Eventually Consistent Service Discovery

@suhailpatel - SRECon 2019
# semaphore

5:02 AM
acquire 1

6:14 AM
acquire 2

9:10 AM
acquire 3

9:23 AM
acquire 4

9:36 AM
acquire 5

10:09 AM
acquire 7

10:09 AM
release 7

10:36 AM
release 5

12:13 PM
acquire 5

1:50 PM
release 5

1:58 PM
back to the void with you, 5!

3:17 PM
Saving you from the void, 5

3:23 PM
Right back in there 5

6:29 PM
Slackbot 4:00 AM
Reminder: ALL SMARTCLIENTS RELEASED 🌟

acquire 1
release 1
acquire 1
acquire 2
acquire 3
release 3
acquire 4 (edited)
acquire 4 again
acquire 1
acquire 2
release 1
acquire 1
Eventually Consistent Service Discovery

@suhailpatel - SRECon 2019
Hi, I’m Suhail

I’m an Engineer at Monzo on the Platform squad. We help build the base so other engineers can ship their services and applications.

Email:  hi@suhailpatel.com
Twitter:  @suhailpatel
Service Discovery?
Service Discovery?

“foo” = “bar”

cassandra
Service Discovery?

```
foo"="bar"
```
Service Discovery?

```
10.0.0.1
  cassandra

10.0.0.2
  cassandra

10.0.0.3
  cassandra

10.0.0.4
  cassandra

10.0.0.5
  cassandra

10.0.0.6

"foo" = "bar"

"foo" = "bar"
"foo" = "bar"
"foo" = "bar"
```

"foo" = "bar"
Strong Consistency

https://etcd.io/
Strong Consistency

https://raft.github.io
Strong Consistency

https://raft.github.io
"foo"="bar"
Strong Consistency

service.account
10.0.10.123

service.transaction
Sequential Consistency
Sequential Consistency

Sequential Consistency

service.account 10.0.10.123

service.transaction
Summary

Remaining issues:
- Positive feedback loop
- Write cost depends on number of backend instances
- Read cost depends on write cost multiplied by number of clients
Gossip Protocols

“foo” = “bar”
Gossip Protocols

- service.account
  10.0.10.123
SWIM: Scalable Weakly-consistent Infection-style Process Group Membership Protocol

Abhinandan Das, Indranil Gupta, Ashish Motivala
Dept. of Computer Science, Cornell University
Ithaca NY 14853 USA
{asdas,gupta,ashish}@cs.cornell.edu

Abstract
Several distributed peer-to-peer applications require weakly-consistent knowledge of process group membership information at all participating processes. SWIM is a generic software module that offers this service for large-scale process groups. The SWIM effort is motivated by the unscalability of traditional heart-beating protocols, which either impose network loads that grow quadratically with group size, or compromise response times or false positive frequency w.r.t. detecting process crashes. This paper re-

1. Introduction
As you swim lazily through the milieu,
The secrets of the world will infect you.

Several large-scale peer-to-peer distributed process groups running over the Internet rely on a distributed membership maintenance sub-system. Examples of existing middleware systems that utilize a membership protocol include reliable multicast [3, 11], and epidemic-style information dissemination [4, 8, 13]. These protocols in turn find use in applications such as distributed databases that need to reconcile re-
(Scalable) Gossip Protocols

N1: PING N2, are you alive?

N1: Also, I’m representing service.account so put it in your state
(Scalable) Gossip Protocols

N2: ACK, I am alive!
N1: I suspect N3 is dead because it hasn’t sent a heartbeat in a while, can you do an INDIRECT PING?
(Scalable) Gossip Protocols

N3

N2: Hi N1, yes, N3 is dead for me too
Your suspicion was correct

N1
Eventual Consistency
Eventual Consistency
Eventual Consistency
Summary

Need Agreement?
- A strongly consistent system may be ideal for this use case

Scalability?
- Eventual consistency will work as long as you acknowledge in your applications that it’s not always perfect
Thanks!

Email:  hi@suhailpatel.com
Twitter:  @suhailpatel