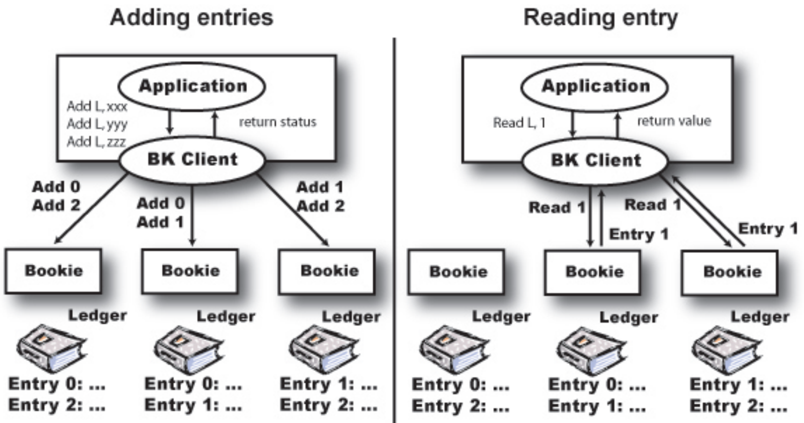
ZAB: Recovery (Leader election)

- elect member with highest zxid
- votes from more then $\frac{n}{2}$ servers
- increment epoch (zxid: $\underbrace{00000011}_{\text{epoch}} : \underbrace{00004213}_{\text{counter}}$)
- replay all not yet committed proposals

Outline

bookkeeper

- originally intended for the Hadoop NameNode write-ahead-log
- high availability, and high throughput
- ledgers readable only after close



Hedwid (Yahoo), Kafka (LinkedIn)

publish-subscribe systems
Hedwig:

- strong durability guarantees
- many topics
- C++

Kafka:

- many subscribers and publishers
- replay already consumed messages
- Scala

Hedwid (Yahoo), Kafka (LinkedIn)

publish-subscribe systems
Hedwig:

- strong durability guarantees
- many topics
- C++

Kafka:

- many subscribers and publishers
- **replay already consumed messages**
- Scala

more users[1] (free software only)

- Eclipse Communication Framework
- Katta (distributed Lucene indexes)
- HBase: master election, server lease management, bootstrapping
- Norbert, LinkedIn, partitioned routing, cluster management
- Mesos, cluster computing management platform
- Neo4j, graph database
- S4, Yahoo, stream processing
- Apache CXF distributed OSGi: discovery
- Apache Solr: configuration, leader election
- Hadoop MapReduce 2.0?

[1]cwiki.apache.org/confluence/display/ZOOKEEPER/PoweredBy

interesting development

- ZOOKEEPER-892 Remote replication of Zookeeper data
- Extracting Zab from Zookeeper[2] - André Oriani

[2] `https://github.com/aoriani/zab`

Outline

Praise

- right balance: functionality vs. usability
- turn key ready server
- bindings to other languages
- large user base
- proven scalability

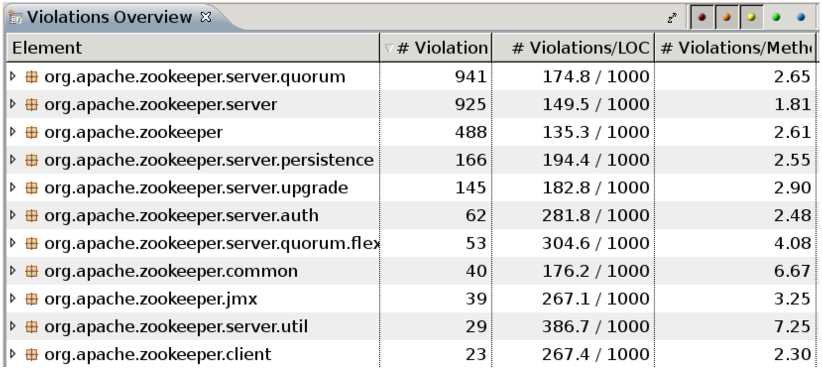
Comparison: JGroups

- Bela Ban (JBoss, Kreuzlingen), 1998
- "toolkit for reliable multicast communication"
- highly complex, highly powerful
- ISIS $\rightarrow$ Horrus $\rightarrow$ Ensemble $\rightarrow$ JGroups
- presumably missing: $\frac{n}{2}+1$ commits, changeable leader election
- failure detection, dynamic groups, auto discovery

google://JGroups ZooKeeper !
others like JGroups[3]: Appia, Spread

[3] `http://jgcs.sourceforge.net/implementations/`

PMDs opinion on ZooKeeper



Element	# Violation	# Violations/LOC	# Violations/Meth
org.apache.zookeeper.server.quorum	941	174.8 / 1000	2.65
org.apache.zookeeper.server	925	149.5 / 1000	1.81
org.apache.zookeeper	488	135.3 / 1000	2.61
org.apache.zookeeper.server.persistence	166	194.4 / 1000	2.55
org.apache.zookeeper.server.upgrade	145	182.8 / 1000	2.90
org.apache.zookeeper.server.auth	62	281.8 / 1000	2.48
org.apache.zookeeper.server.quorum.flex	53	304.6 / 1000	4.08
org.apache.zookeeper.common	40	176.2 / 1000	6.67
org.apache.zookeeper.jmx	39	267.1 / 1000	3.25
org.apache.zookeeper.server.util	29	386.7 / 1000	7.25
org.apache.zookeeper.client	23	267.4 / 1000	2.30

my favourite PMD violations: NPath complexity

possible paths through the method

ClientCnxn.run() NPath complexity of 314

my favourite PMD violations: NPath complexity

possible paths through the method

ClientCnxn.run() NPath complexity of 314
.onConnected() NPath complexity of 750

my favourite PMD violations: NPath complexity

possible paths through the method

ClientCnxn.run() NPath complexity of 314
.onConnected() NPath complexity of 750
.readResponse() NPath complexity of 9750

my favourite PMD violations: cyclomatic complexity

branching points

ClientCnxn.processEvent() Cyclomatic complexity 26

other favourite PMD violations

- Assigning an Object to null is a code smell. Consider refactoring.
- Avoid really long methods.
- This class has too many methods, consider refactoring it.
- A high ratio of statements to labels in a switch statement. Consider refactoring.
- Avoid empty catch blocks.
- Avoid really long parameter lists.
- (Class has) Too many fields.

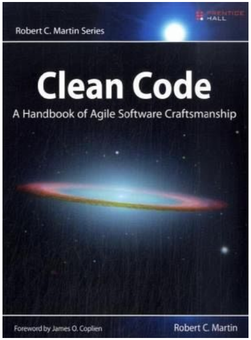
other favourite PMD violations

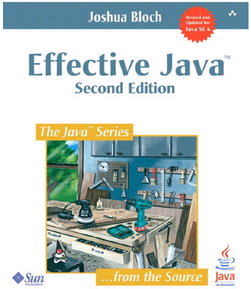
- Assigning an Object to null is a code smell. **Consider refactoring.**
- Avoid really long methods.
- This class has too many methods, **consider refactoring** it.
- A high ratio of statements to labels in a switch statement. **Consider refactoring**.
- Avoid empty catch blocks.
- Avoid really long parameter lists.
- (Class has) Too many fields.

ZooKeeper Devs on refactoring

code quality is important, and there are things we should keep in mind, but in general i really don't like the idea of risking code breakage because of a gratuitous code cleanup. ...

vs.


Robert C. Martin Series
PRENTICE HALL
Clean Code
A Handbook of Agile Software Craftsmanship
Foreword by James O. Coplien
Robert C. Martin



BIG BALL OF MUD[4]

- tight coupling
- therefor no units testable in isolation
- therefor most *unit* tests are actually *acceptance* tests

[4] `http://www.laputan.org/mud/mud.html`

copy 'n paste programming

. . . leads to code duplication
e.g ZOOKEEPER-911 removes 162 lines of duplicate code

client API

your experience?
e.g. ZkClient, cages
no client only jar

feature bloat

- chroot
- automatic event re-subscription
- chroot + automatic event re-subscription = broken
- chroot not fully transparent (ZOOKEEPER-1027)
- multi-update command in the works since half a year

horrible concurrency

XYZ extends Thread

instead of

- implements runnable
- or better: executor framework
- or much better: actors (e.g. Akka)

QA

Live is too short for crap.

`http://www.koch.ro`
`http://identi.ca/thkoch`
`thomas@koch.ro`

QA

Live is short. Strive for Excellence in Everything You Do!

http://www.koch.ro

http://identi.ca/thkoch

thomas@koch.ro

classification

Google		free software
GFS	$\rightarrow$	HDFS (KFS, . . .)
MapReduce	$\rightarrow$	Hadoop MapReduce
BigTable	$\rightarrow$	HBase (Hypertable)
Chubby	$\neq$	ZooKeeper

another distributed file system?

HDFS	ZooKeeper
big files	small “cookies”
dumb	watches, sequential, ephemeral
streaming	random access
	ordered

ZooKeeper API

- String create (path, data, acl, ephemeral|sequential)
- void delete (path, expected Version)
- Stat setData (path, data, expected Version)
- byte[] getData (path, watch)
- Stat exists (path, watch)
- String[] getChildren (path, watch)
- void sync (path)