Project Proposal

Oliver Little
2011802

October 14, 2022

1 General Topic

My project topic falls within the field of distributed systems. I would like to find ways of increasing data processing speed for large datasets (at least 100GB). This sort of processing is generally best applied to ETL *(extract-transform-load)* workflows, where some large dataset needs to be imported, processed in some way, then exported elsewhere for further analysis.

Single systems begin to struggle with this volume of data, as it typically cannot all be loaded into memory at once. Furthermore, often the processing to be performed acts on a small number of rows of data at a time. This means that while the overall dataset is extremely large, the work to be performed lends itself quite well to parallelisation, as only a small number of rows are necessary at any one time.

2 Problem Definition

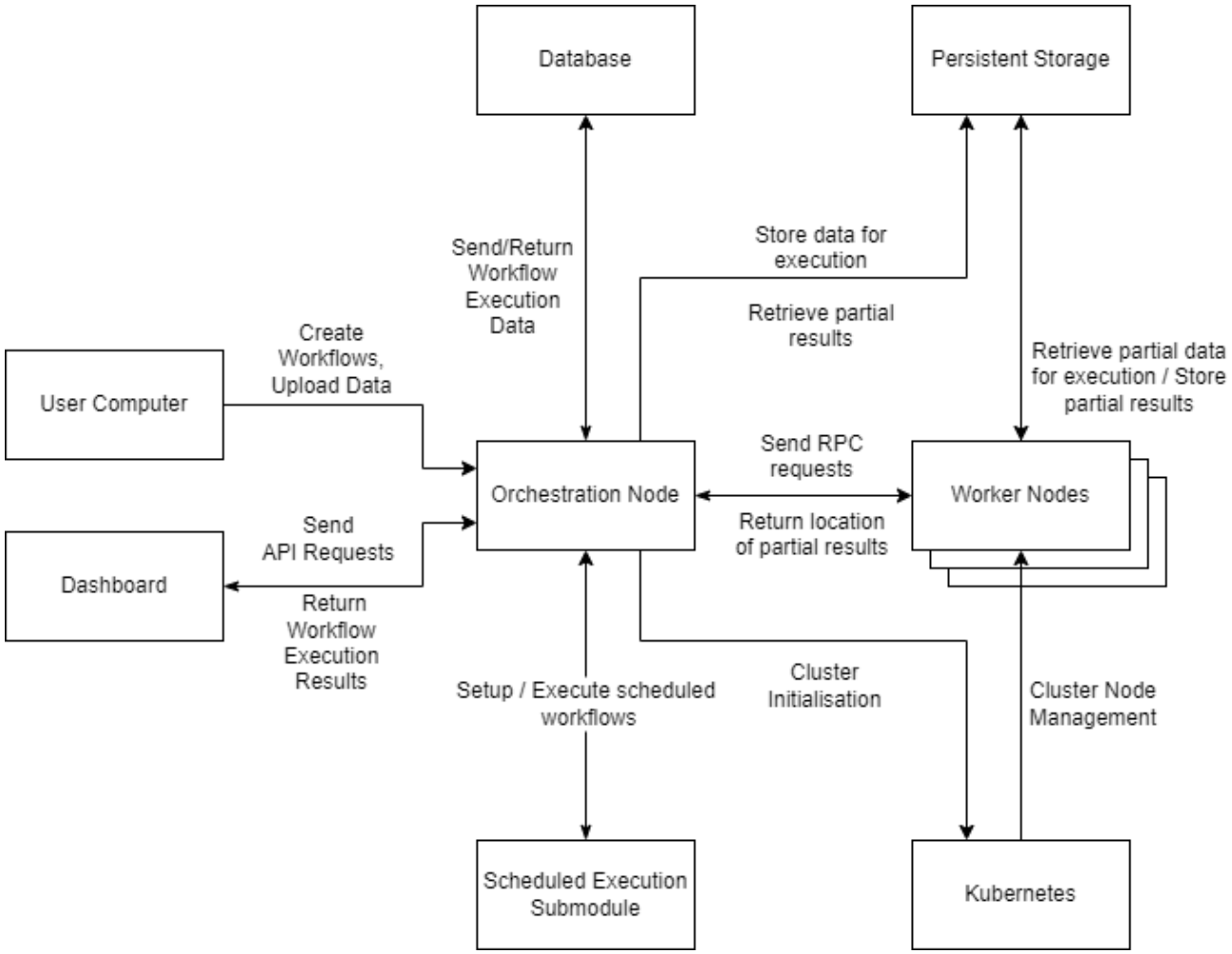
My aim is to build a framework on which data processing operations can be performed on a distributed cluster of nodes. These operations will include typical SQL commands, including:

- Filters
- Joins
- Group Bys

As part of the role I performed on my placement year, I had to spend a significant time actually executing the models we had written, as the data regularly changed format, slowing down the data ingestion process. Therefore, I would also like to build some tools into the framework to automate the following workflow:

- Ingesting new data
- Performing data processing
- Providing alerts for any execution issues
- Outputting the results

Based on the above description, the planned system will feature a number of core components. A UML diagram is included below showing how these will link together, followed by further information on each component:



Orchestration Node An orchestration node will be the main access point for the system, and will provide functionality over a REST API. It will take all data processing instructions (workflows), and the location of all files to execute on. This node will perform all activities related to starting, managing, and collating the results of the calculation.

More specifically, it will send instructions to each worker node (using RPC) on what processing to perform on which parts of the full dataset, it will handle restarting any failed nodes and ensuring calculations are fully complete in this case, and it will combine the partial results from all worker nodes to produce the final output. The orchestration node will also need access to an lightweight persistent database in order to store and update the results of workflow executions.

Furthermore, the orchestration node will feature a submodule that will allow it to perform scheduled and repeated executions of workflows, and potentially watch a given file location for changes to initiate a workflow automatically.

Worker Nodes The system will feature a number of worker nodes. Each node will be provided the instructions to execute, and a subset of the entire dataset. It will execute the instructions on the data, persist the results to disk, then send the location of the results to the orchestration node. Dynamic load balancing can be implemented here, as each node will be provided only a small subset of data to execute, then request more work as required.

The worker nodes will be managed using a Kubernetes or Docker Swarm cluster.

File Storage The system will feature a distributed file storage component. This will aim to provide resiliency against hard drive failure, while also improving file access times by storing data in locations close to the worker nodes. I haven't yet decided whether this will use an existing storage solution or

database, or if I will implement my own. My main goal with this component is to provide quick access times to data that can be grouped by a unique key, which means a tool/framework that supports indexing or partitioning might be suitable.

Dashboard I aim to implement a dashboard, which will provide a convenient frontend for the user to access workflows they have created, including information on previous executions, and will allow the user to initiate instant and scheduled executions on the orchestration node.

3 Previous Work

This area is well researched, and many solutions already exist with varying advantages and disadvantages:

MapReduce Google's MapReduce [1] is an framework designed to perform parallelised data processing using two operations - map, and reduce. It is an extremely simple framework to use, but from the research I have performed so far, contains two key weaknesses. Firstly, it performs many save and load operations to disk. This provides significant resiliency to both node and disk failure (as it utilises HDFS for resilient file storage), but it also significantly slows down processing compared to retaining data in-memory. Furthermore, the API does not transfer well to data that needs to be processed as a group according to some key.

Apache Spark Apache Spark [2] is another alternative which solves problems much closer to what I'm trying to achieve. It provides higher-level API, including both Panda DataFrames, and SQL APIs. However, it again struggles with grouped data, and also has some issues with large datasets as it attempts to store as much of the data as possible in memory

4 Potential Challenges and Difficulties

I foresee a number of interesting challenges I will need to solve as part of this project:

1. Setting up and managing the cluster
    - I expect to use an existing framework for initialising and managing both the main and worker nodes, like Kubernetes or Docker Swarm.
    - However, while this means I will not have to consider challenges like restarting failed nodes from a container perspective, there will still be other resilience considerations like ensuring the workflow recalculates any results that were lost by failed nodes.
2. Storage of Data
    - I haven't yet decided whether I will be using an existing distributed file storage solution or developing my own, as there are a number of problems to solve here, and it could result in significant performance benefits for the framework.
    - I need to spend further time investigating this as soon as I begin work on my project, as including this as a task to complete will be a significant amount of work on its own.
3. Loan balancing and optimisation
    - Loan balancing will need to be implemented to ensure that work is allocated to effectively use all nodes' resources.

- There will need to be a significant amount of optimisation performed across the project, including getting the data from disk, performing the actual processing, and compiling the results.

4. Test data

   - I will not be able to use any data from my placement role due to confidentiality issues.
   - However, it should be relatively straightforward to generate test data for the existing use cases, as they are very well defined and I have prior experience working with the data. These use cases include recalculating interest on loans, and recalculating transaction fees.

5 Methodology

I plan to implement the majority of the framework in a strongly-typed object-oriented language like Java, Scala or C#. However, the framework will target Python as the language for creating workflows, as it is widely used within the team I was part of on my placement year. I will also use either Kubernetes or Docker Swarm to handle the cluster management, and I have yet to decide whether I will be using an existing distributed file system for retaining the data on disk, or implementing my own solution.

I will be following Agile methodology when creating my project, splitting development into weekly sprints. This will allow me to ensure my goals for each week are achievable, and that I am making consistent progress.

6 Major Milestones

Some key milestones for my project include:

- **Project Inspection/End of Term 1:** Storage solution for data implemented, initial setup of Orchestration and Worker Nodes, features implemented to be able to make simple requests and collate the results. Written up first draft of 'Motivation & Aims' section of project report.
- **End of December:** Full setup of Orchestration and Worker Nodes completed, including resilience measures for failed nodes. Written up first draft of 'Methodology' section of project report.
- **End of February:** Frontend Dashboard and Scheduled Execution Submodule written, completed further editing of project report based on recent changes to the system.
- **Project Demonstration/End of Term 2:** Code bugfixes complete, performed performance and user testing. Written up first draft of 'Results' section of project report.
- **Project Submission Date:** Written up 'Evaluation' section, complete final editing of project report.

References

[1] Jeffrey Dean and Sanjay Ghemawat. Mapreduce: simplified data processing on large clusters. *Communications of the ACM*, 51(1):107–113, 2008.

[2] Matei Zaharia, Reynold S Xin, Patrick Wendell, Tathagata Das, Michael Armbrust, Ankur Dave, Xiangrui Meng, Josh Rosen, Shivaram Venkataraman, Michael J Franklin, et al. Apache spark: a unified engine for big data processing. *Communications of the ACM*, 59(11):56–65, 2016.