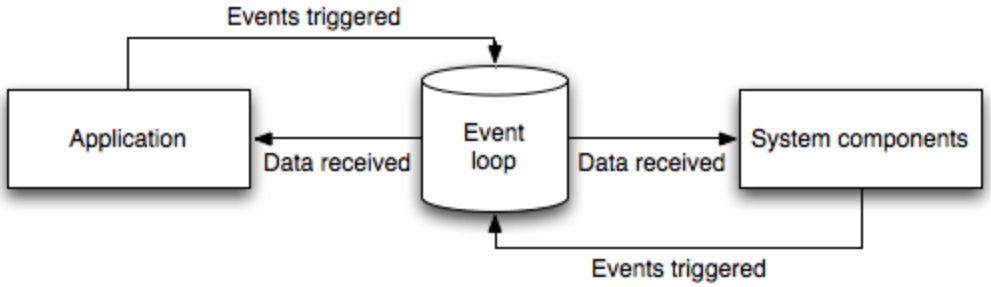
Programming Assignment 1: Asterix and the Olympic Games (Winter Olympics Edition)

Rahul Ramakrishna

Architecture

Event Based Model

The application is based on event based model. Every single action is categorized as an event. A Single Channel registers to a set of events. Once an event is dispatched, channels subscribing to that event perform an action.



Components

1. **Director Server**
   Primary server which holds both the information of events and scores. Its a giant Event Dispatch engine, which gets triggered on various request. It supports both push and pull model of data fetching.

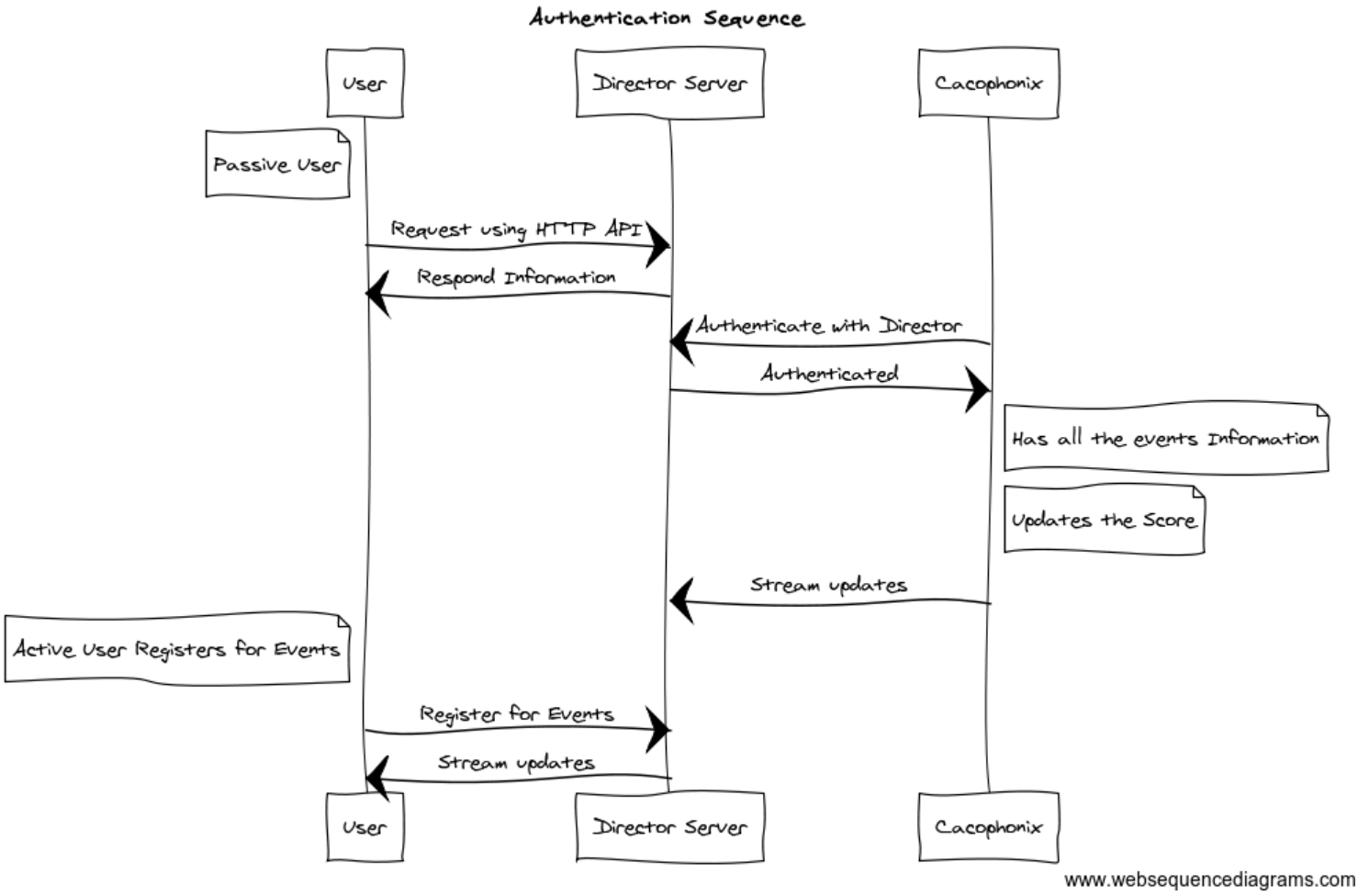
2. **Cacophonix Sever**
   The Cacophonix server Streams updates to Director Server . It offers Command line options where user can mention which event to update etc. and the content is directly streamed to Director Server.

3. **Client(s)**

   There are mainly 2 types of client. Push and Pull based client. Pull based client can be easily executed using **curl command** . Since Director server is processes HTTP based request. Its very easy to get information using HTTP call.

   Push based client is another process which subscribes to a particular event and if the director server receives any of those events, it pushes the content to the subscribed clients.

Sequence Diagram



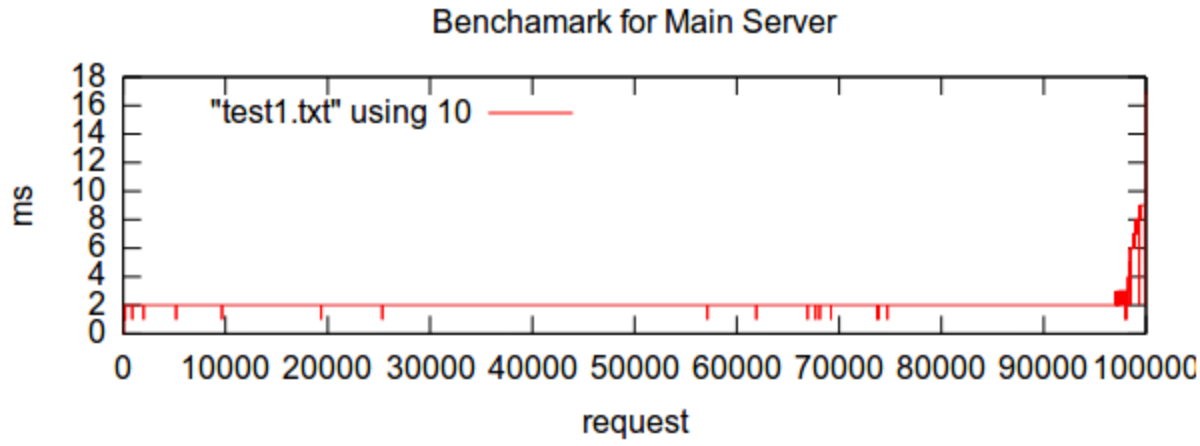
Storage Structure

Score Object : stores : (EventName, Team1_ Score , Team2 _ Score)	**Store Object** Stores: (TeamName, Gold, Bronze, Silver)
Methods Exposed: Score.add() Score.getByEventName()	**Methods Exposed**: Store. Add() Store.getByTeamName() Store.getByGold() Store.incrementMedalTally()

Experiment Setup

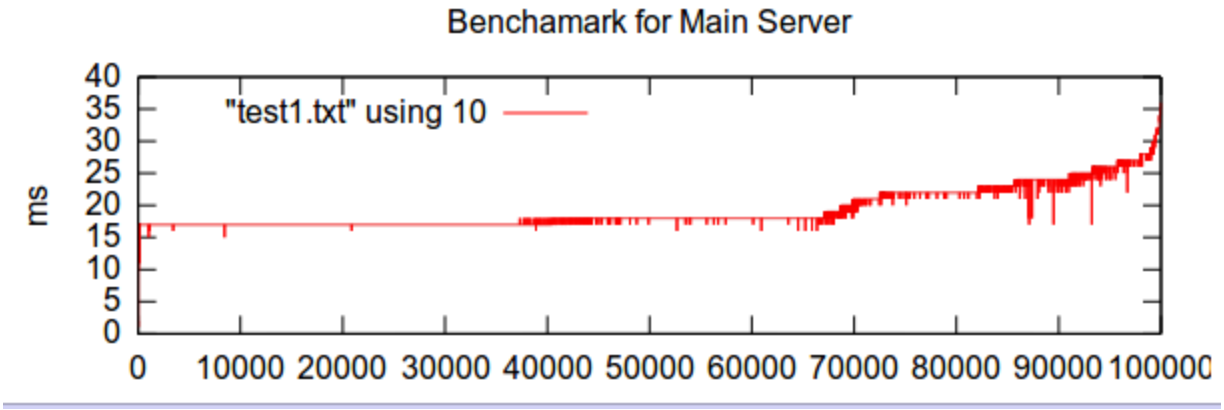
The Apache Benchmark Tool was used in order to hit the server with various get requests . To get team informations and event scores.

Experiment 1 :
Max No of Request / Sec : 100000
Concurrency : 10



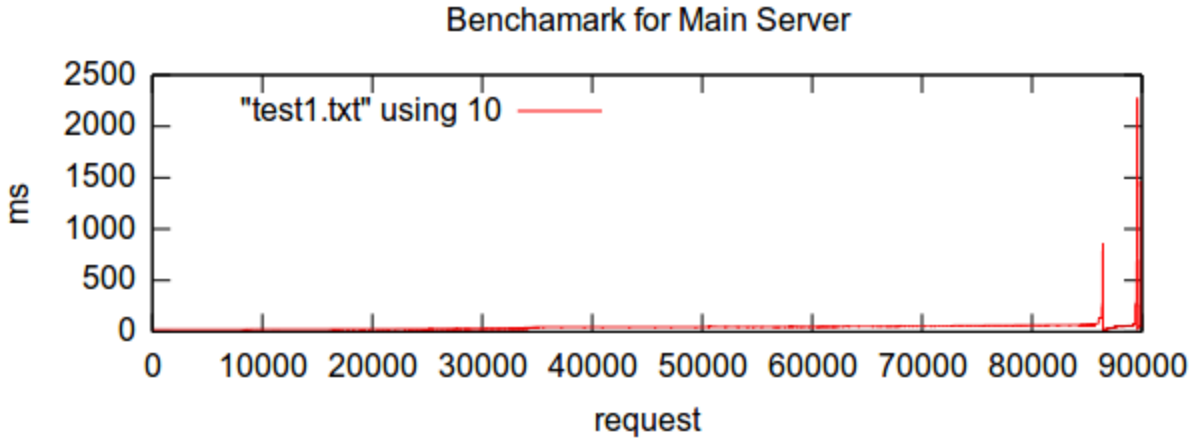
As we can see, the server can easily respond to almost all (10k) the request in less than 10ms.

Experiment 2 :
Max No of Request / Sec : 100000
Concurrency : 100



As we can see, when 100 concurrent clients we present. We see, the response time is increased to a range of 15ms - 30ms and rising.

Experiment 3:
Max No of Request / Sec : 90000
Concurrency: 500



Beyond this, the system was not stable, which is mostly due to hardware limitations.

Test Cases

Jasmine-Node testing framework was used in order to test the various url matching pattern in REST API of the main directory_Server .

Case 1:

```
var request = require('request');

it("should respond with rome info", function(done) {
  request("http://localhost:3000/getinfo/rome", function(error, response, body){
    done();
  });
}, 250);
```

Case 2:

```
var request = require('request');

it("should respond with curling score info", function(done) {
  request("http://localhost:3000/getscore/curling", function(error, response, body){
    done();
  });
}, 250);
```

Similar to this lot many test cases were performed based on the HTTP API response and unit tests for Store and Score Objects

Extensions and Future Work

1. Extensible Storage

   Currently , the storage is very specific to the given problem. It does not scale well for various uses cases like a data store should do. For example, if we need to perform a combined query similar to a database, the current storage method will not be able to do so since they are not very well integrated.

2. Robustness of code

   The Code needs to be little more robust that now. If the System is forced to scale beyond **C10k ,** the nodejs engine tends to core_dump, this could also be due to the fact that I am running on a 32 bit system. It may behave a bit differently on a 64bit system.

Instructions to Run .

1. Start the directory_server first. (Its the main server which handles all the connections from client as well the cacophonix servers). Please refer to director_server instructions. **Default port : 8080** .
2. Start Cacophonix server or Active/ Passive Client irrespective of the order you want. Please refer below for various methods for Passive Client and how to start the other servers.

Passive Client

```
rahul@g3ck0:~$ curl http://localhost:8080/getinfo/rome
[{"TeamName":"rome","gold":0,"bronze":0,"silver":0}]

rahul@g3ck0:~$ curl http://localhost:8080/getinfo/gual
[{"TeamName":"gual","gold":0,"bronze":0,"silver":0}]

rahul@g3ck0:~$ curl http://localhost:8080/getscore/curling
[{"eventname":"curling","rome":0,"gual":0}]

rahul@g3ck0:~$ curl http://localhost:8080/getscore/skiing
[{"eventname":"skiing","rome":0,"gual":0}]
```

Active Client

```
USAGE: node socketio_client.js [OPTIONS] , where OPTIONS are:
  -e, --event_type <ARG1> Name of event (curling,skiing)
  -h, --host <ARG1> Name of the director_server (give localhost if same system)
(mandatory)
```

Director Server

```
rahul@g3ck0:~/programs/dos_homework/hw1$ nodejs director_server.js
                   Welcome to Winter Olympics
                          ---------- __o
                        --------  _ <,_
                      -------    (*)/ (*)
```

Cacophonix Server

```
USAGE: node cacofonix.js [OPTIONS] , where OPTIONS are:
  -a, --action_type <ARG1>      Name of action (inc_medal,set_score)
  -e, --event_name <ARG1>       Name of event
  -t, --teamname <ARG1>         Name of the team
  -m, --medal_type <ARG1>       Type of medal example gold,silver,bronze
  -s, --score <ARG1> <ARG2>     enter scores of 2 teams (rome , gual)
  -i, --interval <ARG1>         No of times you need to send update
  -h, --host <ARG1>             localhost if same server (mandatory)


*Note : Case-sensitive options

-i (Interval options) Could be used to how many times you want to dispatch the data.
```

Case 1: Update score for event curling

```
rahul@g3ck0:~/programs/dos_homework/hw1$ nodejs  cacofonix.js -a set_score -e curling -s
1 1  -h localhost
```

```
Likewise, you can do for skiing.
```

```
Note: *Due to current design limitation, both the team scores needs to be updated at same
time. i.e option -s needs to be give 2 arguments.
```

Case 2: Increment Medal for Rome

```
rahul@g3ck0:~/programs/dos_homework/hw1$ nodejs  cacofonix.js -a inc_medal -t rome  -m
gold -h localhost
Likewise for team “gual” and other medals like “silver” , “bronze”
```

Limitations

1. The options passed to Cacophonix serves are case-sensitive.
2. The command line arguments are not group for Cacophonix servers ie. the program does not enforce option combinations like using { -a , -t , -m } together similar for event updating {-a , -e, -s} .