Journal of Computer Science

Review Article

Review: Fusion Fault Tolerance Replication model and Fragmentation in Grid-cloud Distributed Environments

[1,2,3]Mohammed Adam Kunna Azrag, [1]Noraziah Ahmad, [1]Nurzety A. Azuan, [4]Zarina Mohamad and [5]Julius Beneoluchi Odili

[1]*Faculty of Computing, Universiti Malaysia Pahang Al-Sultan Abdullah, Pekan, Pahang, Malaysia*
[2]*Center for Artificial Intelligence & Data Science (CAIDAS), Universiti Malaysia Pahang Al-Sultan Abdullah, Kuantan, Pahang, Malaysia*
[3]*Institute of Big Data Analytics and Artificial Intelligence (IBDAII), Universiti Teknologi Mara, Shah Alam, Selangor, Malaysia*
[4]*Faculty of Informatics and Computing, Universiti Sultan Zainal Abidin, Terengganu, Malaysia*
[5]*Institute of Digital Humanities, Anchor University Lagos, Nigeria*

Article history
Received: 06-06-2023
Revised: 19-08-2024
Accepted: 05-11-2024

Corresponding Author:
Noraziah Ahmad
Faculty of Computing, Universiti Malaysia Pahang Al-Sultan Abdullah, Pekan, Pahang, Malaysia
Email: noraziah@umpsa.edu.my

Abstract: The study investigates key aspects of data management in distributed systems, focusing on fragmentation, replication and fault tolerance. With the increasing complexity of modern applications, efficient data handling across multiple nodes has become critical. The study begins by reviewing existing literature and moves on to analyze fragmentation techniques, evaluating their role in optimizing performance and resource utilization. Replication methods are discussed next, highlighting how data duplication improves availability and resilience against failures. The results indicate that efficient data management techniques in distributed systems significantly improve performance, availability and reliability. These findings contribute to a deeper understanding of the challenges and opportunities in distributed system environments, offering valuable insights for researchers and practitioners.

Keywords: Grid Computing, Cloud Computing, Data Replication, Fault-Tolerant, Computational Intelligence, Availability

Introduction

Because of their shared philosophical foundations, grid computing and cloud computing are frequently confused. There is a lot of overlap between the two ideas and they both aim to provide services to users by pooling resources. Because of their network architecture and multitasking capabilities, users in different places can access one or more program instances to perform separate but related tasks. The difference between grid computing and cloud computing is that the former involves virtualizing computing resources to store massive amounts of data, while the latter involves applications accessing resources indirectly through an internet service. In contrast to cloud computing, which centralizes resource management, grid computing disperses resources across grids (Srivastava & Khan, 2018). Grid computing is a computational architecture that relies on a network of interconnected computers to aggregate and process massive volumes of data. Grid computing is an enormous network of interconnected computers that address a common problem by dividing it into smaller, more manageable pieces called grids. Due to its decentralized design, it handles job scheduling and management in a dispersed approach, disregarding the passage of time. Figure (1) shows how the cluster of computers works as a virtual supercomputer, which facilitates large-scale tasks like data processing by giving users easy and scalable access to resources located all over the world (Shakarami *et al.*, 2020).

Instead of utilizing resources independently, applications in the cloud contribute to a shared pool. This way, the resources are not directly accessible to the applications. Designed for the remote provision of scalable and quantifiable IT resources, it represents a new paradigm in computing and is based on network technology. Making available a shared pool of configurable computing resources and high-level services on demand removes the need to invest much in local Infrastructure. The distributed computer resources are supervised by central management. In order to use applications and Software from any location, users are not required to know the exact whereabouts of their data. "Pay only for what you require," the old adage says. As can be observed in Figure (2).

Allocating resources over multiple servers in clusters, grid computing, and cloud computing make optimal use

of computing resources and make it easier to build applications without breaking the bank on hardware and Software. Computing in a grid is a method whereby computers from many fields work together to accomplish a common goal.

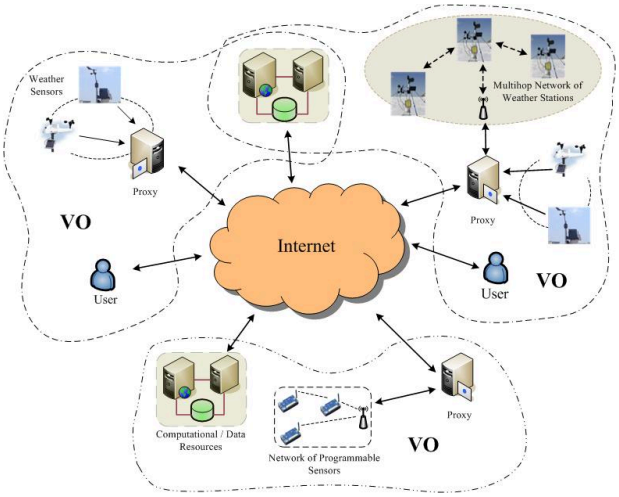

Fig. 1: Grid computing (Shakarami *et al.*, 2020)

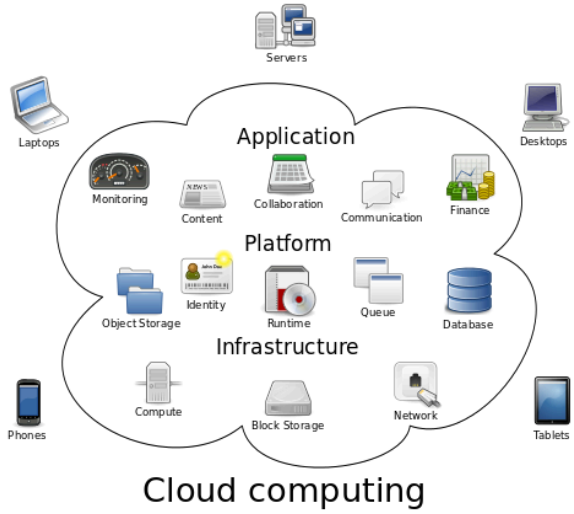

Fig. 2: Cloud computing (Kumar *et al.*, 2019a)

Grid computing, which involves storing massive amounts of data via virtualizing computing resources, is reviewed in this study. Cloud computing, on the other hand, is characterized by an application's indirect, rather than direct, access to resources and data through an internet service. By using either a centralized or decentralized replication technique, data is partitioned into numerous smaller, autonomous bits and then allotted to a preset, geographically dispersed environment.

By keeping numerous copies at each location depending on user access or work behaviour, replication ensures data transparency in a distributed database system. A system loses credibility or crashes in a Grid or Cloud computing setting if it doesn't work as planned. Additionally, the article discusses fault tolerance in fusion systems operating in Grid-Cloud distributed systems.

This study outline is as follows: A literature review is provided in the first section to set the stage for the subsequent debates. In Section 1, we take a look at the fragmentation, replication and fault tolerance. In the latter section, the study's findings are provided.

Literature Review

Data Grid and Cloud environment handle data sharing and data-intensive tasks (Srivastava & Khan, 2018). The high latency and massive amount of data needed by data-intensive activities make data availability and fast data access challenging (Shakarami *et al.*, 2020). Grid computing can be categorized into three main forms, each with its own set of uses and applications. Not only are there no clear-cut differences between the various grid types, but many grids actually use a combination of more than one (JosephNg *et al.*, 2020). The different kinds of grids are as follows.

Computational grid: Allowing resources for processing power is the main emphasis of the computational grid. As an example, the majority of the apparatus consists of high-performance servers (JosephNg *et al.*, 2020).

Scavenging grids: As an example, they are commonly used with large numbers of desktop workstations, each of which is normally given control over when its resources are available for grid participation (Argungu *et al.*, 2020).

A data grid is a system of interconnected computer resources spread out over multiple locations, either within the same country or in other countries (Khan *et al.*, 2021). Consider the following hypothetical technical scenario: Two colleges are conducting a life science study, and each of them has its own dataset. A grid connecting all of these sites improves control over data access rights and communication management (Aral & Ovatman, 2018; Li *et al.*, 2020).

Both the efficacy of replication schemes and the ability to provide a high-level overview of mind grid topologies are significantly affected by the underlying architecture of the grid (Li *et al.*, 2020; Awad *et al.*, 2021). When data needs to be shared across several global partnerships, hierarchical and tree models are employed (Awad *et al.*, 2021; Hamrouni *et al.*, 2016).

Tree topology isn't perfect, either. The grid's hierarchical design allows for the specialized conveyance of data via predetermined routes. Not only that, but it's also impossible for nodes on the same tier or siblings to transfer data to each other (Awad *et al.*, 2021; Hamrouni *et al.*, 2016).

Similarly, P2P systems allow for more versatile component communication and get beyond these restrictions. Programmes running on P2P systems are able to carry out decentralized tasks because they make use of distributed resources. The sharing of resources is where a P2P system and a grid system intersect. What sets a P2P system apart from others that share resources is the symmetric communication architecture between peers, who can act as both serves and clients (Awad *et al.*, 2021; Hamrouni *et al.*, 2016; Khan *et al.*, 2021).

An architecture that incorporates elements from all of the aforementioned topologies can be described as a hybrid topology. When academics are working on a project and wish to make their findings widely accessible for cooperation, they usually use it (Awad *et al.*, 2021; Hamrouni *et al.*, 2016).

One of the most crucial methods of managing data in grid systems is the replication approach. Improving data availability, fault tolerance, scalability, and bandwidth usage are goals of grid optimization, and one way to achieve these goals is by data replication. One way to optimize replicas and cut down on file access times is by directing access requests to the most relevant copies and automatically copying frequently used files based on access data.

The on-demand capacity management paradigm's practical and economic benefits have contributed to cloud computing's meteoric rise in popularity (Khan *et al.*, 2021). There has been a recent shift in computing towards outsourcing data processing to huge, central data centres run by third parties (Subramanian & Jeyaraj, 2018). Cloud computing's elasticity and adaptability, together with its ability to cut down on infrastructure investments and resource management costs, make it a compelling option for consumers (Gai *et al.*, 2020). Cloud computing has been around for a long, according to everyone. However, academics are still debating its precise definition, architecture and models. Web services, Service Oriented Architecture (SOA), virtualization in cloud grid computing and other similar concepts are not novel; rather, they stand as the culmination of relevant and established distributed computing ideas and technology (Subramanian & Jeyaraj, 2018).

In order to evaluate and develop data replica placement methods, Zheng *et al.* (2024) offer a real-world scenario-based approach for data replica placement in hybrid edge-cloud architecture. This model emphasizes features related to latency, dependability and load.

Many cloud providers now provide a wide variety of services, including Infrastructure as a Service (IaaS), Platform as a Service (PaaS) and Software as a Service (SaaS). These services are accessible to users through web browsers and the Internet. Although cloud computing has made many once difficult tasks much easier for clients, there is still a need for cost-effective, specialized cloud services that can adapt to changing Quality of Service (QoS) requirements and prevent Service Level Agreement (SLA) violations. A few of the most common quality of service metrics are reliability, security and response time.

For cloud computing, availability is of the utmost importance. If you believe (Kumar *et al.*, 2018), one of the top 10 challenges with cloud computing is availability. Aside from the obvious negative impact on user experience, service outages also lead to substantial drops in revenue (Abdel-Basset *et al.*, 2018). Many cloud service providers use replication strategies to ensure SLA adherence and tackle this problem. These approaches are believed to be good at preserving computing's availability, fault tolerance, Flexibility and scalability. Cloud replication supports a variety of service levels, each with its own fine-grained replication options. Infrastructure as a Service (IaaS) providers often replicate virtual machines, but Software as a Service (SaaS) providers are more likely to replicate services, applications, or data at the application or data level (Mesbahi *et al.*, 2018).

In order to mitigate the drawbacks of cloud computing, such as slow data access and unavailability, data replication is a crucial tool (Li *et al.*, 2020). There are several benefits to it. Data access time and bandwidth consumption are two areas that it can help with. Also, it might make things more available, scalable and load-balanced (Ramesh Naidu *et al.*, 2021). File replication and storage methods that are close to the data requester are referred to as replication (Gill *et al.*, 2019). Many different types of databases and systems rely on data replication, including DBMS, distributed databases, mobile systems, cloud, P2P and fog (Mansouri & Javidi, 2020).

In some fields of scientific application, including water and rain measurements, massive amounts of data are rapidly becoming a shared resource. Usually, numerous data centres are used to house such massive databases. For distributed management of large datasets, the data replication approach is the method of choice.

Fragmentation

The challenging job of managing fragmented database replication falls on administrators due to the fact that the distributed database is partitioned into multiple replicas. When a database is divided into multiple smaller, autonomous portions, the process is called fragmentation of data. When data is accessed in fragments, a table view context and partial data access are introduced. This is the initial stage in replacing coarse-grained techniques with fine-grained ones when picking data items. After data fragmentation, a preset, geographically dispersed environment is assigned to each fragment using one of four techniques: Centralization,

Fragmentation, Complete Replication, or Partial Replication. By keeping numerous copies at each location depending on user access or work behaviour, replication in a distributed database system maintains data transparency. Availability of data, fault tolerance and reliability at the site level are all guaranteed by these authors reported in Castillo-García *et al.* (2023).

All data distribution methods have one thing in common: they all aim to improve dispersed performance by doing the following:

1. Making data readily available to the workforce at all times and breaking the workload down into smaller, more manageable chunks
2. A modular design ensures that subqueries will run quickly
3. Making it easy to expand the network
4. Assembling the available storage space
5. Easier maintenance of data sites

Due to its many flaws, the prior centralized system was replaced with a decentralized one. Here are a few examples:

1. Load balance performance: In a centralized system, the database is accessed by multiple users simultaneously, which puts a strain on the system. This is because multiple users' queries are handled simultaneously. Database performance suffers, making it harder to respond quickly to queries from multiple users requesting the same data items
2. Intricacy and growth: As a result of the massive workforce, the data-centralized system is growing at an alarming rate. This makes the logical structure harder to understand and necessitates greater room for data storage. Additional database growth in this environment increases the likelihood of inadvertent data deletion
3. Ensuring data availability and maintenance: Data availability and service interruptions caused by maintaining massive amounts of data in a single database might last for quite a while. With a distributed database, this limitation can be easily addressed through data availability and maintenance. In a decentralized system, data is accessible to all nodes on the network
4. Fault tolerance: The central system’s lack of fault tolerance has a negative impact on data availability. However, a distributed approach simplifies this task by incorporating fault tolerance through replication. By ensuring that performance is unaffected by node failure, data is always available

In a distributed setting, a heuristic approach to query fragmentation is proposed to lower the Transmission Costs (TC) of queries. A cost-effective relational model is the foundation of fragmentation, which subsequently progresses to DDBS design. Afterwards, several scenarios based on replication were introduced, such as a Mixed Replication-Based Allocation Scenario (MAS), a Full Replication-Based Allocation Scenario (FAS) and a Non-Replication-Based Allocation Scenario (NAS) (Bittencourt *et al.*, 2018).

They propose a modified Bound Energy Algorithm (BEA), a hierarchical approach to vertically dividing the network into components and placing those components in particular locations within the network, to address these issues. Optimal allocation site selection, cluster cost calculation and attribute affinity clustering are all made easier with this method (Kaur *et al.*, 2019). Everything that can be obtained with a single query has been bundled together. By comparing and analyzing them from a design standpoint, the study sought to identify the pros and cons of current algorithms. All it does is show how a distributed context might benefit from a dynamic architecture for data fragment distribution (Kaur *et al.*, 2019).

An approach to dynamic fragment allocation that does not duplicate data is proposed in order to improve speed. The shifting pattern of access at various locations forms the basis of this. Pieces are redistributed based on the threshold value and the amount of data available on each fragment. This unique method alters the reallocation strategy by changing the read/write data volume factors, adding a threshold time volume and using the Distance Constraints Algorithm. As a result, the write data volume is considered during the reallocation procedure, and fragmentation is approaching at multiple sites. The distributed system's overall performance is enhanced as a result of this study (Nashat & Ameer, 2019).

Additionally, in order to ensure efficient and fault-tolerant data access in the systems, the replication method endeavours to keep numerous replicas. Despite prior research on data management (Qin *et al.*, 2018), few algorithms take a holistic look at the pros and cons of different replication strategies. To make data more accessible and efficient, several of them employ replication. As the system’s copy count increases, these metrics get better. Data replication uses energy and costs money for the provider, which is the most crucial aspect they failed to mention. Consequently, a data replication method that considers balancing many trade-offs is necessary (Qin *et al.*, 2018).

It has been previously investigated how to fragment tables so that local data can be accessed. Some of the studies on distributed file systems are also relevant to this. The usual level of detail of the data being considered (files vs. table) and the necessity of fragmentation features that can be used for partitioning in distributed database systems are two key differences between distributed file systems and distributed database systems.

Allocation of fragments and fragmentation are closely related processes. Fragment allocation is the sole function of some methods, while fragmentation is the

sole function of others (Pandian & Smys, 2020; Castro-Medina *et al.*, 2020a). More than one method combines the two responsibilities (Castro-Medina *et al.*, 2020b). Some methods combine fragmentation, allocation and replication into a single process, whereas others conduct replication independently of the other two steps (Argungu *et al.*, 2020; Awad *et al.*, 2021; Mesbahi *et al.*, 2018; Bittencourt *et al.*, 2018; Kaur *et al.*, 2019; Ahmad *et al.*, 2021a). Also, we still think re-fragmentation and reallocation deserve more attention as replication alternatives, even though dynamic replication methods can optimize for different metrics based on (Argungu *et al.*, 2020; Mesbahi *et al.*, 2018; Bittencourt *et al.*, 2018; Kaur *et al.*, 2019; Ahmad *et al.*, 2021b).

The following are undoubtedly the difficulties and concerns addressed as follows:

1. Managing a database with fragmented replications: Database administrators encounter challenges when partitioning a database into separate portions or duplicates. The database can be "fragmented" into smaller, more manageable pieces to facilitate partial data access and context-specific table views. Data item selection begins with this fine-grained approach, which is distinct from coarse-grained techniques
2. Methods for distributing data fragments: Once data has been fragmented, it must be given to contexts that are geographically spread. Centralized, full-replication, partial replication and fragmentation-based systems are among the mechanisms employed. Assuring data availability, fault tolerance, reliability and openness at individual sites is made possible through replication
3. Goals of data distribution methods: In order to improve distributed performance as a whole, data distribution methods divide workloads, use modular architecture to execute subqueries quickly, build networks that can accommodate new users, manage storage space effectively and make data site maintenance easier
4. Problems with the centralized system: Moving away from centralized to decentralized systems is happening for a number of reasons. Problems with data availability and maintenance in central databases, complexity and growth challenges as data volumes increase, problems with load balancing and performance due to several users' requests once and a lack of fault tolerance are all examples of such concerns
5. A heuristic approach to query fragmentation: A heuristic approach to query fragmentation is offered with the goal of lowering query transmission costs in faraway environments. Here, we propose to use a relational model to efficiently partition the data and then explore different replication-based allocation scenarios, including mixed replication, full replication and non-replication
6. Bond Energy Algorithm (BEA) Data Fragmentation: As a hierarchical approach to network fragmentation and geographical region assignment, the enhanced Bond Energy Algorithm (BEA) is proposed. Optimal allocation decisions, cluster allocation costs, attribute affinity, and clustering are all part of this approach
7. Method of dynamic fragment allocation: To improve efficiency, we provide a dynamic fragment allocation method that is not duplicated. When access patterns at different locations change, pieces are reassigned accordingly. Improved performance in distributed systems is the consequence of factors like read/write data volume, threshold time volume and distance constraints being considered

Database replication, allocation strategies, and moving from a centralized to a distributed system are all complicated by these issues. With the goal of improving distributed data management's performance and reliability, numerous strategies and approaches have been proposed to address these issues.

Replications

Reducing access latency, increasing grid performance and decreasing job response time are the three main goals of nearly all replication systems. Similarly, practically every replication strategy may be used to lessen the load on the network, which in turn boosts data availability and efficiency. The objective is to ensure that it is kept close to the user for easy access. Aiming to distribute the workload evenly among all data servers is one goal of many replication systems. The system's performance is enhanced, and response time is reduced. By making several copies of each job, replication provides fault tolerance (Isukapalli & Srirama, 2024). A master copy will be maintained to track the number of copies of each task and their locations. However, as the number of replicas grows, the expense of maintaining them becomes an overhead for the system. Minimizing the amount of data grid copies is an objective of some techniques. This keeps the replica's maintenance expenses low and guarantees optimal storage utilization. Placing replicas strategically and determining the ideal number of replicas are aspects of some techniques. There are a number of interconnected key aspects, one of which is the strategic placing of copies. By strategically placing the replicas, for instance, it becomes easier to distribute the workload among the available servers. The expense of upkeep is also associated with it. A strategy's load on the system will increase as replica maintenance costs become unsustainable if it continues to blindly replicate popular files (Ahmad *et al.*, 2021a).

The time it takes for a job to run is another important metric. Optimal replica placement is the goal of some replication algorithms, which attempt to decrease job execution time. Replicas should be located closer to users in order to decrease reaction time and, by

extension, the time it takes for jobs to be executed. As a result, the system's throughput will be enhanced. The possibility of using replication to offer fault tolerance and quality assurance has only been considered by a small number of replication systems. All methods of replication require a subset of these parameters.

Other replication algorithms are based on earlier proposals for replication replacement techniques. A few key examples of both old and current replication algorithms are detailed below.

A solution that has been put into practice is the simple optimizer algorithm (Javadpour *et al.*, 2023). Rather than replicating, it reads the necessary duplicate remotely. When compared to competing algorithms, a simple optimizer not only uses less storage space but also completes jobs faster and uses fewer network resources overall. The top client in terms of file size. Every node in the network checks the request count for each of its files at regular intervals; if it finds that the count has surpassed a specific point, it finds the best client to handle that file (Lv and Xiu, 2020). Replication provides support for the Cascading Tree design. The data files are initially created at the top level, and a copy is made at the following level when the number of file visits surpasses a certain threshold. But it starts out on the best client route and continues along it until it reaches the best client (Geng *et al.*, 2019).

Plain Cashing: This is an additional algorithm for replicating data. With this method, whenever a client requests a file, a local copy is made. The client's storage capacity can only hold a single file at a time. Therefore, huge files will be replaced promptly (Geng *et al.*, 2019). Cashing out plus Cascading is a strategy that combines cascading and plain cashing. While the server routinely detects which files are popular and propagates them down the hierarchy, the client stores the file locally in its cache. Remember that clients are always at the very top of the tree, but that every node in the hierarchy has the potential to function as a server. Sibling nodes are those that share a parent (Geng *et al.*, 2019).

Quick distribution: A duplicate of the file is created and then served at each node route to the client using this method. At each level, a copy is created whenever a client requests a file. When a node gets a file but doesn't have enough space, it will delete the one with the lowest popularity rating (Geng *et al.*, 2019).

The Least Frequently Used (LFU) approach does the same thing, constantly duplicating files to local storage systems. To make room for a new copy, if the local storage space is full, the one with the fewest accesses is eliminated. The duplicate with the lowest demand (popularity) gets deleted from local storage by LFU, regardless of how recently it was stored (Kaushik *et al.*, 2021). At the same time, the LRU strategy constantly backs up files to the local storage system. In the LRU technique, the requested site stores the necessary replicas in the cache. When the local storage becomes full, the oldest replica is removed to make room for new ones. If the new replica is larger than the oldest replica, the oldest file is erased, followed by the second oldest file and so on (Kaushik & Santosh, 2020).

In addition, the Proportional Share Replica (PSR) policy developed from the Cascading approach. This heuristic method assigns replicas to the most optimal places, given that the total number of sites and replicas to be distributed is known in advance. If a site's capacity to handle replica requests is slightly greater than or equal to the ideal load distribution, then replicas are distributed to that site (Ciccozzi *et al.*, 2023). An innovative dynamic replication technique, Bandwidth Hierarchy Replication (BHR), reduces the time it takes to retrieve data in a data grid network and avoids network congestion. The BHR approach makes use of "network-level locality," which guarantees that the critical file is kept at a location with high bandwidth in comparison to the location where the job is executed. The OptorSim Simulator was used to test the BHR method. The results show that the BHR method outperforms other optimization techniques in terms of data access time when an internet bandwidth hierarchy is present. By utilizing a network-level strategy, BHR sidesteps site-level optimization replicas (O'brien *et al.*, 2018).

Another two dynamic replication algorithms suggested for the multi-tier data grid design are Simple Bottom-Up (SBU) and Aggregate Bottom-Up (ABU). Any client data file that goes above a certain threshold is replicated by the SBU algorithm. The biggest problem with SBU is that it doesn't care about the connection to records of past access. In order to get to the bottom of the problem, ABU will compile all relevant historical documents and send them to the top tier. The Quick Distribution method proved to be the most effective. Data access distribution, interval checking, and replica sever capacity were calculated from the data access arrival rate and replica server capacity (Ullah *et al.*, 2018).

An operation research-based strategy for replica placement that is worth mentioning is the multi-objective approach. Here, the data request pattern and the present state of the network are considered when a replica is placed. The p-center problem seeks to find a solution that will allow the replica server and user site to respond faster. However, the p-center model's goal is to reduce the overall response time between request and replication sites, according to Ullah *et al.* (2018) and Xiao and Zhang (2019). In the Weight-based dynamic replica replacement approach, replica weight is also calculated using the future access window from the previous access history. The access cost is then calculated based on both the number of copies and the network's current bandwidth. There should be no deletion of low-weight replicas since they do not contribute to improving data access efficiency, but replicas with a high weight should

be retained. The distribution method for access history is similar to that of a zip file (He, 2020).

Another approach that replicates data dynamically is the Latest Access Largest Weight (LALW). Using a popular file as an example, LALW determines how many copies and grid sites are needed for replication. Each record of past data access is given a unique weight. Given the present state of data availability, this suggests that the record is more pertinent (Zhou *et al.*, 2017).

The agent-based replica placement method is another option for replica algorithms. This algorithm is suggested to choose a potential spot to put the duplicate. An agent will be set up for every location to store the master data files for the shared environment. Finding a potential location to set up a replica that minimizes application response times, network traffic, and access charges is an agent's principal objective. In addition, agents build replicas at appropriate locations by prioritizing grid resources according to resource configuration, network bandwidth and agent's insistence on the replica at their sites (Cheng *et al.*, 2018).

Also, APBRP, or Adaptive Popularity Based Replica Placement, is also important. The technique is a replica placement mechanism that uses hierarchical data grids and is directed by "file popularity," by strategically placing replicas near customers; this method aims to decrease data access time while optimizing the utilization of storage and network resources. How well APBRP works depends on the threshold value that is selected in proportion to the popularity of the files. Data request arrival rates are used by APBRP to dynamically establish this threshold (Khaldi *et al.*, 2020).

The Efficient Replication approach is another powerful replica technique; it takes site dynamics into account when replicating dynamic data grids. File availability, response time and bandwidth consumption can all be improved using this method. In order to achieve grid-wide load balancing, it employs replica placement and file requests. Because most grids employ a "load" strategy instead of an "update" one, this approach will prioritize read-only access and will result in minimal dynamic updates. The three stages provided by this algorithm are as follows:

1. Ran a list of all files and chose the ones with the highest number of requests and copies to replicate
2. Choosing the most suitable locations for the files, considering the number of requests and the relative usefulness of each location on the grid, follows from the previous step
3. Picking the optimal copy while keeping in mind the bandwidth and functionality of each site (Tripathi *et al.*, 2020)

On top of that, replication methods would be incomplete without the Value-Based Replication Strategy (VBRS). In an effort to boost system performance and decrease network latency, the VBRS was suggested. To avoid duplicating files, VBRS allows users to choose a threshold before deciding whether to replicate the requested file. There are two portions of VBRS. Finding out if the requested file needs to be copied to the local storage site begins with calculating the threshold. In the second stage, the replacement mechanism is engaged if there isn't enough capacity at the local storage site to copy the requested file. When formulating the policy for replacing replicas, it is necessary to consider the value of each replica, which is established by the file's access time and frequency. Network latency can be reduced by the VBRS algorithm, according to the findings of the trials (Mohammad Hasani Zade *et al.*, 2021).

The Enhanced Fast Spread (EPS) method is also highly regarded. When it comes to data grid replication strategies, the EPS is an upgraded version of Fast Spread. The goal of recommending this approach was to decrease bandwidth usage and increase overall response time. It takes a lot of things into account, such as the size of the replica, the frequency and number of requests and the most recent request for the replica. With the EFS method, you may replace less important duplicates with more important ones while keeping only the most important ones. A dynamic threshold is used to do this by deciding whether the replica should be stored at each node along the route to the requester (Dähling *et al.*, 2021).

Another enhanced variant of the popular fast spread used in multi-tier data grid settings is Predictive Hierarchical Fast Spread (PHFS). In an effort to improve performance that takes spatial proximity into account, the PHFS pre-replicates the minimum hierarchical fashion and seeks to foresee future needs. This technique makes the most efficient use of available storage space by obtaining additional access locales by hierarchical replication of data objects in various layers of a multi-tier data grid. Data grids that are read-intensive are the intended target of this technique. A priority mechanism and component for changing replication configurations are used by the PHFS technique to adapt the replication configuration to the current state of affairs. It also presumes that, with a high degree of certainty, the users operating in the same environment will request certain files (Rabie *et al.*, 2021).

In a similar vein, DHR is a Hierarchical replication algorithm that finds the best places to put copies. Consequently, most people may get that copy through the top site. This algorithm chooses the optimal replica when many sites hold them, minimizing access latency. Users executing jobs are directed to the optimal replica location via the DHR algorithm's replica selection method, which takes data transfer time and the queued number of replica requests into account. It doesn't duplicate files but rather stores them in the most popular and optimal location rather than storing them in several locations (Alzakholi *et al.*, 2020).

When it comes to dynamic data replication strategies, one of the greatest is EMRALW or Enhanced the Most Recent Access Largest Weight. Compared to the Original Latest Access Largest Weight method, this one is far better. EMRALW considers three crucial variables while erasing files:

1. The replicas with the lowest usage rates
2. Replicas that haven't been in use for a long time
3. How big is the replica?

Each duplicate is stored by EMRALW at a suitable location in the region that will have the most future access to it. The EMRALW method achieves better performance than the other algorithms and efficiently uses storage by preventing the formation of unneeded duplicates, according to the testing results (Meng *et al.*, 2019).

Fault Tolerance

The term "cloud computing" describes the practice of providing shared computing resources, including storage, operating systems, and virtualization, over the Internet as a service. Customers still have a lot of concerns about reliability in a cloud computing setting, despite the fact that the benefits of cloud computing include cheap rates, pay-per-use on-demand service, and guaranteed quality of service. The ability of a task to consistently finish within a given time limit is what the term "reliability" means. Systems lose confidence or crash in Grid or Cloud Computing environments when they don't work as planned. When Software or hardware fails to perform as expected, we say that the system has failed. To continue working properly in the face of such setbacks, we need fault-tolerance abilities (Kumar *et al.*, 2019b). A wide variety of errors are possible (Sharif *et al.*, 2020).

Network faults: These can be caused by packet loss, network partitioning, or link failure.

Faults in the Processor: These processor faults happen as a result of the operating system's actions.

Process faults: These occur when there are problems with the program or insufficient resources.

Service expiration faults: When the service timeout duration of the resource expires while the app is executing, this sort of failure occurs.

When it comes to fault tolerance in the cloud, three different methods exist (Alarifi *et al.*, 2019; Fard *et al.*, 2020):

The Proactive Fault Approach: This method replaces distributed components before they fail and finds problems before they happen. It involves avoiding recovery from errors and instead preparing for defects. Following are some of the tactics used in the Proactive Fault Tolerance method:

1. Software rejuvenation: A system that has been restarted at regular intervals will be started in a different state each time
2. Auto-Repair: When a program is operating on many virtual machines, and this technology detects a problem, it takes over the application
3. Preemptive migration: This approach continually assesses an application and moves resources to a different virtual machine if one gets too busy

Reactive fault tolerance: This technique is used to reduce or eliminate the impact of a system failure as soon as it occurs. Here are the techniques that are part of this method:

1. Checkpointing: It is possible to continue a failed task from the most recent checkpoint rather than having to start it from the beginning. The resource cache is where checkpoints are kept. This approach is used for applications on a large scale
2. Task resubmission: This method sends out failed tasks again to the same or a different computer so they can be executed again
3. S-Guard: This technique, which is used in HADOOP and Amazon EC2, guarantees that the job is rolled back if it fails
4. Job migration: When one machine stops working, another one takes over the duty. Errors may be resolved, and batch programs can be migrated over a cloud to several data centres using specific algorithms. Haproxy is used to complete this task
5. Exceptions that are defined by the user: Using this method, the user can dictate what to do in the case of a workflow task failing at any point

Adaptive fault tolerance method: This method works by automatically executing all procedures depending on the system's current state. To ensure that essential modules are reliable, it monitors the dependability of virtual machines per process.

By combining proactive and reactive fault tolerance tactics, the adaptive fault tolerance method can effectively reduce faults. Replication, checkpointing and fragmentation are some of the tactics. By dividing a large file into smaller ones, fragmentation makes it possible to merge them into a single larger file while preserving all of the original data. This approach lessens the quantity of irrelevant data applications retrieved (Fard *et al.*, 2020; Ubaidillah *et al.*, 2021; Castro-Medina *et al.*, 2019; Santos *et al.*, 2019; Qiu *et al.*, 2021; Arunarani *et al.*, 2019; Awad *et al.*, 2019; Ahmad *et al.*, 2021b; Rady *et al.*, 2019; Kaushik *et al.*, 2021).

Automatic Software Self-Healing Through Rescue Points, or Assure, is an additional method of tolerance. Works well for rescue points that handle faults that programmers expect. In the event of a program failure, ASSURE will step in and restore execution to the closest rescue point utilizing virtualization, allowing the application to recover.

Not only that, but HAProxy works wonders. "High Availability Proxy" is the abbreviation of this term. By

distributing the workload over numerous computers, this tool enhances the reliability and efficiency of a server cluster. It is a free and open-source program that helps websites with load balancing. Its efficiency with CPU and memory is commendable. Its steadiness and durability have also brought it fame. Many popular websites, including Twitter (X), Stack Overflow, and GitHub, use this method to reduce their heavy traffic.

The SHelp is an important player in this space. Using error virtualization and rescue points, SHelp is a runtime setup that can handle virtual machines. It reduces duplication of effort and bounces back fast from mistakes caused by the same or similar flaws. SHelp makes it possible for server programs to quickly recover from these issues.

When it comes to cloud storage, Amazon Elastic Compute Cloud (EC2) is a powerful online service platform that caters to customers' individual requirements. This is the most effective method of time management because it gives programmers the resources they need to create programs that never crash.

Hadoop has also recently exploded in popularity. Many cloud computing services rely on Hadoop. It allows a cluster of computers to handle enormous datasets in a distributed fashion; it is an open-source project from the Apache Software Foundation. Infrastructure as a service, platform as a service and Software as a service are all part of its public-private cloud packages.

Many versions are used based on the previously mentioned technology: The idea of replication-based low Latency Fault Tolerance (LLFT) was introduced by (Almutairy *et al.*, 2019) to provide fault tolerance in distributed applications hosted in the cloud. This paradigm uses many replication techniques to protect programs from different types of faults. The primary aspects of cloud computing that are addressed by Kumar *et al.* (2017) are its reliability, availability, and security. Also, the researcher explores several intrusion detection approaches with a focus on how to enhance RAS via the use of virtualization technologies. Use the model suggested by Jhawar and Piuri (2017) to make grid computing more reliable. The core principle is that reliable service providers are the only ones that can provide the data storage, networking and computing power that a cloud infrastructure needs to function properly. Additionally, in order to enhance the integration of different limitations, it has prioritized several dependability criteria. Since the introduction of a virtualized technology-based Fault Tolerance Management (FTM) approach (Bharany *et al.*, 2022) to study fault tolerance, cloud computing systems have become much more dependable and accessible.

An important addition came from (Bala and Chana, 2015), who put out a model of DAFT that is compatible with cloud fault tolerance semantically and ethically. To make this design more reliable, they use checkpointing and replication to make it fault-tolerant. A model for intelligent task failure detection based on proactive fault tolerance was used to forecast the failure task in scientific process applications (Amin *et al.*, 2015).

An agent-based Fault Tolerance and Recovery System (FRAS) was also proposed (Charity & Hua, 2016). When things go wrong, the recovery agent uses the rollback feature. This approach proposes a recovery mechanism for agents to maintain a consistent state in a system.

Furthermore, the concept of migration-based Virtual Data Centers (VDCs) was first proposed by Agarwal and Sharma (2015). To resolve server failures caused by virtual machines becoming overcrowded, resources are transferred to other virtual machines. The idea of Cloud Fault Tolerance (CFT) was put forth by Jamil *et al.* (2021). In this paradigm, the dependability of computer nodes is tracked, and nodes are chosen according to their reliability. They remove the node that keeps giving us false results. Jamshidi *et al.* (2018) introduced a fault tolerance paradigm that relies on decisions. Accuracy and response time are the two metrics that define a node's reliability in this paradigm. When a node stops working, they try to fix it from the past, and they add or remove nodes depending on how reliable they are.

If they want a more reliable cloud computing ecosystem, we should look into the methods suggested by (Agarwal and Sharma, 2015) for fault tolerance and monitoring. This article covers cloud fault tolerance research priorities for the future and gives details on the numerous strategies and approaches used for fault tolerance. For software rejuvenation in the cloud, (Jamil *et al.*, 2021) proposed a reliability model based on dynamic fault trees. System performance may be continuously degraded owing to resource depletion, fragmentation and fault accretion; this model emphasizes the software ageing problem. A method for cloud computing fault tolerance was proposed by Jamshidi *et al.* (2018). The model's developer elaborates on the process by which faults give rise to major problems. In order to anticipate these mistakes and act appropriately before or after a failure happens, this model makes use of a number of fault tolerance mechanisms. The idea of FT-Cloud was first presented by Hasan and Goraya (2018). This part is decided by how well developers of cloud applications are rated. Direct fault tolerance is built into this algorithm. An AFTRC strategy was proposed in the context of real-time cloud computing by Noraziah *et al.* (2021); Ubaidillah *et al.* (2020); Ubaidillah *et al.* (2021); Almjlae *et al.* (2019); Awang *et al.* (2019)). Computing in real-time on the cloud is the foundation of this concept. Forwarding recovery is an advantage it offers. There is a lot of fault tolerance in this model. Additionally, this model highlights reversal recovery through checkpointing.

To sum up, the following are some of the most important issues that were found throughout the literature review on cloud/grid computing fault tolerance:

1. Processing happens via remote machines, which increases the likelihood of errors
2. Effective methods for fault detection must be developed
3. Customers only get limited data since the system is so dense
4. We need to create more technology that can predict faults in real-time
5. Understanding the ever-changing system state is challenging because of the dynamic nature of the cloud environment
6. Fault tolerance methods should be put in place even though the customer's organization isn't worried about data centre faults.
7. We can use new metaheuristic algorithms for detecting fault tolerance, such as in Azrag *et al.* (2018); Kunna *et al.* (2018); Adam Kunna Azrag *et al.* (2020)

Conclusion

In this study, we explore fragmentation, replication and fault tolerance as critical components of data management in distributed systems. The research emphasizes that fragmentation enhances performance and resource allocation by distributing data among remote nodes, which is essential for meeting diverse computational demands. Replication strategies, particularly those balancing consistency and latency, prove indispensable for maintaining data availability and integrity, especially in expanding, complex systems. Finally, fault tolerance is a key factor in ensuring system resilience, as our analysis highlights various adaptive strategies that address failures at both the hardware and software levels. Overall, this research contributes significantly to understanding how these components interact to improve performance, reliability and system scalability. The insights gathered here offer a robust foundation for future work in optimizing distributed data management, benefiting researchers, system designers and developers.

Acknowledgement

The authors would like to thank the University of Malaysia Pahang Al-Sultan Abdullah (UMPSA) for providing laboratory facilities, library services, and financial support for this research. Special thanks to Faculty of Computing, UMPSA and Universiti Sultan Zainal Abidin (UniSZA), Terengganu, Malaysia for the Article Processing Charge (APC) of this study.

Funding Information

This study is formerly supported by the Short-Term Grant (RDU1903122) sponsored by the UMPSA.

Author's Contributions

Mohammed Adam Kunna Azrag: Wrote the original draft of the manuscript.

Noraziah Ahmad: Supervised the research, secured funding, contributed to writing, and served as the corresponding author.

Nurzety Ahmad Azuan: Contributed to writing and proofreading the manuscript.

Zarina Mohamad: Contributed to manuscript writing and provided research funding.

Julius Beneoluchi Odili: Edited and structured the manuscript.

Ethics

The authors don't have any ethical issues that may arise after the publication of this manuscript.

References

Abdel-Basset, M., Mohamed, M., & Chang, V. (2018). NMCDA: A Framework for Evaluating Cloud Computing Services. *Future Generation Computer Systems*, *86*, 12-29. https://doi.org/10.1016/j.future.2018.03.014

Adam Kunna Azrag, M., Asmawaty Abdul Kadir, T., & Mohd Ali, N. (2020). A Comparison of Particle Swarm optimization and Global African Buffalo Optimization. *IOP Conference Series: Materials Science and Engineering*, *769*(1), 012034. https://doi.org/10.1088/1757-899x/769/1/012034

Agarwal, H., & Sharma, A. (2015). A Comprehensive Survey of Fault Tolerance Techniques in Cloud Computing. *2015 International Conference on Computing and Network Communications (CoCoNet)*, 408-413. https://doi.org/10.1109/coconet.2015.7411218

Ahmad, W., Rasool, A., Javed, A. R., Baker, T., & Jalil, Z. (2021). Cyber Security in IoT-Based Cloud Computing: A Comprehensive Survey. *Electronics*, *11*(1), 16-169. https://doi.org/10.3390/electronics11010016

Ahmad, Z., Nazir, B., & Umer, A. (2021). A Fault-Tolerant Workflow Management System with Quality-of-Service-Aware Scheduling for Scientific Workflows in Cloud Computing. *International Journal of Communication Systems*, *34*(1), 4649. https://doi.org/10.1002/dac.4649

Alarifi, A., Abdelsamie, F., & Amoon, M. (2019). A Fault-Tolerant Aware Scheduling Method for Fog-Cloud Environments. *PLOS ONE*, *14*(10), 0223902. https://doi.org/10.1371/journal.pone.0223902

Almjlae, S., Mohamad, Z., & Suryani, W. (2019). Impact Aspectes of it Flexibility Specific to Cloud Computing Adoption on it Effectiveness. *Journal of Theoretical and Applied Information Technology*, *97*(3), 1041-1059.

Almutairy, N. M., Al-Shqeerat, K. H. A., & Al Hamad, H. A. (2019). A Taxonomy of Virtualization Security Issues in Cloud Computing Environments. *Indian Journal of Science and Technology*, *12*(3), 1-19. https://doi.org/10.17485/ijst/2019/v12i3/139557

Alzakholi, O., Haji, L. M., Shukur, H. M., Zebari, R. R., Abas, S. M., & Sadeeq, M. A. M. (2020). Comparison Among Cloud Technologies and Cloud Performance. *Journal of Applied Science and Technology Trends*, *1*(1), 40-47. https://doi.org/10.38094/jastt1219

Amin, Z., Singh, H., & Sethi, N. (2015). Review on Fault Tolerance Techniques in Cloud Computing. *International Journal of Computer Applications*, *116*(18), 11-17. https://doi.org/10.5120/20435-2768

Aral, A., & Ovatman, T. (2018). A Decentralized Replica Placement Algorithm for Edge Computing. *IEEE Transactions on Network and Service Management*, *15*(2), 516-529. https://doi.org/10.1109/tnsm.2017.2788945

Argungu, S. M., Arif, S., & Hasbullah Omar, Mohd. (2020). Compute and Data Grids Simulation Tools: A Comparative Analysis. *Emerging Trends in Intelligent Computing and Informatics*, *1073*, 533-544. https://doi.org/10.1007/978-3-030-33582-3_50

Arunarani, AR., Manjula, D., & Sugumaran, V. (2019). Task Scheduling Techniques in Cloud Computing: A Literature Survey. *Future Generation Computer Systems*, *91*, 407-415. https://doi.org/10.1016/j.future.2018.09.014

Awad, A., Salem, R., Abdelkader, H., & Salam, M. A. (2021). A Novel Intelligent Approach for Dynamic Data Replication in Cloud Environment. *IEEE Access*, *9*, 40240-40254. https://doi.org/10.1109/access.2021.3064917

Awang, W. S. W., Deris, M. M., Rana, O. F., Zarina, M., & Rose, A. N. M. (2019). Affinity Replica Selection in Distributed Systems. *Parallel Computing Technologies*, *11657*, 385-399. https://doi.org/10.1007/978-3-030-25636-4_30

Azrag, M. A. K., Kadir, T. A. A., & Jaber, A. S. (2018). Segment Particle Swarm Optimization Adoption for Large-Scale Kinetic Parameter Identification of *Escherichia Coli* Metabolic Network Model. *IEEE Access*, *6*, 78622-78639. https://doi.org/10.1109/access.2018.2885118

Bala, A., & Chana, I. (2015). Intelligent Failure Prediction Models for Scientific Workflows. *Expert Systems with Applications*, *42*(3), 980-989. https://doi.org/10.1016/j.eswa.2014.09.014

Bharany, S., Badotra, S., Sharma, S., Rani, S., Alazab, M., Jhaveri, R. H., & Reddy Gadekallu, T. (2022). Energy Efficient Fault Tolerance Techniques in Green Cloud Computing: A Systematic Survey and Taxonomy. *Sustainable Energy Technologies and Assessments*, *53*, 102613. https://doi.org/10.1016/j.seta.2022.102613

Bittencourt, L. F., Goldman, A., Madeira, E. R. M., da Fonseca, N. L. S., & Sakellariou, R. (2018). Scheduling in Distributed Systems: A Cloud Computing Perspective. *Computer Science Review*, *30*, 31-54. https://doi.org/10.1016/j.cosrev.2018.08.002

Castillo-García, A., Rodríguez-Mazahua, L., Castro-Medina, F., Arrioja-Rodríguez, M. L., Sánchez-Cervantes, J. L., & Rodríguez-Mazahua, N. (2023). Design of a Dynamic Horizontal Fragmentation Method for Multimedia Databases. *Handbook on Decision Making*, *226*, 71-91. https://doi.org/10.1007/978-3-031-08246-7_4

Castro-Medina, F., Rodriguez-Mazahua, L., Abud-Figueroa, M. A., Romero-Torres, C., Reyes-Hernandez, L. A., & Alor-Hernandez, G. (2019). Application of Data Fragmentation and Replication Methods in the Cloud: A Review. *2019 International Conference on Electronics, Communications and Computers (CONIELECOMP)*, 47-54. https://doi.org/10.1109/conielecomp.2019.8673249

Castro-Medina, F., Rodriguez-Mazahua, L., López-Chau, A., Abud-Figueroa, M. A., & Alor-Hernández, G. (2020). FRAGMENT: A Web Application for Database Fragmentation, Allocation and Replication over a Cloud Environment. *IEEE Latin America Transactions*, *18*(06), 1126-1134. https://doi.org/10.1109/tla.2020.9099751

Castro-Medina, F., Rodríguez-Mazahua, L., López-Chau, A., Cervantes, J., Alor-Hernández, G., & Machorro-Cano, I. (2020). Application of Dynamic Fragmentation Methods in Multimedia Databases: A Review. *Entropy*, *22*(12), 1352-1442. https://doi.org/10.3390/e22121352

Charity, T. J., & Hua, G. C. (2016). Resource Reliability Using Fault Tolerance in Cloud Computing. *2016 2nd International Conference on Next Generation Computing Technologies (NGCT)*, 65-71. https://doi.org/10.1109/ngct.2016.7877391

Cheng, Y., Hao, Z., Cai, R., Wen, W., Wang, L., & Zhou, Z. (2018). Resource-Aware Stream Processing in High Performance Cloud Environment. *2018 IEEE SmartWorld, Ubiquitous Intelligence & Computing, Advanced & Trusted Computing, Scalable Computing & Communications, Cloud & Big Data Computing, Internet of People and Smart City Innovation (SmartWorld/SCALCOM/UIC/ATC/CBDCom/IOP/SCI)*, 381-388. https://doi.org/10.1109/smartworld.2018.00095

Ciccozzi, F., Addazi, L., Asadollah, S. A., Lisper, B., Masud, A. N., & Mubeen, S. (2023). A Comprehensive Exploration of Languages for Parallel Computing. *ACM Computing Surveys*, *55*(2), 1-39. https://doi.org/10.1145/3485008

Dähling, S., Razik, L., & Monti, A. (2021). Enabling Scalable and Fault-Tolerant Multi-Agent systems by Utilizing Cloud-Native Computing. *Autonomous Agents and Multi-Agent Systems*, *35*(1), 10. https://doi.org/10.1007/s10458-020-09489-0

Fard, M. V., Sahafi, A., Rahmani, A. M., & Mashhadi, P. S. (2020). Resource Allocation Mechanisms in Cloud Computing: A Systematic Literature Review. *IET Software*, *14*(6), 638-653. https://doi.org/10.1049/iet-sen.2019.0338

Gai, K., Guo, J., Zhu, L., & Yu, S. (2020). Blockchain Meets Cloud Computing: A Survey. *IEEE Communications Surveys & Tutorials*, *22*(3), 2009-2030. https://doi.org/10.1109/comst.2020.2989392

Geng, X., Mao, Y., Xiong, M., & Liu, Y. (2019). An Improved Task Scheduling Algorithm for Scientific Workflow in Cloud Computing Environment. *Cluster Computing*, *22*(S3), 7539-7548. https://doi.org/10.1007/s10586-018-1856-1

Gill, S. S., Garraghan, P., Stankovski, V., Casale, G., Thulasiram, R. K., Ghosh, S. K., Ramamohanarao, K., & Buyya, R. (2019). Holistic Resource Management for Sustainable and Reliable Cloud Computing: An innovative Solution to Global Challenge. *Journal of Systems and Software*, *155*, 104-129. https://doi.org/10.1016/j.jss.2019.05.025

Hamrouni, T., Hamdeni, C., & Charrada, F. B. (2016). Objective Assessment of the Performance of Data Grid Replication Strategies Based on Distribution Quality. *International Journal of Web Engineering and Technology*, *11*(1), 3-28. https://doi.org/10.1504/ijwet.2016.075948

Hasan, M., & Goraya, M. S. (2018). Fault Tolerance in Cloud Computing Environment: A Systematic Survey. *Computers in Industry*, *99*, 156-172. https://doi.org/10.1016/j.compind.2018.03.027

He, H. (2020). A reliable peer-to-peer storage framework based on virtual peers model. *International Journal of Networking and Virtual Organisations*, *22*(2), 129. https://doi.org/10.1504/ijnvo.2020.105513

Isukapalli, S., & Srirama, S. N. (2024). A Systematic Survey on Fault-Tolerant Solutions for Distributed Data Analytics: Taxonomy, Comparison, and Future Directions. *Computer Science Review*, *53*, 100660. https://doi.org/10.1016/j.cosrev.2024.100660

Jamil, H., Umer, T., Ceken, C., & Al-Turjman, F. (2021). Decision Based Model for Real-Time IoT Analysis Using Big Data and Machine Learning. *Wireless Personal Communications*, *121*(4), 2947-2959. https://doi.org/10.1007/s11277-021-08857-7

Jamshidi, P., Pahl, C., Mendonca, N. C., Lewis, J., & Tilkov, S. (2018). Microservices: The Journey So Far and Challenges Ahead. *IEEE Software*, *35*(3), 24-35. https://doi.org/10.1109/ms.2018.2141039

Javadpour, A., Sangaiah, A. K., Pinto, P., Ja'fari, F., Zhang, W., Majed Hossein Abadi, A., & Ahmadi, H. (2023). An Energy-Optimized Embedded Load Balancing using DVFS Computing in Cloud Data Centers. *Computer Communications*, *197*, 255-266. https://doi.org/10.1016/j.comcom.2022.10.019

Jhawar, R., & Piuri, V. (2017). *Fault Tolerance and Resilience in Cloud Computing Environments*. 165-181. https://doi.org/10.1016/b978-0-12-803843-7.00009-0

JosephNg, P. S., Loh, Y. F., & Eaw, H. C. (2020). Grid Computing for MSE during Volatile Economy. *2020 20th International Conference on Control, Automation and Systems (ICCAS)*, 709-714. https://doi.org/10.23919/iccas50221.2020.9268216

Kaur, A., Gupta, P., Singh, M., & Nayyar, A. (2019). Data Placement in Era of Cloud Computing: a Survey, Taxonomy and Open Research Issues. *Scalable Computing: Practice and Experience*, *20*(2), 377-398. https://doi.org/10.12694/scpe.v20i2.1530

Kaushik, M., & Santosh, M. (2020). A State-of-Art on Cloud Load Balancing Algorithms. *International Journal of Computing and Digital Systems*, *9*(2), 201-220. https://doi.org/10.12785/ijcds/090206

Kaushik, P., Rao, A. M., Singh, D. P., Vashisht, S., & Gupta, S. (2021). Cloud Computing and Comparison based on Service and Performance between Amazon AWS, Microsoft Azure, and Google Cloud. *2021 International Conference on Technological Advancements and Innovations (ICTAI)*, 268-273. https://doi.org/10.1109/ictai53825.2021.9673425

Khaldi, M., Rebbah, M., Meftah, B., & Smail, O. (2020). Fault Tolerance for a Scientific Workflow System in a Cloud Computing Environment. *International Journal of Computers and Applications*, *42*(7), 705-714. https://doi.org/10.1080/1206212x.2019.1647651

Khan, A. W., Khan, M. U., Khan, J. A., Ahmad, A., Khan, K., Zamir, M., Kim, W., & Ijaz, M. F. (2021). Analyzing and Evaluating Critical Challenges and Practices for Software Vendor Organizations to Secure Big Data on Cloud Computing: An AHP-Based Systematic Approach. *IEEE Access*, *9*, 107309-107332. https://doi.org/10.1109/access.2021.3100287

Kumar, G. M., Ramachandram, S., & Gyani, J. (2017). Parameterized Reliable Trust Model in Grid Computing using Regression. *International Journal of Advanced Research in Computer Science*, *8*(3), 423-430.

Kumar, M., Sharma, S. C., Goel, A., & Singh, S. P. (2019). A Comprehensive Survey for Scheduling Techniques in Cloud Computing. *Journal of Network and Computer Applications*, *143*, 1-33. https://doi.org/10.1016/j.jnca.2019.06.006

Kumar, P. R., Raj, P. H., & Jelciana, P. (2018). Exploring Data Security Issues and Solutions in Cloud Computing. *Procedia Computer Science*, *125*, 691-697. https://doi.org/10.1016/j.procs.2017.12.089

Kumar, V., Laghari, A. A., Karim, S., Shakir, M., & Anwar Brohi, A. (2019). Comparison of Fog Computing & Cloud Computing. *International Journal of Mathematical Sciences and Computing*, *5*(1), 31-41. https://doi.org/10.5815/ijmsc.2019.01.03

Kunna, M. A., Kadir, T. A. A., Remli, M. A., Ali, N. M., Moorthy, K., & Muhammad, N. (2020). An Enhanced Segment Particle Swarm Optimization Algorithm for Kinetic Parameters Estimation of the Main Metabolic Model of Escherichia Coli. *Processes*, *8*(8), 963-1006. https://doi.org/10.3390/pr8080963

Li, C., Song, M., Zhang, M., & Luo, Y. (2020). Effective Replica Management for Improving Reliability and Availability in Edge-Cloud Computing Environment. *Journal of Parallel and Distributed Computing*, *143*, 107-128. https://doi.org/10.1016/j.jpdc.2020.04.012

Lv, Z., & Xiu, W. (2020). Interaction of Edge-Cloud Computing Based on SDN and NFV for Next Generation IoT. *IEEE Internet of Things Journal*, *7*(7), 5706-5712. https://doi.org/10.1109/jiot.2019.2942719

Mansouri, N., & Javidi, M. M. (2020). A Review of Data Replication Based on Meta-Heuristics Approach in Cloud Computing and Data Grid. *Soft Computing*, *24*(19), 14503-14530. https://doi.org/10.1007/s00500-020-04802-1

Meng, S., Li, Q., Wu, T., Huang, W., Zhang, J., & Li, W. (2019). A Fault-Tolerant Dynamic Scheduling Method on Hierarchical Mobile Edge Cloud Computing. *Computational Intelligence*, *35*(3), 577-598. https://doi.org/10.1111/coin.12219

Mesbahi, M. R., Rahmani, A. M., & Hosseinzadeh, M. (2018). Reliability and High Availability in Cloud Computing Environments: A Reference Roadmap. *Human-Centric Computing and Information Sciences*, *8*(1), 1-31. https://doi.org/10.1186/s13673-018-0143-8

Mohamed, S. A., & Yousif, A. (2019). Bridging the Gap between Service Oriented Architecture and Cloud Software as a Service. *2019 International Conference on Computer, Control, Electrical, and Electronics Engineering (ICCCEEE)*, 1-6. https://doi.org/10.1109/iccceee46830.2019.9071375

Mohammad Hasani Zade, B., Mansouri, N., & Javidi, M. M. (2021). Multi-Objective Scheduling Technique Based on Hybrid Hitchcock Bird Algorithm and Fuzzy Signature in Cloud Computing. *Engineering Applications of Artificial Intelligence*, *104*, 104372. https://doi.org/10.1016/j.engappai.2021.104372

Nashat, D., & Amer, A. A. (2019). A Comprehensive Taxonomy of Fragmentation and Allocation Techniques in Distributed Database Design. *ACM Computing Surveys*, *51*(1), 1-25. https://doi.org/10.1145/3150223

Noraziah, A., Fauzi, A. A. C., Ubaidillah, S. H. S. A., Alkazemi, B., & Odili, J. B. (2021). BVAGQ-AR for Fragmented Database Replication Management. *IEEE Access*, *9*, 56168-56177. https://doi.org/10.1109/access.2021.3065944

O'Brien, K., Pietri, I., Reddy, R., Lastovetsky, A., & Sakellariou, R. (2018). A Survey of Power and Energy Predictive Models in HPC Systems and Applications. *ACM Computing Surveys*, *50*(3), 1-38. https://doi.org/10.1145/3078811

Pandian, A. P., & Smys, S. (2020). Effective Fragmentation Minimization by Cloud Enabled Back Up Storage. *Journal of Ubiquitous Computing and Communication Technologies*, *2*(1), 1-9. https://doi.org/10.36548/jucct.2020.1.001

Qin, Y., Yang, W., Ai, X., & Chen, L. (2018). Fault Tolerant Storage and Data Access Optimization in Data Center Networks. *Journal of Network and Computer Applications*, *113*, 109-118. https://doi.org/10.1016/j.jnca.2018.04.001

Qiu, H., Noura, H., Qiu, M., Ming, Z., & Memmi, G. (2021). A User-Centric Data Protection Method for Cloud Storage Based on Invertible DWT. *IEEE Transactions on Cloud Computing*, *9*(4), 1293-1304. https://doi.org/10.1109/tcc.2019.2911679

Rabie, A. H., Saleh, A. I., & Ali, H. A. (2021). Smart Electrical Grids Based on Cloud, IoT, and Big Data Technologies: State of the Art. *Journal of Ambient Intelligence and Humanized Computing*, *12*(10), 9449-9480. https://doi.org/10.1007/s12652-020-02685-6

Rady, M., Abdelkader, T., & Ismail, R. (2019). Integrity and Confidentiality in Cloud Outsourced Data. *Ain Shams Engineering Journal*, *10*(2), 275-285. https://doi.org/10.1016/j.asej.2019.03.002

Ramesh Naidu, P., Guruprasad, N., & Dankan Gowda, V. (2021). A High-Availability and Integrity Layer for Cloud Storage, Cloud Computing Security: From Single to Multi-Clouds. *Journal of Physics: Conference Series*, *1921*(1), 012072. https://doi.org/10.1088/1742-6596/1921/1/012072

Santos, N. L., Ghita, B., & Masala, G. L. (2019). Enhancing Data Security in Cloud using Random Pattern Fragmentation and a Distributed NoSQL Database. *2019 IEEE International Conference on Systems, Man and Cybernetics (SMC)*, 3735-3740. https://doi.org/10.1109/smc.2019.8914454

Shakarami, A., Ghobaei-Arani, M., Masdari, M., & Hosseinzadeh, M. (2020). A Survey on the Computation Offloading Approaches in Mobile Edge/Cloud Computing Environment: A Stochastic-based Perspective. *Journal of Grid Computing*, *18*(4), 639-671. https://doi.org/10.1007/s10723-020-09530-2

Sharif, H., Oo, A. M. T., Haroon, K. M., & Kaosar, M. (2020). A Review on Various Smart Grid Technologies used in Power System. *International Research Journal of Engineering and Technology*, *7*(8), 4091-4097.

Srivastava, P., & Khan, R. (2018). A Review Paper on Cloud Computing. *International Journal of Advanced Research in Computer Science and Software Engineering*, *8*(6), 17-20.

Subramanian, N., & Jeyaraj, A. (2018). Recent Security Challenges in Cloud Computing. *Computers & Electrical Engineering*, *71*, 28-42. https://doi.org/10.1016/j.compeleceng.2018.06.006

Tripathi, A. K., Agrawal, S., & Gupta, R. D. (2020). Cloud Enabled SDI Architecture: A Review. *Earth Science Informatics*, *13*(2), 211-231. https://doi.org/10.1007/s12145-020-00446-9

Ubaidillah, S. H. S. A., Alkazemi, B., & Noraziah, A. (2021). An Efficient Data Replication Technique with Fault Tolerance Approach using BVAG with Checkpoint and Rollback-Recovery. *International Journal of Advanced Computer Science and Applications*, *12*(1), 475-480. https://doi.org/10.14569/ijacsa.2021.0120155

Ubaidillah, S. H. S. A., Noraziah, A., & Beneoluchi Odili, J. (2020). Fragmentation Techniques For Ideal Performance In Distributed Database â€“ A Survey. *International Journal of Software Engineering and Computer Systems*, *6*(1), 18-24. https://doi.org/10.15282/ijsecs.6.1.2020.3.0066

Ullah, S., Awan, M. D., & Sikander Hayat Khiyal, M. (2018). Big Data in Cloud Computing: A Resource Management Perspective. *Scientific Programming*, *2018*, 1-17. https://doi.org/10.1155/2018/5418679

Xiao, P., & Zhang, T. (2019). A novel replication scheme based on prediction technology in virtual P2P storage platform. *International Journal of Networking and Virtual Organisations*, *20*(1), 90. https://doi.org/10.1504/ijnvo.2019.096615

Zheng, M., Du, X., Lu, Z., & Duan, Q. (2024). A Balanced and Reliable Data Replica Placement Scheme Based on Reinforcement Learning in Edge-Cloud Environments. *Future Generation Computer Systems*, *155*, 132-145. https://doi.org/10.1016/j.future.2024.02.004

Zhou, A., Wang, S., Hsu, C., Kim, M. H., & Wong, K. (2017). Network Failure-Aware Redundant Virtual Machine Placement in a Cloud Data Center. *Concurrency and Computation: Practice and Experience*, *29*(24), 4290. https://doi.org/10.1002/cpe.4290