

UNIVERSITY OF CALIFORNIA  
SANTA CRUZ

**BENCHMARKING REAL WORLD APPLICATIONS TO FIND  
SCALABILITY LIMITATIONS IN TIDALSCALE**

A dissertation submitted in partial satisfaction of the  
requirements for the degree of

MASTERS OF SCIENCE

in

COMPUTER SCIENCE

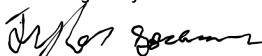
by

**Ananya Das**

June 2023

The Dissertation of Ananya Das  
is approved:

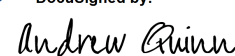
DocuSigned by:



E6842F7C519B427...

Professor Tyler Sorenson, Chair

DocuSigned by:



223B2CC177124AA...

Professor Andrew Quinn

---

Alexander Wolf

Dean and Distinguished Professor of Computer Science

Copyright © by

Ananya Das

2023

# Table of Contents

List of Figures	iv
Abstract	v
Acknowledgments	vii
<b>1 Introduction</b>	<b>1</b>
1.1 Motivation . . . . .	3
1.2 Background . . . . .	5
<b>2 Experimental Setup</b>	<b>8</b>
2.1 1-Node Setup with 30 Cores: . . . . .	8
2.1.1 Scope of the setup . . . . .	10
2.2 3-Node Setup: . . . . .	12
2.2.1 Scope and potential scalability problems of the setup: . . . . .	13
<b>3 Results:</b>	<b>14</b>
3.1 Results for MonetDB: . . . . .	14
3.1.1 Results for 1 Node Setup: . . . . .	15
3.1.2 Results for 3 Node Setup: . . . . .	17
3.2 Results for Gabps: . . . . .	21
3.2.1 Normal Gabps Results: . . . . .	21
3.3 Gabps with "Compare and Swap" Removed: . . . . .	22
<b>4 Conclusion</b>	<b>25</b>
<b>Bibliography</b>	<b>27</b>

# List of Figures

2.1	1-Node Setup with 30 Cores . . . . .	8
2.2	3-Node Setup with 30 Cores . . . . .	12
3.1	<i>query_results</i> for 1 Node Setup . . . . .	17
3.2	<i>query_results</i> for 3 Node Setup . . . . .	20

## **Abstract**

Benchmarking real world applications to find scalability limitations in TidalScale

by

Ananya Das

This report presents an investigation into the scalability and performance of real-world applications such as MonetDB and GAPBS on TidalScale systems, with a focus on identifying and understanding scalability issues. TidalScale's software-defined servers enable the dynamic aggregation of commodity hardware resources, providing a unique environment for studying the behavior of single computer real-world applications in large-scale distributed systems [1]. The objective of this project is to evaluate the scalability and performance characteristics of MonetDB and GAPBS across different core configurations, shedding light on their behavior, potential optimizations, and scalability challenges. To achieve this, a comprehensive set of experiments was conducted using TidalScale configurations with varying numbers of cores. Multiple performance metrics, including query execution time, throughput, and scalability, were measured to assess the application performance under different workloads. The experiments utilized carefully selected datasets, and any necessary preprocessing steps were applied to ensure reliable and meaningful results. The results obtained from the experiments are presented and analyzed in detail. The findings reveal valuable insights into the scalability and performance characteristics of how applications scale on TidalScale systems, with a specific emphasis on identifying and understanding scalability issues. The analysis discusses the

observed trends, bottlenecks, and trade-offs associated with increasing core counts, providing a deeper understanding of their behavior in distributed and parallel computing environments. In addition, a comparison is made with relevant prior studies or benchmarks to highlight similarities, differences, and novel insights uncovered by this research. Limitations encountered during the experiments are acknowledged, and suggestions for future work and improvements to the experimental setup are provided. This study contributes to the understanding of database scalability and performance evaluation in the context of TidalScale systems, with a particular focus on identifying and understanding scalability issues. The insights gained from this research can inform optimizations and guide the selection and configuration of databases for large-scale distributed environments. The findings have practical implications for organizations leveraging TidalScale technology and offer a foundation for further investigations in this field.

## Acknowledgments

I would like to express my sincere gratitude to all those who have supported and contributed to the successful completion of this research project.

I would like to express my deepest gratitude to my advisor, Andrew Quinn, whose guidance and support have been pivotal to the success of this research project. Throughout the course of this study, Andrew has consistently provided valuable insights and ideas on how to approach various problems and challenges. His extensive knowledge of the field and his ability to reference relevant papers and resources have greatly enriched the research process. Moreover, Andrew's keen eye for detail has helped identify potential pitfalls in my approach and provided invaluable suggestions for improvement. I am truly grateful for his mentorship and the countless hours he has dedicated to reviewing and refining my work.

I am grateful to my reader and co-advisor, Tyler Sorenson, for his valuable insights and expertise in reviewing my project. His constructive feedback has immensely contributed to improving the quality and clarity of my work.

Special thanks go to Pooneh Safayenikoo, a PhD student in my lab, for his guidance, collaboration and assistance throughout this project. Her willingness to share ideas have greatly influenced the outcomes of my project. Additionally, I would like to express my gratitude to all the researchers in my lab who have created a stimulating academic environment and provided me with invaluable opportunities for learning and growth.

# Chapter 1

## Introduction

In recent years, the proliferation of big data and the increasing demand for high-performance computing have driven the exploration of scalable and efficient database systems. TidalScale, a leading provider of software-defined servers, offers a unique solution that allows the dynamic aggregation of commodity hardware resources into a single, large-scale system. This technology opens up new possibilities for studying the scalability and performance characteristics of databases in distributed and parallel computing environments.[1] The objective of this project is to investigate the scalability and performance of two prominent real world applications, MonetDB and GAPBS, on TidalScale systems. MonetDB is an open-source columnar database known for its columnar storage model and query optimization techniques. To evaluate MonetDB, large-scale real-world benchmarks representing realistic workloads and data sizes will be used. These benchmarks provide a comprehensive assessment of MonetDB's scalability and performance under real-world conditions, allowing us to understand its behavior



in large-scale distributed systems. GAPBS, on the other hand, is a graph benchmark suite specifically designed to evaluate the performance of graph algorithms on parallel and distributed platforms. For GAPBS, small-scale benchmarks will be employed to assess its scalability and performance on TidalScale systems. These benchmarks capture key graph algorithms and their associated workloads, providing insights into GAPBS's behavior in parallel computing environments. In addition to the small-scale benchmarks, microbenchmarking techniques utilizing compare and swap operations will be employed for GAPBS. This approach enables fine-grained analysis of specific operations and synchronization mechanisms in the graph algorithms, allowing us to gain a deeper understanding of GAPBS's performance characteristics at a lower level.[1]

By evaluating these applications on TidalScale systems using diverse benchmarking approaches, we aim to gain insights into their behavior and identify opportunities for optimization in large-scale, data-intensive scenarios. The experiments will explore the impact of varying core counts on query execution time, throughput, and other performance metrics. By analyzing the results, we can observe how MonetDB and GAPBS leverage the available resources and assess their scalability limits, bottlenecks, and trade-offs in TidalScale environments.

Understanding the scalability and performance characteristics of applications in the context of TidalScale systems is essential for organizations that rely on big data processing and high-performance computing. It enables them to make informed decisions when selecting and configuring database systems for their distributed and parallel computing environments. Furthermore, the findings from this project contribute to the

broader field of database optimization, providing insights that can drive further research in scalability and performance evaluation.

In the following sections of this report, we will describe the experimental methodology, present the results of the conducted experiments, analyze the findings, and discuss their implications. By evaluating the scalability and performance of MonetDB and GAPBS on TidalScale systems using large-scale real-world benchmarks, small-scale benchmarks, and microbenchmarking compare and swap, this research aims to contribute to the understanding of database behavior in distributed computing environments and provide valuable insights for database system optimization and selection.

## 1.1 Motivation

The main motivation of this project is to comprehensively evaluate and address the scalability challenges associated with TidalScale’s distributed system technology. Scalability is a critical factor in determining the effectiveness and performance of distributed systems, and understanding the scalability issues specific to TidalScale is essential for optimizing its utilization.

To achieve this goal, we have adopted a two-fold approach utilizing two different hardware configurations: a single-node setup and a three-node setup. This approach allows us to systematically analyze the scalability characteristics of TidalScale’s distributed system across varying levels of resource availability.

By employing a single-node setup, we aim to identify and understand the scala-

bility limitations of applications when running without TidalScale’s distributed system. This control experiment enables us to isolate the inherent scalability issues within the applications themselves, without the influence of distributed system complexities. By observing how applications scale when utilizing different core counts on a single host, we can uncover any bottlenecks or inefficiencies that hinder their ability to leverage resources optimally.

Once we have established the baseline scalability characteristics in the single-node setup, we proceed to evaluate the performance of TidalScale’s distributed system using a three-node configuration. This configuration allows us to explore the impact of distributed resource allocation and coordination on scalability. By running applications across multiple nodes and observing their scalability patterns, we can identify how TidalScale’s technology addresses or mitigates the scalability issues previously identified in the single-node setup.

By combining the insights gained from both setups, we can develop a comprehensive understanding of the scalability challenges within TidalScale’s distributed system. This knowledge will enable us to propose enhancements, optimizations, or architectural modifications that improve the scalability and performance of TidalScale in real-world deployments.

In summary, the motivation of this project lies in the identification and understanding of scalability issues in TidalScale. By employing both single-node and three-node configurations, we can assess the scalability characteristics of applications and TidalScale’s distributed system technology. Ultimately, this project aims to drive

improvements in TidalScale’s scalability, ensuring its effective utilization in distributed computing environments.

## 1.2 Background

In today’s data-driven world, the efficient processing and analysis of large-scale datasets are critical for organizations across various industries. As data volumes continue to grow exponentially, there is an increasing need for scalable and high-performance database systems that can handle the demands of big data processing. In this context, TidalScale’s software-defined servers offer a compelling solution by enabling the dynamic aggregation of commodity hardware resources into a single, large-scale system.

TidalScale’s software-defined servers provide a flexible and cost-effective approach to scaling computational resources. By combining multiple commodity servers into a unified system, TidalScale allows organizations to create virtual servers with larger memory capacities and increased core counts. This technology enables the creation of a shared-memory architecture, where data and computations can be distributed and parallelized across multiple cores.[2]

MonetDB, an open-source columnar database, has gained attention for its columnar storage model and advanced query optimization techniques. It is designed to efficiently handle analytical workloads and perform complex queries on large datasets. Evaluating MonetDB’s scalability and performance on TidalScale systems can provide insights into its behavior in distributed environments, particularly when leveraging the

aggregated resources made available by TidalScale’s software-defined servers. This understanding is crucial for organizations that require efficient processing and analysis of large-scale data.

GAPBS, a graph benchmark suite, focuses on evaluating the performance of graph algorithms on parallel and distributed platforms. Graph algorithms play a vital role in numerous domains, including social network analysis, recommendation systems, and bioinformatics. By studying GAPBS’s scalability and performance on TidalScale systems, organizations can gain insights into the behavior of graph algorithms in distributed computing environments. This knowledge is essential for optimizing graph algorithm implementations and selecting appropriate parallelization and synchronization mechanisms.

Microbenchmarking compare and swap operations provide a fine-grained analysis of specific operations and synchronization mechanisms in GAPBS. By closely examining the performance characteristics of these operations, researchers can gain deeper insights into the behavior and efficiency of graph algorithms. This analysis helps identify areas for optimization and improves the overall understanding of the performance trade-offs involved in parallel and distributed graph processing.

By conducting experiments with large-scale real-world benchmarks for MonetDB, small-scale benchmarks for GAPBS, and microbenchmarking compare and swap operations, this project aims to explore the scalability and performance characteristics of these databases on TidalScale systems. The insights gained will contribute to the broader understanding of database scalability and performance evaluation in distributed

computing environments, provide guidance for database system optimization, and assist organizations in selecting and configuring database systems that can efficiently handle big data processing requirements.

In summary, the background of this project encompasses the increasing demand for scalable and high-performance database systems, the capabilities of TidalScale's software-defined servers, the features and strengths of MonetDB and GAPBS, and the significance of studying their scalability and performance on TidalScale systems. By delving into these aspects, this research aims to contribute to the optimization and selection of database systems for data-intensive distributed computing environments.

## Chapter 2

# Experimental Setup

### 2.1 1-Node Setup with 30 Cores:

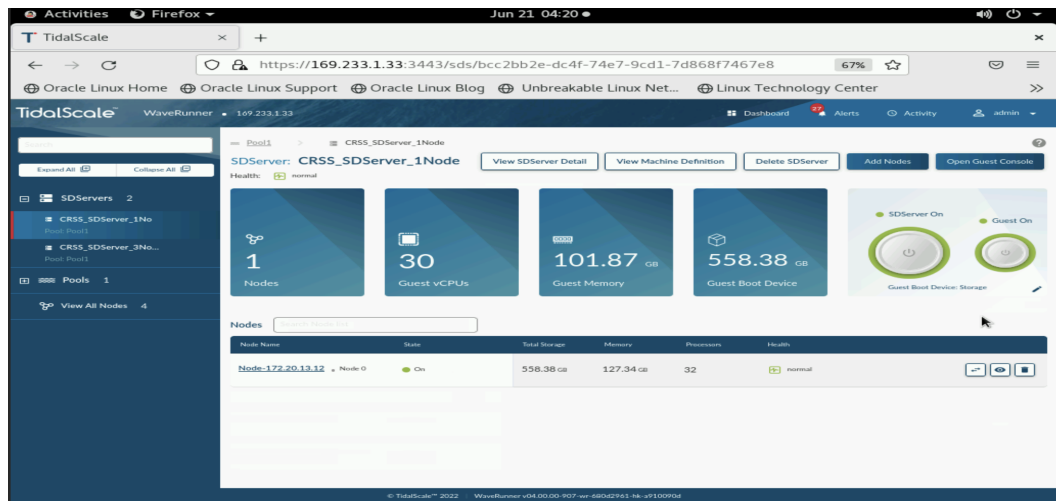


Figure 2.1: 1-Node Setup with 30 Cores

In our experimental setup, I utilized a 1-node system where there was a single physical node with a Guest Operating System (OS) running on it. Here's a brief explanation of

this setup:

In a 1-node system, you had a single physical server or machine that acted as the host for your experimental environment. This physical node served as the foundation for running the experiments and evaluating the performance and scalability of the databases under study.

On this physical node, you installed and configured a Guest OS. A Guest OS, also known as a virtual machine, is a software emulation of a computer system that operates within a host environment. It allows for the creation of multiple isolated virtual instances, each with its own operating system, on a single physical machine.

In your case, you had a single Guest OS running on the physical node. This Guest OS served as a virtual environment where you installed and executed the MonetDB and GAPBS databases. By using a virtual machine, you were able to create an isolated environment for each database, ensuring that they operated independently of the underlying host system.

The 1-node setup allowed you to evaluate the performance and scalability of the databases in a controlled environment. By running the experiments on a single physical node with a Guest OS, you could assess the behavior of the databases under study without the complexities and overhead associated with distributed systems. This setup provided a baseline understanding of how the databases perform in an isolated environment, paving the way for future experiments in larger-scale distributed systems.

It's important to note that while a 1-node setup provides valuable insights into the behavior of the databases on a single machine, it may not capture the full



complexities and performance characteristics that arise in distributed environments. Nonetheless, it serves as an initial step in understanding the performance and scalability of the databases and lays the foundation for further experiments and investigations.

### **2.1.1 Scope of the setup**

The scope of the project encompasses the evaluation of the scalability and performance characteristics of applications (MonetDB and GAPBS) in the TidalScale environment. The investigation focuses on two hardware configurations: a 1-Node setup and a 3-Node setup. The scope also includes the identification and understanding of scalability issues specific to TidalScale systems.

**Scalability Problems in the 1-Node Setup:** The 1-Node setup refers to running the applications on a single TidalScale node, which aggregates and virtualizes multiple commodity hardware resources. While the 1-Node setup provides a unique environment for studying the behavior of applications in large-scale distributed systems, it may also pose certain scalability challenges. Some potential scalability problems in the 1-Node setup could include:

**Resource Constraints:** In a single TidalScale node, there may be limitations on available CPU cores, memory, or other hardware resources. As the workload increases or the dataset size grows, the application may encounter resource bottlenecks that impact its scalability and performance.

**Communication Overhead:** The TidalScale software-defined server dynamically aggregates hardware resources from multiple nodes into a single logical system.

However, in a 1-Node setup, the communication and coordination between these aggregated resources might introduce additional overhead and latency. This overhead could limit the scalability of the application.

**Synchronization and Parallelization:** Achieving efficient synchronization and parallelization in a 1-Node setup can be challenging. As the application scales within a single TidalScale node, coordinating and synchronizing the execution of parallel tasks across multiple cores or threads can become more complex, potentially leading to scalability issues and diminishing returns.

**Scalability Limits:** The 1-Node setup may have inherent scalability limits due to the constraints of a single physical node. These limits could manifest as diminishing returns or diminishing performance gains beyond a certain number of cores or threads.

Identifying and understanding these scalability problems in the 1-Node setup is crucial for optimizing the application's performance, improving resource utilization, and making informed decisions about scalability planning and resource allocation in TidalScale systems.

It's important to note that the specific scalability problems in the 1-Node setup may vary depending on the characteristics of the applications being evaluated, the workload patterns, and the configuration of the TidalScale system. The project's experiments and analysis will aim to uncover and address these scalability challenges to provide insights and recommendations for optimization and improvement.

It's important to consider that while the 1-node setup has its limitations, it serves as an essential component of your overall research. It enables you to establish foundational

knowledge and insights about the performance and scalability of the databases, which can guide subsequent experiments and evaluations in larger-scale distributed environments.

## 2.2 3-Node Setup:

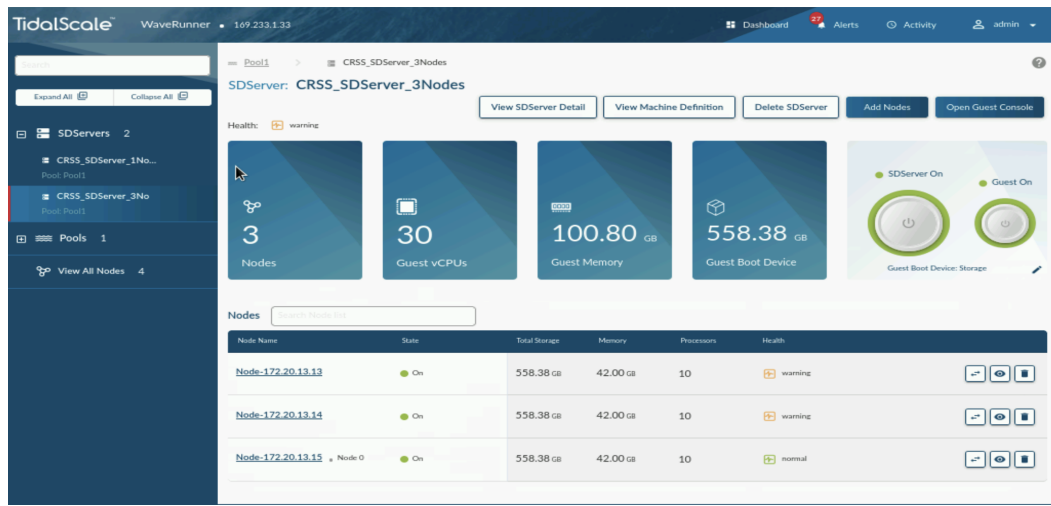


Figure 2.2: 3-Node Setup with 30 Cores

In a 3-node setup with a Guest OS on 1 node and 10 cores in each of the nodes, the experimental environment consists of three physical nodes interconnected to form a distributed system.

Overall, the 3-node setup with a Guest OS on 1 node and 10 cores in each of the nodes provides a distributed environment for running the databases. It allows for increased scalability, parallel processing, and resource utilization compared to the 1-node setup. This configuration enables the evaluation of the databases' behavior and performance in

a distributed computing environment, which more closely resembles real-world scenarios involving multiple interconnected nodes.

### **2.2.1 Scope and potential scalability problems of the setup:**

The scope of the 3-node setup with a Guest OS on 1 node and 10 cores in each of the nodes expands the evaluation and experimentation capabilities compared to the 1-node setup.

The 3-node setup broadens the scope of the experimental environment, encompassing distributed system evaluation, scalability assessments, parallelism analysis, interconnect considerations, fault tolerance investigations, and realistic workload simulations. It allows for a more comprehensive understanding of the behavior and performance characteristics of the databases in distributed computing environments, providing insights relevant to real-world deployments and helping guide system optimizations and configurations.

It's important to consider these potential scalability problems when designing and evaluating systems in a 3-node setup. Proper system design, load balancing strategies, data partitioning approaches, fault tolerance mechanisms, and optimization techniques can mitigate these issues and improve scalability. Addressing these challenges ensures that the system can efficiently handle increasing workloads, datasets, and the demands of distributed computing environments.

## Chapter 3

### Results:

In this section, we present the results of our study on the scalability of MonetDB and Gabps as the number of cores is increased in both a one-node and three-node setup. Our investigation aimed to determine which types of queries on these applications exhibit scalability and perform better in a setup like TidalScale. Additionally, we discuss the possible reasoning behind the observed results and assess whether MonetDB and Gabps applications are suitable candidates for deployment in a TidalScale environment.

#### 3.1 Results for MonetDB:

In this section, we discuss the scalability results for MonetDB in two configurations: a one-node setup and a three-node setup. It should be noted that the one-node setup does not utilize TidalScale's distributed system capabilities.

### **3.1.1 Results for 1 Node Setup:**

In the one-node setup, we studied 22 different queries across various core configurations, ranging from 1 to 30 cores. While all queries showed scalability as the number of cores increased, the rate of performance improvement varied significantly among queries.

#### **3.1.1.1 Observations:**

As observed in figure: 3.1 and table: 3.1 queries 17 and 12 exhibited excellent scalability, showing a substantial improvement in performance with increasing cores. Most other queries reached a point of constant performance after a certain core count. Queries 13, 16, and 18 displayed a consistent rate of growth for almost all core configurations.

Table 3.1: Query Results for 1 Node Setup

Query No	1 Core	2 Cores	4 Cores	8 Cores	16 Cores	20 Cores	30 Cores
1	83.6875	45.65	25.0775	15.43	12.415	12.1825	13.01
2	2.3725	1.225	0.6825	0.405	0.28	0.2725	0.2075
3	17.13	9.3875	5.84	3.865	3.295	3.3125	3.1425
4	16.2725	8.4675	4.685	2.805	2.135	2.1925	2.055
5	25.38	13.5475	7.7225	4.825	3.55	3.445	3.2275
6	4.0425	2.1075	1.1325	0.645	0.4625	0.44	0.47
7	26.49	13.1975	7.325	4.5625	3.6075	3.4725	3.7225
8	18.2725	9.935	5.7125	3.685	2.83	2.7975	2.64
9	41.44	20.86	11.385	6.0925	4.2875	3.9275	3.3625
10	11.93	6.9875	4.4375	3.405	2.84	2.69	2.8175
11	3.505	1.8275	0.9825	0.545	0.345	0.3375	0.2725
12	9.075	4.62	2.3875	1.2875	0.7725	0.6375	0.525
13	38.28	26.745	23.085	21.3375	22.1775	22.82	19.625
14	4.06	2.1325	1.1	0.6075	0.385	0.36	0.34
15	5.4175	3.32	2.3025	1.7675	1.585	1.51	1.77
16	6.875	4.93	3.965	3.44	3.205	3.175	3.305
17	26.775	13.53	6.8475	3.6225	2.25	1.995	1.41
18	21.925	15.1275	10.9875	9.4975	9.54	9.275	8.5
19	5.915	3.225	1.7525	1.02	0.73	0.7125	0.6475
20	10.0225	5.2925	3.1725	2.1775	1.885	1.8	1.78
21	63.945	34.3125	17.8375	10.3025	7.88	7.275	7.495
22	13.9275	7.0075	4.045	2.675	2.075	2.025	1.8325

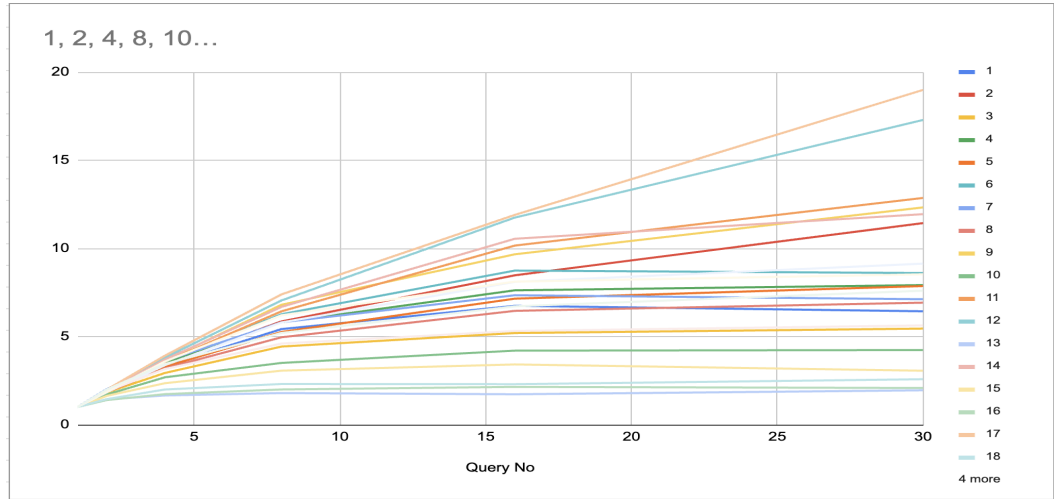


Figure 3.1: *query<sub>r</sub>results* for 1 Node Setup

### 3.1.1.2 Conclusion/Reasoning for 1 Node Setup Results:

The observed variations in scalability can be attributed to the specific characteristics and requirements of each query. Queries 17 and 12 likely possess properties that lend themselves well to parallel execution and efficient resource utilization. In contrast, queries 13, 16, and 18 may have dependencies or bottlenecks that limit their parallelization potential.

### 3.1.2 Results for 3 Node Setup:

In the three-node setup, we evaluated MonetDB's scalability across the same set of queries.



#### **3.1.2.1 Observations:**

Similar to the one-node setup, queries 17 and 12 demonstrated excellent scalability, indicating that their performance improved significantly with increasing cores. Queries 13, 18, and 16 maintained a consistent rate of growth across the core configurations in the three-node setup.

Table 3.2: Query Results

Query No	1 Core	2 Cores	4 Cores	8 Cores	16 Cores	30 Cores
1	393.22	203.21	111.525	67.1825	50.2775	46.265
2	2.4425	1.255	0.6925	0.42	0.3075	0.25
3	57.485	31.0575	19.2625	12.075	10.735	10.6675
4	25.11	14.3475	8.19	5.2725	3.935	3.7025
5	28.77	16.6225	9.8225	6.56	5.26	5.15
6	7.4125	4.1225	2.38	1.46	1.195	1.3325
7	44.7575	25.345	14.405	9.845	7.4225	7.2525
8	230.005	102.175	46.485	29.275	22.12	22.5975
9	19.455	12.4225	8.4875	6.005	4.6125	4.675
10	3.58	1.94	1.015	0.59	0.405	0.335
11	11.1075	5.82	3.08	1.705	1.1275	1.025
12	69.07	45.26	38.66	36.2	34.375	34.6325
13	7.8125	4.68	2.59	1.6075	1.095	1.2025
14	8.67	5.485	3.7725	2.9925	2.57	2.515
15	8.13	6.3025	5.425	4.9325	4.575	4.5075
16	26.465	13.3475	6.8875	3.5925	2.0925	1.435
17	36.215	23.635	24.17	20.675	18.6725	14.9525
18	7.3375	3.98	2.2125	1.2975	0.8375	0.93
19	13.8725	7.4	4.6675	3.075	2.62	2.6775
20	183.998	92.6675	50.515	32.8225	25.2475	28.1675

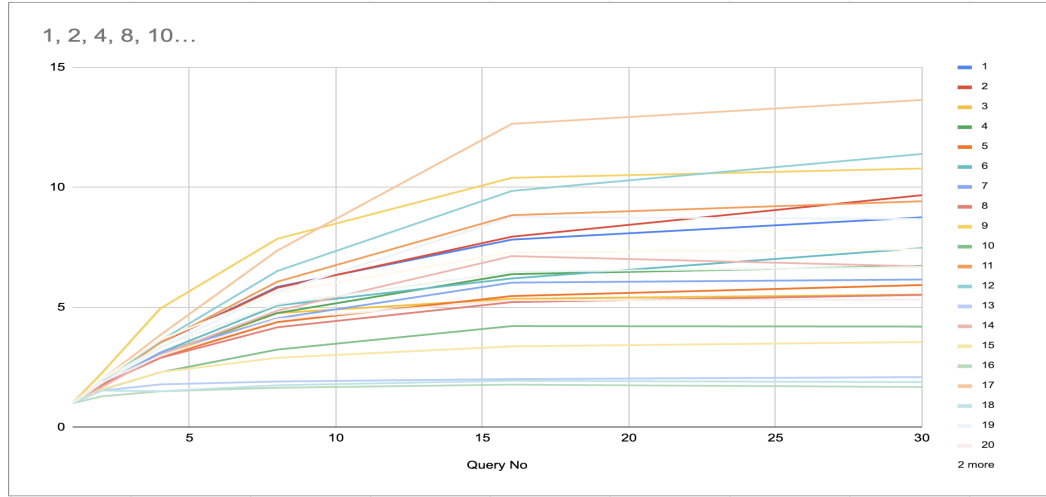


Figure 3.2:  $query_{results}$  for 3 Node Setup

### 3.1.2.2 Conclusion/Reasoning for 3 Node Setup Results:

The similar scalability patterns observed in the three-node setup compared to the one-node setup indicate that MonetDB maintains its scaling capabilities when deployed across multiple nodes. This suggests that MonetDB is a promising candidate for deployment on TidalScale, as it preserves locality and communication characteristics across nodes without sacrificing scalability. However, it is important to note that the absolute performance numbers may decrease in the three-node setup due to increased communication costs.

## 3.2 Results for Gabps:

In this section, we discuss the scalability results for Gabps in two configurations: a one-node setup and a three-node setup. Two experiments were performed: normal execution and execution with the ”*compare<sub>a</sub>nd<sub>s</sub>wap*” code segment removed to increase parallelism.

### 3.2.1 Normal Gabps Results:

For both the one-node and three-node setups, the performance of Gabps was studied using the bc-twitter dataset across 1 core and 30 cores.

#### 3.2.1.1 Observations:

- **Normal Gabps 1 node setup:** Gabps exhibited good scalability on a single node for the bc-twitter dataset. As the number of cores increased from 1 to 30, the execution time reduced significantly from 26 minutes to 3 minutes.

Table 3.3: Normal Gabps 1 node setup

Dataset	1 Core	30 Cores
bc-twitter	26:12.33	2:54.61

- **Normal Gabps 3 node setup:** The performance of Gabps on the three-node setup with the bc-twitter dataset also showed scalability, with the execution time decreasing from 29 minutes to approximately 6 minutes as the number of cores increased from 1 to 30.

Table 3.4: Normal Gabps 3 node setup

Dataset	1 Core	30 Cores
bc-twitter	29:05.36	6:29.76

### 3.2.1.2 Reasoning/Conclusions:

Based on the observations from the one-node setup, we conclude that Gabps is a valid candidate for leveraging scalability benefits. The results from the three-node setup further reinforce Gabps as a suitable candidate for TidalScale, as it maintains scalability even across multiple nodes.

## 3.3 Gabps with "Compare and Swap" Removed:

The second experiment involved removing the synchronized piece of code, "*compare\_and\_swap*," in both the one-node and three-node setups of Gabps. This removal aimed to increase parallelism and avoid code synchronization.

### 3.3.0.1 Observations:

- **Compare and swap removed Gabps 1 node setup:** After removing the "*compare\_and\_swap*" atomic operation in the BFS code of Gabps, a notable improvement in execution time was observed. However, on a single core, the execution time remained the same or worsened, as the code did not fully utilize multiple threads for enhanced parallelism. Nonetheless, with 30 cores, the execution time decreased from 2 minutes and 54.61 seconds to 2 minutes and 40.66

seconds approximately.

Table 3.5: Compare and swap removed Gabps 1 node setup

Dataset	1 Core	30 Cores
bc-twitter	26:16.58	2:40.66

- **Compare and swap removed Gabps 3 node setup:** The removal of the "*compare<sub>a</sub>nd<sub>s</sub>wap*" atomic operation in the BFS code of Gabps resulted in an even better improvement in execution time on the three-node setup. Similar to the one-node setup, the execution time remained the same or worsened on a single core due to the limited utilization of multiple threads. However, with 30 cores, the execution time decreased significantly from 6 minutes and 29.76 seconds to 3 minutes and 9.02 seconds approximately.

Table 3.6: Compare and swap removed Gabps 3 node setup

Node	1 Core	30 Cores
bc-twitter	30:31.75	3:09.02

### 3.3.0.2 Reasoning/Conclusions:

These observations suggest that the horizontal scalability benefits of TidalScale can be leveraged to increase parallelism in codes. Rewriting applications with the design of TidalScale in mind can optimize code for better utilization of parallel resources. In the case of Gabps, the removal of the "*compare<sub>a</sub>nd<sub>s</sub>wap*" atomic operation introduced

potential race condition issues, which were addressed by merging the results after code execution. Although this incurs additional costs, the improvement in execution time was more pronounced with the removal of the atomic operation, particularly when utilizing a higher number of cores and nodes.

## Chapter 4

## Conclusion

In conclusion, this report focused on the evaluation of scalability and performance in the context of MonetDB and GAPBS applications using a variety of experimental setups. Through the experimentation process, we gained valuable insights into the behavior and characteristics of these databases under different conditions.

The motivation behind this project stemmed from the need to assess the scalability and performance of MonetDB and GAPBS in real-world scenarios. By conducting large-scale benchmarks for MonetDB and small-scale benchmarks for GAPBS, we aimed to understand their behavior in different workload scenarios and analyze their scalability potential.

The experiments carried out in the 1-node setup provided a baseline understanding of the databases' performance and scalability in an isolated environment. This setup allowed us to explore the limitations and challenges associated with single-node configurations, including potential scalability problems such as memory constraints, and



limited parallelism.

Expanding the scope, the 3-node setup with a Guest OS on 1 node and 10 cores in each of the nodes provided insights into the behavior of the databases in a distributed system environment. This setup allowed us to investigate scalability, parallelism, communication patterns, and fault tolerance considerations. It enabled us to simulate more realistic workloads and analyze the databases' performance in a distributed computing context.

Throughout the experimentation, we observed the potential scalability problems that can arise in both setups. These problems included communication overhead, synchronization and coordination challenges, load balancing issues, data partitioning and distribution complexities, consistency and replication considerations, and system complexity. Understanding and mitigating these problems are crucial for achieving optimal scalability in distributed systems.

Overall, this report provides a comprehensive evaluation of the scalability and performance characteristics of MonetDB and GAPBS databases in different experimental setups. The findings and insights gained from this project can guide future optimizations, configurations, and decision-making when deploying these databases in real-world scenarios. It highlights the importance of considering scalability and performance factors in database systems to ensure efficient and effective data processing in evolving computing environments.

## Bibliography

- [1] Xingguo Jia, Jin Zhang, Boshi Yu, Xingyue Qian, Zhengwei Qi, and Haibing Guan. Giantvm: A novel distributed hypervisor for resource aggregation with dsm-aware optimizations. *ACM Trans. Archit. Code Optim.*, 19(2), mar 2022.
- [2] Jin Zhang, Zhuocheng Ding, Yubin Chen, Xingguo Jia, Boshi Yu, Zhengwei Qi, and Haibing Guan. Giantvm: A type-ii hypervisor implementing many-to-one virtualization. In *Proceedings of the 16th ACM SIGPLAN/SIGOPS International Conference on Virtual Execution Environments*, VEE '20, page 30–44, New York, NY, USA, 2020. Association for Computing Machinery.

**Certificate Of Completion**

Envelope Id: 8555E3BE015B4CC8A3ED37F47F1CB940

Status: Completed

Subject: Complete with DocuSign: Final\_MS\_Project\_Report.pdf

Source Envelope:

Document Pages: 34

Signatures: 2

Envelope Originator:

Certificate Pages: 2

Initials: 0

Ananya Das

AutoNav: Enabled

1156 High Street

Envelope Stamping: Enabled

Santa Cruz, CA 95064

Time Zone: (UTC-08:00) Pacific Time (US &amp; Canada)

adas13@ucsc.edu

IP Address: 73.202.198.21

**Record Tracking**

Status: Original

Holder: Ananya Das

Location: DocuSign

6/21/2023 1:05:27 PM

adas13@ucsc.edu

**Signer Events**

Andrew Quinn

aquinn1@ucsc.edu

Security Level: Email, Account Authentication (Optional)

**Signature**

DocuSigned by:



223B2CC177124AA...

**Timestamp**

Sent: 6/21/2023 1:07:19 PM

Viewed: 6/21/2023 4:02:23 PM

Signed: 6/21/2023 4:02:43 PM

Signature Adoption: Pre-selected Style

Using IP Address: 71.204.153.91

**Electronic Record and Signature Disclosure:**

Not Offered via DocuSign

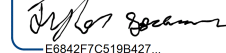
Tyler Sorensen

tysorens@ucsc.edu

University of California, Santa Cruz

Security Level: Email, Account Authentication (Optional)

DocuSigned by:



E6842F7C519B427...

Sent: 6/21/2023 4:02:44 PM

Viewed: 6/21/2023 4:08:29 PM

Signed: 6/21/2023 4:08:32 PM

Signature Adoption: Drawn on Device

Using IP Address: 24.6.88.219

**Electronic Record and Signature Disclosure:**

Not Offered via DocuSign

**In Person Signer Events****Signature****Timestamp****Editor Delivery Events****Status****Timestamp****Agent Delivery Events****Status****Timestamp****Intermediary Delivery Events****Status****Timestamp****Certified Delivery Events****Status****Timestamp****Carbon Copy Events****Status****Timestamp****Witness Events****Signature****Timestamp****Notary Events****Signature****Timestamp****Envelope Summary Events****Status****Timestamps**

Envelope Sent

Hashed/Encrypted

6/21/2023 1:07:19 PM

Certified Delivered

Security Checked

6/21/2023 4:08:29 PM

Signing Complete

Security Checked

6/21/2023 4:08:32 PM

Envelope Summary Events	Status	Timestamps
Completed	Security Checked	6/21/2023 4:08:32 PM
Payment Events	Status	Timestamps