Debugging Distributed Systems

Donny Nadolny
donny@pagerduty.com
SREcon16
What is ZooKeeper

• Distributed system for building distributed systems
• Small in-memory filesystem
ZooKeeper at PagerDuty

- Distributed locking
- Consistent, highly available
The Failure

- Network trouble, one follower falls behind
- ZooKeeper gets stuck - leader still up
Fault Injection Finding

- Leader logs: “Too busy to snap, skipping”
Fault Injection Finding

- Leader logs: “Too busy to snap, skipping”
- Disk slow? let’s test:
  - `sshfs donny@some_server:/home/donny /mnt`
- Similar failure profile
Fault Injection Finding

- Leader logs: “Too busy to snap, skipping”
- Disk slow? let’s test:
  - sshfs donny@some_server:/home/donny /mnt
- Similar failure profile
- Re-examine disk latency... nope, was a red herring
Deep Health Checks

- First warning: application monitoring
- ZooKeeper: used `ruok`
- Added deep health check
"LearnerHandler-/123.45.67.89:45874" prio=10 tid=0x000000000024bb800 nid=0x3d0d runnable [0x00007fe6c3193000]
java.lang.Thread.State: RUNNABLE

1. at org.apache.zookeeper.server.ZKDatabase.serializeSnapshot(ZKDatabase.java:467)
at org.apache.zookeeper.server.quorum.LearnerHandler.run(LearnerHandler.java:493)

2. - locked <0x0000000d4cd9e28> (a org.apache.zookeeper.server.DataNode)
at org.apache.zookeeper.server.DataTree.serializeNode(DataTree.java:1130)

3. at java.net.SocketOutputStream.socketWrite0(Native Method)
at java.net.SocketOutputStream.socketWrite(SocketOutputStream.java:113)
...
Write Snapshot Code (simplified)

```java
void serializeNode(OutputArchive output, String path) {
    DataNode node = getNode(path);
    String[] children = {};
    synchronized (node) {
        output.writeString(path, "path");
        output.writeRecord(node, "node");
        children = node.getChildren();
    }
    for (String child : children) {
        serializeNode(output, path + "/" + child);
    }
}
```
ZooKeeper Heartbeat

• Why didn’t a follower take over?
• ZK heartbeat: message from leader to follower, follower times out
TCP
TCP Data Transmission

Follower: ESTABLISHED → ... SYN, SYN-ACK, ACK ... → ESTABLISHED

Packet 1

Leader: ESTABLISHED

ACK
TCP Data Transmission

Follower

Leader

Packet 1
~200ms
Packet 1
~200ms
Packet 1
~400ms
Packet 1
~800ms
Packet 1
~120sec
Packet 1
~120sec

CLOSED

15 retries

- ESTABLISHED
- ESTABLISHED

~200ms
~400ms
~800ms
~120sec
TCP Retransmission (Linux Defaults)

• Retransmission timeout (RTO) is based on latency
• TCP_RTO_MIN = 200 ms
• TCP_RTO_MAX = 2 minutes
• /proc/sys/net/ipv4/tcp_retries2 = 15 retries
• $0.2 + 0.2 + 0.4 + 0.8 + \ldots + 120 = 924.8$ seconds (15.5 mins)
TCP Close Connection

**Follower**
- ESTABLISHED
- FIN_WAIT1
- TIME_WAIT
- 60 seconds
- CLOSED

**Leader**
- ESTABLISHED
- FIN
- FIN/ACK
- LAST_ACK
- ACK
- CLOSED

60 seconds
TCP Close Connection

Follower

ESTABLISHED
FIN_WAIT1

8 retries

~1m40s
CLOSED

Leader

ESTABLISHED

FIN
FIN
FIN
FIN
FIN
FIN
FIN
FIN
TCP Close Connection

**Follower**
- ESTABLISHED
- FIN_WAIT1
- ~1m40s
- CLOSED

**Leader**
- ESTABLISHED
- FIN
- Packet 1
- ~15.5 mins
- CLOSED
TCP Close Connection

Follower

- ESTABLISHED
- FIN_WAIT1
- ~1m40s
- CLOSED

Leader

- ESTABLISHED
- Packet 1
- FIN
- RST
- CLOSED
syslog - Dropped Packets on Follower

- 06:51:47 iptables: WARN: IN=eth0 OUT= MAC=00:0d:12:34:56:78:12:34:56:78:12:34:56:78:12:34:56:78 SRC=<leader_ip> DST=<follower_ip> LEN=54 TOS=0x00 PREC=0x00 TTL=44 ID=36370 DF PROTO=TCP SPT=3888 DPT=36416 WINDOW=227 RES=0x00 ACK PSH URGP=0
iptables

iptables -A INPUT -m state --state ESTABLISHED,RELATED -j ACCEPT
iptables -A INPUT -p tcp --dport 80 -j ACCEPT
... more rules to accept connections ...
iptables -A INPUT -j DROP
iptables

iptables -A INPUT -m state --state ESTABLISHED,RELATED -j ACCEPT
iptables -A INPUT -p tcp --dport 80 -j ACCEPT
... more rules to accept connections ...

iptables -A INPUT -j DROP

But: iptables connections != netstat connections
conntrack Timeouts

- From linux/net/netfilter/nf_conntrack_proto_tcp.c:
- `[TCP_CONNTRACK_LAST_ACK] = 30 SECS`
TCP Close Connection

**Follower**
- `conntrack LAST_ACK`: 30s
- `kernel TCP FIN_WAIT1`: ~102.4s
- `FIN`: ~51.2s
- `CLOSED`

**Leader**
- `FIN`
- `FIN`
- `FIN`
- `FIN`
- `CLOSED`

**Duration**
- ~102.4s
- ~25.6s
- ~51.2s
The Full Story

- Packet loss
- Follower falls behind, requests snapshot
- (Packet loss continues) follower closes connection
- Follower conntrack forgets connection
- Leader now stuck for ~15 mins, even if network heals
Lessons
Lesson 1

• Don’t lock and block
• TCP can block for a really long time
• Interfaces / abstract methods make analysis harder
Lesson 2

- Leader/follower heartbeats should be deep health checks!
Questions?

donny@pagerduty.com

Link:
“Network issues can cause cluster to hang due to near-deadlock”
https://issues.apache.org/jira/browse/ZOOKEEPER-2201