CS293S Project Summary

Fatih Mehmet Bakir, Kyle Carson

We propose an application of Bitcoin's underlying blockchain technologies as a solution to the issues of consensus and replication in a distributed storage system, specifically for resource constrained web applications. Whereas Bitcoin was concerned with issues such as the double spending problem, proof of work, and honest majority, the architecture we propose inherently solves or reduces the impact of these issues.

The stack for our storage system starts at a simple key-value store. We plan to use a prebuilt solution initially (something like RocksDB, or potentially mongoDB, Redis, etc), then maybe rewriting our own if needed. On top of this will be the core of our project, which consists of modules such as a transaction manager, log manager, and any other logical chunks of code that make up the "brains" of our system. On top of this, we plan to have a client request handler along with an API. The client request handler provides a simple set of methods such as put, get, update and delete, while the API provides a much richer introspection into what's going on internally with the "blockchain network".

As mentioned before, the architecture we imagine simplifies the demands placed on building the blockchain. To both explain and exemplify this, we use a clone of the site Reddit, in which users have account profiles consisting of their own posts as well as comments on posts, where each post belongs to a "subreddit", or subcommunity on the site focusing on a particular topic. When first setting up this system, the developer provides a relatively static configuration file/schema defining how the system should operate in terms of replication, consistency guarantees, storage requirements, etc. They then set up a cluster of server nodes, where server here refers to the datastore servers, not the application servers. Servers can perform both reads and writes of data to the chain, but as more users use the site, a server's workload will almost entirely become writes. This is because when each client joins the site as a user, they effectively agree to act as a read-only replica of some portion of the total data on the chain. By making servers read/write and clients read-only, we solve three major problems:

- We don't need to worry about malicious adversaries since the blockchain can only be modified by our own servers, so maintaining a majority of honest nodes is no longer a major concern
- Since the majority of the site's traffic is comprised of reads,increasing the number of users only causes a small linear increase in load on the servers, as the users themselves are serving nearly all of the read requests, effectively scaling to meet their own demands
- Since a large amount of the network bandwidth is offloaded from the site's creator, the site doesn't have as serious of financial needs, shifting the paradigm from "a platform and its storage" towards just "a platform". Rather than a user dealing with ads or paying a subscription fee, they agree to give some portion of their resources for uninterrupted use of the site

So whereas Bitcoin was 1 CPU = 1 vote, disk space and network bandwidth become the resources of interest, and site's can provide users with flair or additional features as the incentive for contributing more than what’s asked for. When structuring the reddit clone, each user’s profile becomes its own chain, each subreddit is its own chain of posts, and each post is its own chain of comments. By breaking up the data into multiple chains, we can minimize the contention that comes with a single global chain, while using it as a natural means of sharding and caching the data. A single user’s cache of their portion of the site’s data may contain their own profile’s chain, the chains of subreddits they’re subscribed to, and the comment chains of posts they’ve recently read. In this way, the system can leverage the locality of a subreddit community to broadcast content to a user to both replicate/cache this data and use it since the user has expressed interested in it through subscribing, while allowing them to ignore the majority of the site’s content they have no interest in/never visit. If they do make a request outside their locality, they simply experience the added latency of finding the nearest geographical user who has a copy of the chain with that data. That lookup can be performed either via proxy from a server or from public knowledge on the site about who is likely to have what data depending on how the site is designed.