CMPE200 - Assignment 3

1. Reviewed Paper

Subject: Distributed Systems
Title: CORFU: A Shared Log Design for Flash Clusters
Authors: Mahesh Balakrishnan, Dahlia Malkhi, Vijayan Prabhakaran, Ted Wobber, Michael Wei and John D. Davis
Citation: M. Balakrishnan, D. Malkhi, V. Prabhakaran, T. Wobber, M. Wei, and J. D. Davis. Corfu: A shared log design for flash clusters. In NSDI’12, San Jose, CA, April 2012.

I have reviewed the paper on CORFU: A Shared Log Design for Flash Clusters written by Mahesh Balakrishnan, Dahlia Malkhi, Vijayan Prabhakaran, Ted Wobber, Michael Wei and John D. Davis from Microsoft Research Silicon Valley and University of California, San Diego. Following is a link to the paper:


2. Brief Summary

The authors propose the design and implementation of a shared log abstraction over a flash cluster called CORFU. CORFU organizes a cluster of flash devices as a single, global, shared log that can be accessed concurrently by multiple clients over the network. Mapping is maintained at the clients. It makes it easy to build distributed applications that require strong consistency at high speeds, such as:
   I. Databases
   II. Transactional key-value store
   III. Replicated state machine
   IV. Meta-data services.

Other existing systems before CORFU had to make a trade-off between performance and consistency. CORFU enabled multiple clients to read at the same time with high throughput and appends were made at the end of the log using the tail finding function or the position assigned by the sequencer. The sequencer acts like a contention solver for clients that wish to write on the log at the same time. It assigns tokens to the clients that wish to write to the log. Clients can append data to the log as fast as the sequencer can assign them 64-bit tokens, i.e., new positions in the log. Placing functionality at the clients reduces the complexity, cost, latency and power consumption of the flash units.
3. **Strong Points and Main Contributions**

This paper proposes an idea to make use of the capacity of flash devices with the help of a shared log abstraction for multiple purposes like strong consistency and consensus engine. It ensures even wear leveling, network locality, fault tolerance, incremental scalability and geo-distribution. CORFU is designed to work over network attached flash devices, which provides lower latency, higher throughput, low power usage and elimination of slow and bulky servers.

4. **Weak Points**

The main bottleneck of the proposed design is that the sequencer can become a single point of failure. The rate at which the sequencer assigns the tokens to the clients is the rate at which the clients can append to the log. In the current design this rate is 200k tokens/sec. This cannot be viewed as a convincingly high rate as client might want to write much faster than this rate.

5. **Technical Approaches**

A log is perhaps the simplest possible storage abstraction. It is an append-only, ordered sequence of records ordered by time. Records are appended to the end of the log, and reads proceed left-to-right. Each entry is assigned a unique sequential log entry number. In distributed systems, a shared log is a powerful and versatile primitive for ensuring strong consistency in the presence of failures and asynchrony. They serve the following purposes:

- Consensus engine for consistent replication
- Transaction arbitrator for isolation and atomicity
- An execution history for replica creation, consistent snapshots and geo-distribution
- A primary data store that leverages fast appends on underlying media.

With CORFU, the authors have tried to leverage the advantages of a global shared log housed at the clients. This client-centric model has eliminated the need for maintaining high-end servers to store information regarding the location of logs. I believe this approach of the authors was pretty smart which is why this paper received a lot of attention when it was written. They have proposed the idea of network-attached flash devices, which helps clients to directly make reads and writes on the flash devices, using their local shared log.

This idea is a novel because much of the infrastructure already existed. It was just not thought of that network attached flash could be used for a shared log design on flash clusters. Except for the scalability issue of the sequencer, all other technical designs are pretty sound. Even the sequencer, as the authors mention, is not the bottleneck, as it just optimization to reduce contention in the system and is not required for either safety or progress.
6. Experimental Methodology

There are a few things that the authors have experimented with. Firstly, a completely client-centric model was never seen before this paper. Secondly, the capacity of the flash devices to return the tail of the log so that clients can understand where they can write, is another new idea. Finally, the most important idea of maintaining a globally shared and distributed log helped them achieve strong consistency with high throughput, which had been an issue with previous designs.

7. Organization and Presentation

I really liked the way the paper has been organized. The authors began with the importance of shared logs and uses of flash devices. They explained why disks could not be used to implement their shared log design. Moreover, the API of CORFU was explained quite well with a clear explanation of when things could fail and how they could be recovered, without impacting the performance of the system. Another thing I noticed about this paper was, unlike few papers that try to point at the pitfalls of some other existing designs, this paper clearly talks about the areas where it has learned from other designs and how it has tried to solve the issues with earlier designs.

8. Suggestions for Future Work

One challenge with CORFU is its reliance on flash media; especially in real-world deployments where having a dedicated cluster of flash devices may not be cost effective. The Zlog (Zlog: A shared-log on RADOS) project is an implementation of CORFU on top of the Ceph storage system, offering flexibility in the type of hardware used to store the log. The current implementation uses a fixed set of objects over which a log is striped using a round-robin strategy. It may be desirable to change that striping strategy to improve the performance, and do so without affecting the clients that are writing to the log. One simple example of a change to the striping strategy would be an increase in striping width. It would be an interesting experiment to analyze the change in performance based on the striping width.