Apache Ratis
In Search of a Usable Raft Library

Tsz-Wo Nicholas Sze

11/15/2017

Brown Bag Lunch Talk

About Me

- **Tsz-Wo Nicholas Sze, Ph.D.**
  - Software Engineer at Hortonworks

  - PMC member/Committer of Apache Hadoop
  - Active contributor and committer of Apache Ratis

  - Ph.D. from University of Maryland, College Park

  - MPhil & BEng from Hong Kong University of Sci & Tech

Agenda

- **A brief introduction of RAFT**
- **Apache Ratis**
  - Features and use cases
  - Demo
- **Project Status**
  - Current development
  - Future works

Consensus

- **What is consensus?**
  - Multiple servers to agree a value

- **Typical use cases**
  - Log replication
  - Replicated state machines

Consensus Algorithms

- **Paxos (1990)**
  - Work but hard to understand
  - Hard to implement (correctly)

- **Raft (2014)**
- **“In Search of an Understandable Consensus Algorithm”**
  - by Diego Ongaro and John Ousterhout
  - USENIX ATC’14, https://raft.github.io

- **Motivations**
  - Easy to understand
  - Easy to prove
  - Easy to implement

Raft Basic

- **Leader Election**
  - Servers are started as a *Follower*
  - Randomly timeout to become *Candidate* and start a leader election
    - *Candidate* sends requestVote to other servers
  - It becomes the leader once it gets a majority of the votes.

- **Append Entries**
  - Clients send requests to the *Leader*
  - *Leader* forwards the requests to the *Followers*
    - *Leader* sends appendEntries to *Followers*
  - When there is no client requests, *Leader* also sends empty appendEntries to *Followers* to maintain leadership

Raft Library

- **Our Motations**
  - Use Raft in Ozone

- **“In Search of a Usable Raft Library”**
  - A long list of Raft implementations is available
  - None of them a general library ready to be consumed by other projects.
  - Most of them are tied to another project or a part of another project.

- **We need a Raft library!**

Apache Ratis – A Raft Library

- **A brand new, incubating Apache project**
  - Open source, open development
  - Apache License 2.0
  - Written in Java 8

- **Contributions are welcome!**

```
CompletableFuture<?> contributeToApacheRatis() {
  final CompletableFuture<?> yourFuture =
      you.submit(yourIdeas -> new Patch(yourIdeas))
          .thenAccept(invitation -> new Committer(you, invitation));
  return yourFuture;
}
```

Ratis: Standard Raft Features

- **Leader Election + Log Replication**
  - Automatically elect a leader among the servers in a Raft group
  - Randomized timeout for avoiding split votes
  - Log is replicated in the Raft group

- **Membership Changes**
  - Members in a Raft group can be re-configurated in runtime
  - Replication factor can be changed in runtime

- **Log Compaction**
  - Snapshot is taken periodically
  - Send snapshot instead of a long log history.

Ratis: Pluggability

- **Pluggable state machine**
  - Application must define its state machine
  - Example: a key-value map
- **Pluggable RPC**
  - Users may provide their own RPC implementation
  - Default implementations: gRPC, Netty, Hadoop RPC
- **Pluggable Raft log**
  - Users may provide their own log implementation
  - The default implementation stores log in local files

Data Intensive Applications

- **In Raft,**
  - All transactions and the data are written in the log
  - Not suitable for data intensive applications

- **In Ratis**
  - Application could choose to not written all the data to log
  - State machine data and log data can be separately managed

  - See the *FileStore* example in ratis-example

Ratis: Asynchronous APIs

- **Using gRPC bi-directional stream API**
  - Netty and Hadoop RPC can support async but not yet implementated
- **Server-to-server**
  - Asynchronous append entires
- **Client-to-server**
  - Asynchronous client requests
  - RATIS-113: just committed yesterday (11/14/2017)
  - Need to fix out-of-order issues RATIS-140 and RATIS-141

General Ratis Use Cases

- **You already have a service running on a single server.**
- **You want to:**
  - (1) replicate the server log/states to multiple machines
    - The replication number/cluster membership can be changed in runtime
    - It can tolerate server failures.
- **or**
  - (2) have a HA (highly available) service
    - When a server fails, another server will automatically take over.
    - Clients automatically failover to the new server.
- **Apache Ratis is for you!**

Ozone/HDFS Use Cases

- **Replicating open containers**
  - HDFS-11519
    - Ozone: Implement XceiverServerSpi and XceiverClientSpi using Ratis
    - Committed on 4/3/2017
- **Support HA in SCM**
  - HDFS-11443
    - Ozone: Support SCM multiple instances for HA
- **Replacing the current Namenode HA solution**
  - No JIRA yet

Example: ArithmeticStateMachine

- **Maintain a variable map (variable -> value)**
  - Users define the variables and expresions

```
final Variable a = new Variable("a");
final Variable b = new Variable("b");
final Variable c = new Variable("c");
final AssignmentMessage pythagorean = c.assign(
    SQRT.apply(ADD.apply(SQUARE.apply(a), SQUARE.apply(b))));
```

$$c = \sqrt{a^2 + b^2}$$

  - Then, submit assignment messages

```
r = client.send(a.assign(3));
r = client.send(b.assign(4));
client.send(pythagorean);  // reply c = 5.
```

Demo: ArithmeticStateMachine

- **Pythagorean**
  - Put a = 3 and b = 4
  - Find c

$$c = \sqrt{a^2 + b^2}$$

- **Special thanks to Marton Elek for working on RATIS-95 (CLI to run examples) so that this demo is possible!**

Gauss-Legendre

- **A.k.a the arithmetic–geometric mean method**

$$
\begin{aligned}
a_0 &= 1, & a_{n+1} &= \frac{a_n + b_n}{2}, && \text{(arithmetic mean)} \\
b_0 &= \frac{1}{\sqrt{2}}, & b_{n+1} &= \sqrt{a_n b_n}, && \text{(geometric mean)} \\
t_0 &= \frac{1}{4}, & t_{n+1} &= t_n - p_n (a_n - a_{n+1})^2, \\
p_0 &= 1, & p_{n+1} &= 2p_n.
\end{aligned}
$$

- **A fast algorithm to compute pi**

$$\pi \approx \frac{a_{n+2}^2}{t_{n+1}}.$$

Example: FileStore (RATIS-122)

- **Maintain a file map (key -> file)**
- **Support only**
  - Read, Write, Delete
- **But not other operations such as**
  - List, Rename, etc.

- **Asynchronous & In-order**
  - Client may submit multiple write requests to
    - Write to multiple files at the same time
    - Each file may have multiple write requests

- **File data is managed by the state machine**
  - But not store in the raft log

Ratis: Development Status

- **A brief history**
  - 2016-03: Project started at Hortonworks
  - 2016-04: First commit “leader election (without tests)”
  - 2017-01: Entered Apache incubation.
  - 2017-03: Started preparing the first Alpha release (RATIS-53).
  - 2017-04: Hadoop Ozone branch started using Ratis (HDFS-11519)!
  - 2017-05: First Release 0.1.0-alpha
  - 2017-11: Preparing the second Release 0.2.0-alpha

Work in Progress

- **Multi-Raft**
  - General idea: Allow a server to join multiple Raft groups (RATIS-91)
    - When the #groups is large, it is hard to manage the logs
    - May assume SSD for log storage
  - Short term goal: Allow a server to join a small number of groups
    - “Small” means 2 or 3
    - Use a small number of pre-configurated storage locations
    - Big benefit: servers could transition from one group to another group
- **Replicated Map (RATIS-40)**
  - A sorted map similar to ZooKeeper/etcd/LogCabin
  - Intended to be used in production

Ratis: Future Works

- **Performance**
  - Fix gRPC async bugs: RATIS-140 and RATIS-141
  - Use FileStore as benchmarks
- **Metrics**
- **Security**
- **API Specification**
- **Documentations**
  - Project web site
- **Jerkins Builds**
  - Checkstyle configuration

Contributors

- Animesh Trivedi, Anu Engineer, Arpit Agarwal, Brent,
- Chen Liang, Chris Nauroth, Devaraj Das, Enis Soztutar,
- garvit, Hanisha Koneru, Hugo Louro, Jakob Homan,
- Jian He, Jing Chen, Jing Zhao, Jitendra Pandey, Junping Du,
- kaiyangzhang, Karl Heinz Marbaise, Li Lu, Lokesh Jain,
- Marton Elek, Mayank Bansal, Mingliang Liu,
- Mukul Kumar Singh, Sen Zhang, Sriharsha Chintalapani,
- Tsz Wo Nicholas Sze, Uma Maheswara Rao G,
- Venkat Ranganathan, Wangda Tan, Weiqing Yang,
- Will Xu, Xiaobing Zhou, Xiaoyu Yao, Yubo Xu, yue liu,
- Zhiyuan Yang

Contributions are welcome!

- http://incubator.apache.org/projects/ratis.html
- dev@ratis.incubator.apache.org

```
CompletableFuture<?> contributeToApacheRatis() {
  final CompletableFuture<?> yourFuture =
      you.submit(yourIdeas -> new Patch(yourIdeas))
          .thenAccept(invitation -> new Committer(you, invitation));
  return yourFuture;
}
```

Java question: What is <?> in above? 😀

Thank you!