Fall 16 15619 Project Phase 2 Report

Performance Data and Configurations

<table>
<tr><th>Best Configuration and</th><th>Results from the Live Test</th></tr>
<tr><td>Number and type of instances</td><td>HBase Live Test:6 m4.large
MySQL Live Test: 6 m4.large</td></tr>
<tr><td>Cost per hour
(assume on-demand prices)</td><td>HBase Live Test:0.78
MySQL Live Test: 0.752</td></tr>
<tr><td>Queries Per Second (QPS)</td><td>INSERT HERE: (Q1,Q2H,Q2M,Q3H,Q3M)
<table>
<tr><td></td><td>Q1</td><td>Q2H</td><td>Q2M</td><td>Q3H</td><td>Q3M</td></tr>
<tr><td>score</td><td>6.25</td><td>12.435</td><td>12.5</td><td>12.5</td><td>2.3725</td></tr>
<tr><td>tput</td><td>32215.8</td><td>10734.9</td><td>13551.0</td><td>6471.8</td><td>1138.5</td></tr>
<tr><td>ltcy</td><td>2</td><td>4</td><td>3</td><td>7</td><td>43</td></tr>
<tr><td>corr</td><td>100</td><td>93</td><td>95</td><td>100</td><td>100</td></tr>
<tr><td>err</td><td>0</td><td>0.36</td><td>0.36</td><td>0</td><td>0</td></tr>
</table></td></tr>
<tr><td>Rank on the scoreboard:</td><td>Q1(Higher One):10
Q2H:19
Q2M:18
Q3H:1
Q3M:49
HBase Live Test:1
MySQL Live Test:45</td></tr>
</table>

Team :

MyLittlePony

Members :

Jiexin, Guo,jiexing
Wei-Yu, Yen, wyen
Yingxue, Mei, yingxuem

Rubric:

Each unanswered question = -7%

Each unsatisfactory answer = -3%

[Please provide an insightful, data-driven, colorful, chart/table-filled, and interesting final report. This is worth a quarter of the grade for Phase 2. Use the report as a record of your progress, and then condense it before sharing it with us. Questions ending with "Why?" need evidence (not just logic)]

Task 1: Front end (you may/should copy answers from your earlier report-- each report should form a comprehensive account of your experiences building a cloud system. Please try to add more depth and cite references for your answers from the P1Report)

Questions

1. Which front end framework did you use? Explain why you used this solution. [Provide a small table of special properties that this framework/platform provides].

Undertow httphandler because its performance is better comparing to using servlet.

	Reliability	Loading	Performance
Undertow httphandler	Easy to shutdown after test for a while	Lightweight, just peek the part you needed	Better
Tomcat servlet	More reliable	Heavyweight compare to Undertow httphandler	

2. Did you change your framework after Phase 1? Why or why not?

No, because the performance of this framework is enough and once we change to another framework, it will take time for us to learn to develop. And the performance of new framework is not known to us.

3. Explain your choice of instance type and numbers for your front end system.

For this time, for the reason of being stable, we decided to use m4.large on demand. It seems that the m4.large's performance is better than m3.large.

4. Explain any special configurations of your front end system.

In undertow, we set the IOThreads to 8 because the m4.large have two cores.

5. Did you use an ELB for the front-end? Why, or why not? Condense your experience with ELB in the next few sentences. Talk about load-balancing in general and why it matters in the cloud.

Yes. Because I don't need to design and implement the load balancer by myself. What I need is just some server that has the same MySQL database and request handler. But I think maybe I will try to use a load balancer that is developed by myself, because the elb needs to be warm up, which may consume more time and if there is a problem we can not figure it out by watching the performance graphs.
Load balance is quite important since the request maybe a large scale and there can be no way that we can only use one frontend to handle all those requests.

6. Did you explore any alternatives to ELB? List a few of these alternatives. What did you finally decide to use? (if possible) Provide some graphs comparing performance between different types of systems.

I could use only one frontend as the load balancer, but use multiple MySQL connections to different backend database instances.

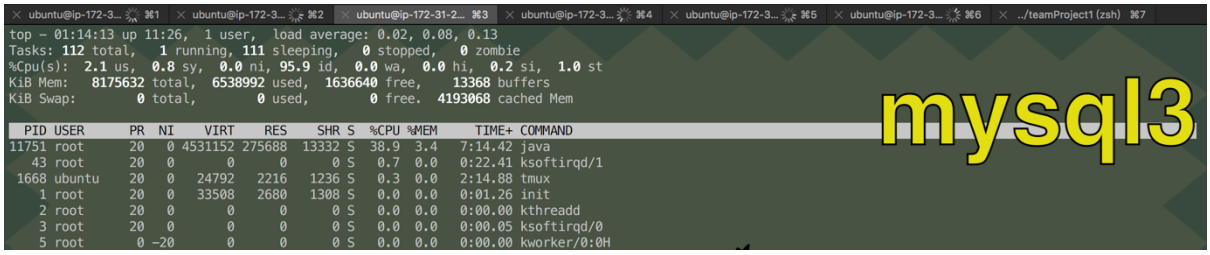
7. Did you automate your front-end instance? If yes, how? If no, why not?

Not really. We have deployed an overall mysql database and frontend server code on one machine and save it as an AMI. Then we can start a new machine and the mysql database is automatically started but we need to run a simple command to start the server. Because I may change the setting of each server, so I do not automate to start the frontend server.

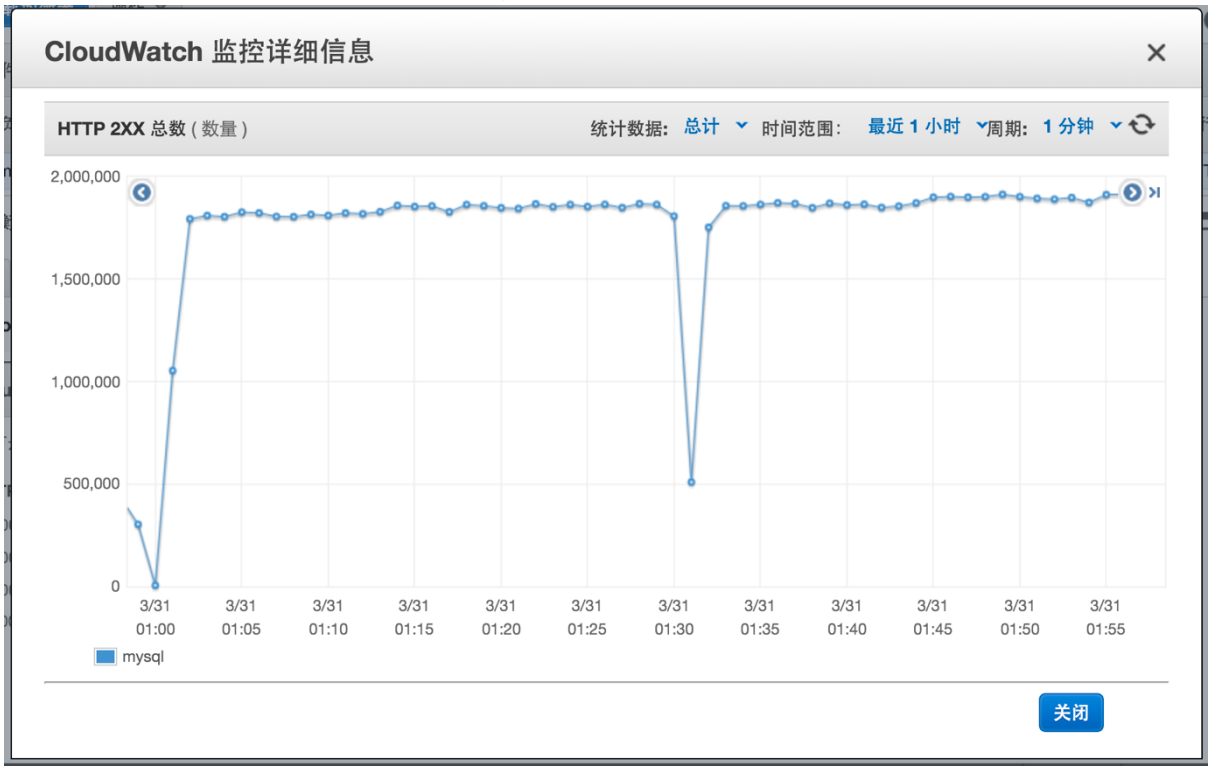
And in HBase, we just have two frontend server so we modify the code frequently. So we choose to deploy our frontend and manually start them, and it is also easily monitor the status when running the (live) test.

8. Did you use any form of monitoring on your front-end? Why or why not? If you did, show us a capture of your monitoring during the Live Test. Else, try to provide CloudWatch logs of your Live Test in terms of memory, CPU, disk and network utilization. Demarcate each query clearly in the submitted image capture.

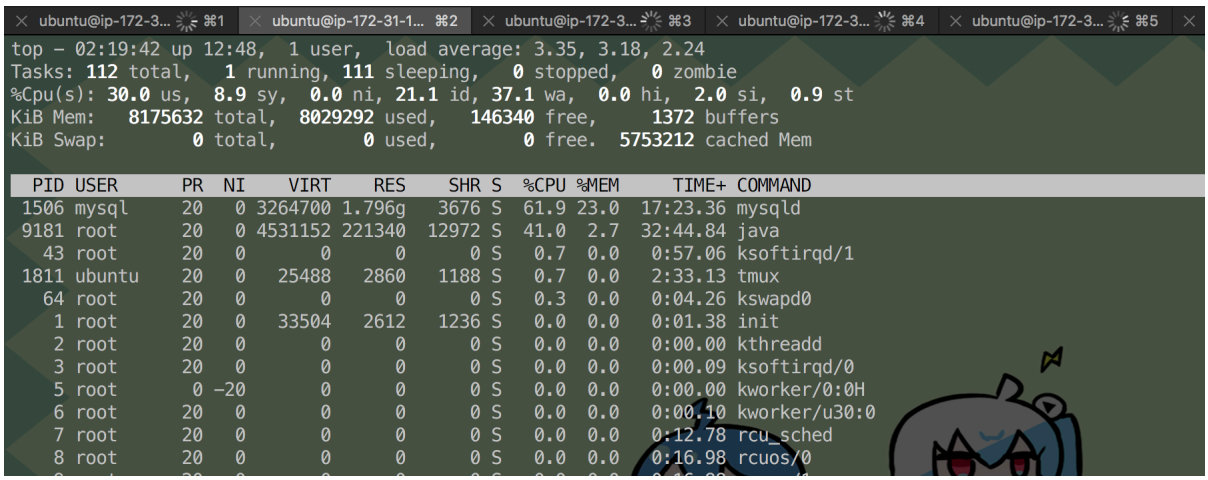
Yes. Like the top command, for MySQL:



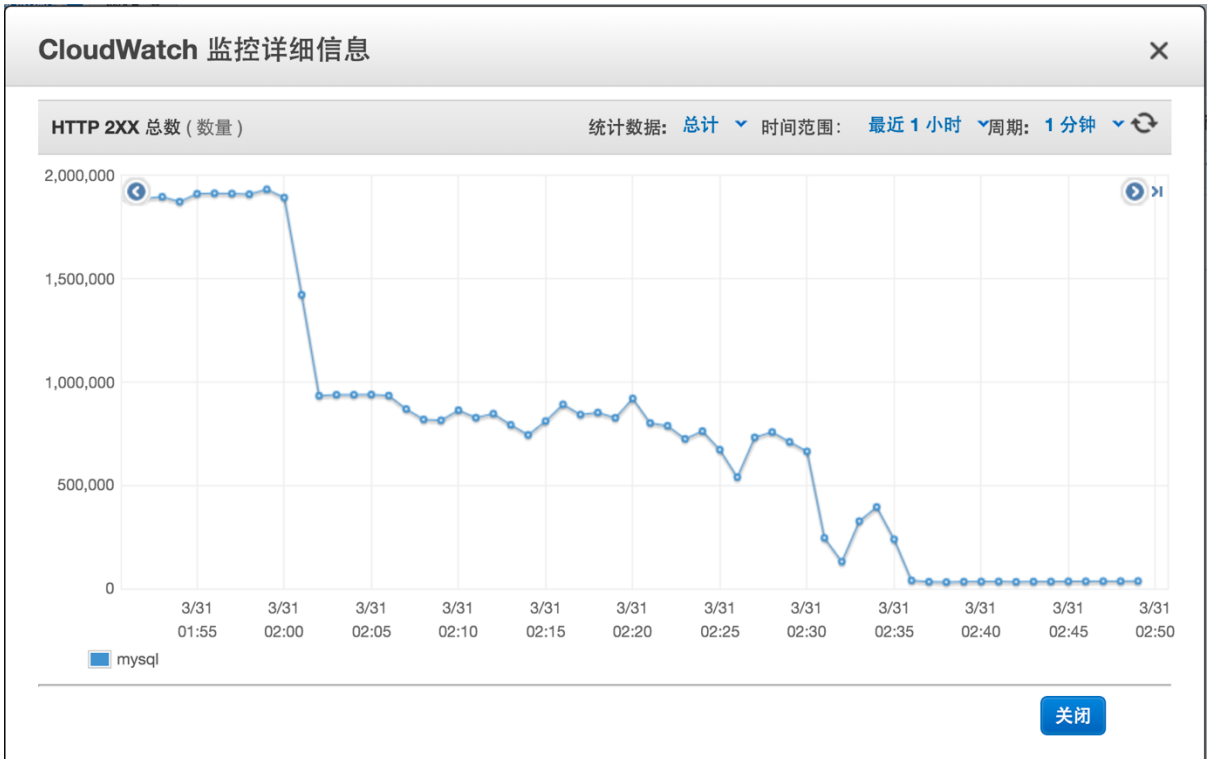
this is in the warm up and the cpu usage of the machine.



this is the cloud watch for http 2XX in both warmup and Q1



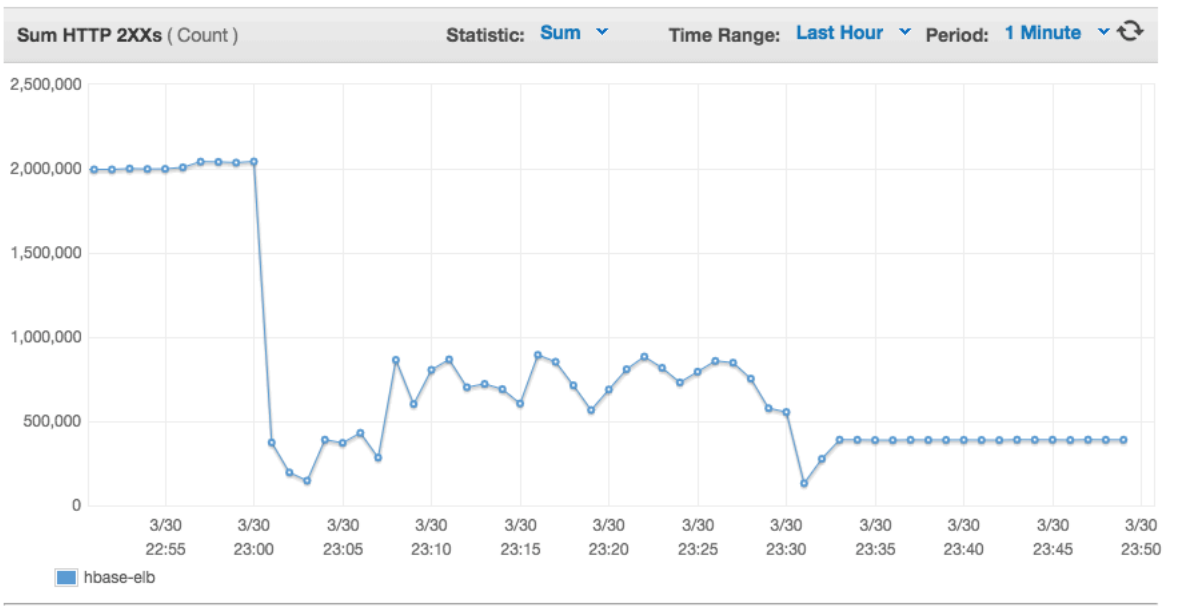
Above is the cpu usage for java and mysql in Q2



Above is the q3 result for http 2XX, our system can work for the first 10 mins, but then it fail like the mysql database is down, but I can not find any result or exception.

PID	USER	PR	NI	VIRT	RES	SHR	S	%CPU	%MEM	TIME+	COMMAND
2398	root	20	0	7907780	128148	15200	S	122.4	1.6	2:19.92	java
43	**root**	**20**	**0**	**0**	**0**	**0**	**R**	**3.3**	**0.0**	**0:50.57**	**ksoftirqd/1**
2365	ubuntu	20	0	105636	1868	896	S	0.3	0.0	0:00.06	sshd
1	root	20	0	33460	2852	1492	S	0.0	0.0	0:01.32	init
2	root	20	0	0	0	0	S	0.0	0.0	0:00.00	kthreadd

The above is CPU usage for HBase Q1.



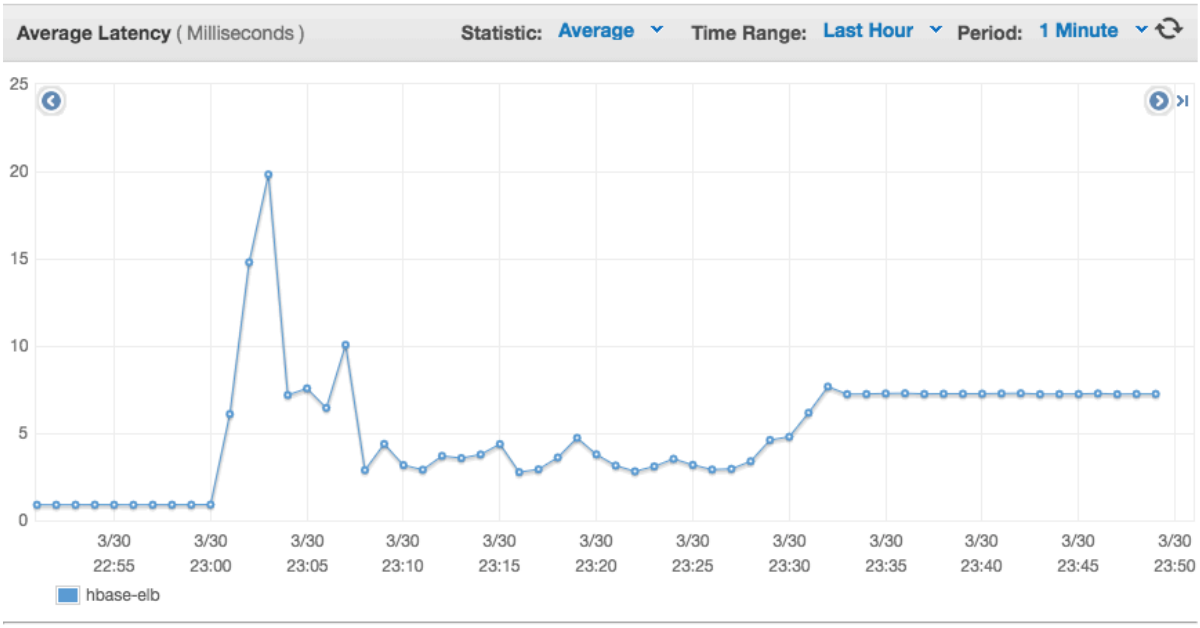
The above graph is the HBase result for http 2XX for Q1(before 3/30 23:00), Q2(between

3/30 23:00 to 3/30 23:30) and Q3 (after 3/30 23:30).

```
top - 23:42:25 up  8:54,  2 users,  load average: 11.59, 11.00, 10.15
Tasks: 110 total,   1 running, 109 sleeping,   0 stopped,   0 zombie
%Cpu(s): 60.8 us, 21.0 sy,  0.0 ni, 12.1 id,  0.0 wa,  0.0 hi,  6.0 si,  0.2 st
KiB Mem:   8175632 total,  2737144 used,  5438488 free,    20428 buffers
KiB Swap:        0 total,        0 used,        0 free.   247484 cached Mem
```

PID	USER	PR	NI	VIRT	RES	SHR	S	%CPU	%MEM	TIME+	COMMAND
3492	root	20	0	7994732	2.241g	15372	S	172.0	28.7	59:00.44	java
43	root	20	0	0	0	0	S	3.3	0.0	4:01.33	ksoftirqd/1
7	root	20	0	0	0	0	S	0.3	0.0	0:08.46	rcu_sched

The above is HBase CPU usage for Q3.



The above graph is the HBase result for Average latency for Q1(before 3/30 23:00), Q2(between 3/30 23:00 to 3/30 23:30) and Q3 (after 3/30 23:30).

9. What was the cost to develop the front end system?

In MySQL, we deploy frontend and backend on the same instance and one ELB. We use 6 m4.large instance. Therefore, the total cost is 0.12 * 6 + 0.025 = $0.745

In HBase, we deploy frontend on 2 instance and use one ELB. Therefore, the total cost is 0.12 * 2 + 0.025 = $0.265.

10. What are the best reference URLs (or books) that you found for your front-end? Provide at least 3.

http://undertow.io/
http://tomcat.apache.org
http://blog.csdn.net/agods/article/details/7801853

In fact, we don't change too much on frontend on phase 2, so the reference is the same as phase I.

[Please submit the code for the frontend in your ZIP file]

Task 2: Back end (database)

Questions

1. Describe your schema for each DB. Explain your schema design decisions. Would your design be different if you were not using this database? How many iterations did your schema design require? Also mention any other design ideas you had, and why you chose this one? Answers backed by evidence (actual test results and bar charts) will be valued highly.

In MySQL, we did not change the table structure in Q2, like:

```
CREATE TABLE `tweetdata` (
   `useridhash` varchar(50) NOT NULL,
   `rawtext` text NOT NULL,
   INDEX index_useridhash (useridhash)
)ENGINE=myisam DEFAULT CHARSET=utf8mb4;
```

We connect the userid and hashtag in one column, and the rawtext column is the response for a specific request. Since one tweet will have multiple hashtag, one tweet may have more than one rows in the database.

Below is our table structure for Q3:

```
CREATE TABLE `q3data` (
   `userid` bigint NOT NULL,
   `datetime` int NOT NULL,
    `wordcount` text NOT NULL,
   INDEX index_useriddate (userid,datetime)
)ENGINE=myisam DEFAULT CHARSET=utf8mb4;
```

First we try to select more integer types because their query performance is better than chars or date. And we build an index on two columns (userid and datetime) because the q3 query is based on both these two columns.

For HBase Q2, we use userid as the key and hashtags1 + raw_texts1 + hashtags2 +raw_texts2 + … concat as the value.
Our table structure for Q2 is

```
create 'tweetdata',{NAME => 'data', IN_MEMORY => true},{SPLITS => ['11','12','13','14','15','16','17','18','19','20','21','22','23']}
```

According to the test data from TPZ, we found that the prefix of most of the query data is between 11 and 23, so we separate the data into 14 regions. One alternative solution is to use userid + hashtag as key and raw_text as value. However, the total number of rows is about 97,000,000 (compare to previous solution is about 17,000,000). Therefore, we decide to use the first one solution.

For HBase Q3, we designed carefully at the beginning. We use user id as key and date + words counts as value.
The stable structure for Q3 is

```
create 'wordcount','data', {NAME => 'data', IN_MEMORY => true},{SPLITS => ['005','0075','01','03','05','075','10','125','15','20','25']}
```

Again, we separate data according to the query data pattern. And this design is easy

to get range data. We add 0 to the prefix of userid if the length of that user id is less than 10 digits. And then we check each value to find the word counts in the date range.

2. Explain your choice of instance type and numbers of your back end system.

For MySQL, we use 6 m4.large, because our frontend and database are deployed on the same machine. And in HBase, we use 2 instance for frontend and 4 instance for HBase cluster (1 Master and 3 Core), all of the instances are m4.large. Because the memory for m4.large is 8G and the CPU is dual core. It is suitable to deal with large amount of cache and calculation.

3. Explain any non-default configuration and parameters you choose to set for your database system.

query_cache_type=1
query_cache_size = 2310720
key_buffer_size=1G
We set these three parameters for MySQL, to use the query cache.
And we use the command in MySQL shell:
LOAD INDEX INTO CACHE q3data,tweetdata IGNORE LEAVES;
To load the index into cache.

In HBase, we set column family IN_MEMORY attribute to true and also do some data partitions,
For Q2,
```
create 'tweetdata',{NAME => 'data', IN_MEMORY => true},{SPLITS => ['11','12','13','14','15','16','17','18','19','20','21','22','23']}
```
For Q3,
```
create 'wordcount','data', {NAME => 'data', IN_MEMORY => true},{SPLITS => ['005','0075','01','03','05','075','10','125','15','20','25']}
```

The in memory cache is the assigned column family can have higher priority to the LRU cache. And the spitted region server and help reduce the workload of each machine.

4. What was the most expensive operation / biggest problem with each DB that you had to resolve for each query? Why does this problem exist in this DB? How did you resolve it? Plot a chart showing the improvements with time.

The query time is too long, so that our db can not get high throughput. I think this is because the data in the database is too large. So we change the database table design and add cache on our frontend. And after our cache is large enough and warmup entirely, the performance improved.

5. Explain (briefly) **the theory** behind (at least) 7 performance optimization techniques for databases. How are each of these implemented in MySQL? How are each of these implemented in HBase? Which optimizations only exist in one type of DB? How can you simulate that optimization in the other (or if you cannot, why not)? Use your own words (paraphrase, this document goes through plagiarism detection software).

1. cache: put all those data from disk into memory since the speed of memory is much higher than disk. This is implemented in MySQL by turning on the query_cache.
2. index: Indexes are used to find rows with specific column values quickly by storing extra information about the data. It is implemented in MySQL using B+ trees.
3. dividing the table: we can divide the whole data into different table, so that the query's

range is limited and could be faster.
4. schema design: we can denormalized the data so the IO will be less and more in-memory calculation in java program. HBase use this technique because it doesn't have index like MySQL.
5. add more instances. add more instance on backend so that each instance has less work load and the total throughput will increase. Both HBase and MySQL use this technique.
6. Tune the data type that store in database. Comparing integer data type in MySQL is faster than other types. But in HBase, the data type is all byte so this optimization can not be performed on HBase.
7. Preload index into database cache. There is one clause in MySQL that can be used directly. For HBase, we can send some request to load some part of the data into the cache.

6. Plot a graph showing results with/without each individual optimization that you used. Extremely impressive will be a timeline of rps v/s submission id (mentioning which optimization was in use at that time).

For the index part, because the database without index is too slow, so we don't have the performance of without index, and all our results are based on database with index.
We didn't have the time to do the dividing table optimization.

optimization name	rps without this optimization	submission id	rps with this optimization	submission id
cache	3239.2	54360	14754.8	61686
index				
dividing table				

7. Would your design work if your web service also implemented insert/update (PUT) requests? Why or why not?

Yes, because if the query's response is cached, we can change it in our cache, but write to the database later, which can maintain our performance. It is also called as write-back method, but it should be applied to the only load balancer otherwise since our database is replication, we have to do some synchronization. But if we can develop a load balancer and distribute the request like using sharding method, we do not have to do synchronization when running the program (but have to do so afterwards).

8. Which API/driver did you use to connect to the backend? Why? What were the other alternatives that you tried? What changed from Phase 1?

JDBC driver. Because it is widely used in online guides. As I know there is hibernate that can also be used to connect to the backend database.

9. Does your usage of HBase maximize the utility of HDFS? How useful is it to have a distributed file system for this type of application/backend? Does it have any significant overhead?

Yes, we distributed all the data to each server evenly. The overhead is that if any server node shutdown, some of the data can not be queried any more.

10. Say you are at any big tech company (Google/Facebook/Twitter/Amazon etc.). List one concrete example of an application/query where they should be using NoSQL versus one where they should be using an RDBMS. Both examples should be based on the same company (you choose).

The company we choose to elaborate on this question is Amazon. The application that should use NoSQL is the log of the error requests/responses or other operations not related to transaction. As the message volume of each log can vary

tremendously, to store the log data in an RDBMS will lead to storage cost that is unnecessary, because an RDBMS requires each column has same data size.

The application should use an RDBMS is the transaction. As the transactions must have the feature as atomicity, which pose the strong-consistency requirement, thus the transaction process should be using RDBMS, which ensures each transaction will complete as a whole or will roll back if any step of the procedure fails, and will ensure read and write query always gets consistent results.

11. How can you reduce the time required to perform scan-reads in MySQL and HBase?

We use joint index for MySQL Q3 requests.

12. Did you use separate tables for Q2-Q3 on HBase and MySQL? How can you consolidate your tables to reduce memory usage?

Yes, but we don't have any issue about memory. So we don't do anything to reduce memory.

13. How did you profile the backend? If not, why not? Given a typical request-response for each query (q2-q3) what percentage of the overall latency is due to:
    a. Load Generator to Load Balancer (if any, else merge with b.)
    b. Load Balancer to Web Service
    c. Parsing request
    d. Web Service to DB
    e. At DB (execution)
    f. DB to Web Service
    g. Parsing DB response
    h. Web Service to LB
    i. LB to LG

    How did you measure this? A 9x4 (Q2H to Q3M) table is one possible representation.

No, because I don't know which kind of measurement can be used to measure these latencies.

14. What was the cost to develop your back end system?

In MySQL, we deploy frontend and backend on the same instance and one ELB. We use 6 m4.large instance. Therefore, the total cost is 0.12 * 6 + 0.025 = $0.745
In HBase, we deploy frontend on 2 instance and use one ELB. Therefore, the total cost is 0.12 * 4 = $0.48.

15. What were the best resources (online or otherwise) that you found. Answer for HBase, MySQL and any other relevant resources.

MySQL
https://www.digitalocean.com/community/tutorials/how-to-use-apachebench-to-do-load-testing-on-an-ubuntu-13-10-vps for apache bench

HBase:
http://hadoopbigdatas.blogspot.com/2013/03/hbase-shell-and-commands.html
http://blog.asquareb.com/blog/2014/11/24/how-to-leverage-large-physical-memory-to-improve-hbase-read-performance/

[Please submit the code for the backend in your ZIP file]

Task 3: ETL

1. For each query, write about:
    a. The programming model used for the ETL job and justification

    Mapreduce model.

    b. The number and type of instances used and justification

    1+8 m3.xlarge

    c. The spot cost for all instances used

    0.15 each, but the actual price is 0.03 each

    d. The execution time for the entire ETL process

    6 hours

    e. The overall cost of the ETL process

    5 dollars

    f. The number of incomplete ETL runs before your final run

    3

    g. Discuss difficulties encountered

    Can not do network stream programming in the mapper, which will lead to out of memory error.

    h. The size of the resulting database and reasoning

    the result data file of etl is 8.6G

    i. The size of the backup

    About 24 G for each backup.

2. What are the most effective ways to speed up ETL? How did you optimize writing to MySQL and HBase? Did you make any changes to your tables after writing the data? How long does each load take?

Add more machines, do less thing in ETL.

3. Did you use EMR? Streaming or non-streaming? Which approach would be faster and why? How can you parallelize loading data into MySQL?

Yes, we use streaming. I think streaming is faster because it is easier to develop. There is a parameter to set the threads that used to load data into MySQL

4. Did you use an external tool to load the data? Which one? Why? If you were to do Q2 (Phase 1) ETL again, what would you do differently?

No, we use the load data clause in MySQL to load the data, because our data format is standard and it can be directly load into the database.

5. Which database was easier to load (MySQL or HBase)? Why?

MySQL, because we can directly import the data csv file, and MySQL is also easier to deploy.

[Please submit the code for the ETL job in your ZIP file]

General Questions

1. Would your design work as well if the quantity of data would double? What if it was 10 times larger? Why or why not?

I don't think so. We are now using replication method to maintain our database. If the data becomes 2 or more times, the scan of the whole database will cost unacceptable time. And in this situation we should use shading method rather than replication.

2. Did you attempt to generate load on your own? If yes, how? And why?

Yes, we can grape the requests from the real request, and we can use apache bench to generate load to our machine.

3. Describe an alternative design to your system that you wish you had time to try.

I can store all the tweets for one userid and one hashtag in one row, and it should reduce the query time because we got exactly one answer each time.

4. Which was/were the toughest roadblock(s) faced in Phase 2?

To warmup and tune.
To make our database stable.

5. Did you do something unique (any cool optimization/trick/hack) that you would like to share with the class?

We make up a large scale of cache on our frontend.
We use connection pool to manage our database connections.

[NOTE:

- **IN YOUR SUBMITTED ZIP FILE, PLEASE DO NOT PUT MORE ZIP OR GZ FILES (NO NESTED ARCHIVES)**
- **USE A SIMPLE FORMAT**
  - **/etl**
  - **/frontend**
  - **/backend**

]