

TP TLC :

Google App Engine : Advertisement Board

Github Link :

<https://github.com/masoumiaa/GoogleCloudScalabilityMeasure>

Students : SOUMIAA Mhamed-Amine & Brossault Guillaume
Professor : FREY David
Université de Rennes 1 - ISTIC - M2 ILA
2017/2018

Introduction :

- Google App Engine is a Platform as a Service (PaaS) and it is Google's internal infrastructure exposed as a cloud platform. It stands quite different from the rest of the cloud platforms. It allows hosting of web applications in Google managed data centers , applications are executed virtually across multiple servers and data centers . The architecture of GAE is very complex and it spawns over a million of servers, which are distributed geographically across the globe.
- GAE uses Bigtable, which is a distributed system for storing data. Bigtable is also being used internally at Google for different products such as Google Analytics, Google Finance, Google Earth, Google Search Index etc. It has been designed to provide sustained performance with scalability and availability.

Task 1 :

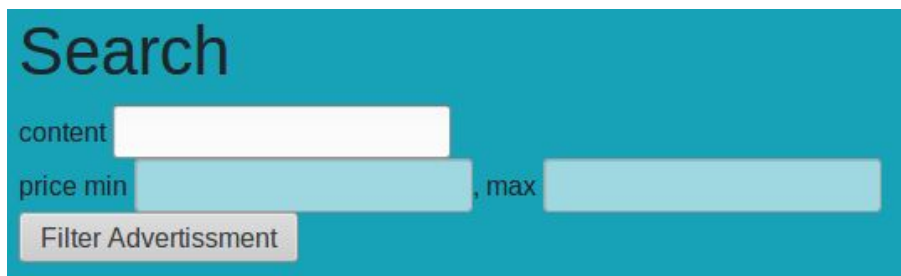
- To develop this application, we use the base we have implemented at the Task0. We have modified it to present advertisement.

- **The URL to access the application is :**

<https://tlcproject-194513.appspot.com/advertisementBoard.jsp?advertisementName=default&filterValue=&priceMin=&priceMax=>

- The view is composed with three parts :

- **A search form :**



We have created a form allowing users to filter the results. If the content of ads is equal to content field, it will be displayed. For the price, the settings are managed by the application but we did not succeed in fixing the index errors generated by appengine.

```
EntityQuery query =
    Query.newEntityQueryBuilder()
        .setKind("Advertisement")
        .setFilter(hasAncestor(key))
        .setFilter(PropertyFilter.eq("content", filterForName))
        .setOrderBy(desc("date"))
        .setLimit(5)
        .build();
```

- A list view of the ads :

Advertisements			
User	Ads Description	Price	
An anonymous person	greg	50.0	Delete
An anonymous person	coucou	999.0	Delete
An anonymous person	coucou	15.0	Delete
An anonymous person	retgerqg	10.0	Delete

This part shows the ads that have been filtered, they can be deleted.
We have revised the servlet management to split the treatment of the
different actions. Here is the delete action :

```
private void deleteAds(HttpServletRequest req, HttpServletResponse resp) throws IOException {
    String advertissmentName = req.getParameter("advertissmentName");
    // Create a Key given its URL safe encoded form.
    Key adsKey = Key.fromUrlSafe(req.getParameter("key"));
    Greeting gr = new Greeting();
    gr.delete(adsKey);
    resp.sendRedirect("/advertissmentBoard.jsp?advertissmentName="+advertissmentName+"&filte
}
```

- A form allowing to add advertisement :

Add advertisement

hello world

999

Post Greeting

It contains two fields to enter the title and the price of the add.

Advertisements			
User	Ads Description	Price	
g.brossault@hotmail.fr (You)	hello world	999	Delete

The advertisement has been added.

Task 2 :

- Measure the Scalability of Google App Engine :
 - In this task, we tried to investigate the scalability of the Google App Engine . we submit many http requests to your advertisement board application to perform bulk add,search, and bulk delete functions that we have implemented in the first part.
 - For each request, we measured the round-trip latency for receiving responses from the advertisement board application.
 - This is an example of java code used for one http request to search for ads which content is equal to 'facebook' :

```
private static void measureSearchingAds() throws IOException{
    String url = "https://tlc-tp1-194513.appspot.com/advertismentBoard.jsp"
        + "?advertismentName=default&filterValue=facebook";
    URL search = new URL(url);
    long startTime = System.currentTimeMillis();
    BufferedReader in = new BufferedReader(
        new InputStreamReader(search.openStream()));
    String inputLine;
    while ((inputLine = in.readLine()) != null)
        System.out.println(inputLine);
    in.close();

    long endTime = System.currentTimeMillis();
    System.out.println("\n - Round trip latency " + (startTime - endTime) + "ms");
}
```

- The round trip latency of http requests, is calculated by subtracting the recorded start time (that is just before the program sends an http request) and the end time immediately after the program receives the response from the web server.

- Results :

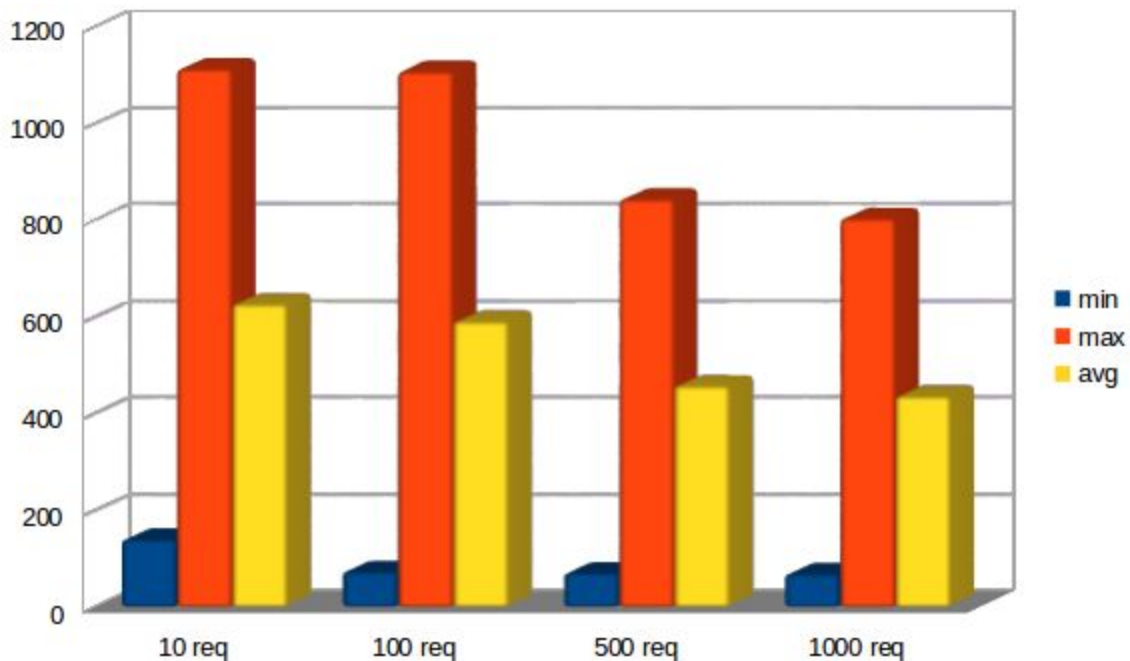
```
static PrintWriter writer ;
public static void main(String[] args) throws IOException {

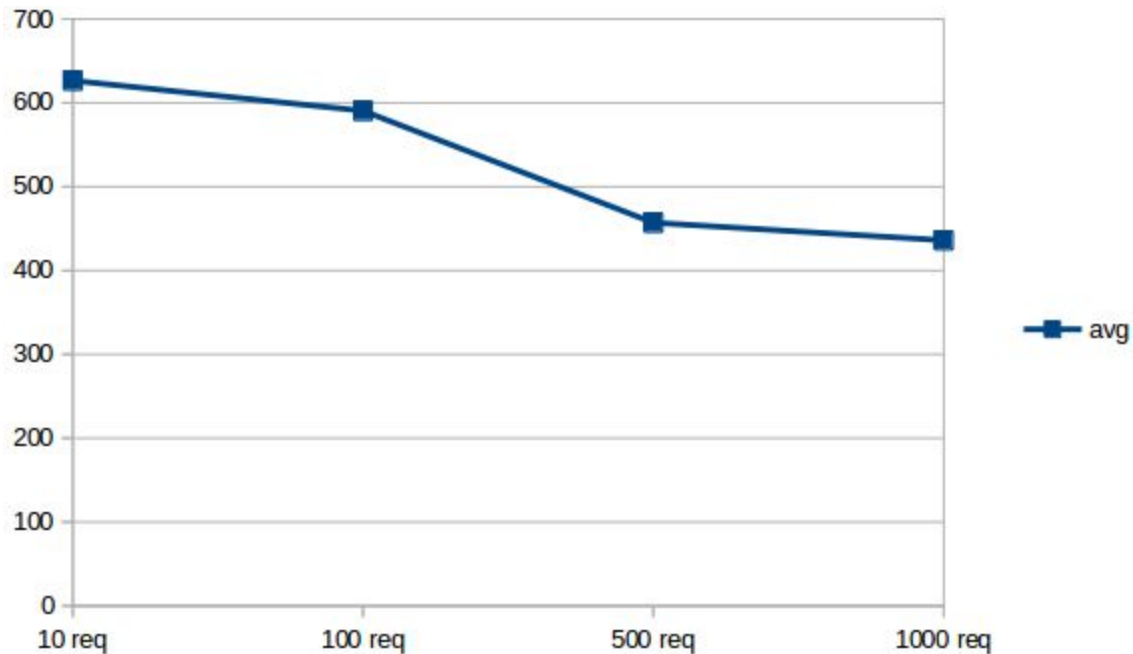
    writer = new PrintWriter("Searchresults1000.txt", "UTF-8");
    writer.println("");

    for(int i=0;i<1000;i++){
        measureSearchingAds();
    }
    writer.close();
}
```

- The total number of concurrent requests used in the different scenarios ranges from 10 to 1000. Average, maximum and minimum response time was noted in files(.txt) during these scenarios for different number of concurrent request generate.
- **First scenario- Search :** (*time with ms)

	10 req	100 req	500 req	1000 req
min	140	73	70	68
max	1113	1108	844	804
avg	626.5	590.5	457	436





- **The first chart shows the maximum and minimum response time of the prototype application on the GAE platform. The maximum response time is 1113 ms, recorded for 10 request scenario, and the minimum response time is 68 ms, recorded for the 1000 request scenario.**
- **From the second chart, we can see that average time of response decrease when we increase requests number.**
- Note : decreasing the amount of logging in the application, decrease immediately the requests latency and can help to avoid reaching the 60 second deadline (Google App engine have a request handler that limits the amount of time to generate and return a response to a request, typically around 60 seconds. Once the deadline has been reached, the request handler is interrupted)
- Quotas :

- Google App Engine automatically allocates resources to your application as traffic increases. However, this is bound by the following restrictions:
 - App Engine reserves automatic scaling capacity for applications with low latency, where the application responds to requests in less than one second. Applications with very high latency (over one second per request for many requests) and high throughput requires paying account. Customers with this level of support can request higher throughput limits by contacting their support representative.
 - Applications that are heavily CPU-bound may also incur some additional latency in order to efficiently share resources with other applications on the same servers. Requests for static files are exempt from these latency limits.
- *Secure incoming bandwidth :*

The amount of data received by the application over a secure connection from requests. Secure incoming bandwidth also counts toward the Incoming Bandwidth quota.

Resource	Free Default Limit		Billing Enabled Default Limit	
	Daily Limit	Maximum Rate	Daily Limit	Maximum Rate
Outgoing Bandwidth (billable , includes HTTPS)	1 GB	56 MB/minute	1 GB free; 14,400 GB maximum	10 GB/minute
Incoming Bandwidth (includes HTTPS)	1 GB; 14,400 GB maximum	56 MB/minute	None	None

- Search API resources are also limited :

Resource or API Call	Free Quota
Total Storage (Documents and Indexes)	0.25 GB
Queries	1000 queries per day
Adding documents to Indexes	0.01 GB per day

- Limitations :
 - **Response size limits**
 - Dynamic responses are limited to 32MB. If a script handler generates a response larger than this limit, the server sends back an empty response with a 500 Internal Server Error status code. This limitation does not apply to responses that serve data from the Blobstore or Google Cloud Storage .
 - **Streaming Responses**
 - App Engine does not support streaming responses where data is sent in incremental chunks to the client while a request is being processed. All data from your code is collected as described above and sent as a single HTTP response.