Raft User Study

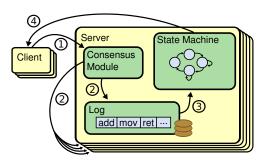
Diego Ongaro & John Ousterhout Stanford University SEDCL Retreat

June 6, 2013

Intro

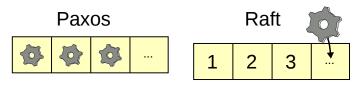
- Last year's retreat: talk on Paxos
- John started a competing algorithm
- Designed Raft to be easier to understand
- Our reviewers didn't believe us
- Conducted an experiment to demonstrate that Raft is easier to understand than Paxos

Context: replicated state machines



- State machine defines data structure
 - Interface is application-specific
- Replicated log feeds commands to state machine
- ▶ Same log \Rightarrow same sequence of states, outputs
- Raft and Multi-Paxos are two consensus algorithms to manage the replicated log

1. Serial operation



- Basic Paxos defines consensus on just one log entry
- Multi-Paxos forms a log and optimizes across entries
- Each log entry can proceed concurrently
- What's the advantage of concurrent operation?
 - Ultimately the state machine must consume entries serially
- Raft appends entries to the log in order

2. Strong leader

- Raft first elects a cluster leader
 - Only the leader appends to the replicated log
 - Inconsistencies arise only on leader changes
- Basic Paxos is symmetric (p2p)
- Multi-Paxos introduces a leader as an optimization

A few tips from Scott

- Don't just ask people their opinion: measure it
- Record the lectures
- Pilot everything twice
- Doing Psychology Experiments by David W. Martin

From the participants' view

- Participants: undergrad and grad students
 - Stanford's Advanced OS class: 32 (5% participation grade)
 - Berkeley's Distributed Computing class: 16 (obvious bluff)
- Get randomly assigned to group
- Log onto web site, watch 1 hr Paxos (Raft) video, take 1 hr Paxos (Raft) quiz
- sleep $(60 \times 60 \times 24 \times 5)$
- ► Log onto web site, watch 1 hr Raft (Paxos) video, take 1 hr Raft (Paxos) quiz
- Take short survey

Lecture challenges

- Same lecturer or expert on each algorithm?
- Which Paxos do we teach? How much do we improve it?
- What material do we include?

Paxos lecture:

Implementing Replicated Logs with Paxos

Ousterhout and Diego Ongaro



Strawman: Single Acceptor Problem: Split Votes

. Simola (incorract) approach Acceptor accepts only first value it receives? a single acceptor chooses What if acceptor crashes amplied amphibid Solution: quorum Till Assessed amplified ample(tim)

Acceptors must sometimes accept multiple (different) **Basic Paxos Examples**

market (ma)

Three possibilities when later proposal prepares:

Multi-Paxos Issues

Full Disclosure

. Solution part 1/4: keep retrying Accept RPCs until all

Configuration Changes, cont'd

Log entities not fully replicated (majority only) Clost full replication

acceptors respond (in background)

Solution part 2/4: track chosen entries

. Which log entry to use for a given client request

m-5

• Clean number of

Configuration changes

Performance optimizations:

 Elimonte mont Presare requesto Ensuring full replication

employed appealment

2. Previous value not chosen, but new proposer sees it 4 (631) · (EII (EIII (EIII) (EIII (EIII) (EIIII) (EIII) (EIII) (EIII) (EIII) (EIII) (EIII) (EIII) (EIII) (EIIII) (EIII) (EIII) (EIII) (EIII) (EIII) (EIII) (EIII) (EIII) (EIIII) (EIII) (EIII) (EIII) (EIII) (EIII) (EIII) (EIII) (EIII) (EIIII) (EIII) (EIII) (EIII) (EIII) (EIII) (EIII) (EIII) (EIII) (EIIII) (EIII) (EIII) (EIII) (EIII) (EIII) (EIII) (EIII) (EIII) (EIIII) (EIII) (EIII) (EIII) (EIII) (EIII) (EIII) (EIII) (EIII) (EIIII) (EIII) (EIII) (EIII) (EIII) (EIII) (EIII) (EIII) (EIII) (EIIII) (EIII) (EIII) (EIII) (EIII) (EIII) (EIII) (EIII) (EIII) (EIIII) (EIII) (EIII) (EIII) (EIII) (EIII) (EIII) (EIII) (EIII) (EIIII) (EIII) (EIII) (EIII) (EIII) (EIIII) (EIIII) (EIII) (EIIII) (E m-5

Selecting Log Entries

Goal: Replicated Log

RESERVE

♠ ♠ ♠

Consensus module ensures orsser los reolication

Acceptor accepts every value it receives?

(i---

Could choose multiple values

and the same

. Fallers model fathers (not Experient, delayed but messages

Problem: Conflicting Choices

Basic Paxos Examples, cont'd

Three possibilities when later proposal prepares:

 When request arrives from client: Run Basis Places to propose client's command for this index

Full Disclosure, cont'd

. Solution part 34: proposer tells acceptors about

Configuration Changes, cont'd . Paxos solution: use the log to manage configuration

erisi i i i i i i i i

The Paxos Approach Decompose the problem

Conflicting Choices, cont'd

-

-

. s. needn't propose red (it hasn't been chosen vet . s,'s proposal must be aborted (s, must reject it)

Basic Paxos Examples, cont'd

Selecting Log Entries, cont'd

. Must apply commands to state machine in log order

Full Disclosure, cont'd

Pproposer's tradition/scentrales = tradition/scentrales from response, then proposer sends (Suppose RPC III) background

Successfindex, vi: notifies acceptor of chosen entry;

Servers can handle multiple client requests

Three possibilities when later proposal prepares:

New proposer chooses its own value

4 (731)

1. Previous value not chosen, new proposer doesn't

a married (mg

. Basic Payos ("sinola decree") Distrace value is everythoses

 Liveness (as long as majority of servers up and communicating with reasonable timelinessi: Condone several redances of Basic Places to agree on a series.
 of values forming the loc

The term "consensus problem" typically refers to this

. Only a single value may be chosen. . A server never learns that a value has been chosen unless to

Proposal Numbers

Requirements for Basic Paxos

 Each proposal has a unique number It must be possible for a proposer to choose a new proposal number higher than anything It has seen used before One simple approach: Proposal Standar Standarder General . Each server stones manfound, the targest Round Number II has

a Increment machineral proposal numbers after crash/restart

Liveness . Competing proposers can livelock



. Multi-Paxos will use leader election instead

Improving Efficiency Usino Basic Paxos is inefficient:

 With multiple concurrent proposers, combins and restarts are there triples tool — more conflicts: 1 Dirk a leader

 N any given time, only one server acts as Propose 2. Eliminate most Prepare RPCs.

Client Protocol

 Foundated senser not leader, if will redired to leader Leader does not respond until command has been chosen for log entry and executed by leader's state

 If request times out (e.g., leader crash): . Client recours command to some other server

Retry request with new teader

Paxos Summary

- David Prepar Multi-Paxos Choosing tog erities

Client protocol

Paxos Components

Author and furth contrader values to be shown

For this presentation:

. Phase 1: broadcast Prepare RPCs Find out about any shoren values . Shops It broadcast Accord SSCs

Multi-Paxos

Separate instance of Basic Paxos for each entry in

. Add Index argument to Prepare and Assept (selects entry in log

· Multiple acceptors (3, 5, ...)

Value vis choses if accepted by

Other Notes with their own proposal

. If server not leader

Client Protocol, cont'd

. What if leader crashes after executing command but

Solution: client embeds a unique id in each

. Desuit assettunence sementics as been as client

before responding?

ommand

Elever includes at in log entry

Elste machine records mod re

Leader Election **Eliminating Prepares** . Let the server with highest ID act as leader

. Each server sends a heartbeat message to every other server every T ms . If a server hasn't received heartheat from server with higher ID in last 2T ms. it acts as leader:

. If acceptor responds to Prepare with noMoreAccepted, skip future Prepares with that Once leader receives noMoreAccepted from majority

of acceptors, no need for Pregare RPCs.

configuration

Configuration Changes

During configuration changes, it must not be possible for different manufact to change different up on the finance. 00000

Raft lecture:

Raft: A Consensus Algorithm for Replicated Logs

Diego Ongaro and John Ousterhou





Heartbeats and Timeouts

. Followers expect to receive RPCs from leaders or

AppendEntries Consistency Check

Follower must contain matching entry; otherwise in

Implements an induction step, ensures coherency

New Commitment Rules

--- (10)

×

Miller Market

. For a leader to decide an

Must be stored on a majority of servers.

. Each AppendEntries RPC contains index, term of

AppendEntries RPCs) to maintain authority

. If electionTimeout elapses with no RPCs:

. Servera start up as followers

. Summarrie landardese · Asymmetric, leader-based: · Clerk communities with the trade

Approaches to Consensus Two general approaches to consensu

Election Basics

1. Leader election:

. Select one of the servers to act as hooler 1. Safety and consistency after leader changes 4. Neutralizing old leaders. 5. Client interactions

Raft Overview

Addressed removes server

Elections, cont'd

 Change to Candidate state . Send Requestions SSCs to all other sequery retry == 00000=

Safety Requirement

Leader Changes

Once a log entry has been applied to a state machine · At beginning of new leader's term: · Raft safety property: Ittl eventually make follower's logs identical to leader's. Multiple crashes can leave many extraneous tog entities

If a leader has decided that a tig entry is committed, that entry will be present in the loss of all future headers. . This guarantees the safety requirement.

00000 . Leaders never overwrite entires in their tops . Entires must be committed before applying to state machine



Repairing Follower Logs Leader changes can result in log inconsistencies: . Leader Resus restindes for each follows ----When Appendituries or neutroles and by again - BEB - - E E BB B



. Consensus mechanism must support changes in the configuration:

Configuration Changes, cont'd

another: conflicting majorities could arise

Conver States . At any given time, each server is either

. Leader handles all altert interactions, log replication Published completely passive (sesses no RPCs, responds to Normal operation: 1 leader, N-1 followers

Log Structure

MMMAMM

Log stored on states starage (disk), survives crashes

Picking the Best Leader

Voling senior V denies vole filts tog is "more complete" (section, » textform,) [(section, » section),) [

 Time divided into terms: . Each server maintains current term value. . Key role of terms: identify obsolete information

. Client sends command to leader

Crashedislow followers?

. Leader appends command to its loo

· Performance is optimal in common case

Landar sands AnnandEntries DDCs to tollower

Leader notifies billowers of committed entires in subsequent Ascendibilities RPCs

Committing Entry from Current Term

. Case #1/2: Leader decides entry in current term is

Terms





. If log entries on different servers have same index



Committing Entry from Earlier Term

. Case #2/2: Leader is trying to finish committing entry ____ teacher for · THE . Entry 2 not safely committed

Client Protocol

logged, committed, and executed by leader's state

 If contacted server not leader, if will redirect to leader. . Leader does not respond until command has been

. If request times out is o. leader crash)

Netry request with new teader.

. Client residues command to some other server



Once entry 4 committed: Times

Client Protocol, cont'd What if leader crashes after executing command, but

before responding? . Solution: client embeds a unique id in each

 Sever includes at in tag every Betare accepting command, leader shecks its tag for entry with . If it found in big. sprore new command, return response from oil

Result: exactly-once semantics as long as client doesn't crash

Repairing Logs, cont'd - When follower overwrites inconsistent entry, it deletes all subsequent entries:

. Balt upon a Lobana soccoach

C_{strong} entry surrenized

--------Manager patients | | | | | | | | | | | |

Tolot Consensus

Deposed leader may not be dead

Other servers elect a new leader

. Election undates terms of majority of servers

. Safe: leader for term 3 must contain entry

Neutralizing Old Leaders . Terms used to detect stale leaders (and candidates)

Encoura and updates its term
 Encourar's term is older, it reverts to follower, updates its term,
then processes RPC normally.

Tolot Consensus, cont'd

Additional details

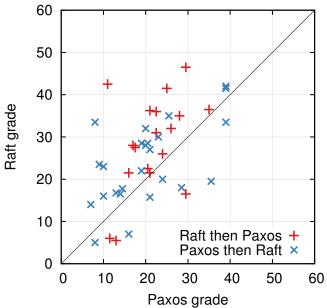
Raft Summary 1. Leader election

2. Normal operation 3. Safety and consistency 4. Neutralize old leaders 5. Client protocol Configuration changes

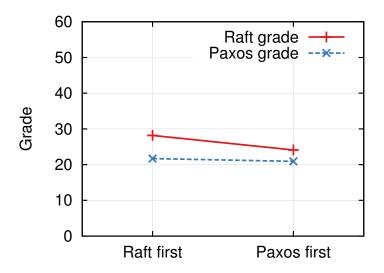
Quiz challenge: maintain equal difficulty

- Easy questions (4 points): warm-up
- Medium questions (26 points): apply algorithm
- Hard questions (30 points): not clear which algorithm to apply
- Paired question difficulty across exams

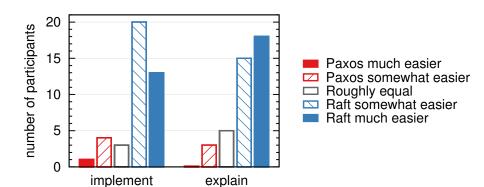
Quiz results



Ordering effects



Survey results



Recent popularity

A bunch of open-source Raft implementations:

```
Bloom 3
C++ 1
Erlang 4
F# 1
Go 2
Haskell 1
Java 2
```

Upcoming talks:

- StrangeLoop (Ben Johnson of go-raft, September)
- RICON West (Diego, October)

Conclusions

- Really hard to measure understandability
 - ▶ 99% of effort before getting any results
- Students averaged 23% better on Raft quiz
- Survey showed overwhelming support for Raft
- Recent academic and industrial interest is encouraging
- Under submission...

http://ramcloud.stanford.edu/raft.pdf

Acknowledgements

- Scott Klemmer for help with experimental design
- David Mazières and Ali Ghodsi for their guinea pigs
- "Volunteers" for pilots: Aleks, Amit, Ankita, Arjun, Daniel, David, and Vimal
- The 48 participants
- Nelson Ray for his stats help