# Team Project Report - Phase 1 Blockchain Service Checkpoint

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

## Phase 1, Blockchain Service Checkpoint Summary

Team name: CCprojectFor3
Members (names and Andrew IDs): Zhitong Guo zhitongg, Kewen Zhao kewenz, Yu Wei yuwei2

Please color your responses red.

## Overview:

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

1. Blockchain Service and QR Code Service

| **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 deployment orchestration using Kubernetes, Helm and Terraform | Report, manual grading |
| Profile and optimize computationally expensive tasks | Report, throughput |

2. Twitter Service

| **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 |
| Identify **MySQL 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%

## Guidelines

- Use the report as a record of your progress and then condense it before submitting it.
- When documenting an observation, we expect you to also provide an explanation for it.
- Questions ending with "Why?" require evidence, not just reasoning.
- It is essential to express ideas in your own words, through paraphrasing. Please note that we will be checking for instances of plagiarism.

## Question 0: Knowing each other

After your first meeting, each member is required to submit the following Google Form before the Blockchain Service Checkpoint Report submission deadline:
Team Introduction Survey

## 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:

- What skills have you learned in previous projects that will come into play here?

Terraform, kubernetes, Docker

- 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 Spring before, but this time we are consulting from the https://www.techempower.com/benchmarks/#hw=ph&test=fortune&section=data-r22 benchmark, and find the top-ranking web frameworks. Also, we need to make sure that the web framework is compatible with MySQL database.

- We want you to learn the cloud-native development and deployment approach. Explain the differences between a monolithic application and having multiple microservices.

Monolithic: the entire service as a single, indivisible unit
Multiple microservices: each component can be installed and uninstalled without affecting the other ones

- Use one or two short sentences to explain how the index in MySQL works.

It allows for faster retrieval of rows from a table by providing quick access to the rows based on the indexed columns' values.

## Question 2: Team

As mentioned in the Team Formation Primer, 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? Tell us more about yourselves and your plan of collaboration by answering the following questions. For each bullet point, please explain in 1 to 2 sentences:
    - Did each member simply accept an invitation to join this team?

No, we asked for the other team members' availability and past experience on software engineering, and carefully chosen ones who are proactive in communication.

    - Did members observe the performance and behavior of other team members? If so, what traits did you observe? Describe the skill sets of each of your team members.

Yes, we observed the performance and behavior of team members, and all of the members are good communicators, hard workers, and started early on individual projects. Skill sets of Zhitong: Data

ETL and MySQL and MongoDB databases, Java and Python; Skill sets of Yu: Spring boot, Java, mySQL, mybatis; Skill sets of Kewen: Java, Javascript, backend, frontend

- Did your team follow the recommendations provided by the course staff?

Yes, we carefully examined each team members' Time Commitment, Semester Plan, Skillsets, Domains of Interests, Working Style and Team Expectations.

- Did you form the team based on friendship?

No, although two of us know each other from our major, we chose team mates based on our workstyle, time commitment and skill sets.

- Did you follow the questionnaire posted on Piazza to find your teammates?

No, we did not.

- 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)

Zhitong: ETL and Web Tier; Yu: Web Tier and backend; Kewen: ETL, rust (web tier), deployment

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

- When will your team hold regular team meetings?
  - Every Tuesday, Thursday, Friday, Saturday
- Are you using a common calendar to notify members about the meetings?
  - Yes
- Is there a set of action items for each member to work on after each meeting?
  - Yes
- Which project management tool and collaboration platform do you plan to use for the project? (some options: Jira, Asana, Slack, Trello, Notion, etc
  - Notion

## 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 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.

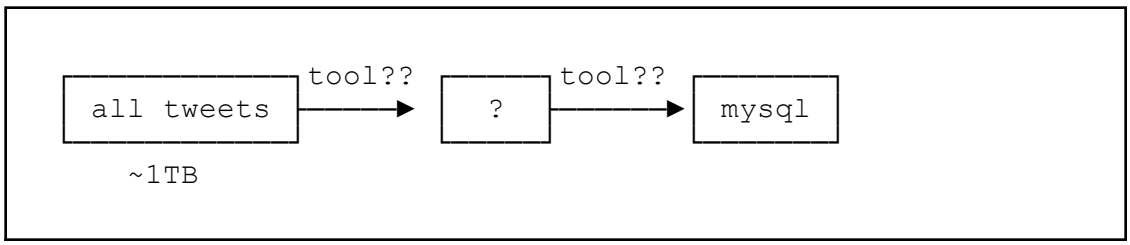
- Which combinations of programming languages and web frameworks have you explored or do you plan to explore? Name two combinations. Use one or two sentences to explain the underlying implementation of each of them (e.g. Compiled language? Event-loop? Multi-Thread? Other? Note that the combinations you choose to explore can be in the same programming language).
  - We planned to try ntex with rust language, and axum with rust.
  - Axum:
    - based on asynchronous/await syntax and Tokio runtime.

        - focuses on ergonomics, type safety, and composability.
    - Ntex
        - built on top of Tokio, an asynchronous runtime for Rust, provides tools and abstractions for building scalable and high-performance networking applications.

- What parameters are you going to use to compare the different web frameworks? Provide at least 3 parameters.
    - JSON serialization, Single query, Multiple query
- 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(ALB) is based on Layer 7(Application Layer) and allows for content-based Routing which can route traffic based on path, host headers and HTTP headers. It also has application-level features like sticky sessions and support for microservices.
    - Network Load Balancer(NLB) is based on Layer 4(Transport Layer) and has high throughput and low latency. It also can route traffic to primary IP addresses as targets and supports assigning a static elastic IP address per availability zone.
    - HERE IS ANOTHER TECHNIQUE: DNS load balancing. It distributes traffic across multiple servers by responding to DNS queries with different IP addresses. However, it does not have the sophisticated routing and management features as ALB or NLB.
    - WE THINK NETWORK LOAD BALANCER IS A BETTER IDEA FOR US. Because we still need to make our RPS larger with higher throughput and low latency which matters the most in this project.

- What's your preliminary plan for the ETL process? Can you express the process as a directed graph (see below)? What measures 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?
    - All tweets (~1TB) -> extract with ??? -> Clean and transform on Azure/GCP platform -> Loading with Spark sql -> mysql

Example Directed Graph (you may illustrate your idea in any way you like):



- 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 will be using Apache Spark, which fits well due to its in-memory processing capabilities, scalability, and support for complex transformations. It has libraries such as PySpark and SparkSQL, which makes it convenient for development and it's also suitable for processing large datasets efficiently.
- In order to return the response, you need to store the processed data in MySQL for the Twitter Service. How do you plan to design the database schema? What are some considerations behind your schema design?
    - The schema will be designed to optimize query performance and storage. Considerations include indexing important fields (like user ID, tweet ID), partitioning the data based on time or geography, and normalizing data to reduce redundancy. The schema design will focus on supporting the queries expected by the Twitter Service, such as searching tweets, aggregating user data, and analyzing tweet patterns.
- 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 resources of your Kubernetes cluster will be distributed among different parts of your web service and across different machines. (You can provide an illustration of your architecture and of your cluster design here)
    - We'll follow a similar architecture as individual P2. We'll use an Ingress controller as the load balancer to different pods, which are managed by kubernetes. We chose to use ntex as it's very efficient to handle API calls and data serialization.
- Describe at least two methods to provision a self-managed Kubernetes cluster. Compare the methods and list their advantages and disadvantages.
    - Kops
        - Ease of Use: provides a simple command-line interface (CLI) for creating and managing Kubernetes clusters
        - Complexity for Custom Configurations: configuring advanced features requires deeper knowledge of Kubernetes..
    - Terraform
        - supports provisioning Kubernetes clusters across multiple cloud providers
- 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 will set the max pods to 20, as scale up quickly by setting a lower threshold of the cpu utilization
- 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: Provisioning and managing the infrastructure in a cloud provider; Defining infrastructure as code, making it easier to create, update, and maintain the resources needed for the project.

- Kubernetes: Manages the deployment, scaling, and operations of application containers across clusters of hosts. It provides a platform for automating deployment, scaling, and operations of application containers.
- Helm: A package manager for Kubernetes, allowing for the definition, installation, and upgrade of Kubernetes applications. Helm charts help manage Kubernetes applications with pre-configured templates, simplifying the deployment and management process.

- 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?
  - The following hints are useful: Git workflow and git operations is essential, time and team management skills are required to ensure efficiency, ETL process is error-prone and we need to prevent it; We need to consider some parameters for deciding the web framework to use.

## Question 4: Time

After getting a big picture by answering the above questions, you may start thinking about how to distribute the workload amongst you and your teammates. Conducting sufficient exploring and optimization will not only help you achieve the target RPS in Phase 1 but also greatly expedite your process in Phases 2 and 3. The following questions provide a guideline for you and your teammates 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). Explain how you plan to divide the work and why it is split in this manner.

| Tasks | Deadline | Person |
|---|---|---|
| Blockchain Logic V1 | 2.20 | Yu, Zhitong |
| Blockchain service deployment | 2.22 | Kewen |
| Blockchain Logic Tuning | 2.25 | Yu, Kewen |
| Checkpoint Report | 2.25 | Yu, Zhitong, Kewen |

- 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?

- Prevent: implement testing and checks the validity for every small step of ETL, doing frequent code review and peer code review to make sure that the implementation is correct;
- Detect: Detect unexpected patterns with anomaly detection and perform data quality checks;

- Recover: implementing checkpoints, using data backup, and using git version control to recover.

## 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 adopt these best practices fully.

- Please provide the commands on how to do the following:
- How do you create a pull request on git?
  - **git request-pull** [-p] <start> <URL> [<end>]
  - The upstream project is expected to have the commit named by <start> and the output asks it to integrate the changes you made since that commit, up to the commit named by <end>, by visiting the repository named by <URL>.
- How do you create a new branch on git?
  - git checkout -b new-feature-branch
- How do you merge conflicts on git?
  - Step1: use **git status** to check **both modified files**
  - Step2: use **cat** to take a look at files with conflict which should look like this:
    - <<<<<<< HEAD
    - Conflict content version 1
    - =======
    - Conflict content version 2
    - >>>>>>> new_branch_to_merge_later
  - Step3: use **vim** to make modifications on this file and only save the version you want to keep in the text above with other lines all removed.
  - Step4: use **git commit** to commit changes
- What's the difference between a git commit and a git push?
  - **Git commit** operates locally but **git push** interacts with the remote repository. What's more, **git commit** is about saving changes in the local repository but **git push** is about sharing the committed changes with the remote repository.
- When should you push and pull? (e.g., when my feature completes, every few days, every day)
  - When to push:
    - After committing local changes

        - Before making a pull request
        - Regular to back up your work
        - Trigger (CI/CD) pipelines
    - When to pull:
        - Before start new work
        - After being notified of changes
        - Before pushing local changes
        - To stay updated with the project process

- What should you look for when conducting a code review?
    - Correctness and functionality
    - Code quality
    - Performance
    - Documentation and Comment
- How should you report bugs found in a code review?
    - Provide a clear and descriptive title
    - Include detailed steps to reproduce
    - Describe the expected and actual behavior
    - Add relevant context and screenshots
    - Prioritize the issue
    - Propose possible solutions

## Question 6: Profiling, Debugging, and Performance Optimization

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 numbers, timestamps, 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?
    - Language and framework support
    - Platform compatibility
    - Overhead
    - Features and flexibility
    - Ease of use and configuration
    - Scalability and extensibility
    - Reliability and maintenance
- How do you monitor your VM's CPU usage, and memory usage if you have direct access to them? If the VMs are in a Kubernetes cluster and you don't have SSH access to them, how would you monitor their CPU and memory usage?
    - IF WE HAVE DIRECT ACCESS TO VMS:
        - Using command line tools like: **tops, htop, vmstat, free**
        - Using monitoring tools like: Nagios, Zabbix

    - IF WE DO NOT HAVE DIRECT ACCESS TO VMS:
        - Kubernetes Metrics Server: using instructions like **kubectl top node** or **kubectl top pod**
        - Kubernetes dashboard: web-based user interface
        - Prometheus and Grafana
- 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?
    - MONITOR:
        - kubectl top pods -n <namespace>
        - kubectl top pod <pod-name> -n <namespace>
    - IDENTIFY BOTTLENECKS:
        - High CPU utilization and high memory usage are the symptoms for bottlenecks in systems
- How can you find out the execution details of a MySQL query? (e.g., what indexes it will use etc.)
    - CODE:
        - **EXPLAIN ANALYZE SELECT * FROM your_table WHERE your_column = 'some_value';**
    - After adding EXPLAIN and ANALYZE in front of the sql we'd like to execute, we can interpret the results as follows:
        - Using index: this extra column indicates which index we are using
        - Check the "type " column: this is for the join type. When it is "ALL", this indicates that we are not using any indexes.
- Read through the MySQL primer carefully. Quote the terminologies that you think will impact the database tier performance.
    - Mysql Indexing: slow down the deletion and addition operations, but can speed up the queries a lot if created rationally.
    - Mysql Storage Engine: this significantly affects database performance because each engine is designed with different use cases, capabilities and performance optimizations.

## Question 7: Integration

The integration between the web-tier and storage-tier will play a crucial role in getting good performance in the Twitter Service. Integration is typically achieved through database libraries. There are multiple libraries for integration with MySQL. 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)
    - Connection Pooling: this will minimize the overhead of establishing connections to the database by reusing a pool of connections, reduce latency and improve throughput.

    - Prepared Statement: this will allow SQL statements to be compiled once and executed multiple times with different parameters, reducing parsing time.
    - Asynchronous Operations: this enables non-blocking database calls and improves application responsiveness and scalability.
    - Batch Operations: this will greatly improve performance when inserting or updating large volumes of data.
    - Configurable Query Timeouts: this prevents the system from waiting indefinitely for a query to complete which can be critical under heavy load.
- Name a few libraries that you might consider.
    - As we are using rust to implement our web tier and storage tier. Here are the libraries we are going to use:
        - Mysql_async: this offers non-blocking APIs for interacting with MySQL, supporting future and async/await programming models. It also supports connection pooling and SSL.
        - Diesel: this is an ORM and query builder for Rust that supports MySQL.

## 5% Early Bird Bonus for Blockchain Service

Fill in the form if your team was able to achieve the Blockchain Service target by 2/25 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-blockhain” so that we know you completed this bonus at that commit.

| | Blockchain Service |
|---|---|
| score | 22.50 |
| submission id | 920147 |
| throughput | 24581.73 |
| latency | 22.89 |
| correctness | 84.97 |
| error | 7.67 |
| date/time | 2024-02-25 12:39:49 |

## 5% First Mover Bonus for Blockchain Service

If your team was among the first 10 to reach the Blockchain Service checkpoint by 2/25 and thus eligible for the first mover bonus, please provide the following information to allow the teaching staff to verify and grant you the bonus.

You need to fill the table below to describe the submission that achieved the Blockchain Service checkpoint, label the code that you used for the submission and push to your team's GitHub repository, and provide a screenshot of the complete scoreboard to serve as evidence of your team's ranking. Note that you need to take the screenshot immediately after you achieve the bonus, otherwise making more submissions may overwrite the scoreboard.

| | Blockchain Service |
|---|---|
| score | |
| submission id | |
| throughput | |
| correctness | |
| latency | |
| error | |
| date AND time of submission (e.g., 2024-02-22 19:11:08) | |
| ranking for first mover bonus (e.g., 1, 2, …, 10) | |

Please push the code that you used to reach the target RPS and tag your git commit as "firstMover-blockchain" so that we know you completed this bonus at that commit.

In order to claim the bonus, you must submit a screenshot of the scoreboard at the moment of submission showing your team's ranking among the other submissions. The screenshot should include all present submissions on the scoreboard, not just your own. This will help us verify that your team was one of the top 10 teams on the scoreboard for Blockchain Service at the time of submission.

## Penalty Waiver for Blockchain Service Correctness

Fill in the form if your team was able to make a 10-minute Blockchain Service submission with above 95% correctness by 2/25 and hence eligible for waiving one most significant penalty for each team member.

| | Blockchain Service |
|---|---|
| score | |
| submission id | |

| | |
|---|---|
| throughput | |
| latency | |
| correctness | |
| error | |
| date/time | |