Team Project Report - Phase 1 Microservice 1 Checkpoint

Note: We strongly recommend reading through both checkpoint and final report template before starting the project!

Phase 1, Microservice 1 Checkpoint summary

Team name: **ThreeCobblers**
Members (names and Andrew IDs):
Leo Guo (jiongtig)
Benny Jiang (xinhaoji)
Yukun Jiang (yukunj)

Overview:

In Phase 1, our objective is to make sure all the teams acquire the following skills through various tasks:

1. Microservice 1 and Microservice 2

Learning Objective	**How it will be graded**
Comparing web frameworks	Test submissions, report, throughput
Comparing different types of load balancers	Test submissions, report, throughput
Exploring instance types and number, and architecture	Report, throughput
Web-tier resilience: Malformed requests	Test submissions, manual grading
Web-tier deployment orchestration using Kubernetes, Helm and Terraform	Report, manual grading
Profile and optimize computationally expensive tasks	Report, throughput

2. Microservice 3

Learning Objective	**How it will be graded**
Break down requirements and **design functional database schemas** based on a given dataset	Test submissions, report, throughput

Design and implement effective **ETL pipeline** given the desired output	Budget usage, report
Implement **TDD** by writing unit tests on small data subsets	Report, manual grading
Storage-tier **deployment orchestration** using Kubernetes, Helm, and Terraform	Report, manual grading
Develop **character encoding-aware** solutions	Test submissions
Recognize **skewness** in the Twitter dataset and implement how to address it/remove it	Test submissions
Identify **MySQL/HBase APIs** to use and integrate the APIs into the web service	Test submissions, throughput, manual grading

3. Peripheral skills

Learning Objective	**How it will be graded**
Using git effectively	Report, git statistics
Carrying out code reviews regularly	Report
Efficient task partition to enable parallel development and testing	Report
Timely and effective communication with the team	Report
Effective monitoring resources during the live test	Report
Clear concise documentation through reports	Report, manual grading

Phase 1 Checkpoint 1 Rubric

- Each unanswered bullet point = -4%
- Each unsatisfactory answer = -2%
- Use the report as a record of your progress, and then condense it before submitting it.
- Always use your own words (paraphrase); we will check for plagiarism.

Question 0: Knowing each other

After your first meeting, each member is required to submit the following Google Form before the Microservice 1 Checkpoint Report submission deadline:
https://forms.gle/Co9wnFHSWMuPsVXw6

Question 1: Scope of the project

You may need to read the write-up several times to get a general understanding of tasks. Discuss the following questions as a team:

1. What skills you have learned in previous projects will come into play here?
    1. Container, Docker, and Kubernetes: to achieve platform-independent deployment and large-scale micro-service management
    2. Autoscaling: to best utilize the budget based on the current traffic flow
    3. Data Storage: backend database storage and schema design, optimization
2. Have you used any web framework before? How will you compare the different web frameworks and choose one suitable for this phase? (Answer in less than 100 words)
   We have used Flask and Spring before, but not in-depth.
   For this project, we prefer Java frameworks to Python because we all have experience with Java, and compiled languages are generally faster. Choosing specific frameworks, we prefer ones that are known to be faster. After going through Techempower and tech blogs, we lock on Veter.X and Spring frameworks to try out in Microservice 1.
3. We want you to learn the cloud-native development and deployment approach. Explain the differences between a monolithic application and having multiple microservices.
   A monolithic application is deployed as a whole unit of application and contains all the required functionality and components together. In contrast, microservice is that each individual functional component of the application is broken down into pieces and developed and deployed individually to allow maximal flexibility.
4. The project requires you to work with either MySQL or HBase. List at least three fundamental differences between these two database systems.
   1) MySQL is a relational database with structured data, while HBase manages unstructured data
   2) MySQL typically stores data and serves on a single backend, while HBase supports distributed data storage and query
   3) MySQL could build indexing on the frequently-visited column, while HBase could not since it is not structured and not every data row has that required column.
5. Use one or two short sentences to explain how the index in MySQL works.
   The index in MySQL is like the page mark in a book. We can set and create an index on one or several specific columns, and when we need to query based on that column, we use the index to do a “pointer-dereference-like” operation instead of scanning blindly through the whole database.

6. Use one or two short sentences to explain how HBase finds a row using its data structure.
   Typically in Hbase, a row has a row key, column family, column, and contents. To find a row, the HBase needs to filter based on the column family and to check if a row has the required column. If so, then further check if the content of that column is what the query wants.

Question 2: Team

As mentioned on Piazza, team formation is critical for the success of the team.

- What criteria did you use to choose team members, and when did you begin that process? For each bullet point, please explain in 1 to 2 sentences:
  - Did each member simply accept an invitation to join this team?
    We had a few meetings before to discuss and briefly introduce each other. After having a rough sense of how the team could be well-roundedly constructed, we decide to group together.
  - Did members observe the performance and behavior of other team members? If so, what traits did you observe?
    Mostly each member is very active in participating in the project discussion and implementation. And we all come from Computer Science backgrounds, so we are well aligned on technicality.
  - Did your team follow the recommendations provided by the course staff?
    Yes, we have different expertise before and expect us to complement each other in the project.
  - Did you form the team based on friendship?
    No.
  - Did you follow the questionnaire posted on Piazza to find your teammates?
    No, we reached out to each other through the channel of our program, we asked each other about each member's interest and skillset, and decide if we would be a good match for each other in terms of this team project

Regular meetings are necessary to review status updates, to integrate code and to discuss solutions to possible challenges.

- When will your team hold regular team meetings?
  We hold meetings every Tuesday and Thursday afternoons.
- Are you using a common calendar to notify members about the meetings?
  Yes, and we all have courses on campus on Tuesday and Thursday afternoons, so we naturally meet then.
- Is there a set of action items for each member to work on after the meeting?
  Yes, we plan out the actions and progress as detailed as possible each week.
- Which project management tool will you use to keep track of these tasks? (some options: Asana, Trello)

We use google calendar to set up deadlines and teamwork split each member. We typically use slack and WhatsApp for discussion when remote. And since we are in the same in-person class, we could often discuss the team project after that class.

The team members should have met several times by now. You might have different skill sets and interests in working on various parts of the project. Tell us more about yourselves and your plan of collaboration by answering the following questions.

- Describe the skill sets of each of your team members.
  Benny: experience in software development, taken CS courses in algorithms
  Yukun: experience in database designs and optimization
  Leo: experience in system benchmarking and profiling

- Describe the interests of each of your team members in the project tasks. (we encourage you to go outside your comfort zone and work on tasks that might challenge you and give you the opportunity to learn new skills)
  Benny: Backend web dev, and cluster architecture.
  Yukun: Web frame development and database optimization
  Leo: Microservice pipeline design and optimization, and ETL
- Which collaboration platform do you plan to use for communication (e.g., Asana, Slack, Trello, etc.)? Why did you choose this platform? How do you plan to use this platform?
  We typically use slack and WhatsApp for discussion when remote since they are the most popular social media for daily usage. We will set checkpoints and milestones and synchronize with each other on the collaboration platform. And since we are in the same in-person class, we could often discuss the team project after that class.

The team project is known to require a heavy workload. The division of labor has to be done in a careful manner to make sure that everyone is contributing to the progress of the team.

- How do you plan to divide the work and why?
  The ideal case would be to evenly divide the work and assign the one to each other that they are most confident with, though a certain level of flexibility will be allowed here.
- Which part will each of you work on?
  Right now we roughly decide that Benny will mostly work on Java backend application development, Yukun and Leo will each take care of one web frame for experiments. Yukun will take database design and Leo ETL.

Question 3: Design and exploration

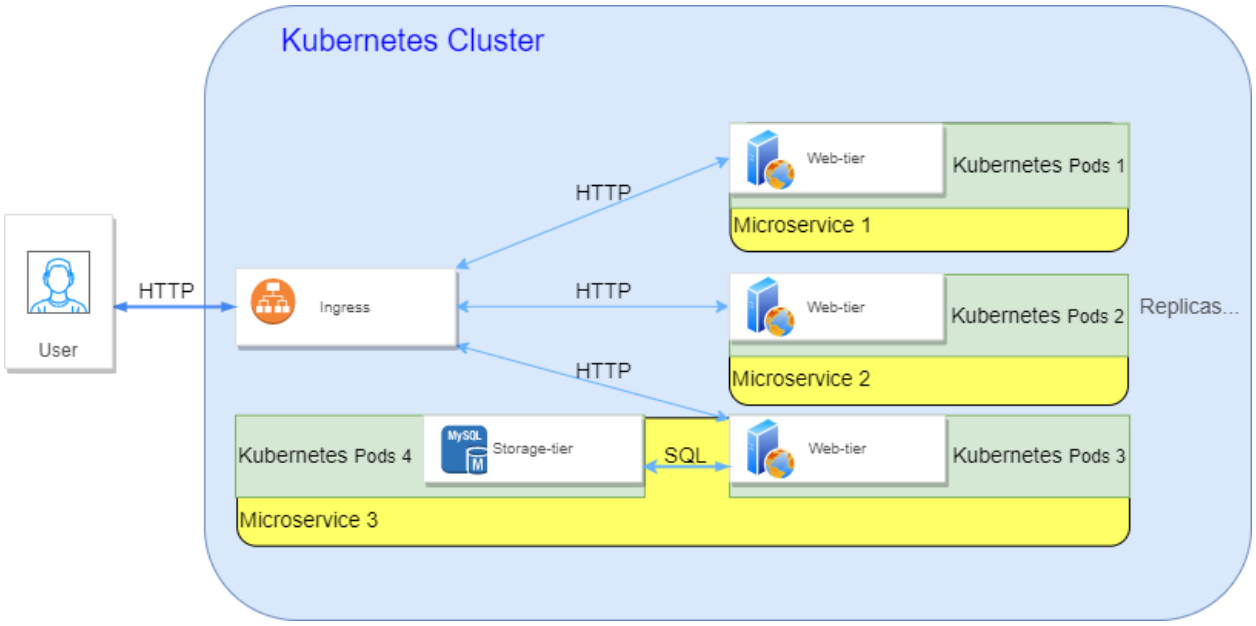
During the implementation of the team project, you have the chance to explore different ETL tools, web frameworks, database schema designs as well as Kubernetes and EMR cluster configurations. Show us your plan of exploration and how it will impact your design decisions. You don't necessarily need to implement them or apply them to your actual implementation.

- Which programming language/web framework combinations did you or will explore? Name two combinations that you are considering exploring. Use one or two sentences to explain the underlying implementation of each of them (e.g. Compiled language? Event-loop? Multi-Thread? Other?). Note that the combinations you choose to explore can be in the same programming language.
  We have explored Spring and Vert.X in Microservice 1, both using Java.
  Spring: Spring is a multithreaded web framework.
  Vert.X: Vert.X uses an event-loop, and it is multithreaded. We are able to create multiple Vert.x instances and each Vert.X instance can maintain multiple event loop threads.
- What parameters are you going to use to compare the different web frameworks? Provide at least 3 parameters.
  1. Throughput per second
  2. The time needed to warm up the framework
  3. Average latency to finish a request
- What are the differences between Application and Network Load Balancers? Describe one other technique that achieves load balancing and request routing. How are you going to compare them in order to decide which one is suitable for your web service?
  Application Load Balancer will examine the content of the web request to determine where to route the request, while Network Load Balancer will directly forward requests to deployed servers. Therefore, NLB will take in a much higher load of traffic.
  Another technology that achieves load balancing and request routing is the combination of Ingress and NodePort.
  We would first build up the service and the Kubernetes cluster, then try to examine how the performance of our service changes as we change the load balancing tool of our system.
- What's your preliminary plan for the ETL process? Can you express the process as a directed graph (see below)? What measure are you going to take to ensure the correctness of the ETL result? What can you do to enable reuse if you had to rerun the ETL process?
  Example Directed Graph (you may illustrate your idea in any way you like):
  We will first run ETL using only the first part of the data set, and compare the result with the reference data. We also plan to break down the ETL process into multiple substeps and save the intermediate results so that we can reuse the most amount of work if we have to rerun the ETL process.

- Which ETL tools (Hadoop Streaming, Hadoop Batch Processing, Apache Spark, etc.) do you plan to use? How does it fit into your ETL process? What about its development convenience, expressiveness, runtime performance and tools for debugging given the large dataset?

We plan to use Apache Spark. The ETL process is quite complex in our case, and it may require multiple MapReduce. In this case, Apache Spark is more suitable for the ETL process. It is very convenient to deploy, as many cloud providers provide Apache Spark as a service. For example, we can use Azure HDInsight. It is very expressive as well: it has map and reduce functions like MapReduce framework, but it also has other functions like filter and groupby. It uses RAM more than disk to store the intermediate result, so it should be faster than Hadoop. We can use Spark's web UI to debug the program.

- In order to return the response, you need to store the processed data in MySQL or HBase for Microservice 3. Which data is needed? How do you plan to design the database schema? What are some considerations behind your schema design?
  The data fields that are needed are tweet_id, sender_uid, content, reply_to_uid, retweet_to_uid, and hashtags. We plan to store three tables in total. The first table stores every field and uses tweet_id as the primary key. This table is used to compute the keywords_score since that score is calculated in a dynamic way based on the phrase provided by the user. The second table is used to retrieve the interaction score. It would have four columns: user_id_1, user_id_2, interaction score, and an additional dummy counter as the primary key. The interaction score is static based on reply_count and retweet_count, so we can compute them once and store them to speed up the query. The third table is used to retrieve the hashtag score. It would have four columns: user_id_1, user_id_2, hashtag score, and an additional dummy counter as the primary key. The hashtag score between two users is static based on the number of common hashtags, so we can compute them once and store them to speed up the query.
- Provide the preliminary architectural end-to-end design of your microservices for this project, including your Kubernetes cluster, load balancer, web-tier, database, etc. Discuss how the computational resource of your Kubernetes will be allocated to which part of your web service? How would you distribute the load across different machines? (You can provide an illustration of your architecture and of your cluster design here)



We will use the self-management property of Kubernetes to allocate computational resources to web services. We would use Ingress to distribute the load across different machines.

- Describe at least two methods to provision a self-managed Kubernetes cluster. Compare the methods and list their advantages and disadvantages.
  Method 1: manually deploy the Kubernetes cluster using the web console. Advantage: easy to use. Disadvantage: the deployment process can be tedious each time we try to deploy the cluster.
  Method 2: using kops to programmatically deploy the Kubernetes cluster. Advantages: fast. Disadvantage: hard to write the YAML files since there are many parameters to consider.
- Since one pod in your Kubernetes cluster is not enough to reach the target throughput, how do you plan to scale up your web service? Explain which method you will adopt and how you reached this decision.
  We have decided to use the Horizontal Pod Autoscaler (HPA) to scale up the web service. We would monitor the CPU utilization of each pod, and scale up when CPU utilization is high. We reached this decision based on our experience in project 2.
- Your team will be required to orchestrate the deployment of your microservices using Terraform, Kubernetes and Helm. Briefly explain the functional role of these components.
  Terraform is a cloud-resource deployment script that automates the process and steps. Kubernetes is a docker container orchestration platform that automatically manages the status of a cluster of application containers. Helm relies on the Kubernetes cluster. It is a tool to allow easy installation and update of applications (many pods) on a Kubernetes cluster.
- Reading the Phase 1 report template will provide your team with many hints on how to successfully complete this project and not waste many hours. What hints were useful to your team after reading the report template?
  1. Consider different types of load balancing techniques.
  2. Let us figure out the advantages of MySQL in MicroService 3.

Question 4: Time

After getting a big picture by answering the above questions, you may start thinking how to distribute the workload among you and your teammate. Conducting sufficient exploring and optimization will not only help you achieve target RPS in Phase 1 but also greatly expedite your process in Phase 2 and 3. The following questions provide a guideline for you and your teammate to plan your time wisely.

- What tasks do you plan to divide phase 1 into? Show us the internal team deadlines you have decided for each task. (provide a timeline as a table with tasks, who will work on which tasks by which days/weeks)
  Timeline:
  Feb 14 - Feb 20: Design the overall architecture and start working on part 1. Benny is in charge of writing Java codes related to decoding and encoding QR codes. Yukun is in charge of implementing the Vert.X framework, the terraform code used to deploy S3 buckets, and

the YAML file used to deploy the Kubernetes cluster. Leo is in charge of implementing the Spring framework, the helm charts used to deploy service on the Kubernetes cluster and the docker file.
Feb 21 - Feb 27: Combining every component together. Evaluating different load balancing techniques. Exploring advantages and disadvantages of different web frameworks. Finishing the checkpoint for microservice 1. Trying to get the early bird bonus. We will sit down and work together this week so that we can solve the potential problems together.
Feb 28 - Mar 6: Implementing microservice2 together. How we are going to divide work will be based on how we are going to design the backend architecture of microservice2. Finishing the checkpoint for microservice 2. Trying to get the early bird bonus.
Mar 7 - Mar 13: Starting working on microservice 3. Leo will be working on ETL. Yukun will be in charge of database design and implementation of tables in the MySQL database using the reference data. Benny will be in charge of the web tier of microservice 3. Combining every component together. Trying to finish checkpoint.
Mar 14 - Mar 20: Finishing checkpoint. Trying to optimize the database schema and get the early bird bonus. Finishing the report.

- ETL will be very time-consuming. It's very common that you have to rerun the ETL job because of a trivial error in your code. In light of this almost unavoidable event, what will you do to prevent, detect and recover from such errors?
  We will first run the ETL process using only the first part of the data set, and compare the result with the reference data to make sure that the ETL process has minimal errors. We will break down the ETL task into multiple sequential steps, and try to save the intermediate result. When we find a problem later, we could use the most recent correct intermediate result to ensure that we can reuse as much computation as possible.

Question 5: Git and Code Review

In this project, we will set up a GitHub repository for the team. We expect you to follow best practices in your development effort. We have a primer on the best practices of using GitHub, please go through it carefully.

Teams are required to conduct code reviews for every piece of code written by a team member. This is usually done using the Pull Request feature on GitHub. Upon receiving a pull request, another member of the team is supposed to review the code and approve the pull request. This makes sure that the production code has been reviewed and works as expected. Skipping this process usually leads to many hidden bugs, wasted time, and frustrations. We require you to fully adopt this best practice.

Please provide the commands on how to do the following:

- How do you create a pull request on git?
  We create pull requests only when we have a new function that is bug-free in testing environments. Then we need another member to review approve the request.
- How do you create a new branch on git?

We create a new branch for each independent function/feature.

- How do you merge conflicts on git?
  We merge conflicts by mostly choosing the latest version and three of us need to agree on the commit.
- What's the difference between a git commit and git push?
  Git commit saves the work locally. Git push updates the origin(remote) version with all the commits we have saved.
- When should you push and pull? (e.g., when my feature completes, every few days, every day)
  Push: when important features are implemented and pushed every workday.
  Pull: When checkout to another, or there are updates on the Master/Development branch.
- What should you look for when conducting a code review?
  Look whether there are bugs, code style problems, or the possibility of further optimization.
- How should you report bugs found in a code review?
  Comment and mark the line number, explain why it is a bug.

Question 6: Profiling and Debugging

Debugging capabilities are going to be crucial to achieving the set targets for this project. The easiest way for debugging is logging. Logging requests will help you understand where the bottleneck might be. Logging libraries offer additional features like log levels, log line number, timestamp, etc. So it's advisable to use a logging library. **Caution: logging in production systems can lead to a drop in performance, so only use logging while developing and debugging.**

- What should you look for when selecting a logging library?
  1. Usability: Whether it has many dependencies, whether the framework has support for the logging tool, whether it is easy to be integrated.
  2. Informativeness: Whether it is able to log the information we need.
  3. Performance: Does it make the program a lot slower.
  4. Reliability and stability.
- How do you monitor your VM's CPU usage, memory usage if you have direct access to them? If the VMs are in a Kubernetes cluster and you don't have SSH access into them, how would you monitor their CPU and memory usage? How do you monitor the CPU and memory usage of a Kubernetes pod? Give specific commands. How will you identify if one of them is the bottleneck of your system?
  For general VMs: we can use "top" to assess CPU usage given the direct access to it.
  For clusters: We can use HPA. The command is: kubectl describe hpa.
  If the CPU is underutilized (generally less than 70%). Then the bottleneck should be network, load balancer, or IO/Database. Otherwise, the bottleneck might be computation, and then we need to optimize algorithms and computation frameworks.

- How can you find out the execution details of a MySQL query? (e.g., what indexes it will use etc.)

We can use the "EXPLAIN" query before the execution of the query to see the plan and prediction of the query, which includes the number of rows, indexing, and other optimizations.

- What is the time complexity of an HBase GET operation? (Hint: it should be related to the number of HFiles.)
  The GET operation:
  First find the metafat: zookeeper, correct region server in O(1) time.
  Find the corresponding block in the Hfile block in O(b) time where b is the number of HFile blocks.
  Find the keyValue pair in O(e) time where e is the number of key-value entries in an Hfile block.
  At the same time, data in Blockcache or Memstore will be retrieved first and faster.
- Read through the MySQL primer carefully. Quote the terminologies that you think will impact the database tier performance.
  - Indexing: appropriate creation and usage of the index helps speed up the retrieval of data from the table.
  - For the composite index, the column order matters.
  - Index slows down writes: each index needs to be modified during writes.
  - Choose between MyISAM and InnoDB, compare in many scenarios:
    - **E.g. "MyISAM** saves data as table level so you can easily read the saved row number while **InnoDB** does not save data as table level so counting the number of rows needs to scan the whole table."
- Read through the HBase primer carefully. Quote the terminologies that you think will impact the database tier performance.
  - "Get and Scan operations always return data in sorted order."
  - "Therefore, to improve operational efficiency, it is preferable to place columns with common I/O characteristics in one column family."
  - Choose the write cache: "HBase provides three options for BlockCache: LruBlockCache, SlabCache, BucketCache."
  - Make use of Memstore: "MemStore is a write cache, which stores updates in memory as sorted KeyValues. The data written by the user will first be put into MemStore."
  - Make use of compaction:
    - Minor Compaction and Major Compaction
    - "If you let HBase accumulate many HFiles without compacting them, you'll achieve better write performance (the data is rewritten less frequently) but get worse read performance"
    - "Besides monitoring the compaction progress metrics, you should also monitor how the locality changes for each region during the compaction."
  - Compression and Data Block Encoding

Question 7: Integration

The integration between the web-tier and storage-tier will play a crucial role in getting good performance in Microservice 3. Integration is typically achieved through database libraries. There

are multiple libraries for doing integration with MySQL or HBase. The performance can vary between libraries and configurations so it will behoove the team to experiment with different libraries and configurations.

- What features would be crucial for achieving the target throughput? (e.g., connection pooling, prepared statements)
  Firstly, in MySQL building a correct and efficient Indexing would be very important since we are always querying a few critical columns. Secondly, maintaining an active connection between the backend application and the database would be critical for handling a large number of queries in a short period of time. Thirdly, as mentioned above, prepared statements could save some time for compilation and this effect will exacerbate itself with the large volume of traffic we will be handling.
- Name a few libraries that you might consider?
  In Java, we would consider JBDC, Hibernate, MyBatis, and Apache Cayenne.

5% Bonus of Microservice 1

Fill in the form if your team was able to achieve the Microservice 1 target by 2/27 and are hence eligible for the 5% early bird bonus. Please push all the files you wrote to reach the target RPS and tag your git commit as “earlyBird-m1” so that we know you completed this bonus at that commit.

	Microservice 1
score	**10(weighted: 13.258122081538462)**
submission id	**689094**
throughput	**86177.79**
latency	**5.55**
correctness	**100.00**
error	**0**
date/time	**2022-02-27 02:51:51**