CPSC416 Capstone Project

Distributed Load Balancer Reflection

Vincent Kohm

Introduction

This document provides an in-depth examination of my capstone project, "Distributed Load Balancer," focusing on its design and implementation phase. The report aims to articulate the challenges faced, the strategies employed, and the learning outcomes of this significant academic endeavor. The initial section addresses the design phase, highlighting how the imposed constraints and focus on a specific core function—identifying the most frequent integer in a dataset—guided my approach towards mastering distributed computing concepts. This part of the report also evaluates the use of the Unified Modeling Language (UML) Class Diagram in the design process, acknowledging its role as a strategic roadmap despite some limitations. A significant portion of the text is dedicated to discussing the intricacies of my engagement with gRPC (gRPC Remote Procedure Calls). This includes the complexities involved in its installation and configuration, the underestimation of the proto file's significance, and the challenges in implementing the header files generated from the proto file. Furthermore, the report reflects on some unresolved issues within the project's timeline, such as network environment considerations and the system's synchronous operation using gRPC. These sections not only describe the hurdles encountered but also outline my intentions for future improvements and adaptations of the system. Towards the end, the document shifts to a reflective tone, emphasizing the importance of problem-solving skills in computer science, especially in complex project settings like distributed systems. I also ponder the potential advantages of collaboration in future projects, considering the benefits of teamwork for in-depth analysis and effective workload distribution.

Objectives of the Original Design

The project aims to address the challenges of computational resource demands in Machine Learning tasks, with a focus on hyperparameter tuning and the use of grid search cross-validation. Machine Learning tasks often require significant computational resources, which can be a bottleneck in the development and deployment of models. Hyperparameter tuning, the process of selecting the optimal configuration of model parameters, is a crucial step in improving model performance. However, this process can be time-consuming and computationally expensive, particularly when using grid search cross-validation, which involves testing multiple combinations of hyperparameters.
To address these challenges, the project proposes a Distributed Load Balancer System Design, which distributes the computational workload across multiple nodes to improve efficiency and reduce processing time. The system is designed to read a dataset from a disk and execute a series of bulk 'put' operations to insert this data into a distributed storage system. The application layer then generates and distributes subtasks across a network of worker nodes, each of which fetches the required data from the appropriate storage node and processes it to compute an intermediate result. The final result is then aggregated and displayed in the system's console.

While the aforementioned tasks related to Machine Learning (ML) models are of considerable relevance, the scope of this investigation was intentionally constrained to simplify the problem. This approach is adopted to maintain a focused examination on the architecture of the distributed system, which constitutes the primary objective of this project. To elaborate, engaging with complex tasks distributed across nodes could inadvertently shift the emphasis towards resolving the intricacies of the task itself, rather than concentrating on the nuances of the distributed design. Nevertheless, it is imperative to note that the simplified problem under consideration still maintains a conceptual parallelism with more complex tasks in the domain of ML. The task selected for this study involves traversing a dataset comprising 1 billion integers and identifying the most frequently occurring integer. This task, while simplified, effectively illustrates the key principles of distributed computing relevant to this research.

Overall, the project aims to improve the efficiency and reduce the processing time of Machine Learning tasks by distributing the computational workload across multiple nodes. By doing so, the project seeks to address the challenges of hyperparameter tuning and grid search cross-validation, which can be time-consuming and computationally expensive.

Successes and Triumphs

In the design phase of my capstone project, entitled "Distributed Load Balancer," a critical juncture was reached that significantly influenced the subsequent implementation process. My initial skepticism about the rigorous constraints imposed on this project eventually gave way to an appreciation for their role in refining and focusing my efforts. These constraints not only facilitated a more streamlined implementation but also ensured a clear path through the complex terrain of development. A prime example of these beneficial constraints was the deliberate limitation placed on the system's core function: the identification of the most frequent integer within a large dataset. This specific focus allowed me to channel my efforts towards exploring and mastering distributed computing concepts, including initial data transfer, equitable data distribution among worker nodes, and the intricacies of node-to-node communication. This targeted approach was a strategic deviation from my original plan, which revolved around the complexities of hyperparameter optimization. By narrowing the project's scope, I was able to delve deeper into the nuances of distributed systems rather than becoming mired in the complexities associated with hyperparameter optimization. This pivot in focus is indicative of a significant self-evaluative process; it reflects a judicious balance between embracing a challenging problem and remaining within the realm of my current skill set. This aspect of my project underscores a critical aspect of my academic journey: the ability to critically assess and align project objectives with personal competencies and academic goals. It stands as a testament to the dynamic interplay between ambition and practicality, a balance that is essential in the field of computer science.

Additionally, one of the most beneficial aspects of my design phase was the incorporation of a Unified Modeling Language (UML) Class Diagram. This diagram delineated the requisite classes for constructing the system as envisioned. Admittedly, the diagram occasionally fell short in precision, particularly in aspects such as return types, helper functions, and encapsulation details. Despite these limitations, it served as an invaluable roadmap, underpinning the implementation strategy. The class diagram afforded me a comprehensive overview of the system's components, both in terms of their development status and interconnectivity. This comprehensive perspective was particularly advantageous when determining access patterns to individual classes. Furthermore, the design anticipated numerous challenges that might emerge during implementation. These predictions significantly mitigated the occurrence of unforeseen complications. Being forewarned of potential issues, although not exhaustively addressed in the design, rendered the process considerably less daunting.

Additionally, my academic background, encompassing courses like Computer Networks, Computer Organization, and Object-Oriented and Functional Programming, as well as Operating Systems, profoundly influenced the implementation phase. Theoretical concepts, such as network port availability, thread safety, process management, and mutex locks, were not only relevant but proved to be crucial in the practical application. The knowledge gleaned from these courses facilitated a swift and effective application of these concepts, thereby enhancing the efficiency of the implementation process. Exemplary of this, is the thread pool implementation that I used to advance parallelism. Here, I needed to facilitate prior knowledge about server queues, threads, and locks to implement a correct thread pool class. This prior knowledge also allowed me to use this class effectively without conducting more research. This was helpful since my design did not specify any use of thread pools and therefore no planning had gone into this. This leads me to the shortcomings of my design.

Addressing the Challenges

Despite the overall effectiveness of my project design, certain aspects fell short, particularly in facilitating a seamless implementation phase. I wish to delineate these shortcomings, notably in my engagement with gRPC (gRPC Remote Procedure Calls) and reflect on the consequent challenges and learning opportunities.

The initial challenge lay in the installation and configuration of gRPC. Aligning the versions of the protocol compiler (protoc) with the gRPC version proved to be a laborious task. The intricacies involved in selecting compatible versions and navigating the correct installation pathways precipitated several dependency issues, significantly impeding progress for a few days. Nonetheless, the comprehensive documentation of gRPC eventually provided the necessary guidance, enabling me to operationalize a basic gRPC example. Additionally, a particular blog post was instrumental in resolving these issues, especially through its

modified CMakeLists file, which facilitated the generation of headers from the .protoc file and compiled my source files efficiently.
Another critical deficiency in my design was the underestimation of the importance of the proto file in the gRPC framework. This file, which outlines all the services and data types for the gRPC protocol, was overlooked in my initial design. This oversight necessitated additional research and delayed the development process, as it is crucial for defining service methods like PUT and GET, along with their respective input and output data types.

Furthermore, the practical application of the header files, generated from compiling the proto file, posed a significant challenge. These headers are vital for implementing the service functions defined in the proto file. My lack of foresight in researching the nuances of gRPC and its communication mechanisms led to a deceleration of the development process.

Regarding the implementation guidance provided by my design, it was somewhat limited. While it served as a basic framework, it lacked specific directives on the sequence of implementing various components. This absence of a clear implementation order and a detailed description of system behavior complicated the testing process. It became particularly apparent after the initial implementation of components, as there was no predefined methodology to test these elements in isolation. Debugging a complete system, especially as it increases in complexity, is a formidable task. However, I managed to devise client files for testing individual components, such as the Storage Node and its GET and PUT methods. Testing the Storage Coordinator required extensive use of debug statements for verification. In hindsight, the design should have included comprehensive guidelines for testing individual components, detailing their expected behavior and suggesting a sequential roadmap for implementation.

Lastly, the design inadequately addressed the sizing of subtasks distributed among worker nodes. While a static size for every problem scenario is impractical, the design should have contemplated a dynamic approach to task sizing relative to the dataset's complexity. For instance, a subtask size too small relative to a large dataset, or too large for a smaller dataset, would lead to inefficiencies. A more adaptive strategy for determining task size could have significantly optimized system performance.

In summary, while my design laid a foundational framework, it required additional refinement in several key areas, particularly concerning the integration and optimization of gRPC, testing methodologies, and task sizing strategies. These learnings have been instrumental in my development, underscoring the importance of thorough research and adaptive planning in complex project execution.

Unresolved Issues

While I successfully addressed certain design shortcomings through additional research and time investment, there were other issues that remained unresolved within the implementation phase's allotted timeframe. A notable oversight in my design was the lack of consideration for the specific network environment in which the system would operate. The design did not specify whether the system would run locally or across different networks. My initial strategy involved utilizing containerization to deploy instances of my Storage and Worker Nodes and executing computations on disparate hardware components. However, this approach encountered connectivity challenges within the University's network, likely due to security protocols that restricted access to network ports. Efforts to circumvent this through port forwarding revealed the issue's complexity, leading to a decision to run the system locally on my machine. This pivot refocused my attention on implementing the core concepts of the distributed system, especially given the substantial time invested in resolving gRPC-related challenges. Nonetheless, I am optimistic that, with additional time, I can enhance future iterations of my system to operate on various physical hardware components, more accurately reflecting a genuine distributed system.

Another unresolved challenge was related to the operational architecture of gRPC within my system. Upon implementing most service functions, I observed that the system operated synchronously. While effective in several scenarios, this synchronous model was suboptimal for the intended parallel access pattern to Storage Nodes. This architecture introduced a bottleneck, as clients were required to await the completion of preceding requests. Preliminary research into transforming the system into an asynchronous model suggested that such modifications would be too extensive to implement within the existing timeframe. However, to enhance the project's efficacy, I intend to revisit this aspect and adjust the system to alleviate the bottleneck issue, moving towards an asynchronous framework.

Embracing Challenges as Learning Opportunities

While my design project incorporated numerous beneficial guidelines, it also exhibited limitations in certain areas. Such shortcomings were anticipated, given that this was one of my initial forays into crafting a blueprint for a complex system. It was foreseeable that the design might not provide comprehensive guidance for every conceivable scenario, a common occurrence when venturing into new territories of system design. Despite these gaps, I take pride in the manner I approached and resolved these challenges. A fundamental skill for a computer scientist is the ability to isolate specific problems, conduct a thorough analysis, and devise potential solutions. While these solutions may not always be successful on the first attempt, they are critical stepping stones towards resolution. A case in point was the challenge of testing individual components, such as the Storage Node. The absence of specific instructions in my design for testing this component led me to innovate an external client component, which was not originally specified in the system. This makeshift component enabled me to connect to the Storage Node and rigorously test its functionalities for accuracy, a method that also proved beneficial in assessing the Worker Node. Such scenarios are almost inevitable in the realm of system development, and it's important not to be disheartened by generic guidelines or incomplete designs. Instead, these instances should be viewed as opportunities for creative problem-solving and learning. I am confident that with continued practical experience, my proficiency in addressing and resolving these challenges will enhance. Developing this skill set is crucial in the process of designing and implementing complex systems. It fosters a capacity for logical and efficient problem-solving, which is invaluable in the ever-evolving field of computer science.

Preconditions and Assumptions

In reflecting upon the preconditions of my project, I cannot pinpoint any specific knowledge that required unlearning; however, I did find that I had somewhat underestimated several facets of the project's process. A primary revelation was the significant efficiency gains attainable through a well-structured and detailed design. The comprehensive nature of the design was not just an auxiliary tool but a critical component in the project's success. Without this foundational element, achieving a correctly functioning end product would have been doubtful.

Moreover, the project illuminated the complexities inherent in what might seem like straightforward distributed computing concepts. The act of messaging a node, a concept that appears simple in theory, revealed itself to be quite intricate in practice. Managing the influx of high-traffic messages presented a challenge, especially in discerning the importance of each message to ensure efficient system runtimes. This aspect of the project was unexpectedly challenging, underscoring the complexity hidden within seemingly basic operations of distributed systems.

Lastly, ensuring expected system behavior through error handling proved to be a daunting task. The countless of potential failure points in complex systems necessitates thorough and meticulous testing of each function. The difficulty in preemptively identifying and addressing these potential failure scenarios was a surprising aspect of the project. This experience highlighted the criticality of robust error handling and testing protocols, particularly in intricate systems where numerous variables and interactions can lead to unforeseen issues.

Personal Reflection

In reflecting upon my recent project, I am profoundly satisfied with the outcomes and the extensive learning experience it provided, particularly in the realm of planning and executing a complex project. My contributions to this project, undertaken independently, have been a source of personal fulfillment. However, during the course of this endeavor, I recognized numerous instances where collaboration with a team member would have been advantageous. This was most apparent during the research phases involving intricate concepts such as gRPC and networking.

Despite undertaking this project solo, I am pleased with the extent of research and the level of detail I achieved, considering my limited experience in design processes and the constrained timeframe allocated for the project. One aspect of my project that warrants particular discussion is its focus on horizontal scaling to enhance system efficiency and computational speed. Contrary to expectations, the system's performance was slower compared to a conventional monolithic architecture. This issue was addressed in my design phase, leading to repeated contemplations about the viability and effectiveness of the scaling approach adopted. Initially, the results, while accurate, were disappointing in terms of efficiency.

However, a pivotal conversation with Tony shed light on the true essence of the project. It became clear that the primary objective was not to achieve superior computational efficiency, but rather to delve into the exploration of distributed systems with an emphasis on the design phase. From this perspective, the project was immensely successful, significantly enriching my understanding of numerous complex concepts. Tony's insights were instrumental in arriving at this conclusion.
For future projects of a similar scope, I am inclined to seek collaboration with a like-minded individual. Such teamwork would not only enhance the enjoyment and depth of the project but also prove essential for in-depth discussions, particularly when navigating the complexities inherent in distributed systems. The presence of a teammate would enable an effective division of labor, akin to a load balancer, allowing for a more focused exploration of specific aspects and providing a clearer implementation roadmap.

I would also like to reflect on feedback I acquired throughout the project. While I did not receive much specific feedback on my project, I attempted to participate during lecture discussions, and attend some office hours from the teaching assistants or Tony himself. These discussions helped to deepen my understanding of several distributed systems concepts. However, I must admit that throughout the design phase I tended to be rather shy when it came to asking specific questions. Exemplary here was my limited knowledge on the communication between nodes which I should have asked more questions about when I had the chance. Not only did this slow down my implementation process but it also ended up as a missed opportunity to gain a deeper insight into an essential distributed system concept. I believe the reason for not seeking help in such instances could be explained with the imposter syndrome which gave me the feeling that in many cases I should know better and seeking for help seemed a little embarrassing. For future projects, I will try to avoid these scenarios. A potential solution would be to collaborate on a project like this as in group environment as the imposter syndrome usually does not come into effect with the same intensity in such settings.

In addition to these reflections, I am also planning to make several adjustments to the project, addressing areas identified during this report. This continuous process of review and improvement is vital for my growth and development in the field and I hoping to achieve better results upon completion of my points of improvement.

Relevance to Course Concepts

Throughout the course we have discussed many essential distributed system core concepts such as data replication, fault tolerance, data storage, storage access patterns, consensus protocols, or leader elections. As this was first introduction to distributed systems most of these concepts were completely new to me and quite challenging to grasp at times. However, through diligent reading and research I would argue that I managed to understand the core idea of each concept which allowed me to understand distributed systems a lot better on a high level. However, due to the natural intricacies in the subject I deliberately choose to limit the scope of my project to only a fraction of these concepts. Most notably, data distribution across several nodes, dynamic task distribution across worker nodes, and the general idea of scaling up a project and designing a concept that allows the shift from a monolithic architecture to a distributed architecture or in other words, scaling up horizontally. In addition to applying these above-mentioned concepts, my project also included appropriate discussions with regards to limitations, scalability, and fault tolerance, all of which represent intrinsic within the realm of distributed systems. While it is debatable how well I implemented these concepts into my project, it is fair to say that enough understanding was presented since the system produced a correct result after finalizing the implementation phase.

I also believe this project provides a strong foundation to add additional functionalities and illustrate my understanding of other concepts such as data replication and consistency. In future, I am not opposed of the idea of keep working on this project and adding more complexity to further test my knowledge on distributed systems.

Conclusion

In conclusion, the "Distributed Load Balancer" capstone project has been a profound learning journey, merging theoretical knowledge with practical application. It presented numerous challenges, from the intricacies of gRPC installation to the complexities of network environments, each contributing significantly to my personal development with regards to these concepts. The project underscored the importance of detailed planning, problem-solving, and the adaptability required in complex system design. While certain aspects did not go as anticipated, the experience gained in addressing these hurdles was invaluable. Looking forward, I am eager to apply these learnings to future projects, potentially collaborating with peers for a more comprehensive exploration and implementation. This reflection not only marks the completion of a fun capstone project but also lays the groundwork for continued growth and development in the field of computer science, and distributed systems in particular.