1 Warm Up

Our objective is to design a scalable consensus protocol using LOT, RDMA, and Optical Networks. We are using ZooKeeper as our case study. A simple benchmark revealed that ZooKeeper is not scalable with the number of servers.

We changed ZooKeeper’s server-to-server communication from TCP to RDMA. With jVerbs and RDMA send/receive verbs, we are able to achieve an average round trip latency of 11 $\mu$s . As a result, ZooKeeper’s throughput has increased up to 3.8x. However, it is still not scalable with the servers due to its atomic broadcast protocol.

We implemented LOT and integrated it with the ZooKeeper. We replaced the ZooKeeper’s consensus protocol, Zab, with LOT consensus protocol. We found that as we add more servers in the system, LOT consensus protocol performs better than ZooKeeper’s Zab protocol.

We are currently finalizing failures handling for LOTKeeper and its consistency model.

2 Goals for the weeks

- Write failures detection section in the paper
- Add failure scenarios and their handling in the paper

3 Activities

- Studied failures detection and sequential consistency in literature

4 What I learnt (insights)

4.1 Consistency Model

We can provide three levels of consistency with LOTKeeper: linearizability, sequential consistency, and weak consistency. The level of consistency is changed by changing the way read requests are served by the servers. A read request
can be served at the end of a consensus cycle, during a consensus cycle, or immediately regardless of the consensus cycles.

Linearizability: To provide linearizability, a system has to serve the requests in a specific time order such that it looks like there is one FIFO queue of the requests. In LOTKeeper, we can achieve linearizability if the read requests are served at the end of a consensus cycle, and in the order the servers received them.

Let’s say a server S1 has a set of four requests K1=W1,R1,R2,W2 at the start of a consensus cycle. The server creates a requests proposal P1=W1,W2 and share with others for consensus. At the end of the consensus cycle, let’s say the received proposals are P2, P1, and P3. Proposals P2 and P3 are received from two other servers. The server updates it log by applying the update requests in the order P2, P1, and P3. However, when applying its own proposal, it also serves the read requests in the order the requests were received, i.e., W1,R1,R2,W2.

If we assume the other servers have the set of requests K2=W3,R3 and K3=R4,R5, at the start of the consensus cycle, then the three servers has the following sequence of requests at the end of the consensus cycle:

- Server S1: W3,W1,R1,R2,W2
- Server S2: W3,R3,W1,W2
- Server S3: W3,W1,W2,R4,R5

In a global or unified view, the request have the order: W3,R3,W1,R1,R2,W2,R4,R5. Therefore, the requests have a specific order as if the requests were placed in a FIFO queue.

Note that the read requests are not shared among the servers during the consensus because the read requests are only useful for the servers which received those requests.

Sequential consistency: A system provides sequential consistency if the following two conditions are satisfied by the system [1, 2]: i) Update requests are totally ordered across all the servers in the system. ii) The order of a client’s requests is preserved.

LOTKeeper satisfies the first requirement through total order of the update requests. To satisfy the second requirement, read requests are served immediately if there is not pending update operation for the requested key in the request. In case there is a pending update request, the read request is served after the consensus of the update request. Moreover, the sequence of the requests is preserved by each server.

Weak consistency: LOTKeeper can be configured for weak consistency as well, if the read requests are served immediately by each server, regardless of the pending update requests. This model provides weak consistency and optimizes the system for read requests. It can be useful for the applications that can tolerate stale reads and have strict latency requirements.

N.B. Word "weak" is used in its literal meaning here, not as a consistency term. We need to verify the actual consistency level.

4.2 Failure detection

Chandra and Toueg [4] introduced failure detectors and classified them in eight different classes depending on the completeness and accuracy of the detectors. They showed that a failure detector in one class can be reduced to a failure detector in another class, making a hierarchy of the classes. For implementation, we can use "adaptive failure detector" given in section 15.8 of the book from Kshemkalyani and Singhal [3] (need to verify that is it the same as given in a paper [5]). It belongs to "eventually perfect" class of the failure detectors. A failure detector is eventually
perfect if there is a time $t$ such that after $t$, every faulty (crash failed) node is suspected by every correct node (strong completeness), and no correct node is wrongly suspected (eventual strong accuracy).

Although we assume asynchronous model for our consensus protocol, failures detection requires timeouts. Therefore, we have to relax the synchrony model for failures detection.

### 4.3 Server failures

If a server fails before sharing its pending requests with other peers, it cannot add any inconsistency in the system. The failure will be detected by other peers in the super-leaf, and the failed server will be removed from the system. The client requests will also fail, but the clients need to know whether their requests were successful or not.

If a server fails after sharing its requests, the requests will be shared with all the nodes in the system, regardless of the current round number in the consensus cycle. Although the system will still be in a consistent state, the client connected to the failed server cannot verify if the last request was completed or not.

A possible way to solve this problem is to keep track of last successful request of each client. A client is required to have a unique client ID and request ID. The client ID and the request ID are given to the server with each request. As done in Raft [6], the servers keep track of the last successful request and its response for each client. A client can later query any server to retrieve the request ID of its last succeeded request. Unlike Raft, clients are not given session IDs for this purpose. Session maintenance further complicates the system.

As an optimization, the client request tracking can be limited to a super-leaf. In that case, a client is required to reconnect to any node in the same super-leaf with which it was previously connected.

### 5 Proposed goals for next week

- Write failures detection and handling section in the paper

### 6 Meeting agenda

- Consistency models

### 7 References


Exemption Amounts for Alternative Minimum Tax. For taxable years beginning in 2016, the exemption amounts under § 55(d)(1) are. Reporting Exception for Certain Exempt Organizations with Nondeductible Lobbying Expenditures. For taxable years beginning in 2016, the annual per person, family, or entity dues limitation to qualify for the reporting exception under § 6033(e)(3) (and section 5.05 of Rev. High Times November 2015. Item Preview. remove-circle. Share or Embed This Item. EMBED. magazine_rack; additional_collections. Language. English. High Times November 2015. Addeddate. 2016-01-02 05:25:25. Foldoutcount. 0. Identifier. High_Times_November_2015. Identifier-ark. ark:/13960/t18m16f9s. Ocr. ABBYY FineReader 11.0. Looks like there are some pretty good matchups on this week's slate. But what do I know? I'm just a robot! --- **Helpful...Â this post was submitted on 02 Nov 2015. 7 points (100% upvoted). shortlink: remember me reset password. login. Submit a new post. Get an ad-free experience with special benefits, and directly support Reddit. â€” Sakir Khader (@sakirkhader) November 2, 2015. Robbed of the chance of growing up: This baby killed in a #Russia|n air strike on #Aleppo|s Al-Kalasa neighbourhood. pic.twitter.com/feqKqbtFgW. â€” Sakir Khader (@sakirkhader) October 31, 2015. But you saidâ€¦ Here we can see that The Telegraph in January 2015 claimed that only 5% of Syria was held by the â€œmoderateâ€™ rebels. The question is: were they lying then, or now? Dispatch: Syria rebels â€œburned down churches and destroyed Christian gravesâ€™.Â Feel free to take a break from reading this report and resume when you are not rolling on the floor laughing. True Story: Pigs Flied. The fairytale written below is actually not a better love story than Twilight â€“ it is on par because they are both fictional.