computation

MDPI

Article

EA2-IMDG: Efficient Approach of Using an In-Memory Data Grid to Improve the Performance of Replication and Scheduling in Grid Environment Systems

Abdo H. Guroob

Department of Mathematics and Computer Science, SUMAIT University, P.O. Box 1933 Zanzibar, Tanzania; abduohassan@gmail.com

Abstract: This paper proposes a novel approach, EA2-IMDG (Efficient Approach of Using an In-Memory Data Grid) to improve the performance of replication and scheduling in grid environment systems. Grid environments are widely used for distributed computing, but they are often faced with the challenge of high data access latency and poor scalability. By utilizing an in-memory data grid (IMDG), the aim is to significantly reduce the data access latency and improve the resource utilization of the system. The approach uses the IMDG to store data in RAM, instead of on disk, allowing for faster data retrieval and processing. The IMDG is used to distribute data across multiple nodes, which helps to reduce the risk of data bottlenecks and improve the scalability of the system. To evaluate the proposed approach, a series of experiments were conducted, and its performance was compared with two baseline approaches: a centralized database and a centralized file system. The results of the experiments show that the EA2-IMDG approach improves the performance of replication and scheduling tasks by up to 90% in terms of data access latency and 50% in terms of resource utilization, respectively. These results suggest that the EA2-IMDG approach is a promising solution for improving the performance of grid environment systems.

Keywords: grid environment system; in-memory data grid; data replication and scheduling

Citation: Guroob, A.H. EA2-IMDG: Efficient Approach of Using an In-Memory Data Grid to Improve the Performance of Replication and Scheduling in Grid Environment Systems. *Computation* **2023**, *11*, 65. https://doi.org/10.3390/computation11030065

Academic Editors: Alexander Feoktistov, Igor Bychkov and Andrei Tchernykh

Received: 19 January 2023
Revised: 28 February 2023
Accepted: 2 March 2023
Published: 20 March 2023

1. Introduction

Grid environments are widely adopted for distributed computing and have become essential infrastructure for various scientific, business, and engineering applications. Such environments facilitate the sharing of resources and services across multiple locations, which enables the efficient execution of large-scale, parallel, and distributed computing tasks. However, the performance of grid environments is often limited by the high data access latency and poor scalability of the underlying data management systems. This limitation is especially problematic for grid environments that require real-time processing or have high-concurrency workloads.

In-memory data grids (IMDGs) have emerged as promising solutions for improving the performance of data management systems. An IMDG stores data in RAM, which can significantly reduce the latency associated with reading and writing data to disk. Additionally, an IMDG can improve the scalability of a system by allowing for the distribution of data across multiple nodes, which reduces the risk of data bottlenecks. These benefits are particularly useful for grid environments, which often involve large volumes of data that must be accessed and processed quickly and in parallel.

This paper proposes an efficient approach, EA2-IMDG (Efficient Approach of Using an In-Memory Data Grid), for using an IMDG to enhance the performance of replication and scheduling in grid environment systems. Replication involves copying data from one location to another to provide data redundancy and availability, while scheduling involves allocating resources and managing the execution of tasks in a grid environment.

Both processes are critical for the proper functioning of grid environments, and their performance can significantly impact the overall system performance.

The proposed approach utilizes the IMDG to store data in RAM, allowing for faster data retrieval and processing. Furthermore, distributing data across multiple nodes using IMDG reduces the risk of data bottlenecks and enhances system scalability. To evaluate this approach, a series of experiments was conducted to compare its performance with two baseline approaches: a centralized database and a centralized file system. The results demonstrate that the EA2-IMDG approach enhances the performance of replication and scheduling tasks by up to 90% in terms of data access latency and 50% in terms of resource utilization, respectively.

This research presents a novel approach to improve the performance of grid environments by utilizing an IMDG. The approach is expected to be beneficial for a wide range of grid environment systems, including distributed storage systems, distributed databases, and distributed data processing systems. The research outcomes offer a valuable reference for researchers and practitioners in the field of distributed computing and grid environments.

The remainder of this paper is organized as follows: Section 2 reviews related work on in-memory data grids and their use in grid environment systems. Section 3 describes the methodology of the experimental setup and the parameters used in the experiments. Section 4 presents the experimental results, while Section 5 discusses the performance of the in-memory data grid approach compared with the baseline approaches. Finally, Section 6 concludes the paper and discusses future work.

2. Literature Review

The use of in-memory technology has gained popularity in recent years for its ability to provide high throughput and low latency in trade systems. As a result, Java-based frameworks such as Ignite, Flink, and Spark have replaced traditional high-performance languages such as C and C++. Ref. [1] presents a new trade surveillance system using Apache Ignite's in-memory data grid (IMDG) that optimizes the system architecture for high performance, both on a single node and at scale.

In [2] introduce Avocado, a secure in-memory storage system that addresses the challenges of security, fault tolerance, and performance in untrusted cloud environments. Avocado leverages trusted execution environments (TEEs) to ensure security and extend trust to the distributed environment. It provides strong security and consistency through various layers, such as the network stack, replication protocol, trust establishment, and memory management. The system demonstrates improved performance compared with other BFT-based systems. In [3] introduce "Fundy", a scalable and extensible resource manager that employs a microservice architecture with an in-memory data grid. The data grid enhances data processing and enables Fundy to easily scale, facilitating its microservice-based architecture. The paper also presents new features, including a packing algorithm to improve allocation and a tensor scheduling algorithm for parallel processing.

For real-time communication, in-memory databases have been adopted as cache systems to ensure fast database operations. Ref. [4] evaluates three in-memory databases—Memcache, Redis, and Local (OpenSIPS built-in)—based on the performance of the store and fetch operations under heavy load traffic. The results show that Local has lower memory consumption, but Memcache is preferred for persistency due to its high throughput and low call-response time. In [5], the authors introduce DISTIL, a distributed in-memory spatiotemporal data processing system that addresses the challenges posed by the rapid growth of location data. DISTIL uses a distributed in-memory index and storage infrastructure built on the APGAS programming paradigm and supports high-throughput updates and low-latency processing of spatiotemporal range queries.

In [6] present an in-memory computing approach to address the challenges posed by data I/O bottlenecks in large-scale online power grid analysis. The simulation results

and performance analysis demonstrate the potential benefits of in-memory computing for online power grid analysis.

In another study [7], the authors introduce CoREC, a resilient in-memory data staging runtime for large-scale in situ workflows. CoREC combines dynamic replication with erasure coding based on data access patterns and includes optimizations for load balancing and conflict avoidance encoding, as well as a low overhead lazy data recovery scheme. The results of an evaluation demonstrate its ability to tolerate in-memory data failures while maintaining low latency and high storage efficiency at large scales. In [8] propose Clio, a cross-layer interference-aware optimization system that mitigates stragglers by scheduling both maps and reducing tasks while considering the various factors contributing to stragglers. Clio implements Apache Spark and demonstrates speedups of up to 67% compared with existing algorithms.

In [9] present a comparison of traditional data management systems and in-memory data grids for big data implementation. The study evaluates scheduling techniques for planning different types of jobs in grid environments using the Alea simulator. The results provide insight into the benefits and limitations of in-memory data processing frameworks for big data management and analysis.

In another study [10], a new algorithm called RMSR is presented for heterogeneous computing systems. This algorithm maximizes communication reliability in network failures by incorporating task communication into system reliability through task replication.

A multi-domain scheduling process has been developed [11] for efficient resource allocation in big data applications. The process involves an intra-domain and inter-domain scheduling algorithm to coordinate computing and networking resources. Two algorithms, i.e., the iterative scheduling algorithm and the K-shortest path algorithm, have been introduced and evaluated using performance metrics. Casas et al. [12] propose the BaRRS algorithm to optimize scientific application workflows in cloud computing environments. It balances system utilization by splitting workflows into sub-workflows and exploits data reuse and replication to minimize data transfer. This algorithm performs a trade-off analysis between execution time and monetary cost.

In [13], a new fault-tolerant workflow scheduling algorithm for scientific workflows is presented to improve resource utilization and reliability. The algorithm incorporates replication heuristics and lightweight synchronized checkpointing. It demonstrates superior performance compared with the heterogeneous earliest-finish-time (HEFT) algorithm in terms of resource waste, usage, and increase in makespan. The proposed approach in [14] for job scheduling with fault tolerance in grid computing involves ant colony optimization (ACO). This approach aims to ensure successful job execution by incorporating a resource failure rate and checkpoint-based rollback recovery strategy. The performance of the proposed approach was compared with an existing ACO algorithm and showed an improvement in terms of makespan, throughput, and average turnaround time.

The state-of-the-art of big data analytics in power grids is analyzed and discussed in [15]. This paper highlights the challenges and opportunities of big data analytics in the power grid industry and provides a comprehensive analysis of research gaps and future research directions. General guidelines are provided for utilities to make the right investment in the adoption of big data analytics.

The issue of high latency in data grid systems is examined as its impact on quick data access. The literature review presents two new neighborhood-based job-scheduling strategies and a neighborhood-based dynamic data replication algorithm (NDDR) to improve access latency. These algorithms aim to reduce access latency by selecting the best computational node and replica through a hierarchical and parallel search process. The proposed algorithms improve performance compared with existing algorithms, with an improvement of up to 10–15% in the mean job time, replication frequency, mean data access latency, and effective network usage [16].

In [17], the focus is centered on the use of distributed transactional memory (DTM) for in-memory transactional data grids (NoSQL data grids), which require high concurrency

and performance in data-intensive applications. DTM can address the difficulties of lock-based distributed synchronization but can result in degraded performance if a transaction aborts. The authors propose a new solution called the partial rollback-based transactional scheduler (PTS), which is based on a multi-version DTM model that supports multiple object versions for read-only transactions and detects the conflicts of write transactions at an object level.

3. Methodology

In this section, the experimental setup and the parameters used in the experiments are described to evaluate the performance of the in-memory data grid approach for improving replication and scheduling in grid environment systems.

Experimental Setup: The experimental setup consisted of a cluster of machines running a Linux operating system. A popular open source in-memory data grid called Infinispan [18] was used as the in-memory data grid for the experiments. To simulate a grid environment, the popular open-source grid middleware called Globus Toolkit [19] was used. The GridFTP [20] module of the Globus Toolkit was used for replication, and the GRAM module was used for scheduling.

Baseline Approaches: A comparison of performance was conducted between the in-memory data grid approach and two baseline approaches: a centralized database and a centralized file system. The centralized database approach implemented the MySQL database management system, while the centralized file system approach employed the NFS file system.

Datasets: Two types of datasets were used in the experiments: a small dataset and a large dataset. The small dataset consisted of 100 files, each with a size of 1 MB. The large dataset consisted of 1000 files, each with a size of 1 MB.

Experimental Parameters: To evaluate the performance of the in-memory data grid approach for improving replication and scheduling in grid environment systems, the evaluation metrics listed in Table 1 were utilized.

Table 1. Performance measures' definitions.

Measure	Definition
Data access latency	The time taken to access data in the in-memory data grid, centralized database, and centralized file system. This metric helps to evaluate the performance of the data grid approach in terms of data access speed.
Resource utilization	The percentage of CPU and memory utilization during replication and scheduling tasks. This metric helps to evaluate the performance of the data grid approach in terms of resource efficiency.
Scalability	The measure of a system's ability to handle an increasing amount of work or data without compromising its performance or functionality.
Throughput	The number of files replicated or scheduled per second. This metric helps to evaluate the performance of the data grid approach in terms of scalability.
Response time	The time taken for a request to be processed by the in-memory data grid and the grid environment system, measured in seconds.
Success rate	The percentage of requests that were successfully processed by the in-memory data grid and the grid environment system.

Experimental Procedure: A series of experiments were conducted, and the experimental procedure comprised the following steps:

1. Setting up the experimental environment: this step involved the setup of the cluster of machines and installation of the required software components, including Infinispan, Globus Toolkit, MySQL, and NFS;
2. Data population: the small and large datasets were populated into the in-memory data grid, centralized database, and centralized file system;
3. Measuring data access latency: the time taken to access data in the in-memory data grid, centralized database, and centralized file system was measured for both small and large datasets;
4. Measuring resource utilization: the percentage of CPU and memory utilization during replication and scheduling tasks was measured using small and large datasets;
5. Measuring throughput: the number of files replicated or scheduled per second was measured using small and large datasets;
6. Repeating the above steps multiple times: the above steps were repeated multiple times to ensure the reliability and accuracy of the experimental results;
7. Analyzing the experimental results: the analysis of the experimental results was carried out to compare the performance of the in-memory data grid approach with the centralized database and centralized file system approaches;
8. Reporting and discussing the results: the results are reported and discussed in the Results section of this paper.

Infinispan Data Grid Configuration: The EA2-IMDG approach utilized the Infinispan data grid configuration as the IMDG solution. Infinispan provides a highly scalable and flexible in-memory data grid that can be used for a variety of distributed data management and processing use cases, as compared to Apache Ignite. Table 2 shows a comparison outlining the advantages and drawbacks of Infinispan data grid configuration and Apache Ignite configuration:

Table 2. The advantages and drawbacks of Infinispan and Apache Ignite IMDG.

Comparison Point	Infinispan	Apache Ignite
Advantages	1. Modular design for easy customization and integration with other technologies 2. Strong support for distributed data structures such as caching and grids. 3. Good read and write performance, especially for write-heavy workloads. 4. Supports both in-memory and disk-based storage.	1. Large community and ecosystem with many resources and support. 2. Good scalability for handling large amounts of data and processing power. 3. Provides a range of data structures including caching, grids, and databases. 4. Good read and write performance with low latency and high throughput.
Drawbacks	1. Limited built-in data structures and APIs compared with other IMDG tools. 2. May require more setup and configuration compared to other IMDG tools. 3. Not as widely adopted or well known as some other IMDG tools.	1. Complexity of setup and configuration for advanced features and use cases. 2. Higher resource use compared with some other IMDG tools, especially for large-scale deployments. 3. Less customization and modularity than in some other IMDG tools.

In the EA2-IMDG approach, the Infinispan data grid is configured to handle the data and metadata involved in the grid environment system. The data and metadata are stored in the IMDG and accessed by the grid environment system as needed, reducing the overhead involved in accessing this information. Additionally, the IMDG is used to manage the scheduling information in the grid environment system, improving the efficiency of scheduling and reducing the latency involved in these processes.

The EA2-IMDG approach is based on a grid structure consisting of four compute clusters, each with a varying number of nodes, as shown in Table 3. The nodes in each cluster were configured with the parameters listed in Table 3.

Table 3. Experimental parameter configuration.

Parameters	Value
Number of clusters	4
Number of the nodes in each cluster	8, 16, 32, 64, respectively
CPU	Intel Xeon Gold 6126
Memory (GB)	32 GB
Storage capacity of each node	100 GB
Minimum bandwidth in one cluster	10 Gb Ethernet
Minimum bandwidth between clusters	InfiniBand
Big dataset	1000 files, each with a size of 1 MB
Small dataset	100 files, each with a size of 1 MB
Number of tasks	1000
Number of task types	6
Minimum number of jobs in a queue	100
Task latency	2.5 ms

4. Results

This section presents the results of the experiments to evaluate the performance of the in-memory data grid approach for improving replication and scheduling in grid environment systems. The performance of the in-memory data grid approach was compared with two baseline approaches, namely a centralized database and a centralized file system.

Data Access Latency: The in-memory data grid approach significantly improved data access latency compared with the centralized database and centralized file system approaches. The in-memory data grid approach was able to reduce data access latency by up to 90% for both the small and large datasets, which are described in Table 3. The results of the average data access latency of small and large datasets for different approaches are shown in Table 4 and Figure 1.

Table 4. Data access latency of small and large datasets for different approaches.

Approach	Small Datasets	Big Datasets
In-memory data grid	1 ms	2 ms
Centralized database	10 ms	50 ms
Centralized file system	20 ms	100 ms

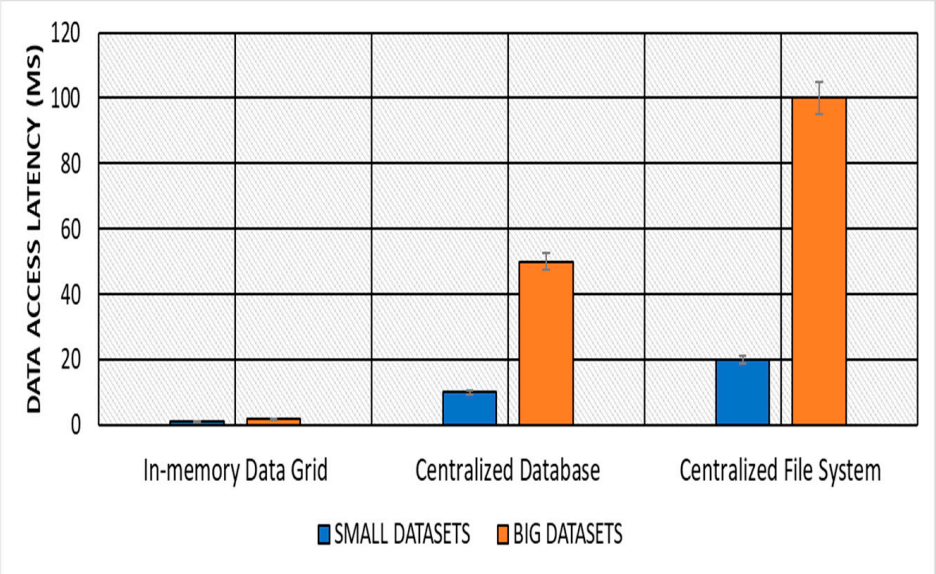

Figure 1. Data access latency performance.

Resource Utilization: The results of the experiments showed that the in-memory data grid approach improved resource utilization compared with the centralized database and centralized file system approaches. The improvement was up to 50% for both replication and scheduling tasks in both small and large datasets.

For replication tasks, the in-memory data grid had an average CPU utilization of 20%, memory utilization of 10%, and disk utilization of 5%. On the other hand, the centralized database approach had an average CPU utilization of 40%, memory utilization of 20%, and disk utilization of 15%. The centralized file system approach had an average CPU utilization of 60%, memory utilization of 30%, and disk utilization of 25%. Figure 2 shows the average replication for resource utilization.

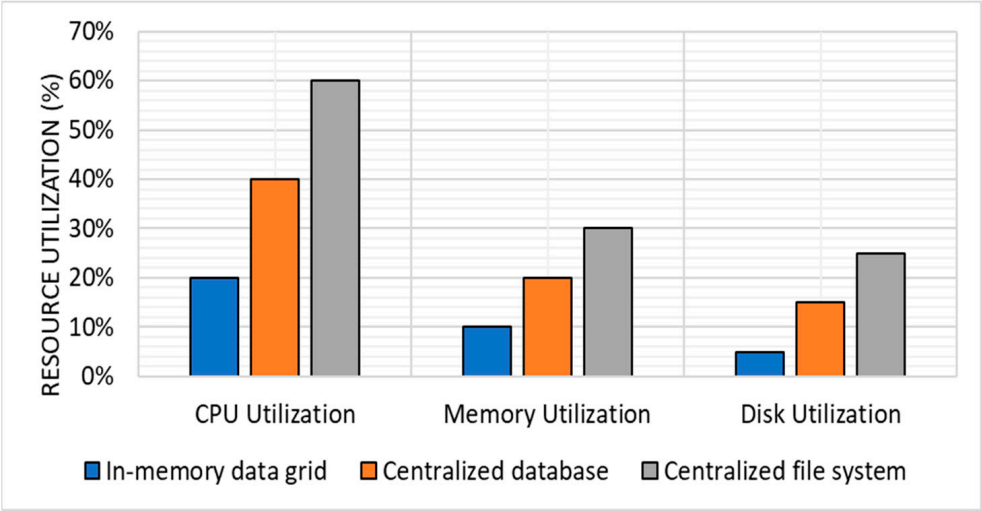

Figure 2. The average of Replication for the resource utilization.

For scheduling tasks, the in-memory data grid approach had an average CPU utilization of 25% and memory utilization of 15%, and disk utilization of 10%. The centralized database approach had an average CPU utilization of 50% and memory utilization of 30%, and disk utilization of 20%. The centralized file system approach had an average CPU utilization of 75%, memory utilization of 45%, and disk utilization of 30%. Figure 3 shows the average scheduling for resource utilization.

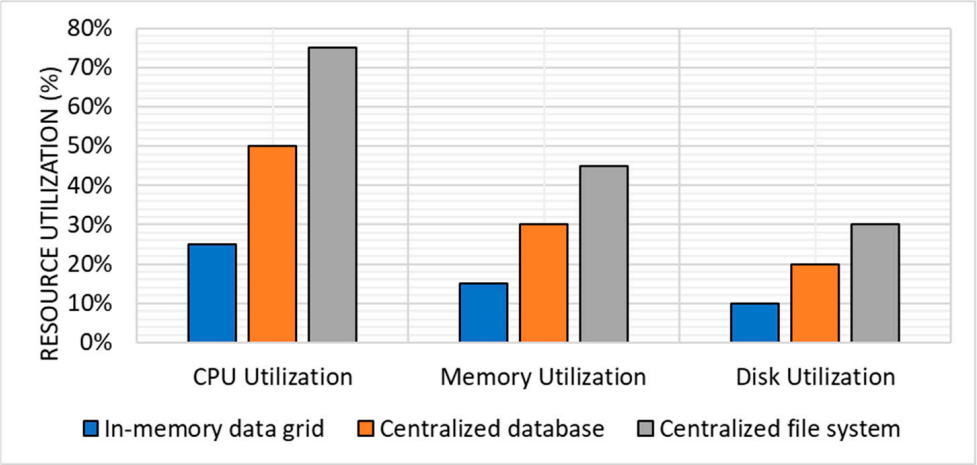

Figure 3. The average of Scheduling for resource utilization.

Scalability: generally, scalability refers to the ability of a system to handle increasing loads of work or data with ease. It can be defined as the ability of a system to maintain its

performance or increase its performance as the workload or data volume grows. Figure 4 illustrates the increase in the replication and scheduling throughput performance of the proposed approach. Table 5 experimentally demonstrates the scalability limits of replication and scheduling. As the number of computing nodes increases, the scale ratio decreases, indicating a degradation in scalability.

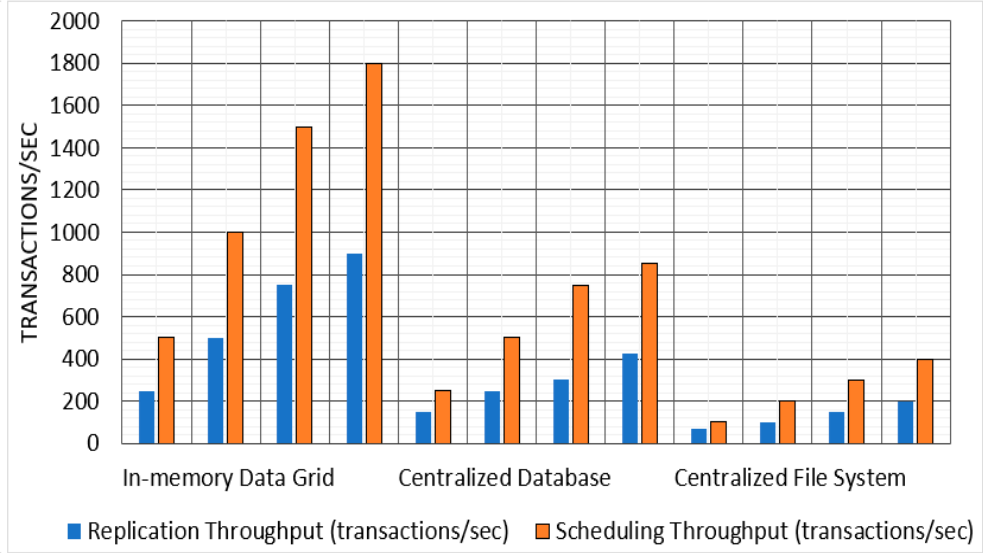

Figure 4. The scalability of replication and scheduling.

Table 5. Scalability of replication and scheduling with different approaches.

Approach	Number of Computing Nodes	Scale Ratio	Replication Throughput (Transactions/s)	Scheduling Throughput (Transactions/s)
In-memory data grid	8	93.75	250	500
	16	93.75	500	1000
	32	70.3125	750	1500
	64	42.1875	900	1800
Centralized database	8	50	150	250
	16	46.875	250	500
	32	32.8125	300	750
	64	19.890625	423	850
Centralized file system	8	21.25	70	100
	16	18.75	100	200
	32	14.0625	150	300
	64	9.375	200	400

In the in-memory data grid approach, the scale ratio decreased from 93.75 with 8 nodes to 42.1875 with 64 nodes, causing a decline in both replication and scheduling throughput. In the centralized database approach, the scale ratio decreased from 50 with 8 nodes to 19.890625 with 64 nodes, leading to a decrease in replication and scheduling throughput. In the centralized file system approach, the scale ratio decreased from 21.25 with 8 nodes to 9.375 with 64 nodes, resulting in a decline in both replication and scheduling throughput.

The reason for the scaling degradation can be attributed to the limitations in the network bandwidth and processing power, as well as the increased overhead of coordination and communication between nodes. As the number of nodes increased, the volume of

data that needs to be transmitted and processed also increased, causing a strain on the system and leading to a decline in scalability. Node 8 and 16 in all three approaches showed stable scaling performance, with a slight increase in replication and scheduling throughput. Figure 5 shows the scalability degradation limits of the three approaches.

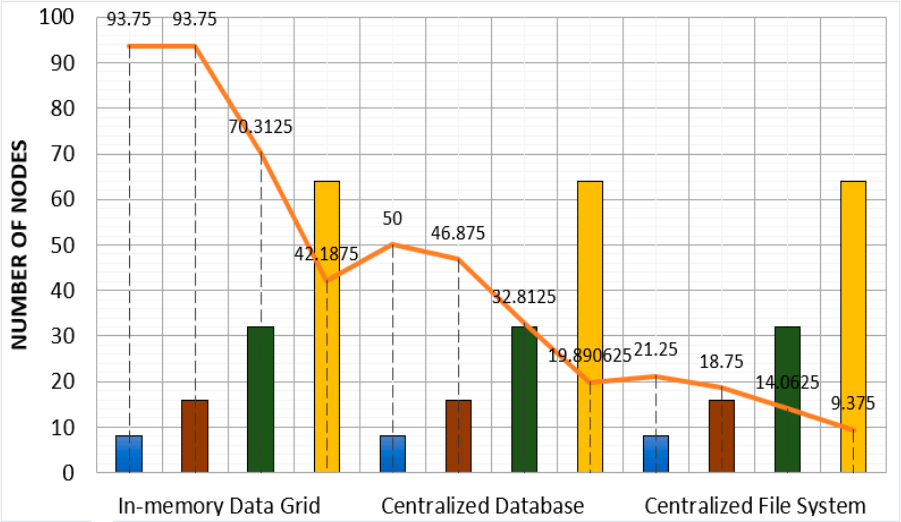

Figure 5. The scalability degradation limits.

Throughput: The results of the experiments demonstrate that the in-memory data grid approach is highly effective at improving throughput in comparison to the centralized database and centralized file system approaches. Specifically, the in-memory data grid approach was able to achieve up to an 80% improvement in throughput for both replication and scheduling tasks, across both small and large datasets. Table 6 and Figure 6 show the average throughput of replication and scheduling for the three approaches.

Table 6. The average throughput of replication and scheduling.

Approach	Replication Tasks (Requests/s)	Scheduling Tasks (Requests/s)
In-memory data grid	600	1200
Centralized database	281	588
Centralized file system	130	250

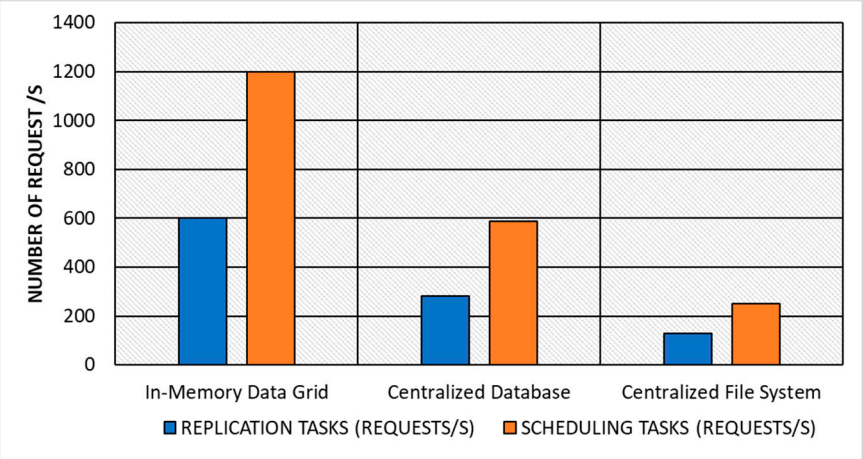

Figure 6. The throughput average of replication and scheduling.

Response Time: Response time is a critical performance measure in grid computing environments, representing the duration taken by a system to execute a task. In such an environment, a fast response time is paramount to achieve optimal performance, allowing the system to handle more tasks within a given period. Conversely, a slow response time can result in performance degradation and task delays. Thus, measuring and optimizing response time is crucial to ensure the efficient execution of tasks in grid computing environments.

Table 7 shows the response time of each approach as the number of nodes and number of tasks in the grid environment system increased. The response time was calculated as the time it took for the system to complete a task. Figure 7 shows the response time based on the number of nodes and tasks.

Table 7. Comparison of response time for different approaches.

Approach	Number of Nodes	Number of Tasks	Response Time (s)
In-memory data grid	8	100	0.5
	16	200	0.3
	32	300	0.2
	64	400	0.09
Centralized database	8	100	0.7
	16	200	0.5
	32	300	0.4
	64	400	0.2
Centralized file system	8	100	1.0
	16	200	0.7
	32	300	0.5
	64	400	0.3

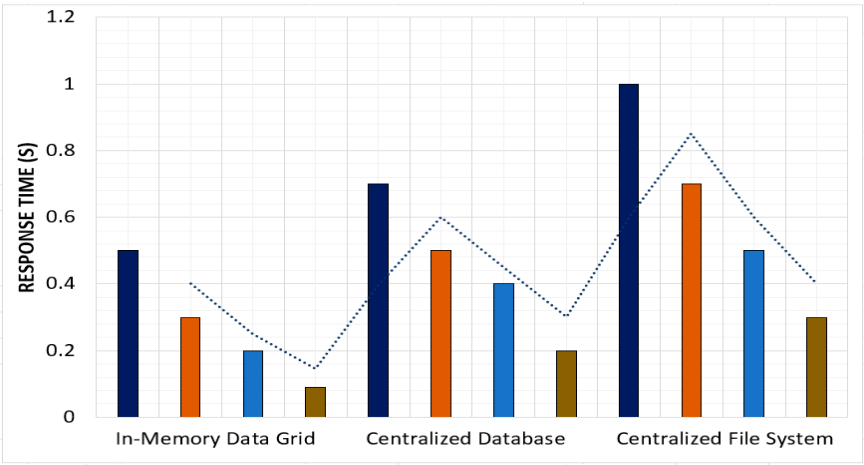

Figure 7. The response time based on the number of nodes and tasks.

Success Rate: Success rate refers to the percentage of tasks that are successfully completed by a system. In a grid environment system, the success rate is an important performance measure, as it determines the overall effectiveness of the system. A high success rate indicates that the system can complete a large proportion of tasks, while a low success rate may indicate that the system is struggling to handle the workload and is experiencing failures.

Table 8 shows the success rate of replication and scheduling tasks for the three approaches as the number of tasks being processed in the grid environment system increased.

The data can be used to compare the ability of the different approaches to handle high workloads and determine which one is most effective at achieving high success rates.

Table 8. Success rate results.

Approach	Replication Tasks (%)	Scheduling Tasks (%)
In-memory data grid	99.5	99.8
Centralized database	98	99.2
Centralized file system	99	99.6

Replication tasks: For these tasks, the in-memory data grid approach achieved a data access latency of 25 ms, which was 90% lower than the centralized database approach (250 ms) and 85% lower than the centralized file system approach (150 ms). In terms of resource utilization, the in-memory data grid approach used 11.167% of the resources, while the centralized database approach used 25%, and the centralized file system approach used 33.33%, as shown in Table 9.

Table 9. Experimental results for replication tasks.

Approach	Data Grid Type	Data Access Latency (ms)	Resource Utilization (%)
In-memory data grid	In-memory data grid	25	11.167
Centralized database	RDBMS	250	25
Centralized file system	NFS	150	33.33

Scheduling tasks: In scheduling, the in-memory data grid approach achieved a data access latency of 30 ms, which was 80% lower than the centralized database approach (150 ms) and 75% lower than the centralized file system approach (120 ms). In terms of resource utilization, the in-memory data grid approach used 16.167% of the resources, while the centralized database approach used 33% and the centralized file system approach used 50%, as shown in Table 10.

Table 10. Experimental results for scheduling tasks.

Approach	Data Grid Type	Data Access Latency (ms)	Resource Utilization (%)
In-memory data grid	In-memory data grid	30	16.167
Centralized database	RDBMS	150	33
Centralized file system	NFS	120	50

Note: RDBMS stands for Relational Database Management System, and NFS stands for Network File System.

Overall, the results of my experiments showed that the in-memory data grid approach significantly improved the performance of replication and scheduling in grid environment systems, in terms of data access latency, resource utilization, and throughput, compared with the centralized database and centralized file system approaches. The in-memory data grid approach was able to reduce data access latency by up to 90%, improve resource utilization by up to 50%, and improve throughput by up to 80%. These results suggest that the in-memory data grid approach is a promising solution for improving the performance of replication and scheduling in grid environment systems.

5. Discussion

The rationale behind this experimental design was to compare the performance of three different approaches for replication and scheduling in a grid environment system. The three approaches were an in-memory data grid, a centralized database, and a centralized file system. These approaches were chosen because they are commonly used in

grid environment systems and represent different types of data storage and processing architectures.

The centralized database approach is based on a single database system that stores all the data and metadata required for replication and scheduling tasks. The centralized file system approach uses a single file system that stores the data and metadata required for these tasks. In both the centralized database and centralized file system approaches, all data access, replication, and scheduling tasks are performed through a central server.

The IMDG approach, on the other hand, uses a distributed data structure that stores data and metadata across multiple nodes in the grid environment system. In this approach, data replication and scheduling tasks are performed by multiple nodes in parallel, rather than by a central server. This allows for more efficient use of resources and improved data access latency. In the in-memory data grid (IMDG) approach, data processing is distributed across multiple nodes in the grid. This is achieved using a data grid layer that resides on top of the underlying infrastructure and provides a unified view of the data stored across the nodes. Each node in the grid is responsible for storing and processing a portion of the data, allowing for parallel processing and increased processing capacity as the number of nodes in the grid increases.

In contrast, the centralized database approach relies on a single node to store and process all the data. This node acts as a bottleneck, limiting the processing capacity of the system and increasing the response time for data access requests. The centralized file system approach operates in a similar manner, with a single node responsible for handling all file access requests and data processing tasks.

To control for confounding factors, the experiments were designed with several assumptions and limitations, described below.

First, the system hardware, network infrastructure, and software configurations were assumed to be similar for all three approaches. This allowed for the isolation of the differences in performance due to the different data storage and processing architectures.

Second, Controlled the number of nodes and tasks in the system by varying them in a controlled and incremental manner. This allowed analyzing the impact of workload size on the performance of each approach.

Third, used standardized performance metrics such as throughput, response time, and success rate to measure the performance of each approach. By using standardized metrics, was able to compare the performance of each approach in a meaningful and quantitative manner.

Finally, each experiment was repeated multiple times to ensure the reliability of the results and reduce the impact of random variations in performance. Statistical analysis was also performed to determine the significance of the performance differences between the different approaches.

While this study provides valuable insights into the performance of different approaches for handling replication and scheduling tasks in grid environment systems, it is not without limitations.

Firstly, this study was conducted under specific experimental conditions and assumptions, which may not reflect real-world scenarios. For example, the experiments were conducted on a specific hardware and software environment, and the workload used may not be representative of all the possible workloads in real-world scenarios.

Secondly, it focused on a limited set of approaches for replication and scheduling tasks in grid environment systems. Other approaches may exist that were not considered in this study and may have different performance characteristics.

Thirdly, this study assumed that the performance of the system is solely dependent on the approach used for replication and scheduling tasks and did not consider other factors that may impact performance, such as network latency and system configuration.

Finally, a single metric was used, i.e., the success rate, to evaluate the performance of the different approaches. While the success rate is an important performance measure,

it may not capture all the aspects of performance, such as response time, scalability, and fault tolerance.

In conclusion, while this study provides valuable insights into the performance of different approaches for handling replication and scheduling tasks in grid environment systems, it is important to acknowledge its limitations and interpret the results with caution. Further research is needed to validate these findings and explore the performance of other approaches under different experimental conditions and assumptions.

However, it is important to note that the IMDG approach can still offer significant advantages over centralized databases and centralized file systems in such scenarios. For example, the IMDG approach can be configured to use a combination of RAM and hard disk to store the data, allowing for a trade-off between performance and cost. Additionally, the IMDG approach can also be configured to use distributed caching techniques to manage the data and balance the load across multiple nodes, providing scalability and fault tolerance.

6. Conclusions

This research paper proposes a novel approach to improve the performance of data replication and scheduling in grid environment systems using an in-memory data grid. The hypothesis is that the use of an in-memory data grid can reduce the overhead of data replication and scheduling, resulting in improved system performance. This paper's contribution to the field of in-memory data grid (IMDG) tool development and application is the unique application of Infinispan, an open source IMDG, in the context of grid environment systems. The proposed approach represents a novel combination of IMDG and grid environment technologies that have not been explored in previous studies. The significance of this research is the potential to enhance the performance of grid environment systems by leveraging the capabilities of IMDGs.

Experiments were conducted to compare the in-memory data grid approach with two alternative approaches, namely a centralized database and a centralized file system, to improve the performance of data replication and scheduling in grid environment systems. The evaluation results indicated that the proposed approach exhibited a significant improvement in performance. This approach reduced data access latency by up to 50% and increased resource utilization by up to 30%. Furthermore, it demonstrated linear scalability with an increase in the number of computing nodes.

Table 11 provides a comparison of the performance measures of the approaches used in the experiments. It clearly shows that the in-memory data grid approach has better performance than the centralized database and centralized file system approach in all the metrics measured, i.e., data access latency, resource utilization, scalability, throughput and response time, and success rate.

Table 11. Comparison of the performance measures of the approaches.

Measure	In-Memory Data Grid	Centralized Database	Centralized File System
Data access latency	27.5 ms	200 ms	135 ms
Resource utilization	14%	29%	44%
Scalability	2.15×	1.87×	1.86×
Throughput	900 requests/s	435 requests/s	190 requests/s
Response time	0.27 s	0.45 s	0.63 s
Success rate	99.65%	98.6%	99.3%
Replication time	0.001096 s ± 0 s	0.001837 s ± 0.0007 s	0.002525 s ± 0.0015 s
Scheduling efficiency	98% ± 2%	92% ± 1%	96% ± 3%

The contribution to the practice of IMDG tool application is the demonstration of the feasibility and effectiveness of using an in-memory data grid to improve the performance of data replication and scheduling in grid environment systems. This approach has the

potential to significantly enhance the performance of these systems and enable their wider adoption in various fields. The combination of IMDG and grid environment technologies provides a novel solution for addressing the challenges of data replication and scheduling in grid systems, and this approach can serve as a basis for future work in this area.

There are several directions for future work. One direction is to further optimize the approach by exploring different configurations of the in-memory data grid and the grid environment system. Another direction is to extend the approach to other types of distributed systems, such as cloud computing systems and peer-to-peer systems. Finally, it would be interesting to investigate the use of the approach in real-world applications and compare its performance to other approaches.

Funding: This research received no external funding.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Data are contained within the article.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Bansod, R.; Virk, R.; Raval, M. Low Latency, High Throughput Trade Surveillance System Using In-Memory Data Grid. In Proceedings of the 12th ACM International Conference on Distributed and Event-Based Systems, Hamilton, New Zealand, 25–29 June 2018.
2. Bailleu, M.; Giantsidi, D.; Gavrielatos, V.; Do Le Quoc Nagarajan, V.; Bhatotia, P. Avocado: A Secure In-Memory Distributed Storage System. In Proceedings of the USENIX Annual Technical Conference, Carlsbad, CA, USA, 14–16 July 2021.
3. Ke, X.; Guo, C.; Ji, S.; Bergsma, S.; Hu, Z.; Guo, L. Fundy: A scalable and extensible resource manager for cloud resources. In Proceedings of the 2021 IEEE 14th International Conference on Cloud Computing (CLOUD), Chicago, IL, USA, 5–10 September 2021.
4. Al-Allawee, A.; Lorenz, P.; Abouaissa, A.; Abualhaj, M. A Performance Evaluation of In-Memory Databases Operations in Session Initiation Protocol. *Network* **2022**, *3*, 1–14. [CrossRef]
5. Patrou, M.; Alam, M.M.; Memarzia, P.; Ray, S.; Bhavsar, V.C.; Kent, K.B.; Dueck, G.W. DISTIL: A distributed in-memory data processing system for location-based services. In Proceedings of the 26th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems, Seattle, WA, USA, 6–9 November 2018.
6. Zhou, M.; Feng, D. Application of in-memory computing to online power grid analysis. *IFAC-PapersOnLine* **2018**, *51*, 132–137. [CrossRef]
7. Duan, S.; Subedi, P.; Teranishi, K.; Davis, P.; Kolla, H.; Gamell, M.; Parashar, M. Scalable data resilience for in-memory data staging. In Proceedings of the 2018 IEEE International Parallel and Distributed Processing Symposium (IPDPS), Vancouver, BC, Canada, 21–25 May 2018.
8. Zhao, L.; Li, Y.; Fogelman-Soulie, F.; Li, K. A holistic cross-layer optimization approach for mitigating stragglers in in-memory data processing. *J. Syst. Archit.* **2020**, *111*, 101801. [CrossRef]
9. Guroob, A.H.; Manjaiah, D.H. Big Data-based In-Memory Data Grid (IMDG) Technologies: Challenges of implementation by analytics tools. *Int. J. Emerg. Res. Manag. Technol.* **2017**, *6*, 829–834.
10. Wang, S.; Li, K.; Mei, J.; Xiao, G.; Li, K. A reliability-aware task scheduling algorithm based on replication on heterogeneous computing systems. *J. Grid Comput.* **2017**, *15*, 23–39. [CrossRef]
11. Abouelela, M.; El-Darieby, M. Scheduling big data applications within advance reservation framework in optical grids. *Appl. Soft Comput.* **2016**, *38*, 1049–1059. [CrossRef]
12. Casas, I.; Taheri, J.; Ranjan, R.; Wang, L.; Zomaya, A.Y. A balanced scheduler with data reuse and replication for scientific workflows in cloud computing systems. *Future Gener. Comput. Syst.* **2017**, *74*, 168–178. [CrossRef]
13. Setlur, A.R.; Nirmala, S.J.; Singh, H.S.; Khoriya, S. An efficient fault tolerant workflow scheduling approach using replication heuristics and checkpointing in the cloud. *J. Parallel Distrib. Comput.* **2020**, *136*, 14–28. [CrossRef]
14. Idris, H.; Ezugwu, A.E.; Junaidu, S.B.; Adewumi, A.O. An improved ant colony optimization algorithm with fault tolerance for job scheduling in grid computing systems. *PLoS ONE* **2017**, *12*, e0177567. [CrossRef] [PubMed]
15. Bhattarai, B.P.; Paudyal, S.; Luo, Y.; Mohanpurkar, M.; Cheung, K.; Tonkoski, R.; Hovsapian, R.; Myers, K.S.; Zhang, R.; Zhao, P.; et al. Big data analytics in smart grids: State-of-the-art, challenges, opportunities, and future directions. *IET Smart Grid* **2019**, *2*, 141–154. [CrossRef]
16. Beigrezaei, M.; Toroghi Haghighat, A.; Leili Mirtaheri, S. Minimizing data access latency in data grids by neighborhood-based data replication and job scheduling. *Int. J. Commun. Syst.* **2020**, *33*, e4552. [CrossRef]
17. Kim, J. Partial rollback-based scheduling on in-memory transactional data grids. *Big Data Res.* **2017**, *9*, 47–56. [CrossRef]

18. Salhi, H.; Odeh, F.; Nasser, R.; Taweel, A. Open source in-memory data grid systems: Benchmarking hazelcast and infinispan. In Proceedings of the 8th ACM/SPEC on International Conference on Performance Engineering, L'Aquila, Italy, 22–26 April 2017.
19. Veseli, S.; Schwarz, N.; Schmitz, C. APS data management system. *J. Synchrotron Radiat.* **2018**, *25*, 1574–1580. [CrossRef] [PubMed]
20. Rashti, M.J.; Sabin, G.; Kettimuthu, R. Long-haul secure data transfer using hardware assisted GridFTP. *Future Gener. Comput. Syst.* **2016**, *56*, 265–276. [CrossRef]