APACHE CASSANDRA

Research and Implementation

ABSTRACT
Summarize on Apache Cassandra database including introduction, architecture, data model and applications.

Yu Fu
CS 7280

Contents

Research and Implementation on Apache Cassandra

Introduction

Apache Cassandra

Cassandra is an open-source distributed NoSQL database system, which was originally developed by Facebook, aims to improve the search performance of email systems for simple format data, combining the data model of Google BigTable with the fully distributed architecture of Amazon Dynamo. The name Cassandra came from Greek mythology, the name of a tragic prophetess of Troy, so the logo of the project is a shining eye.

Logo of Apache Cassandra

In 2008 summer, Cassandra was discharged by Facebook on Google code as an open-sourced project. Then the project got to be an Apache Hatchery extend in Spring 2009. After graduation from Facebook, Cassandra become an official top-level project in ASP (Apache Software Foundation) on Feb 17, 2010.

The Official Website Link: https://cassandra.apache.org/_/index.html

Architecture of Apache Cassandra

Bigtable & Dynamo

At first, Apache Cassandra was planned to execute making a combine of Amazon's Dynamo (Distributed capacity and replication techniques) together with Google's Bigtable (Data and storage engine model) by applying a staged even-driven architecture (SEDA).

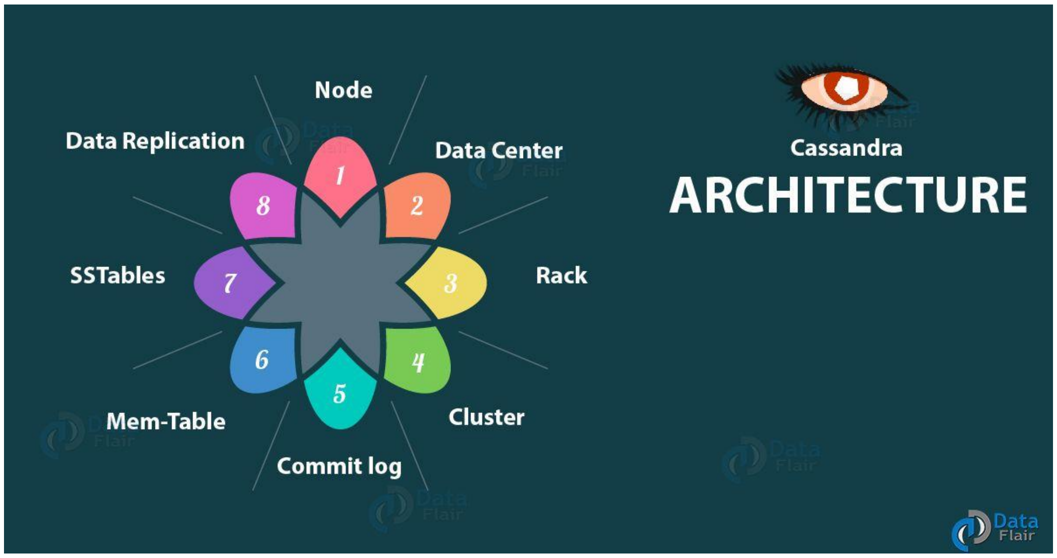
Even though Dynamo and Bigtable is designed for scalable, reliable and highly available storage systems, they still have limitations in these aspects.

Apache Cassandra has similar architecture like Amazon's Dynamo for its distributed system.

Consistent Hashing

The system architecture of Apache Cassandra is a node-to-node distributed system which based on the consistent Hashing. All nodes in the cluster are equal. The data is distributed among the nodes in the cluster, and each node exchanges information every second. Each row of data is hashed to determine which node or nodes it should exist in.

The system is a ring-type architecture which makes its nodes are logically distributed like a ring. The state synchronization between clusters is carried out through the Gossip protocol for communication. Each node stores data locally, and each node accepts requests from clients. It consists of a ring structure, and its nodes are logically distributed like a ring. So, it has no master or leaf nodes. It makes copies of data on several homogenous nodes of the cluster.



Cassandra Architecture a Complete Guide

CAP Theorem Problem

Cassandra is as flexible as Dynamo when it comes to the compromise of consistency, availability, and partition tolerance (CAP). Since high availability comes first in web-based applications, Cassandra choose Availability and Partition Tolerance as its target goal while data Consistency may be lost. Each keyspace of Cassandra can configure how many nodes a row of data will be written to ensure that data is not lost due to machine downtime or disk damage.

Data Modeling

Column-based NOSQL

Data modeling is query-driven in Apache Cassandra. The application queries can make simple selections on key and decide structure and organization of the data. The data model of it is one key per row. After each row of data is uniquely identified by the row key, there can be up to 2 billion columns, each column is identified by a column key, and each column key corresponds to several values.

All columns in Apache Cassandra must be grouped together in a single table as Table Join is not supported in it just like relational database model. Tables in Apache Cassandra can also be called as column families.

Data Model Components

The data model components of Cassandra are three key points**: keyspace, tables**, and **columns**, which put together to make the data storage.

Keyspace

Keypace makes up the data model in highest data level. It was built up by group of data, which is like the scheme in a relational database. Simply speaking, one keyspace can have several column families, which can also be called tables.

Table (column families)

The tables are created after keyspace are defined. Data is stored as horizontal and vertical formats in it, which can be called as rows and columns. Column families encompass many columns and a primary key, where data can be put in storage in a set of rows.

Column

Columns identify the structure of data in column families, which denotes a single piece of data in Cassandra and has a data type defined, for instance, integer, text, double, and Boolean. Besides, Cassandra offers collection types like set, list, and map. Additional, column values come up with an associated time stamp representative the time of update. This timestamp can be recalled applying the function **writetime**.

Features of Apache Cassandra

Apache Cassandra has a lot of great features, which made it so popular among the worlds. I will discuss about some of its main features to explain why I choose Apache Cassandra as my research target with some application cases.

Popular features of Cassandra

Firstly, Cassandra is highly scalability can be added more hardware to adapt more and more clients and data according to the needs of requirements. This kind of scalable interests me as I always tend to a system which has more freedom on adding things.

Secondly, Cassandra is a distributed system, so every node can serve any request. It does not have any single point of failure, and the data can be copied to many other nodes to make it suitable for these applications that cannot afford this failure. I believe a system which owns the fault tolerant would be more fitting for the business application.

Besides, Cassandra can support a lot of kinds of data structure, which ensure its flexibility on the data storage. Also, it can run very fast to input data and store huge amounts of data devoid of lowing data read proficiency. Cassandra's data consistency strategy is configurable, and we can choose between strong consistency or higher-performance eventual consistency according to the client needs.

Application Use Cases

Applications which need to store user information based on the time index like internet user clicks on web page, the GPS location. Etc. These applications need a huge number of data input and data reading, which make Cassandra be a perfect choice for them.

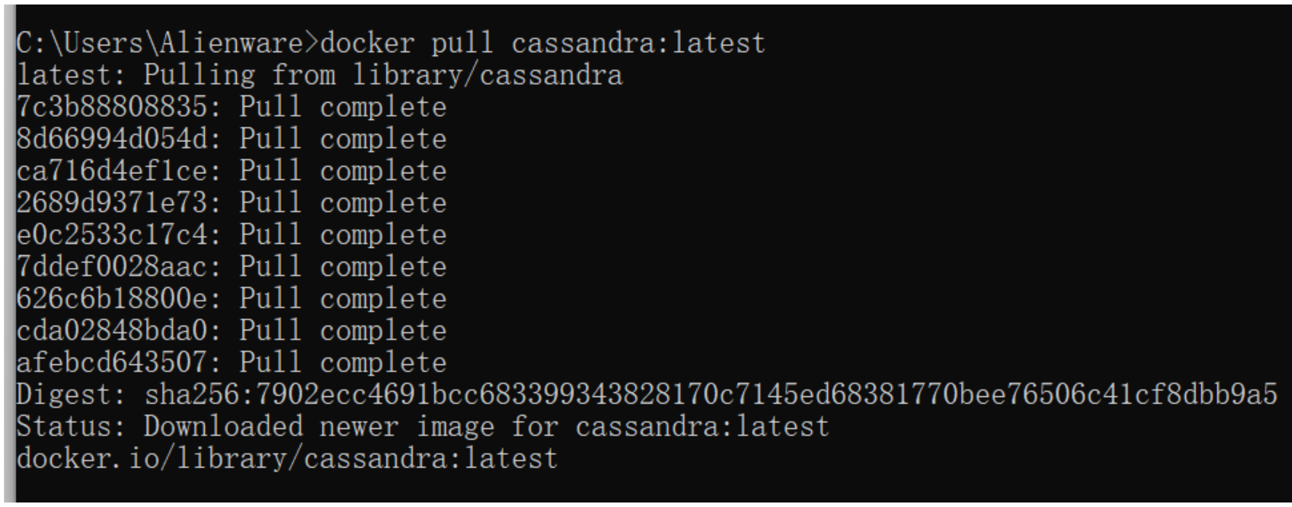
Social Media applications may also need Cassandra to help them analysis the data from different resources. Corporation can use it to search and analysis the data information from the users, so that they can recommend more related advertisements to the users.

The chatting application can use Cassandra as a database management system since people send a lot of messages and the features of Cassandra make it good to handle the large message data to offer the messaging service.

Database Installation

1. Firstly we went to the official website download link: https://cassandra.apache.org/_/quickstart.html
   Follow the step instructions on getting started:

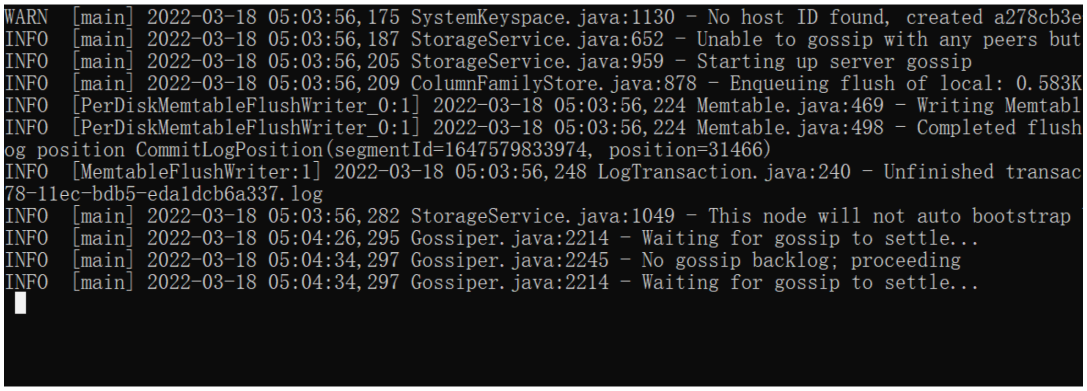
   DOCKER PULL CASSANDRA:LATEST



```
C:\Users\Alienware>docker pull cassandra:latest
latest: Pulling from library/cassandra
7c3b88808835: Pull complete
8d66994d054d: Pull complete
ca716d4ef1ce: Pull complete
2689d9371e73: Pull complete
e0c2533c17c4: Pull complete
7ddef0028aac: Pull complete
626c6b18800e: Pull complete
cda02848bda0: Pull complete
afebcd643507: Pull complete
Digest: sha256:7902ecc4691bcc683399343828170c7145ed68381770bee76506c41cf8dbb9a5
Status: Downloaded newer image for cassandra:latest
docker.io/library/cassandra:latest
```

2. Then we start the Cassandra by running

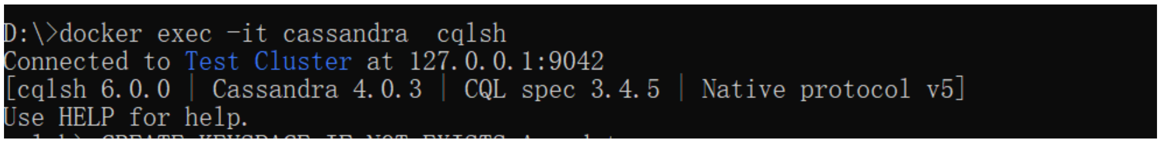
   docker run --name cassandra cassandra



```
WARN  [main] 2022-03-18 05:03:56,175 SystemKeyspace.java:1130 - No host ID found, created a278cb3e
INFO  [main] 2022-03-18 05:03:56,187 StorageService.java:652 - Unable to gossip with any peers but
INFO  [main] 2022-03-18 05:03:56,205 StorageService.java:959 - Starting up server gossip
INFO  [main] 2022-03-18 05:03:56,209 ColumnFamilyStore.java:878 - Enqueuing flush of local: 0.583K
INFO  [PerDiskMemtableFlushWriter_0:1] 2022-03-18 05:03:56,224 Memtable.java:469 - Writing Memtabl
INFO  [PerDiskMemtableFlushWriter_0:1] 2022-03-18 05:03:56,224 Memtable.java:498 - Completed flush
og position CommitLogPosition(segmentId=1647579833974, position=31466)
INFO  [MemtableFlushWriter:1] 2022-03-18 05:03:56,248 LogTransaction.java:240 - Unfinished transac
78-11ec-bdb5-eda1dcb6a337.log
INFO  [main] 2022-03-18 05:03:56,282 StorageService.java:1049 - This node will not auto bootstrap
INFO  [main] 2022-03-18 05:04:26,295 Gossiper.java:2214 - Waiting for gossip to settle...
INFO  [main] 2022-03-18 05:04:34,297 Gossiper.java:2245 - No gossip backlog; proceeding
INFO  [main] 2022-03-18 05:04:34,297 Gossiper.java:2214 - Waiting for gossip to settle...
```

3. Lastly, we install the docker image to start the CQL shell:

   docker exec -it cass_cluster cqlsh



```
D:\>docker exec -it cassandra  cqlsh
Connected to Test Cluster at 127.0.0.1:9042
[cqlsh 6.0.0 | Cassandra 4.0.3 | CQL spec 3.4.5 | Native protocol v5]
Use HELP for help.
```

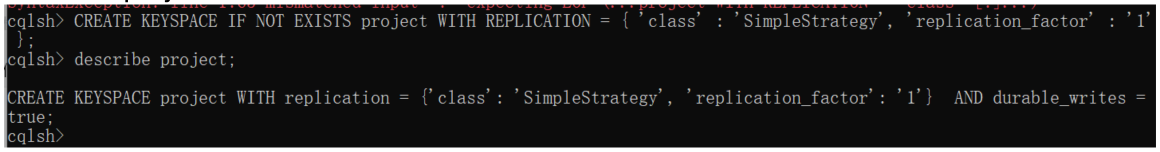
Data Definition and Data Manipulation

CQL shell

As we mentioned before in Data Model section, keyspace and tables are the central points. This part will list and discuss the statement which can create, update information and remove the data in keyspace and tables.

Create Keyspace

CREATE KEYSPACE IF NOT EXISTS project WITH REPLICATION = { 'class' : 'SimpleStrategy', 'replication_factor' : '1' };
describe project;

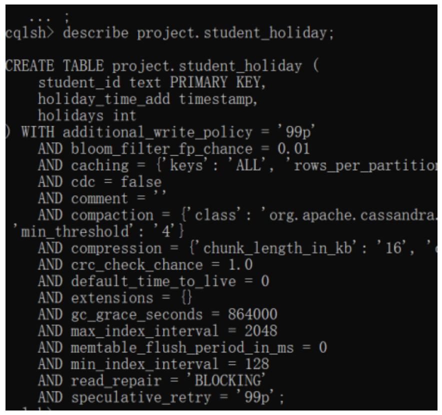


```
cqlsh> CREATE KEYSPACE IF NOT EXISTS project WITH REPLICATION = { 'class' : 'SimpleStrategy', 'replication_factor' : '1'
 };
cqlsh> describe project;

CREATE KEYSPACE project WITH replication = {'class': 'SimpleStrategy', 'replication_factor': '1'}  AND durable_writes =
true;
cqlsh>
```

Create a table

CREATE TABLE IF NOT EXISTS project.student_holiday (
student_id text PRIMARY KEY,
holidays int,
holiday_time_add timestamp
);
describe project.student_holiday;



```
   ... ;
cqlsh> describe project.student_holiday;

CREATE TABLE project.student_holiday (
    student_id text PRIMARY KEY,
    holiday_time_add timestamp,
    holidays int
) WITH additional_write_policy = '99p'
    AND bloom_filter_fp_chance = 0.01
    AND caching = {'keys': 'ALL', 'rows_per_partitio
    AND cdc = false
    AND comment = ''
    AND compaction = {'class': 'org.apache.cassandra.
 'min_threshold': '4'}
    AND compression = {'chunk_length_in_kb': '16', '
    AND crc_check_chance = 1.0
    AND default_time_to_live = 0
    AND extensions = {}
    AND gc_grace_seconds = 864000
    AND max_index_interval = 2048
    AND memtable_flush_period_in_ms = 0
    AND min_index_interval = 128
    AND read_repair = 'BLOCKING'
    AND speculative_retry = '99p';
```

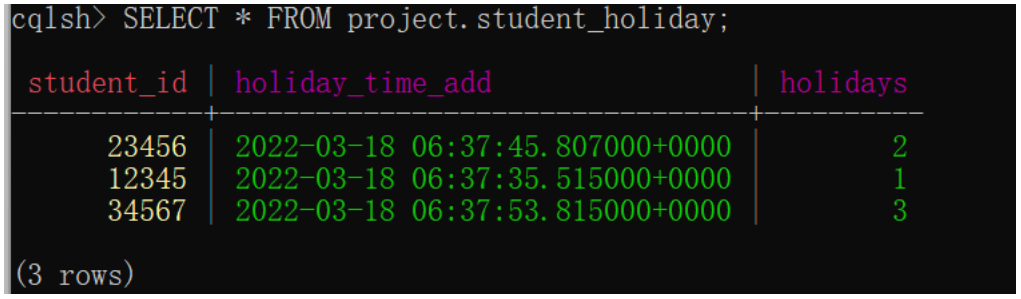
Insert some data

INSERT INTO project.student_holiday
(student_id, holidays, holiday_time_add)
VALUES ('12345', 1, toTimeStamp(now()));

INSERT INTO project.student_holiday
(student_id, holidays, holiday_time_add)
VALUES ('23456', 2, toTimeStamp(now()));

INSERT INTO project.student_holiday
(student_id, holidays, holiday_time_add)
VALUES ('34567', 3, toTimeStamp(now()));

SELECT * FROM project.student_holiday;



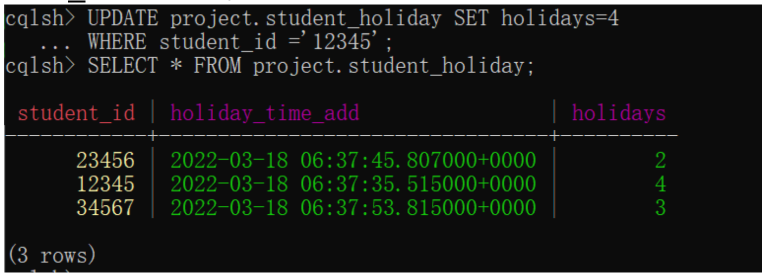
```
cqlsh> SELECT * FROM project.student_holiday;

 student_id | holiday_time_add                | holidays
------------+---------------------------------+----------
      23456 | 2022-03-18 06:37:45.807000+0000 |        2
      12345 | 2022-03-18 06:37:35.515000+0000 |        1
      34567 | 2022-03-18 06:37:53.815000+0000 |        3

(3 rows)
```

Update Operation

UPDATE project.student_holiday SET holidays=4
WHERE student_id ='12345';



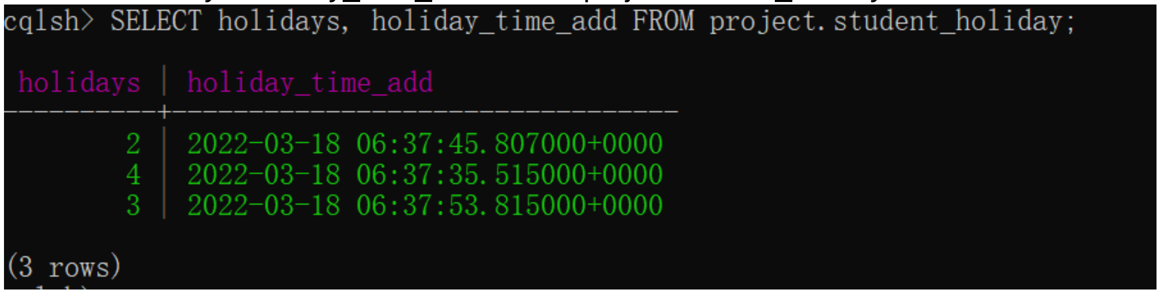
```
cqlsh> UPDATE project.student_holiday SET holidays=4
   ... WHERE student_id ='12345';
cqlsh> SELECT * FROM project.student_holiday;

 student_id | holiday_time_add                | holidays
------------+---------------------------------+----------
      23456 | 2022-03-18 06:37:45.807000+0000 |        2
      12345 | 2022-03-18 06:37:35.515000+0000 |        4
      34567 | 2022-03-18 06:37:53.815000+0000 |        3

(3 rows)
```

Read Data

SELECT holidays, holiday_time_add FROM project.student_holiday;



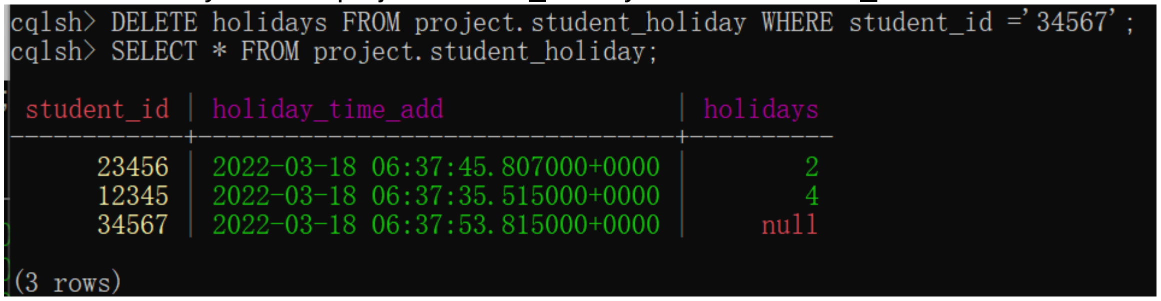
```
cqlsh> SELECT holidays, holiday_time_add FROM project.student_holiday;

 holidays | holiday_time_add
----------+---------------------------------
        2 | 2022-03-18 06:37:45.807000+0000
        4 | 2022-03-18 06:37:35.515000+0000
        3 | 2022-03-18 06:37:53.815000+0000

(3 rows)
```

Delete data from table

DELETE holidays FROM project.student_holiday WHERE student_id ='34567';



```
cqlsh> DELETE holidays FROM project.student_holiday WHERE student_id ='34567';
cqlsh> SELECT * FROM project.student_holiday;

 student_id | holiday_time_add                | holidays
------------+---------------------------------+----------
      23456 | 2022-03-18 06:37:45.807000+0000 |        2
      12345 | 2022-03-18 06:37:35.515000+0000 |        4
      34567 | 2022-03-18 06:37:53.815000+0000 |     null

(3 rows)
```

Drop Keyspace

DROP KEYSPACE project;

Summary

There have been many examples of Cassandra being used effectively. Banks and other financial institutions are using it to store large amounts of financial data. Analytics companies are using Cassandra to store web analytics data. Medical companies are using Cassandra to store sensor data and other time-series data. There are also many companies applying Cassandra to store IoT data. Cassandra not only absorbs the successful experience of how to do distribution, do copy replication, and fault tolerance from Amazon, but also grasps the essence of the google Bigtable. Therefore, since its release, it has been constantly popular among people.

Meanwhile, it also has unpredictable performance because all background tasks are executed in a random way, not scheduled by the user. The data is modeled around the query, not around its structure. There are many different pros and cons to using Cassandra, and it depends on how we will use it.

Reference List

- Simplilearn.com. (2021). Apache Cassandra Data Model: Components And Statements. [online] Available at: https://www.simplilearn.com/tutorials/big-data-tutorial/cassandra-data-model.

- DataFlair. (2018). Cassandra Architecture and It's Key Terms - Complete Guide. [online] Available at: https://data-flair.training/blogs/cassandra-architecture/ [Accessed 18 Mar. 2022].

- cassandra.apache.org. (n.d.). Data Definition | Apache Cassandra Documentation. [online] Available at: https://cassandra.apache.org/doc/latest/cassandra/cql/ddl.html [Accessed 18 Mar. 2022].

- cassandra.apache.org. (n.d.). Data Manipulation | Apache Cassandra Documentation. [online] Available at: https://cassandra.apache.org/doc/latest/cassandra/cql/dml.html [Accessed 18 Mar. 2022].

- classics.mit.edu. (n.d.). The Internet Classics Archive | The Aeneid by Virgil. [online] Available at: http://classics.mit.edu/Virgil/aeneid.7.vii.html [Accessed 18 Mar. 2022].

- maps-legacy.org. (n.d.). Cassandra in the Classical World. [online] Available at: http://maps-legacy.org/poets/a_f/bogan/classical.htm [Accessed 18 Mar. 2022].

- The Role of Women in the Art of Ancient Greece. (n.d.). Cassandra, Ancient Princess of Troy, Priestess and Prophetess. [online] Available at: http://www.rwaag.org/cassandra [Accessed 18 Mar. 2022].

- Pausanias, Description of Greece. (n.d.). Pausanias, Description of Greece, Corinth, chapter 16, section 6. [online] Available at: http://www.perseus.tufts.edu/hopper/text?doc=Perseus:text:1999.01.0160:book=2:chapter=16:section=6 [Accessed 18 Mar. 2022].