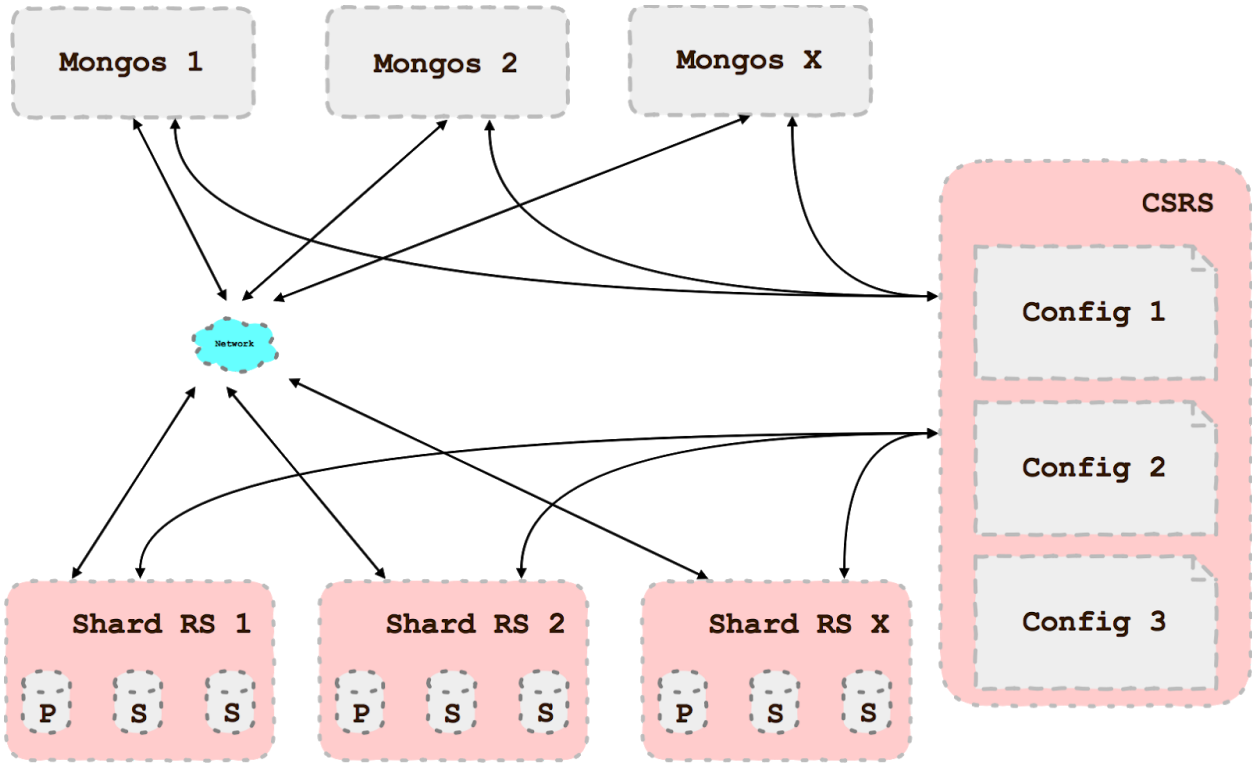
MongoDB Sharded Architecture

Overview

As our MongoDB data grows we are evaluating various options to scale the infrastructure. Initially we were running out of Disk space but as the single replica set was being used by other parts of the stack we needed a way to scale Memory, Disk and CPU horizontally rather than beefing up the servers vertically. In this document we will explore MongoDB in a sharded architecture so rather than adding more resources to the same node we could add more nodes with less resources.

Diagram



https://drive.google.com/open?id=1I5J_Ci2Rlg5O0gHlLelDK4zg6xRImgTu

Terminology

Replica Set: A Replica Set is a copy of the same data placed across multiple nodes. This will allow for redundancy, if the Primary goes down the Secondary would get promoted to Primary.

Shard: Shards comprise of multiple Replica Sets with each set holding a portion of the data. This will allow for horizontal scalability as you can add multiple shards to the cluster to scale it to hundreds of Replica Sets.

Shard Key: The Shard Key is probably the single most important thing to be determined when sharding a cluster. In order to ensure the data gets distributed evenly among shards the shard key needs to have high cardinality. What does this mean? It means the shard key must be as unique as possible between each document to give MongoDB the opportunity to split the chunks into smaller chunks as they grow larger.

For instance let's take the following set, [hello22, hello33, hello44, hello33, hello44, hello55], this set has a cardinality of 4 because there are only 4 unique values, the other 2 are duplicates. The higher the cardinality the more evenly data gets distributed into shards. https://en.wikipedia.org/wiki/Cardinality_(data_modeling)

CSRS: Config Server Replica Set (CSRS) is a special type of replica set that stores the configs of the sharded cluster across 3 nodes. At the moment only 3 nodes are supported.

Monitoring

Today we monitor over 20 services on MongoDB nodes, below are a few MongoDB specific ones that we monitor on all the nodes. As we do sharding we will add sharding specific checks as well.

mongodb-collections	OK
mongodb-connect	OK
mongodb-connections	OK
mongodb-databases	OK
mongodb-memory	OK
mongodb-memory-mapped	OK
mongodb-opcounters	OK
mongodb-oplog	OK
mongodb-replication-lag	OK
mongodb-replset-state	OK

Backups

The same LVM backup strategy we use for backing up the current Replica Set can be used with Sharded Replica set, with a few minor differences. In addition to **db.fsyncLock()** we also need to stop Balancer as well **sh.stopBalancer()**. The link below further details this procedure along with a procedure to backup CSRS as well.

https://docs.mongodb.com/manual/tutorial/backup-sharded-cluster-with-database-dumps/

Cost

There are a number of ways we can approach this from an infrastructure standpoint; Physical Hardware, Cloud, Atlas.

Before we evaluate lets setup some base parameters.

VMs per Mongos Servers: 3
VMs per CSRS: 3
High Performant Shards: 3

The most important part of this is the Cost per Shard, based on that let's discuss from pros and cons.

Physical Hardware: PowerEdge R740XD Server
CPU: 28
Memory: 128 GB
Disk: 2 x 1TB SSD
Cost per server: $19,200
Cost per Shard: $57,600 (not including support renewal cost)
Lifespan: 2-3 yrs

Once you buy the server you own it. Once they run their course they can be repurposed as Dev or non-critical servers. Doing maintenance or upgrades means one of the nodes must be down for longer than it has to be, it's not possible to keep the upgraded node primed and ready to go because we do not have additional servers. Hardware failures are inevitable and fixing such issues takes significant effort and resources for operations, this is an unmanaged solution. Procuring and setting up servers takes a long time, from the moment the request for

quote is made to servers are provisionable it can take 1-2 months, sometimes longer.

Atlas: M80
CPU: ???
Memory: 120GB
Disk: 750 GB
Cost per Shard: 6.13/hr x 720 hrs = $4413.60/mo = **$52,963.20/yr**

Atlas is a PaaS solution, the Managed version of the Mongo from MongoDB Inc. It supports fully managed backups, comprehensive monitoring, real-time performance, alerting, among other cool managed features. MongoDB engineers completely take care of the operations and scaling of the cluster, we just point to the CSRS. The one uncool thing about it is it very expensive, almost cost prohibitive. I think it's for people who cannot or do not want to manage MongoDB, we are in neither category because we are a smart and dedicated bunch.

Cloud (GCP): n1-standard-32
CPU: 32
Memory: 120GB
Disk: 1TB
Cost per Shard: $2,815.98/mo or **$33,791.81/yr** or $21,531.45/yr (on 3 yr commitment)

Cloud is neither fully managed nor unmanaged. The underlying hardware is managed by Cloud provider, in this case GCP, but the service launched atop is managed by us. We are not committed to a single configuration or instance type, based on our load and traffic we can swap out existing instances with powerful ones or add new shards on the fly. When the load goes back down we can swap with smaller nodes or remove the additional nodes. Maintenance is no more a chore and nodes are moved in and out of clusters swiftly. Mostly importantly there is no more hardware to deal with, if a hardware failure does occur GCP moves the instance transparently to a healthy hypervisor.

Only one of these options hits the sweet spot in terms of Price, Scalability and Tech Debt, I'll let you figure out which one.

Questions

Questions and comments welcome.