Mastering Cassandra

Jon Haddad

Table of Contents

This book will focus on the latest stable version of Apache Cassandra. I'm targeting the most current release, which at the time of this writing is 4.0. Once the next version is released, this book will be updated.

This is an opinionated book. I will not advise using features that perform poorly in production or have so many caveats that failure is almost unavoidable. This is not the result of marketing, promises, or speculation.

This book is the culmination of everything I've learned about running Apache Cassandra in production. I have been lucky enough to have run it for several years at a startup, worked at DataStax as a technical evangelist, and while working at The Last Pickle I've helped fix and tune a wider variety of Cassandra clusters than almost anyone else on the planet.

I'd like to thank a handful of people to have made this book possible. I've made a lot of friends as the direct result of a technology choice. A special thanks to Blake Eggleston, Patrick McFadin, Nate McCall, Aaron Morton, Caleb Rackliffe, Luke Tillman, and Jonathan Ellis. Thank you to many others who I've had the privilege of working with and learning from over the years.

The Scale Problem

Chapter 1. Scaling, Old School

When I got interested in Cassandra, I was working at some startups that had tried to build scalable systems with tight SLAs and had run into some trouble. This was a point in time when using a relational database and vertical scaling was the tried and true solution.

Once we hit the limits of vertical scaling, the next step was to chip away at the advantages of the RDBMS.

1.1. Caching

First, we would introduce caching whenever possible. Caching servers are easily scaled by partitioning the data among all the servers. Splitting data among caching servers is easy. This works great for read heavy workloads and scales linearly for a long time.

You've probably heard of one of the most well known caching projects of all time, Memcached. Memcached made it possible to massively scale read heavy workloads by exposing a simple api. The Memcached api only exposes a simple key/value api for setting and retrieving data. The following (crappy) code shows the basic logic:

```
def get_something(id):
    tmp = memcached.get(key)
    if(tmp):
        return tmp
    result = db.query("SELECT * from stuff WHERE id = " + id)
    memcached.put(key, result)
    return result
```

Starting with aggregations and joins.

precalculate expensive queries rather than doing them on the fly

basic statistics

We start thinking about partitioning our data at this point. Without joins, we can move tables to different machines without much problem. We lose the ability to d

querying single table

querying single rows

Hello

Once we've

The Fundamentals of Cassandra

Chapter 2. Shared Nothing Architecture

One of the

We

SAN

VMS

nodes

Chapter 3. Token Ring

distributed hash table

each node has one or more tokens

Chapter 4. Write Path

Chapter 5. Read Path

Data Modeling

Chapter 6. Keyspaces

6.1. Replication Strategies

6.1.1. Simple Strategy

6.1.2. Network Topology Strategy

Chapter 7. Tables

Much like a relational database, we create structured tables with schema in Cassandra to store our data.

Tables contain rows

Unlike a traditional relational database, we need to specify how we want our data partitioned across

What can we use as a sort date

You might recognize this syntax:

```
CREATE TABLE mytable (
    id uuid PIRMARY KEY,
    data blob
);
```

7.1. Partition Key

Partition key can be composed of one or more fields. These fields are hashed to give us a token, and that token determines the replicas that a given partition lives on.

7.2. Clustering Keys

Clustering keys determine sorting order

```
CREATE TABLE comlex (
    id uuid,
    bucket int,
    data blob,
    PRIMARY KEY (id, bucket)
);
```

Partitions sorted lists of rows

```
CREATE TABLE sensor_data (
  id uuid,
  ts timestamp,
  data blob
) PRIMARY KEY (id, ts);
```

Getting Started

Chapter 8. Download

test

Common Operations

Chapter 9. Starting a New Cluster