Blockstore

A data storage solution built with Blockchain

Fatih Mehmet Bakır
Kyle Carson
CS 293S, UCSB Winter 2018

Motivation

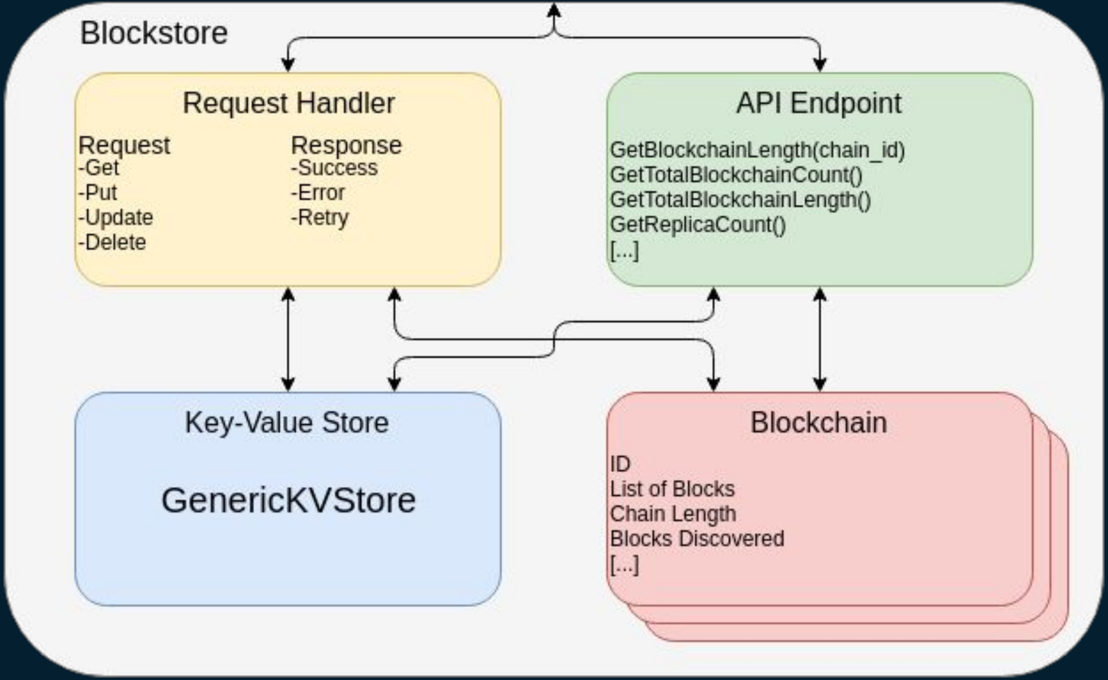
- Consensus, replication, and sharding are exceedingly difficult in the context of data storage
- Blockchain takes an interesting approach to distributed communication/consensus
- A permissioned environment such as owning all of the mining/storage nodes provides some room for experimentation
  - A semi-permissioned environment has the potential to cut costs
- ~~Blockchain is all the hype~~

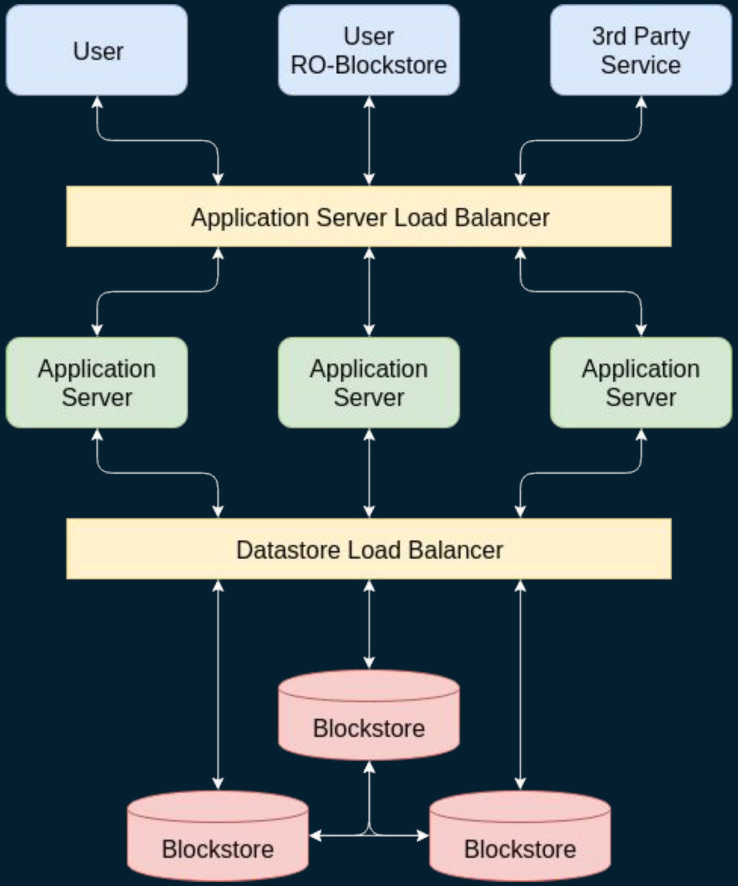
What does Blockstore aim to achieve?

Provide a simple key-value storage solution using a modified Blockchain implementation for managing operation logs and distributed communication

Overview

- Modified Blockchain-variant using Generics
- C++ for intensive operations such as mining, validating
- NodeJS + TypeScript
  - HTTP for client communication
  - Socket.io for internal communication
- Simple in-memory KV store using a dictionary
- Request and API Handlers
- Client module as an application-programmer's interface


Blockstore
Request Handler
Request
-Get
-Put
-Update
-Delete
Response
-Success
-Error
-Retry
API Endpoint
GetBlockchainLength(chain_id)
GetTotalBlockchainCount()
GetTotalBlockchainLength()
GetReplicaCount()
[...]
Key-Value Store
GenericKVStore
Blockchain
ID
List of Blocks
Chain Length
Blocks Discovered
[...]


User
User RO-Blockstore
3rd Party Service
Application Server Load Balancer
Application Server
Application Server
Application Server
Datastore Load Balancer
Blockstore
Blockstore
Blockstore

Typical Flow

- Request is made by some client to Blockstore
- Reads get sent directly to the KV-store, while any sort of Write is created in the blockchain as a unit of Operation
- Operation added to a block and mined
- Once mined, it is appended to the blockchain, distributed to all other known replicas, and a response is sent to the client
- Mining is synchronous, rest is asynchronous

Blockchain Modifications

- No merckle-tree, need to maintain full history of operations
- Significantly lower bounds on honest node count and mining difficulty requirements since there are no attackers
- Full view/control of nodes in the network
- Flexibility in the data size of Operations, Blocks, Async vs. Sync communication, etc via a cluster-wide configuration file

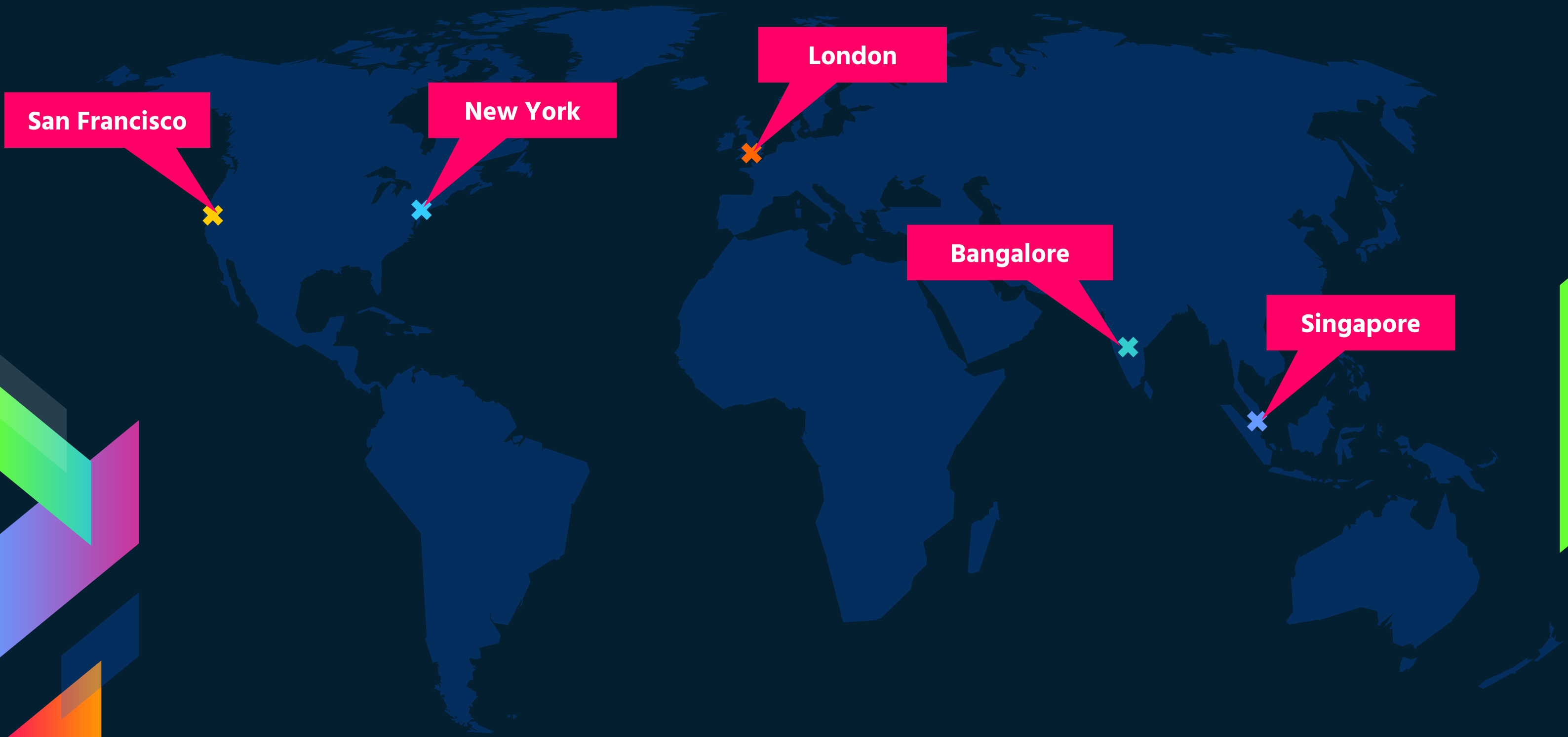
Experimental Setup

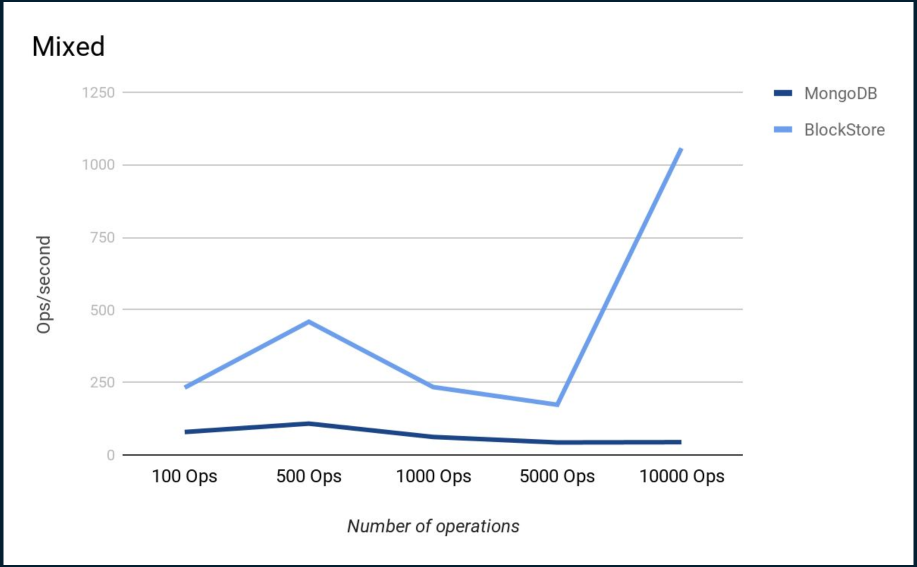
- Remotely test MongoDB vs Blockstore with YCSB
  - Latency
  - Throughput
- Multi-server cluster
- Simple Demo Application

Cluster Deployment

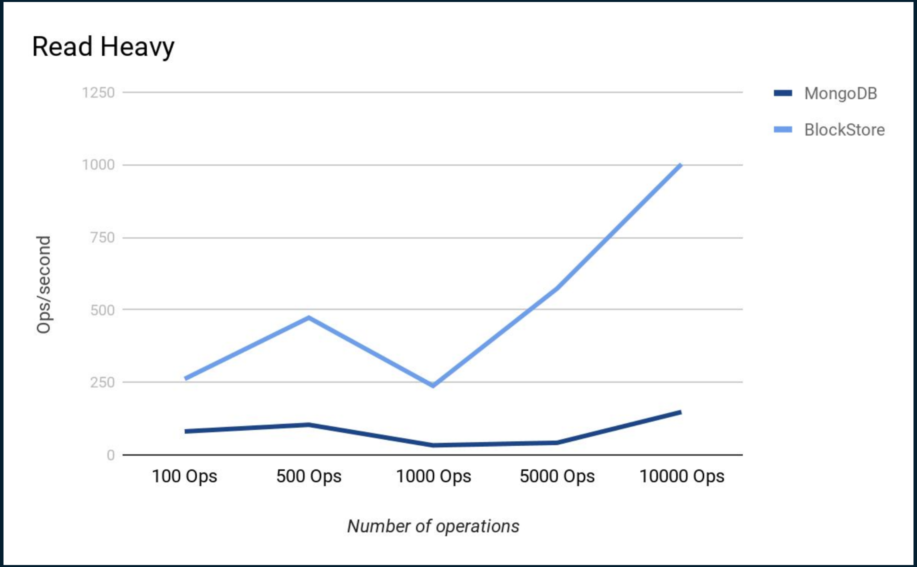
- Docker images running on DigitalOcean
- Servers located in multiple regions
- VM Capacity:
  - 1 CPU
  - 1 GB RAM
  - Ubuntu 16.04

Geographical Deployment

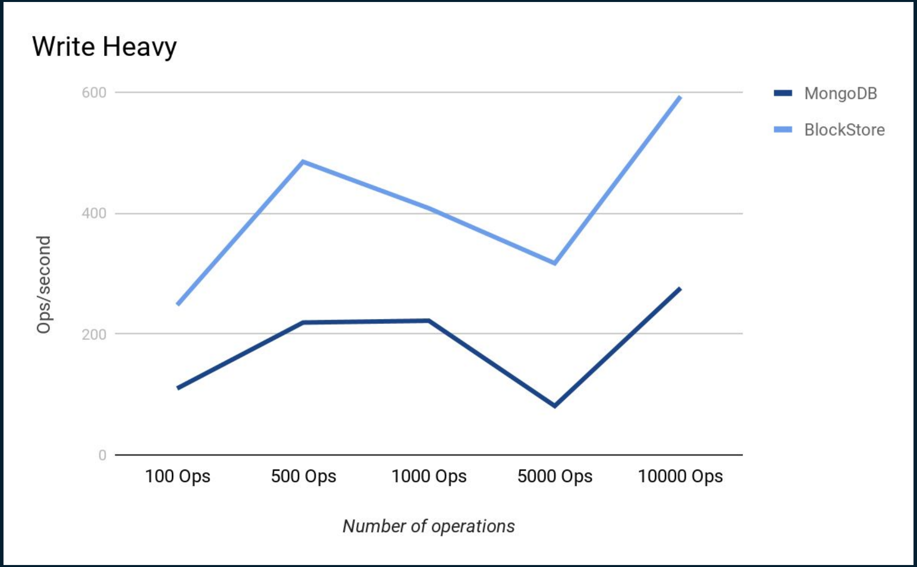




65% Get, 25% Put, 10% Update



90% Get, 5% Put, 5% Update



45% Put, 45% Update, 10% Get

Lessons Learned

- Bitcoin is a very simplistic approach to Blockchain
- Coupling of transaction commital to mining is *slow*
- Strong consistency guarantees are difficult in a reasonable timeframe
  - Decays into 2PC
- Redundancy between contents of Blockchain and KV comes with overhead

Additional Ideas to Explore

- Read-only replica functionality
- Garbage collecting the blockchain
  - Checkpointing
- Leveraging permissioned environment to do leader-based transaction committal (ByzCoin)
- Build schema definition API for ease of development
- Expand request query types to support more advanced queries on multiple items

http://bs.fatihbakir.net

WE’LL DO IT LIVE!

THANKS!

Questions?

@FatihBAKIR

@carsonkk

https://github.com/FatihBAKIR/blockstore