Trilobita: A Distributed Graph Computing System
Checkpoint 2

1. Group Members

Guo Yuchen	1004885
Guo Ziniu	1004890
Liang Junyi	1004891
Wang Yanbao	1004865
Xiang Siqi	1004875

2. Introduction

Trilobita is a distributed graph processing system designed to efficiently handle large-scale graph processing tasks in a distributed environment. This report is the checkpoint 2 report of the course 50.041 group project. We will demonstrate the working implementation of our system by running a page rank task using multiple machines. In addition, we will outline a group of features that we are going to implement from checkpoint 2 until the final project demonstration.

2.1. Problem Statement

Our problem statement stays still compared to Checkpoint 1.

The problem that lies in this project is achieving both efficient and consistent processing of large graphs within a distributed, fault-tolerant, and adaptable framework. Large-scale graph computing is prevalent in many practical domains, necessitating distributed processing due to their size and complexity.

Inspire by and following the idea proposed by Pregel, *Trilobita* endeavors to incorporate several critical features, including:

1. **Fault Tolerance**: Ensuring the system can gracefully manage failures of both the master and worker servers, without imperiling data integrity and enabling a seamless continuity in server failure handling.
2. **Flexibility**: Enabling users to customize graph processing tasks through user-accessible and user-definable interfaces.
3. **Efficiency**: Optimizing communication and processes to efficiently handle data transfer across distributed resources. Gracefully implement “Superstep” logic to minimize time spent on computing.
4. **Consistency**: Guarantee a consistent data model and processing logic across the entire distributed computation process.

3. Demonstration

We will use a page rank job implemented by us to demonstrate the functionality of our system.

3.1. Introduction to Page Rank

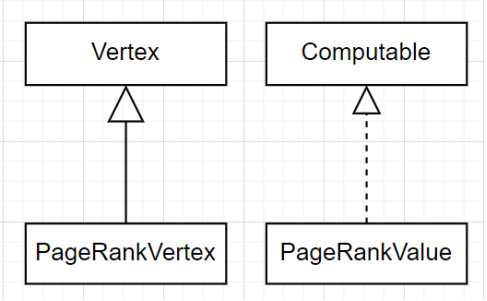
PageRank is an algorithm used to assess the importance of web pages on the internet developed by Google, in 1996. The concept of PageRank is that the importance of a web page can be measured by the quantity and quality of links from other web pages. Specifically, the PageRank algorithm operates based on two principles:

- Link Quantity: The more web pages that link to a particular web page, the higher its importance. This means that web pages with more inbound links are considered more important.
- Link Quality: Links from web pages with higher importance are more valuable than those from less important web pages. Therefore, PageRank takes into account the importance of the source web pages that provide the links.

The PageRank algorithm views the internet as a vast graph, where web pages are nodes in the graph, and hyperlinks between them are the edges. It then employs an iterative process to calculate the PageRank value of each web page until it converges. In each iteration, a web page's PageRank value is updated based on the quantity and quality of its links.

In our demonstration, we hardcoded a graph to represent the web pages. Each vertex represents a website page, and links between vertices enable them to transfer PageRank value and update their local PageRanks in each superstep.

Fig 1 shows the class implementation of the page rank job.



[fig 1]. page rank implementation

4. Remaining Features

We have several key features that remain to be implemented in the Trilobita system before the final demonstration. These features are listed in order of priority:

- Heartbeat Mechanism and Fault Tolerance:

    - The master should be capable of detecting the failure of worker servers.
        - Detection mechanism: heartbeat messages.
        - Handling procedure: The master will reassign vertex groups to operational workers, and all workers will reload their partition states from the most recent available checkpoint.
    - Workers should be able to detect the failure of the master.
        - Detection mechanism: Heartbeat messages.
        - Handling procedure: A backup master will take control. The backup master should be able to load states from the previous master at the most recent available checkpoint.

- *Functionable* Module (Combiner & Aggregator)
    - *Functionable* module contains widgets that optimize the overall performance of the system, for example, Combiner and Aggregator.
    - While we have already implemented the *Combiner*, the implementation of the *Aggregator* and its integration into the system remains ongoing.

- Implement Runtime Environment
    - In our design, *Trilobita* hides servers from the user, providing system-level APIs for defining and launching jobs.
    - We strictly followed this principle and have left appropriate user-accessible interfaces in our system, such as *Vertex*, *PartitionStrategy*, and *Computable* Entities. However, the implementation of *TrilobitaEnviroment*, which serves as the running environment for a user-defined job, is still a work in progress.

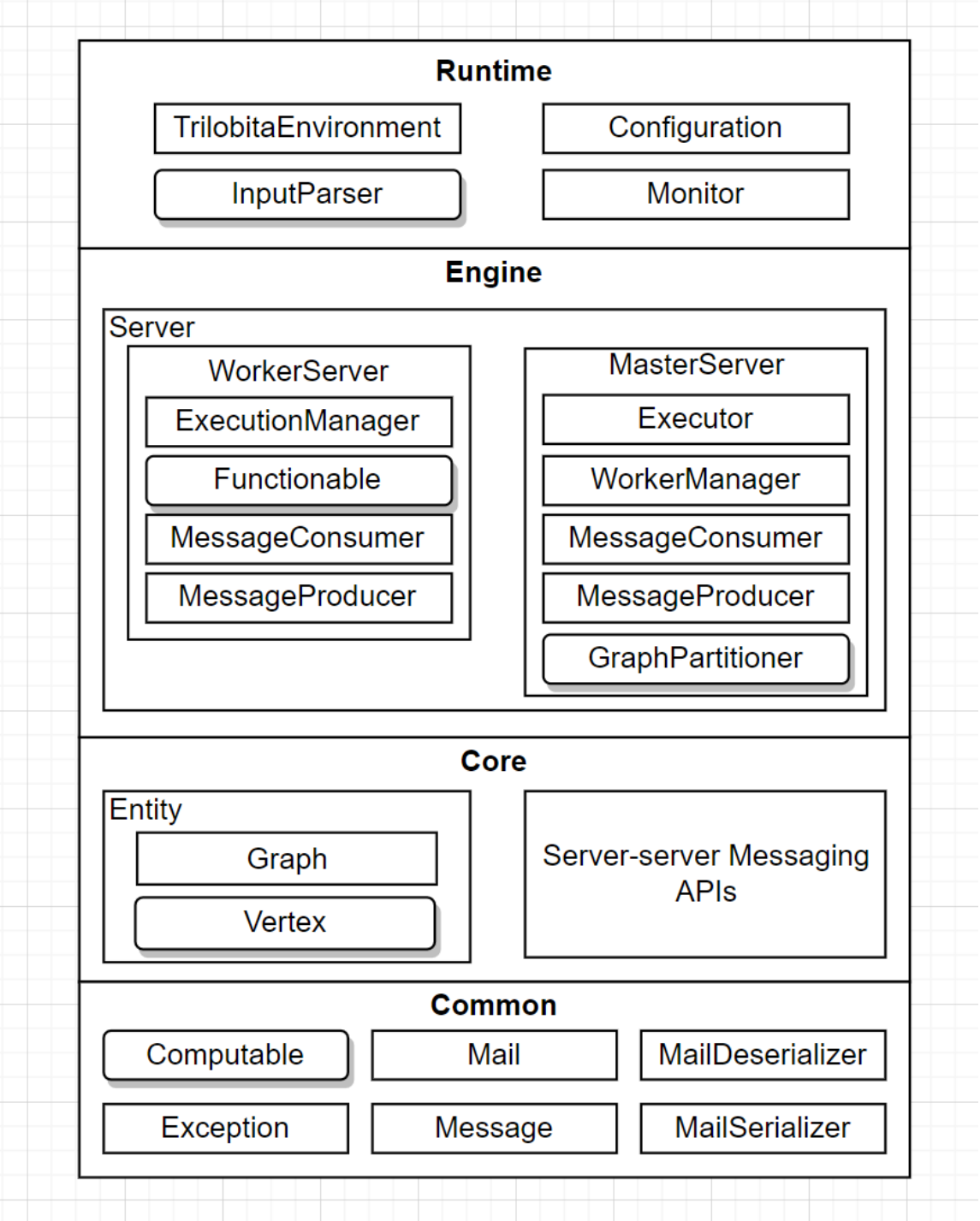
- Standardize and parameterize input and output
    - We need to define how users can input random graphs into the system. For this demonstration, we have hard-coded the graph.
    - We have implemented a JsonInputParser to convert JSON-format files into graphs. This feature, along with the complete runtime environment, will be showcased in the final demonstration.
    - Standardize the output and logs.

- Other Validation and Optimization
    - Improving the performance of a single server, both in the master and worker roles.
    - Implementing a monitor to track the state and performance of the system.
    - Implementing more use cases such as Breadth-First Search and Neural Network Forward Inference to demonstrate the capability of *Trilobita*.

5. Conclusion

Trilobita is a Pregel-like distributed graph processing system that prioritizes adaptability, efficiency, and robustness. In this report, we have showcased the current state of Trilobita, demonstrating its working process using the page rank example and highlighting the remaining features to be implemented.

[Appendex - Structure of the *Trilobita* System]



[fig 2]. Structure of *Trilobita*