Engineering a BigData Pipeline for DotA2

CSCI 5751 - Big Data Engineering and Architecture

NoSQL Proof of Concept - Proposal

by

Data Diggers

Bhaargav Sriraman - srira048@umn.edu
Prabhjot Singh Rai - rai00027@umn.edu
Suresh Siddharth - siddh010@umn.edu
Vishwesh Mishra - mishr167@umn.edu

Table of Content

1. Overview

Through this project, we are planning to design and develop a system that can take a look at the data generated by people playing the game of DotA2 and perform analytics-at-scale on it. The aim is to provide general global-level statistics on the game (descriptive analysis), insight on the latest trends followed by the players, as well as look into how the system can provide suggestions and strategy on how to play better (predictive analysis).

1.1 What is DotA2?

Before we dive any further into the how and the why of things, a formal introduction to the game from Wikipedia[1] would be a good starting point - DotA2 stands for **Defence of the Ancients 2** and is a multiplayer online battle arena (MOBA) video game developed and published by Valve Corporation. DotA2 is played in matches between two teams of five players, with each team occupying and defending their own separate base on the map (Figure 1.1). Each of the ten players independently controls a powerful character, known as a "hero", who all have unique abilities and differing styles of play.

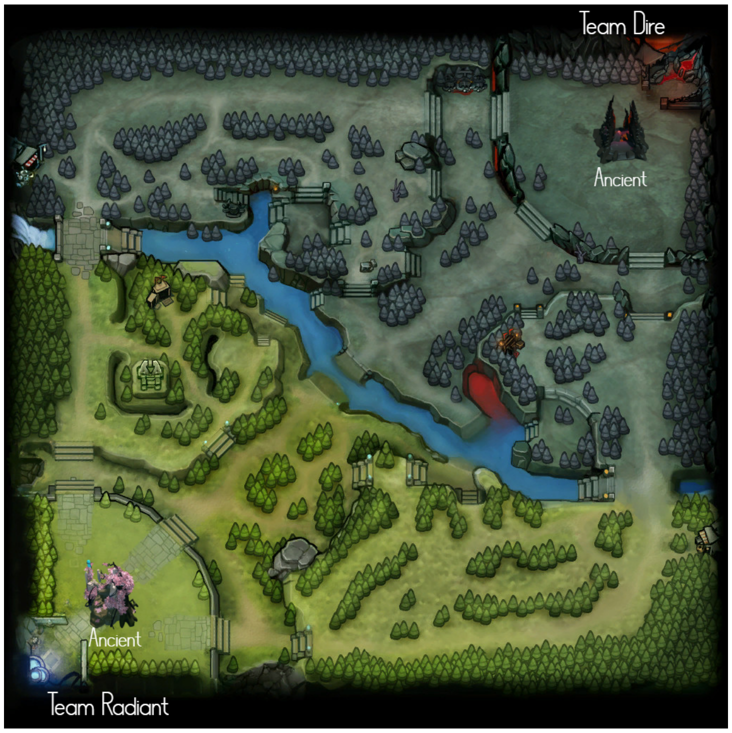


Figure 1.1 - The miniature map of DotA2

In order to understand the complexity of the game, we need to look at this in a little more detail. These heroes are the most essential elements of the game. A player needs to pick one out of 117 heroes for the duration of the game. They each have 4 unique abilities that can range from passive effects to “devastating explosions of energy, to complex, terrain changing feats”[2]. Figure 1.2 gives a representation of the different heroes in the games along with their primary class. In-game equipment or “Items” are another part of the equation. Heroes gain gold as they play the game, and can use these to purchase items. These items provide heroes with bonuses and/or special abilities.

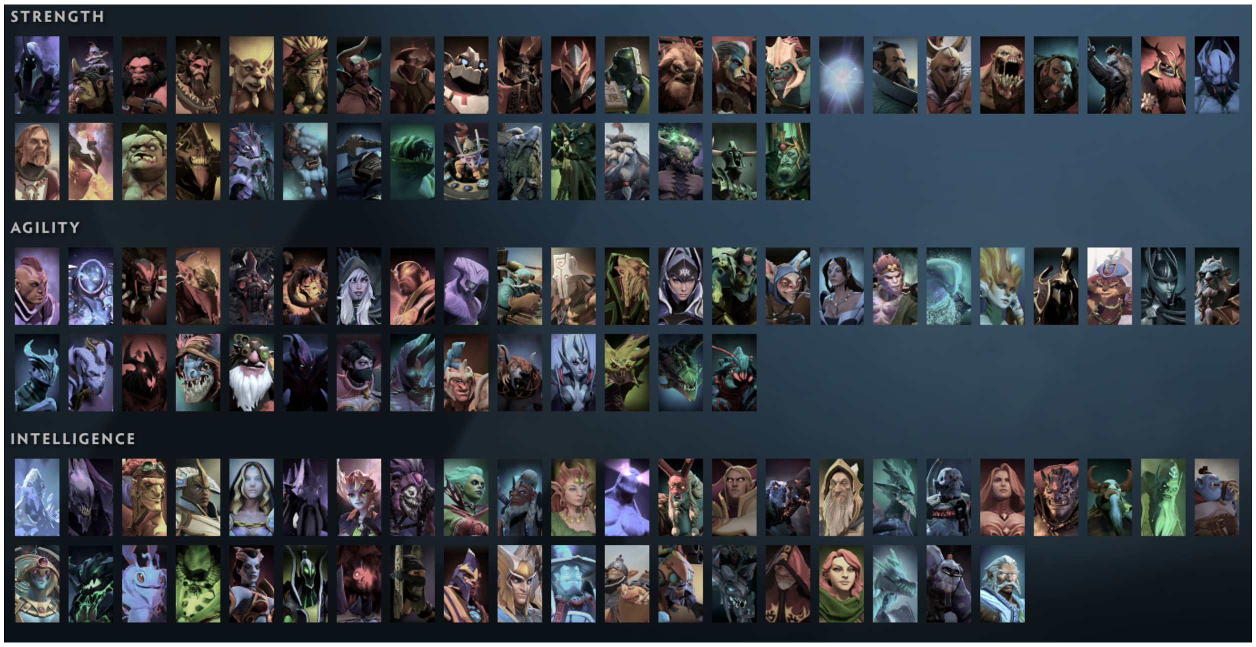


Figure 1.2 - The Heroes of DotA2

The win condition is a lot less complicated to understand. A team wins by being the first to destroy the other team's "Ancient", a large structure located within the enemy base.

1.2 Business Questions

Now that we have an understanding of a few of the core concepts of the game, here are the business questions we are looking to solve:

Easy business questions

1. Can an abandoned game be identified from the dataset?
2. What is the average number of games hosted in a day?
3. What is the average duration of a game?
4. How many games is a hero featured in? What is the win rate and the pick rate?
5. What is the average first blood time seen in a match?

Medium business questions

1. Identify pairs of heroes that work well together. Quantify the synergy.
2. Who is a hero most and least effective against?
3. What are the most frequent items bought by a hero?

Hard business questions

1. What are the factors that affect the result of the game (first blood etc)?
2. Identify pairs of heroes that are good and bad against another particular pair.

Ambitious business questions

1. Which team is more likely to win, if all 10 heroes involved are given?
2. Include a replay parser in the data pipeline. What new data do we get? Can we enhance any of the previous business questions?

1.3 Why is this a Big Data problem?

DotA2 has been a game that has been around since 2011 and is still going strong. With a global player base, it has people playing the game around the clock. As of last month, the game has 9.9 million unique users login to the game[1] and averages around 400k people playing the game at any point in time. This gives us the first V of Big Data - *Volume*.

It is also the nature of the game that there are times at which split-second decisions are made and hence our solution would also need to respond just as fast in order to satisfy our ultimate goal of providing real-time tracking and support, giving the second V of Big Data - *Velocity*.

Currently, we are only looking to ingest data about the game. However, if possible we might be looking at downloading actual game replays and parsing them to generate our own data from them. This would introduce a new format of data and brings in the third V of Big Data - *Variety*

[1] As per in-game dashboard as of October 3rd, 2019

2. Data Source

When exploring the data sources, we found two in particular which could answer our business questions.

	OpenDota[3]	**Steam**[15]
Definition	OpenDota is an open-source platform that provides an API to fetch DotA2 related data.	SteamAPI is hosted by Valve Corporation, who are the creators of DotA2. They provide information on completed DotA2 games.
Number of Attributes	Data provided by OpenDota provides a higher number of attributes in a single call	Native Steam API provides less number of attributes.
Originality of Data	OpenDota employs a replay parsing mechanism to obtain a richer data set. Attributes generated are added to the original data.	Steam's API is the original source of data and provides raw match data only
API Constraints	50,000 requests for OpenDota API in free tier per month	100,000 API calls per day for Steam API

Other notable sources which we found but did not rank as high as these were:

i. Kaggle - Not considered as it was a very old dataset.
ii. UCI ML Library - Same issues as with Kaggle.

2.1 Our choice of API

After going through both the APIs, we have decided to initially focus on the Steam API because of the following reasons:

- We believe that the Steam API's veracity is higher than the OpenDota dataset as the former is provided directly by the game's developers.
- We can get more data from Steam's API than OpenDota. Although OpenDota may give us greater number of attributes, we believe that having more data is crucial as the volume is more critical for solving our business problems than the number of attributes.
- We also have the option of using an open-source DotA2 replay parser (Manta) to complement our data pipeline.

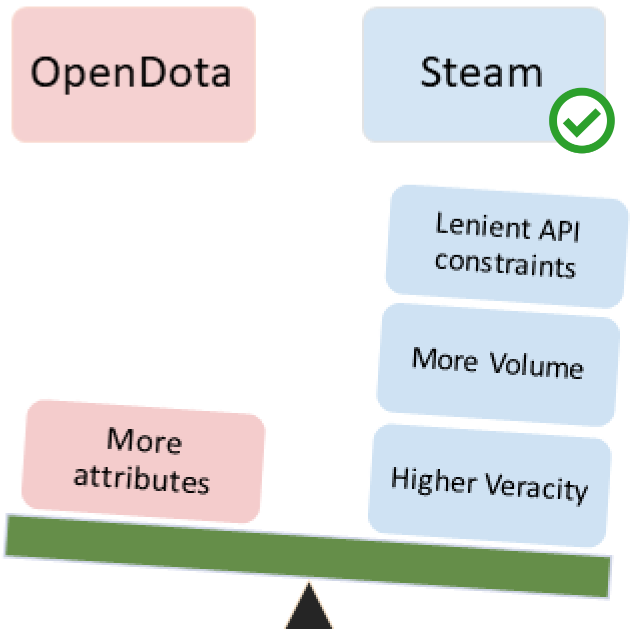


Figure 2.1 - "Steam" Overweighs "OpenDota"

2.2 Constraints of using Steam's API

1. Obtaining a Steam WebAPI key requires an account that has at least one paid game. For now, we have one such account but as we are looking to make multiple calls at the same time, we will need more API keys.
2. We will need to manually limit our requests to one request per second per key to reduce the strain on the servers.
3. In case we get a 503 error, the matchmaking server is busy or we exceeded limits. This kind of errors will have to be handled in code[4]
4. Finally, we are limited to 100,000 API calls per day under the Steam API conditions[5].

2.3 Inconsistencies in data

Although the data was mostly consistent, sometimes we got the server busy error stating that *"Too many requests, please try again in a while"* or sometimes we would get an OK response (200 status), but with an empty array. We are looking to handle it in our application code and track these through data provenance.

2.4 Other datasets explored

There were two other data sources that we explored before finalizing DotA2 API. These were the following:

1. **MLS Listings Database**
   We wanted to analyze and answer business questions around real estate datasets in the United States, hence we reached out to many listing providers like MLSPin, NWMLS, MNCAR, etc. None of them agreed to provide data for university projects since their terms and conditions involve the accessor to be a real estate agent.

2. **SENTINEL-2**
   This dataset contains data from a constellation of two polar-orbiting satellites placed in the same sun-synchronous orbit. After exploring this space dataset, there were a number of challenges that we could think of. For example:
   a. Exploring the dataset was becoming very difficult on our commodity laptops as the images provided were around 1.4 Gb per image, making it difficult for us to form a feasible business question
   b. We couldn't find a free API to fetch data as a batch process and inject it into a NoSQL storage

3. NoSQL Storage Tech

We have chosen FaunaDB for the purpose of processing, storing and analyzing DotA2 data. FaunaDB is an indexed document store that allows access to the stored data using multiple models - relational, document, graph and key-value.[6] FaunaDB core functions are inspired by Calvin: a clock-less, strictly-serializable transactional protocol for multi-region environments.[7]

3.1 Features[8]

Multi-API querying

- FaunaDB offers polyglot APIs for data access, including native GraphQL and FQL for advanced tasks and more productive development.
- ***Our thoughts:*** We can exploit the FQL features like join to our advantage instead of implementing it at the application level.

Flexible data modeling

- Access all data via a data model that best suits your needs - relational, document, graph or composite.
- ***Our thoughts:*** The various features will help us answer different business questions appropriately. Like, graphs can be used for modeling the relationship between hero pairs.

Powerful indexing

- A unique approach to indexing makes it simpler to write efficient queries that scale with your application.
- ***Our thoughts:*** This may help us in creating indexes according to our objective for better querying the attributes which we require while fetching the data.

Built-in temporality

- Travel back in time with temporal querying. Run queries at a point-in-time or as change feeds. Track how your data evolved.
- ***Our thoughts:*** This can be used for live-game tracking. It will help us answer business questions related to the start and end of an event in-game.

Note: We also considered using ArangoDB. However, ArangoDB is not great for transactional support, which is needed to answer some of our business questions. In addition to that, we faced issues while bringing up a cluster of nodes. As a result, we decided to go with FaunaDB.

4. System Setup

4.1 FaunaDB Setup

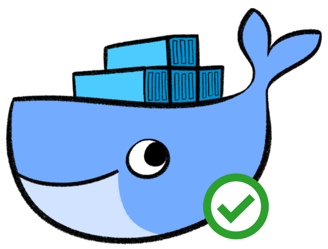
From our analysis, FaunaDB can be implemented in 2 ways:

1. **Use the public Docker[2] image**
   The significance of the docker approach is simplification and acceleration of deployment and configuration of FaunaDB. However, some of the key features which are available in the cloud platform are not supported in the Docker. For example, GraphQL is not supported in the dockerized version.

2. **Use the Cloud PaaS**
   FaunaDB also provides a cloud platform to build an application. Although all the features are present in it, it is not free. This is a problem if the application needs to be considerably scaled in the future.

Our choice

Currently, we are planning to use the Docker image in a local environment to understand the database, develop the data ingestion and processing programs/queries. This approach will not be feasible in the long run due to system availability and specifications.



Once our code is stable, we plan to migrate the database to the GCP. This will let us use what we have implemented locally as well as also improve the availability.

Note: While setting up FaunaDB, the native join command was not working, and we had to come up with a work around to create a Docker network as well as split the process into initialize the node, and then join the node to the cluster via the admin command

4.2 GCP configuration

While we have not yet decided on the exact configuration of the instances we are planning to use in GCP, we have decided on restricting the number of containers per system to be less than or equal to the number of virtual cores available. Since the nodes will simultaneously process the data, it is better to minimize the chance of threads interleaving.

[2] Docker is a tool designed using containers. Containers allow a developer to package up an application with all of the parts it needs, such as libraries and other dependencies, and ship it all out as one unit.

5. Project Plan

So far in our proposal, we have spoken about the first and second stages of the Big Data Analytics Lifecycle.

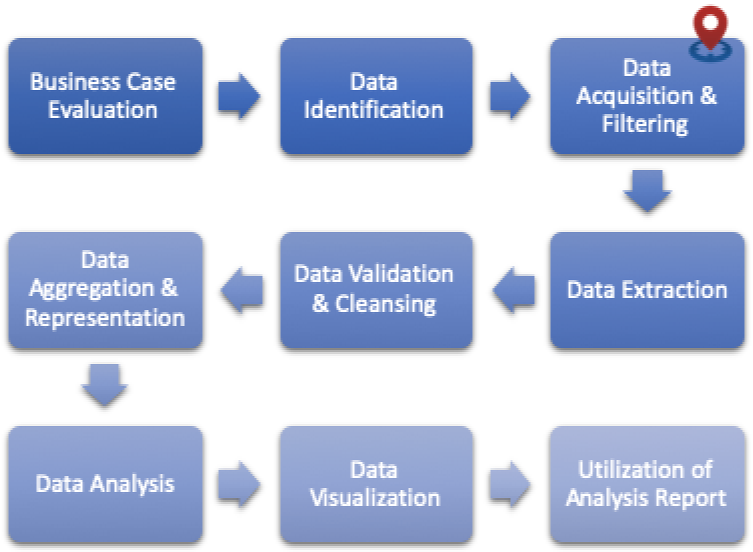

Business Case Evaluation
Data Identification
Data Acquisition & Filtering
Data Extraction
Data Validation & Cleansing
Data Aggregation & Representation
Data Analysis
Data Visualization
Utilization of Analysis Report

Figure 5.1 - Big Data Analytics Life Cycle

The current problem we are tackling is to come up with a synchronous multithreaded approach to simultaneously write the data returned by the API. Any errors in the API need to be handled via code and the data should be stored both in its raw form as well as filtered.

Future Steps

The next steps of our data pipeline would be to identify and remove games which were abandoned[3] and games that lasted less than 10 mins. This would mark the start of data exploration and the journey to answer our business questions.

We can loosely capture what the different stages of the Life Cycle would look like for our project.

- **Data Aggregation and Representation:** Explore different representation formats to see how our data can be used to efficiently answer our business questions
- **Data Analysis:** The data model required to answer the hard business questions can be improved based on the feedback from the user. This step is iterative in nature.
- **Data Visualization:** Create a dashboard to visualize our analysis and showcase our final product.
- **Utilization of Analysis Report:** Feedback from the users can be fed back to the data analysis stage to improve our results.

[3] A game is considered abandoned when a player has been disconnected for longer than 5 minutes

6. Potential Issues and Challenges

Through the life cycle of the project, here are some of the potential issues and challenges we might face:

1. Since there is a limit on the number of records fetched by a particular API call, we need to come up with a synchronized multithreaded program to fetch the required quantities of data
2. Docker is subject to performance overhead due to overlay networking, interfacing between containers and the host system
3. The unusual domain of data chosen will need some extra time to understand
4. Handling errors and empty data from API calls
5. Understanding the inner workings of FaunaDB and differentiating what features are provided in the public Docker image and what is specific to the enterprise PaaS solution

Appendix I - About DotA2

DotA2 is a massively multiplayer online based battle arena game that is considered to be one of the biggest games in the esports (electronic sports) industry. There is a lot of money being put into the ecosystem and DotA2, in particular, has the distinction of having the tournament with the highest prize pool of all time of $34.3 million. The below figure shows estimates from Newzoo's 2019 esports market report.

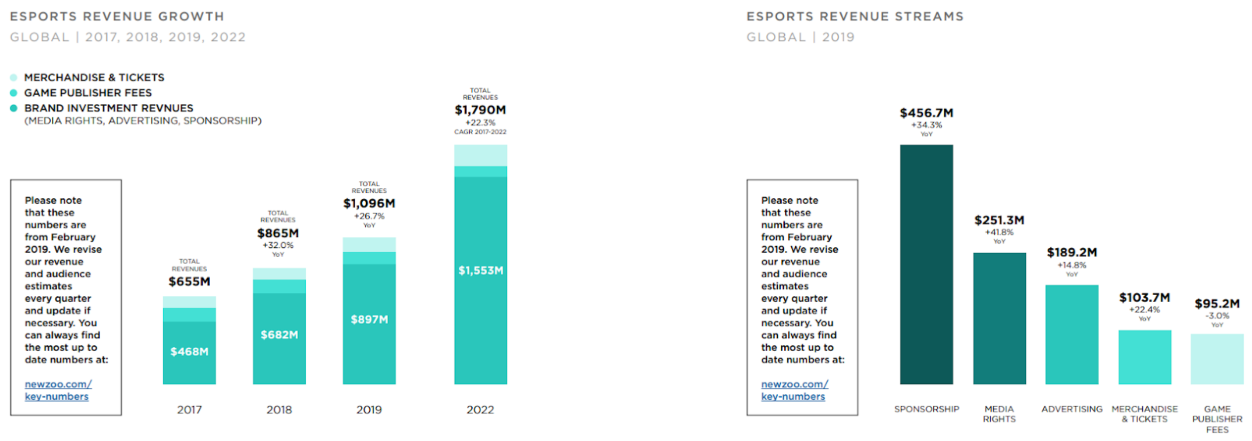


Figure - Newzoo's 2019 Esports Market Report

Heroes[1]

Heroes are divided into two primary roles, known as the core and support. Cores, which are also called "carries", begin each match as weak and vulnerable, but are able to become more powerful later in the game, thus becoming able to "carry" their team to victory. Supports generally lack abilities that deal heavy damage, instead having ones with more functionality and utility that provide assistance for their carries, such as providing healing and other buffs. Players select their hero during a pre-game drafting phase, where they can also discuss potential strategies and hero matchups with their teammates.

Items

There are close to 200 items in the game that are built on the 10 heroes that are in the game at different times. Items are crucial to how the game is played and at times can be as important and powerful as a hero's ability.

Ranked Matchmaking

Matchmaking is the process through which the system groups players into opposing teams for public games. With the exception of Bot games, matchmaking is mostly determined by matchmaking ratings (MMR).[12] There are two kinds of match making, normal and ranked. Only All Pick and Captains Mode are available for ranked matches.

Game Modes

Game modes are a set of restrictions within which the game of DotA 2 can be played. Players are given the option of picking what game modes they want to play. Current game modes for public matchmaking include:

- All Pick
- Captains Mode
- Single Draft
- Random Draft
- All Random
- Ability Draft
- All Random Deathmatch

Out of these, we will be preferring data from All Pick and Captains Mode due to these being the most played games as well as the most "complete" games.

Captain's Mode[13]

Captains mode is the standard format for tournament games. The captains ban certain heroes, up to six per team, preventing either team from picking the hero. The captain also chooses five heroes for their team. After the captains choose five heroes, each player chooses a hero from their captain's selections. The starting team is randomly selected in matchmaking; if playing in a private lobby a starting team may be specified. Some heroes that were recently created or tweaked are unavailable in Captains Mode; they will be added eventually.

All Pick[14]

In the All Pick mode, players can choose from the full hero pool. Players can choose any hero, random a hero, re-pick a hero, or swap heroes with teammates. Teams alternate picking. Whenever it is a team's turn to pick, anyone on that team can pick their hero. Once a selection is made, it immediately switches to the other team to pick. Initial starting team is random, but known at the start of the strategy period.

Appendix II

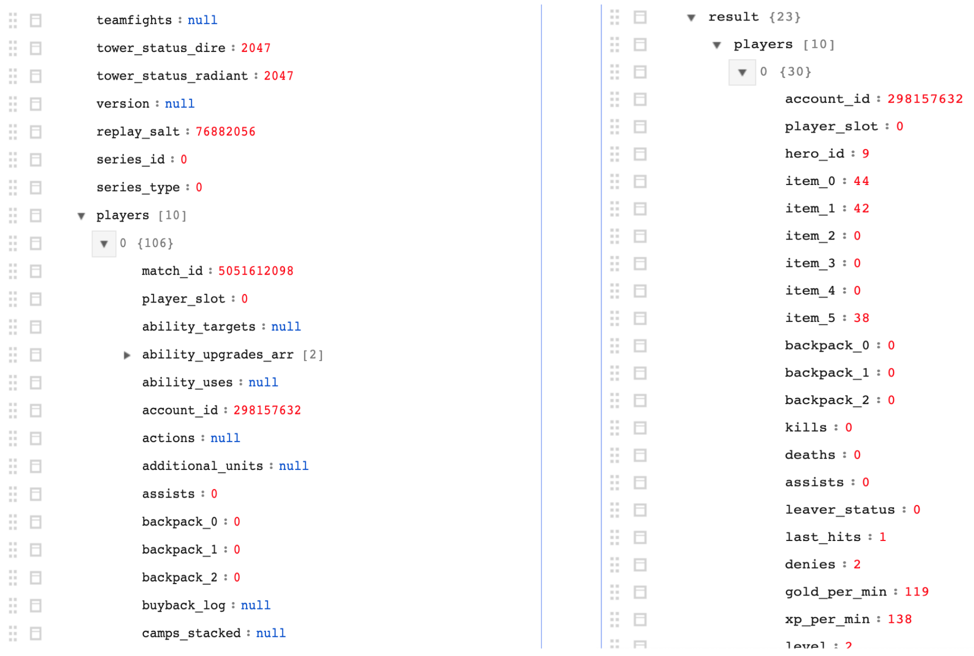
Sample Match Data[9]

```
{
    "result": {
        "players": [
            {
                "account_id": 4294967295,
                "player_slot": 1,
                "hero_id": 99,
                "item_0": 44,
                "item_1": 0,
                "item_2": 182,
                "item_3": 0,
                "item_4": 0,
                "item_5": 0,
                "backpack_0": 0,
                "backpack_1": 0,
                "backpack_2": 0,
                "kills": 0,
                "deaths": 0,
                "assists": 0,
                "leaver_status": 0,
                "last_hits": 8,
                "denies": 5,
                "gold_per_min": 227,
                "xp_per_min": 250,
                "level": 3,
                "hero_damage": 768,
                "tower_damage": 0,
                "hero_healing": 0,
                "gold": 682,
                "gold_spent": 590,
                "scaled_hero_damage": 933,
                "scaled_tower_damage": 0,
                "scaled_hero_healing": 0,
                "ability_upgrades": [
                    {
                        "ability": 5549,
                        "time": 132,
                        "level": 1
                    },
                    {
                        "ability": 5550,
                        "time": 274,
                        "level": 2
                    },
                    {
                        "ability": 5549,
                        "time": 361,
                        "level": 3
                    }
                ]
            }, …
        ],
        "radiant_win": true,
        "duration": 177,
        "pre_game_duration": 90,
        "start_time": 1570047664,
        "match_id": 5051612098,
        "match_seq_num": 4237617145,
        "tower_status_radiant": 2047,
        "tower_status_dire": 2047,
        "barracks_status_radiant": 63,
        "barracks_status_dire": 63,
        "cluster": 184,
        "first_blood_time": 6,
        "lobby_type": 0,
        "human_players": 10,
        "leagueid": 0,
        "positive_votes": 0,
        "negative_votes": 0,
        "game_mode": 4,
        "flags": 0,
        "engine": 1,
        "radiant_score": 3,
        "dire_score": 0
    }
}
```

OpenDota - Sample Data of a Match[10]

Sample data available here

Comparison of the two sources of data



```
teamfights : null
tower_status_dire : 2047
tower_status_radiant : 2047
version : null
replay_salt : 76882056
series_id : 0
series_type : 0
▼ players [10]
  ▼ 0 {106}
      match_id : 5051612098
      player_slot : 0
      ability_targets : null
    ▶ ability_upgrades_arr [2]
      ability_uses : null
      account_id : 298157632
      actions : null
      additional_units : null
      assists : 0
      backpack_0 : 0
      backpack_1 : 0
      backpack_2 : 0
      buyback_log : null
      camps_stacked : null
```

```
▼ result {23}
  ▼ players [10]
    ▼ 0 {30}
        account_id : 298157632
        player_slot : 0
        hero_id : 9
        item_0 : 44
        item_1 : 42
        item_2 : 0
        item_3 : 0
        item_4 : 0
        item_5 : 38
        backpack_0 : 0
        backpack_1 : 0
        backpack_2 : 0
        kills : 0
        deaths : 0
        assists : 0
        leaver_status : 0
        last_hits : 1
        denies : 2
        gold_per_min : 119
        xp_per_min : 138
        level : 2
```

Figure - Per player, OpenDota(left) has 106 attributes versus 30 from Steam(right)

Appendix III

FaunaDB Clustering Basics[11]

FaunaDB uses a mesh-oriented approach to clustering: Deploy a node, point it to its peer, and go. The system takes care of everything else.

The key components of a FaunaDB cluster are:

1. Node
2. Replica
3. Shard/partition

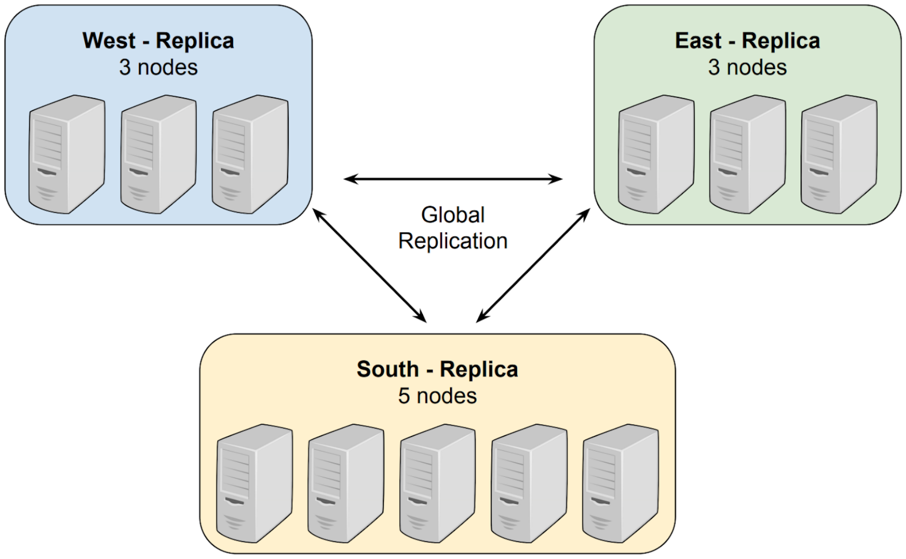


Figure - FaunaDB example cluster

Fauna's physical data layout can be described entirely by three simple concepts, node, replica, and cluster.

Node

A node is a single computer with its own IP address in which FaunaDB is installed and running. The Fauna nodes can run on all public cloud services, including Amazon, Azure, Bluemix, and Google, in addition to running on a private cloud, an on-premise solution, a virtual machine, a docker image, and several other platforms.

Replica

A replica is comprised of a group of one or more node(s). A node can belong only to a single replica. A fundamental property of a replica is that it contains a complete copy of the Fauna data, including end user data, system data, and the transaction log. Within a replica, a particular piece of data is normally found on exactly one node; there is no data redundancy provided by Fauna within a replica. Having multiple replicas, each containing the full set of data, is what provides redundancy in a Fauna database cluster. Another common term for a replica would be a datacenter.

Cluster

A FaunaDB cluster is the highest entity. Under the cluster is both a physical layout and logical layout. To review, the physical layout is comprised of individual nodes. Every node is a computer with a unique IP address. Nodes are grouped into replicas, with every node belonging to exactly one replica. The main significance for this grouping is that a replica contains a full copy of the data and can vary in size. Within a replica, a particular piece of data is normally found on exactly one node; there is no data redundancy within a replica.

The FaunaDB Cluster has a logical data model comprised of databases, collections, and documents. This data model is a semi-structured, schema-free object-relational data model, a strict superset of the relational, document, object-oriented, and graph paradigms. The model starts with the database which can be laid out in traditional flat fashion, in a hierarchical or a combination of flat and hierarchical as seen in Figure 2.

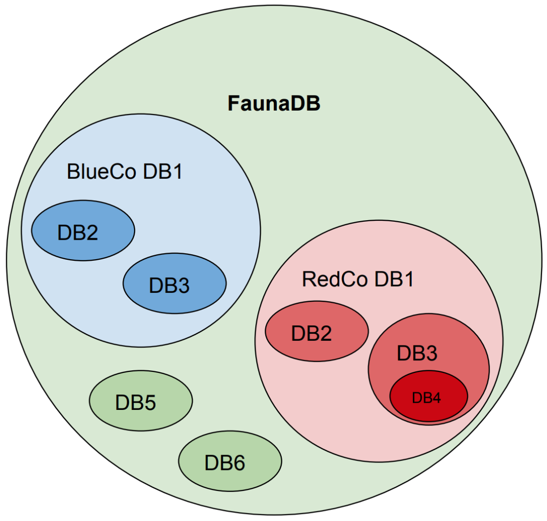


Figure - Database which can be laid out in traditional flat fashion, in a hierarchical or a combination of flat and hierarchical

Appendix IV

Docker Setup

```
# Docker setup in Debian
# Reference - https://www.puzzlr.org/install-docker-on-a-google-cloud-virtual-machine/

sudo apt update
sudo apt install --yes apt-transport-https ca-certificates curl gnupg2
software-properties-common
curl -fsSL https://download.docker.com/linux/debian/gpg | sudo apt-key add -
sudo add-apt-repository "deb [arch=amd64] https://download.docker.com/linux/debian
$(lsb_release -cs) stable"
sudo apt update
sudo apt install --yes docker-ce

# To allow docker commands from non-root users
sudo usermod -aG docker $USER
logout
```

FaunaDB and Dashboard Setup

```
# Install Node.js for developer dashboard
curl -sL https://deb.nodesource.com/setup_12.x | sudo -E bash -
sudo apt-get install -y nodejs

# FaunaDB setup with Docker images
# Reference - https://gist.github.com/CaryBourgeois/ebe08f8819fc1904523e360746a94bae

# Fetch image
docker pull fauna/faunadb

# Test image
docker run fauna/faunadb --help

# Create a Docker network
docker network create fauna-network

# See the list of networks (fauna-network should be in the list)
docker network ls

# Initialize the cluster
docker run --network=fauna-network -d --rm --name faunadb_node0 -p 8443:8443
fauna/faunadb

# Inspect the network created
docker network inspect fauna-network (Replace fauna-network with your-network-name)

# Login to the node to find the internal IP used
docker exec -it faunadb_node0 /bin/bash
```

```
# Check status and get the internal IP
# Can be found under the Address column
/faunadb/bin/faunadb-admin -k secret status

# Start second node and join it to the cluster
# --run command is used to start the node, but not initialize it
docker run --network=fauna-network -d --rm --name faunadb_node1 -p 8444:8443
fauna/faunadb --run
docker exec -it faunadb_node1 /bin/bash
/faunadb/bin/faunadb-admin join -r NoDC <ip-address-here>

# Repeat for third node
docker run --network=fauna-network -d --rm --name faunadb_node2 -p 8445:8443
fauna/faunadb --run
docker exec -it faunadb_node2 /bin/bash
/faunadb/bin/faunadb-admin join -r NoDC <ip-address-here>

# Run the developer dashboard
# Login to the cluster using port http://<public-ip-address>:8443 and 'secret' as they key
git clone https://github.com/fauna/dashboard
cd dashboard/
npm install
nohup node start > dashboard-output.log &
```

References

[1] https://en.wikipedia.org/wiki/Dota_2

[2] https://dota2.gamepedia.com/Abilities

[3] https://docs.opendota.com

[4] Steam Community Documentation: https://steamcommunity.com/dev/apiterms

[5] FAQ section under https://dev.dota2.com/showthread.php?t=58317

[6] https://dbdb.io/db/faunadb

[7] https://docs.fauna.com/fauna/current/introduction

[8] https://fauna.com/faunadb

[9] Full JSON - https://jsonblob.com/c65dd64c-e63e-11e9-8b82-0b0ad04e74ce

[10] Full JSON - https://jsonblob.com/f44f33d2-e63f-11e9-8b82-dd807557ebd2

[11] https://fauna.com/blog/introduction-to-faunadb-clusters

[12] https://dota2.gamepedia.com/Matchmaking

[13] https://dota2.gamepedia.com/Game_modes#Captains_Mode

[14] https://dota2.gamepedia.com/Game_modes#All_Pick

[15] https://dev.dota2.com/showthread.php?t=58317