Hierarchical Virtual Storage

Evgeniy Ibatullin[1], Valery Khvatov[2], Alexander Bogdanov[1], and Nadezhda Shchegoleva[1(✉)]

[1] St. Petersburg University, St. Petersburg, Russia
{a.v.bogdanov,n.shchegoleva}@spbu.ru
[2] DGT Technologies AG., Toranto, Canada

Abstract. The increasing complexity of data management and storage systems, coupled with the growing demand for flexible and efficient solutions, has led to the emergence of data virtualization technologies. This study investigates the potential for enhancing data storage methodologies through virtualization approaches, particularly focusing on the integration of hierarchical storage systems. Additionally, the research explores methods for organizing the management of such systems. The principles, methodologies, and architectures underlying distributed storage systems employed for handling big data tasks are analyzed. Experimental validation was conducted through the physical implementation of a prototype system on hardware. The results demonstrate a hierarchical data storage system leveraging virtualization, facilitating seamless data access and integration from disparate sources independently of their structure or storage method. Furthermore, a management approach based on reinforcement learning is proposed for controlling the developed storage system.

Keywords: Data management · Storage systems · Data virtualization technologies · Data Lake approach · Distributed File System · Containers cluster · Data-mesh system

1 Introduction

1.1 Relevance of the Topic

The emergence of data virtualization has become increasingly critical due to the rapid evolution of data management practices. Its significance lies in addressing the following dynamic challenges:

1. Managing the Ever-Growing Complexity and Volume of Data:
   (a) Applications today generate and rely upon an unprecedented volume and diversity of data.
   (b) The proliferation of varied data organization methods complicates storage, retrieval, and efficient utilization.
   (c) Contemporary systems frequently experience performance constraints, particularly bottlenecks at preprocessing and data storage layers, hindering timely and efficient data delivery.

© The Author(s), under exclusive license to Springer Nature Switzerland AG 2026
O. Gervasi et al. (Eds.): ICCSA 2025 Workshops, LNCS 15894, pp. 231–248, 2026.
https://doi.org/10.1007/978-3-031-97648-3_16

2. Tackling Novel Challenges in Big Data Management:
   (a) Big data environments introduce unique complexities absent in traditional data management frameworks.
   (b) Storing metadata in raw form inhibits direct data interactions by applications, requiring intricate and resource-intensive preprocessing.
   (c) Multiple preprocessing stages significantly escalate resource demands, resulting in higher operational latency and complexity.
3. Unlocking Potential through Data-as-a-Service (DaaS) Architecture:
   (a) Implementing a Data-as-a-Service architecture strategically distributes workloads between data extraction and storage processes.
   (b) This architectural model provides an agile foundation that scales effectively, adapting seamlessly to rapidly changing business and technological landscapes.

1.2 Defining the Research Object

This research addresses comprehensive systems designed explicitly for processing, storing, and managing large-scale data (big data), focusing on maximizing efficiency and adaptability.

1.3 Clarifying the Research Subject

The study specifically targets systems that streamline data accessibility within big data ecosystems through virtualization. The virtualization approach is being studied in [1] and [2], which explains why the virtualization approach is relevant for big data environments. The goal is to achieve a unified and efficient access point that minimizes structural dependencies.

1.4 Research Objectives

The central objective of this research is the conceptualization and development of a hierarchical data integration system powered by virtualization technology. The system designed will:

- Facilitate seamless access to and integration of data originating from diverse and distributed sources.
- Ensure efficient data delivery regardless of original storage methodologies or structural complexities.
- Establish and implement an effective, adaptive management methodology for robust operational control.

1.5 Detailed Research Tasks

To accomplish the research objectives, the following tasks have been outlined:

1. **Comprehensive Evaluation**: Critically assess contemporary data storage methods and architectures to identify strengths, weaknesses, and opportunities for virtualization.
2. **Exploration of Advanced Technologies**: Investigate and evaluate cutting-edge technologies that could facilitate efficient and scalable data virtualization systems.
3. **Architectural Development**: Formulate an innovative architecture employing proven technologies to create a virtualization system well-aligned with current and emerging IT industry trends.
4. **Technological Innovation**: Identify technologies not currently available or sufficiently mature and describe the development processes necessary for their creation and integration.

1.6 Scientific Novelty and Practical Significance

The scientific innovation of this research is evident in:

- **Integration of Virtual Core Technology**: Leveraging virtual cores to dynamically scale systems, optimizing load balancing, and improving responsiveness.
- **Advanced Multi-layered Data Access Model**: Developing sophisticated, multi-tiered access architecture significantly enhances the efficiency and effectiveness of data retrieval operations.

Practical contributions of this study include:

- **Empowering Business through Enhanced Solutions**: Meeting critical business needs by delivering effective, scalable storage solutions that address the exponential growth in data.
- **Improving System Performance**: Offering organizations robust, reliable, and scalable methodologies for managing extensive data operations, significantly boosting operational efficiency and strategic agility.

2 Main Section

2.1 Problem Statement

The rapid advancement of technology has led to the emergence of Big Data, significantly reshaping the landscape of data management. In work [3] it is shown that the transition of the system to the big data environment requires completely new approaches capable of adapting to new demands The escalating demands for precise measurements and sophisticated modeling techniques require the handling of increasingly vast datasets, frequently surpassing hundreds of petabytes.

This trend is especially evident in international experimental research domains. As sensors and detection equipment continue to evolve, generating data with greater precision and higher frequency, the volume of collected data is set to grow exponentially. Consequently, the challenge emerges to develop advanced data storage solutions that can not only accommodate such massive datasets efficiently but also enable rapid, seamless, and reliable user access.

2.2 MPD Experiment Case Study

To illustrate the contemporary challenges associated with Big Data analytics, the MPD (Multi-Purpose Detector) experiment at the Joint Institute for Nuclear Research provides an insightful case study. The primary goal of this experiment is to examine particle collision processes by collecting, processing, and analyzing extensive datasets generated by specialized sensor arrays. The computational and storage requirements for such experiments have been thoroughly analyzed in prior research [4].

As outlined in [4], accurate estimations of data transfer rates, event sizes, and reconstruction speeds are crucial for determining the necessary computing resources: «Assuming no preliminary data selection at the initial L3 production stage (prior to reconstruction), the expected data transfer rate will be around 1.3 GB/s, equating to approximately 70 TB per day. Considering an individual event size of 0.45 MB, a predicted trigger frequency of 6 kHz, and a 0.5 reduction factor due to past-future protection, the annual storage demand for RAW data alone is estimated at 8.4 PB. Furthermore, accounting for around 20% simulated events and additional compact data files produced at expert request, total annual storage requirements may rise to approximately 10 PB. Event reconstruction, estimated at 2 s per event, translates to a total computational demand of about $37.4 \leq 10^9$ seconds measured in 1K SI2K units, corresponding to roughly 1480 standalone 1K SI2K category PCs. Including additional computational demands arising from online processing farms and approximately 20% extra resources needed for simulation activities, the total requirement increases to around 1800 such machines.»

Table 1 in reference [4] further elaborates on the disk space needed for data preprocessing and storage, providing concrete insights into the substantial computational and storage infrastructure necessary for supporting international collaborative experiments. A significant challenge arises from the continuous influx of semi-structured data streams produced by diverse sensors, necessitating highly adaptable storage solutions customized for unique and evolving data handling requirements. Consequently, traditional storage systems fall short in addressing these specialized and rigorous demands, underscoring the critical need for innovative, flexible, and high-performance data management infrastructures in experimental contexts such as MPD.

Table 1. Estimation of the necessary disk space and power of PCs.

Parameter	Value	Unit
Data rate from L2	4.7	GB/s
Experimental run duration	120	day
Efficiency	60	%
Event rate	6	kHz
Past-future protection	50	%
Total number of events	$1.87 \cdot 10^{10}$	event
Mean number of tracks per event	500	track
Mean number of hits per particle (TPC + eCAL)	20	hit
Mean number of bytes per hit	45	byte
Total size of RAW data	8.4	PB
Total necessary disk space	10	PB
Mean reconstruction time per event by 1K SI2K PC	2	s
Total time per year for 1 PC	7000	hour
Number of 1K SI2K PCs for reconstruction	1480	
Total number of necessary PCs	1800	

2.3 Methods of Virtualization

Virtualization Overview. In this research, virtualization refers to the abstraction layer over physical hardware that facilitates treating distributed resources as a cohesive whole. It should also be noted that virtualization can be applied to logical constructions to unify separate systems effectively.

The main virtualization technologies include virtual machines (VMs) and containers:

- Virtual Machine (VM): A VM represents either hardware or software emulation of a computing environment, providing isolation from the host operating system. This approach can be implemented via a hypervisor directly on hardware or atop an existing operating system. While advantageous due to its operating system independence, a notable downside is the slower startup time.
- Container: Containers provide isolated software packages capable of running applications in tailored environments. Compared to VMs, containers offer quicker deployment as they do not require full operating systems. However, container execution necessitates specialized software known as a container engine.

The advancement of virtualization has demonstrated that container-based environments offer superior adaptability and speed, contributing to their widespread adoption in business environments. Research [5], investigating the feasibility of constructing supercomputers via virtualization, highlights containers as the optimal choice for distributed system implementations.

Advantages of Virtualization are extensively utilized within distributed systems, offering several distinct benefits relevant to this research:

1. Scalability and Management: Allocating CPU resources to specific tasks can be complex due to varying system loads and limited resources. Virtualization enables dynamic resource allocation, allowing precise control over resources consumed by individual applications within containers.
2. Efficient Resource Utilization: Virtualization optimizes load distribution based on computational capabilities, effectively mitigating system overload and minimizing idle resources.
3. Isolation and Security: Virtualization ensures each task operates within separate namespaces, significantly enhancing system security and eliminating conflicts among concurrent processes.
4. Cost Efficiency: Optimized resource allocation through virtualization results in cost savings, particularly noticeable in large-scale systems where identifying underutilized resources can be challenging.

Logical Parallelism in Processors. Virtualization enables applications to function within isolated environments, addressing critical load-balancing challenges associated with classical cluster systems. Virtualization supports vertical scaling via virtual CPUs (vCPUs) created through logical parallelism.

Creating vCPUs involves dividing the processor's operational time into segments, each representing a distinct vCPU's execution cycle. Figure 1 illustrates the cumulative operational time of two vCPUs, demonstrating how logical parallelism efficiently facilitates parallel processing within virtualized systems.

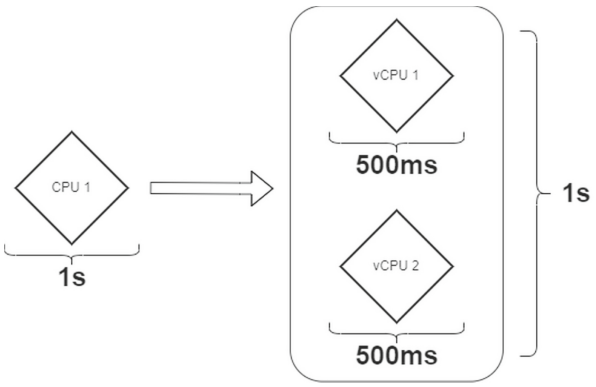

Fig. 1. Splitting a Single Physical CPU into Two vCPUs.

This allows having a network of virtual cores based on such a virtual cluster to organize a data access service in which the user receives the required number of virtual cores to extract data from storage.

In this case, there is only a layer interface for the user, which provides data through the virtualization described above, which simplifies access to the system in the case of users, and this approach also expands the possibilities for administering such virtual clusters.

2.4 Data Storage Methods

This section discusses classical distributed storage systems, highlighting their limitations in meeting modern business requirements for comprehensive data processing, especially as Big Data evolves and integrates various data domains.

Current Approaches to Data Organization. Among the most prevalent contemporary data storage methods are structured data replication in Data Warehouses and unstructured data storage in Data Lakes. However, as data handling continues to advance, Data Warehouses have demonstrated limitations, particularly concerning scalability. High costs and bottlenecks associated with preprocessing and data structuring in large-scale operations pose significant operational risks. Figure 2 illustrates the evolution of key data storage approaches.

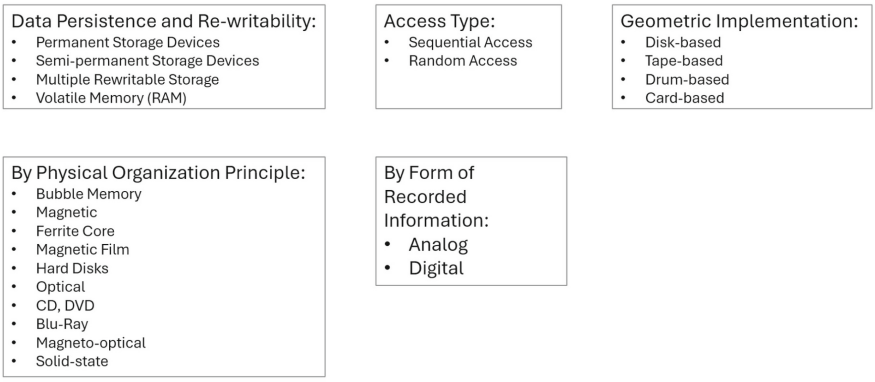

Fig. 2. Evolution of Data Storage and Processing Technologies.

Data Lakes. Following the Data Warehouse era, Data Lakes emerged as a prominent solution, focusing on unstructured data to eliminate preprocessing bottlenecks. Nevertheless, this approach necessitates meticulous metadata management and continuous maintenance. Rapid data accumulation can lead to clutter with redundant or useless data, degrading overall data quality.

Key advantages of the Data Lake approach include:

1. Scalability: Data Lakes efficiently manage data volumes reaching several petabytes.

2. Cost Efficiency: Experience indicates Data Lakes are more economical than traditional Data Warehouses.
3. Flexibility: Data Lakes accommodate various data formats, both structured and unstructured.
4. Reliability and Accessibility: Distributed data mechanisms ensure secure storage and rapid data access.

If we consider the case study of MPD from previous section, there is a research in high energy physics [6] where it being described why and how to move to data-lake systems in this cutting-edge computing field. This confirms that such systems are in demand in the most heavily loaded systems.

Hadoop Distributed File System (HDFS). The Hadoop Distributed File System (HDFS) represents a foundational Data Lake architecture, leveraging data clusters for orchestrating metadata management, data control, distribution, and availability. Core HDFS principles underpin many modern data storage solutions. Figure 3 demonstrates an example of an HDFS-based storage system. HDFS, as open-source software, initially addressed essential business requirements for implementing Data Lake architecture. Fundamental distributed storage methodologies, including replication, sharding, and consensus-based recovery, exemplify advanced techniques integral to contemporary storage solutions.

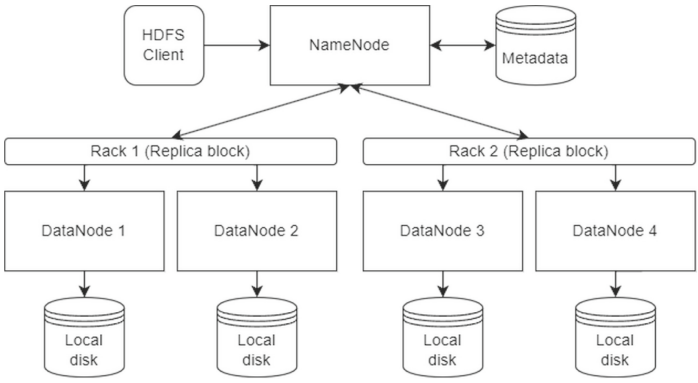

Fig. 3. HDFS-Based Storage Architecture.

Hierarchical Approach. The specific challenges in Mega Science projects have given rise to an approach similar to distributed Data Lake systems, but one that addresses the particular needs of preprocessing and long-term storage. This approach implements storage systems that work with disk spaces of varying structures and speeds. The hierarchical methodology necessitates organizing levels that handle specific data types, allowing for more efficient use of existing

resources. However, administering such a system becomes a distinct challenge requiring detailed consideration.

A classic example of such a system is CERN's data storage implementation, which uses active storage for «hot data» and tape libraries for data archiving. This allows users to access current information quickly, while less frequently accessed «cold data» is transferred to tape storage, the most cost-effective method in terms of price per gigabyte of storage.

Similarly, extending the multi-tier architecture concept to full-scale storage systems, both local and global, enables the formation of Grid architecture used in worldwide scientific experiments. The foundation of such a system is communication between distributed organizations that provide data processing and storage. Today, the standard architecture consists of three or four layers:

- Level 0: Ensures acquisition and distribution of experimental data
- Level 1: Receives data blocks from the initial center and processes them before transferring to Level 2 nodes
- Level 2: Responsible for research and analysis
- Level 3: Comprised of smaller centers responsible for information storage

Extraction with Virtualization. With user requests and the necessary number of virtual cores, it's possible to organize access to data nodes for extraction and data organization. Streaming extraction tools assume a highly parallel distributed architecture that enables efficient data extraction. However, problems arise with new artificial intelligence requests that require access not to specific data areas within an organization, but to complete datasets. In such situations, extraction systems must work with heterogeneous storage environments represented by relational database tables as well as Data Lake architecture. Therefore, virtualization of logical clusters aims to solve this problem.

The following sections examine the main extraction technologies used in modern data processing pipelines.

MapReduce. MapReduce is a distributed computing model used for parallel computations on very large datasets, including volumes reaching several petabytes.

The advantage of MapReduce lies in its ability to perform distributed data preprocessing and reduction operations. The preprocessing operations (Map phase) work independently of each other and can be executed in parallel (though in practice this is limited by the input data source and/or the number of processors available). Similarly, multiple worker nodes can perform the reduction operations (Reduce phase) - this only requires that all preprocessing results with a specific key value be processed by a single worker node at one time.

Although this process may be less efficient compared to algorithms implementing partial sequential operations, MapReduce can be applied to massive datasets that can be processed by a large number of servers. Figure 4 illustrates the classic workflow of the MapReduce model across three computational nodes.

The MapReduce paradigm consists of two primary functions:

- **Map**: processes input key/value pairs to generate intermediate key/value pairs
- **Reduce**: mergers all intermediate values associated with the same intermediate key

This model has become fundamental to many big data processing frameworks, offering a balance between processing capability, scalability, and programming simplicity.

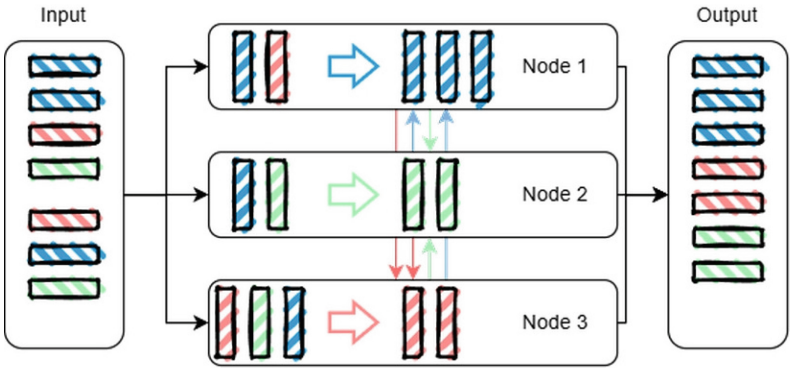

Fig. 4. The Process of the Map Reduce Model.

Apache Spark. Apache Spark is an open-source framework for distributed processing of unstructured and semi-structured data, part of the Hadoop ecosystem.

Unlike the classic processor from the Hadoop core that implements the two-level MapReduce concept with intermediate data storage on disk, Spark operates in the in-memory computing paradigm—processing data in RAM. This approach provides significant speed advantages for certain classes of tasks, particularly the ability to repeatedly access user data loaded into memory, making the library well-suited for machine learning algorithms.

Data Repository Queries. For end users, through virtualization, the entire storage system appears as a single access point through which they make data requests. After a user receives a virtual cluster with allocated resources from the system, they can make requests to the Spark Master node:

- Users write Spark applications (in Scala, Python, or Java)
- Applications are sent to the cluster manager (e.g., standalone Spark manager, YARN, or Mesos)

An alternative approach is preprocessing data streams into traditional tabular or other storage types from which users can retrieve information in their preferred way. PostgreSQL can serve as an example of such a database.

2.5 Practical Implementation

Implementing the proposed system requires a distributed network of multiple nodes. Additionally, virtualization and cluster organization are needed, along with an abstraction layer that provides end users with simplified access to the entire system. Below is a set of technologies and their organization method to achieve the necessary architecture.

Containerization. A container-based system requires deploying an operating system on a physical machine and a means for launching and supporting containers. Docker, an open-source product, can serve as this tool. This approach allows for: isolating programs within a single machine, flexible resource management, and automated deployment.

Clustering. After selecting the container launch mechanism, it's necessary to organize the resulting containers into a cluster. For this, the Kubernetes orchestration tool is required. This tool is designed to maintain a fault-tolerant cluster. In its architecture, Kubernetes uses a declarative cluster deployment system implemented through a scheduler. The advantages of this system include scalability, fault tolerance, and availability. Through this orchestrator, master nodes are deployed to manage worker nodes. The result is a container cluster or virtual cluster, as the number of nodes depends on the number of deployed containers. On top of this system, a storage or extraction system can be deployed. Figure 5 shows the architecture of a virtual cluster in the context of a two-layer hierarchical storage system.

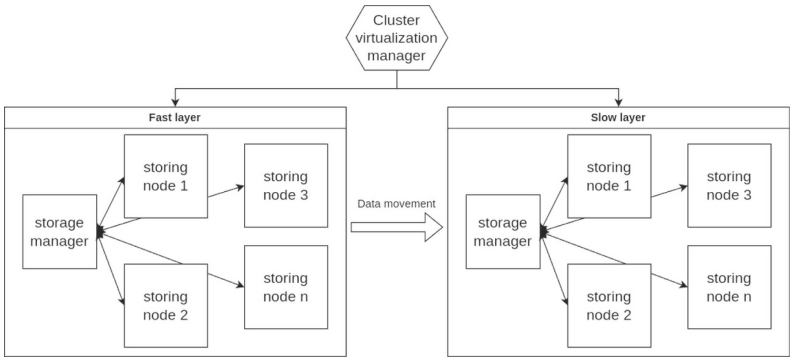

Fig. 5. Virtual Cluster Architecture.

Data Extraction Layer. As shown above, new challenges arise due to evolving business requirements in large-scale organizations. Extraction systems like Spark are designed to extract data from a single storage system, which leads to the problem of ensuring data completeness in machine learning tasks. To address this issue, the Data-Mesh concept has been proposed, aimed at working with heterogeneous data domains. In this system, the data extraction layer plays a key role. It consists of a set of tools for working with various storage systems and, through virtualization, presents information from different domains as a unified whole.

Data-Mesh Architecture. The proposed solution involves an intermediate layer that unites different data domains. Data is also virtualized using the method described above. Let's first examine one of the data domains locally. This environment represents a data storage system, in this case, a data lake. Data flows into this environment from various data streams, and the resulting information is used by the department for operations.

To implement local virtualization, it's necessary to present data stored in a distributed data center as a single entity through metadata. This also requires organizing high-speed communication between storage system nodes. After creating a virtual representation of each local data domain, they must be combined to obtain a complete data-mesh system.

In practice, an intermediate layer is proposed to organize this virtual representation. The virtual representation allows us to organize data domains as cluster nodes. On the layer that sits between the consumer and the data system, sets of metadata are organized to determine the location of data corresponding to each virtual node. Each node has a representation for its local cluster, so determining data location can be organized through access to this cluster.

Once the location is determined, data needs to be extracted from the storage system nodes. This requires a highly parallel extraction system such as Apache Spark, described above. Ultimately, this entire system—from local Data Lakes to the computational cluster for data extraction—can be virtualized, presenting the entire architecture to the end user as a single machine with a single storage repository.

This creates an abstraction layer that works with metadata and the extraction system to provide the user with a unified, virtualized view of the entire complex system. Figure 6 illustrates the architecture of the proposed system.

2.6 Data Transferring

For the effective organization of data at different levels of work, which have a different structure due to the equipment used, a dynamic system is needed that can adapt to user requests. To solve this problem, [7] proposes to use reinforcement learning, representing the data movement model as a Markov process.

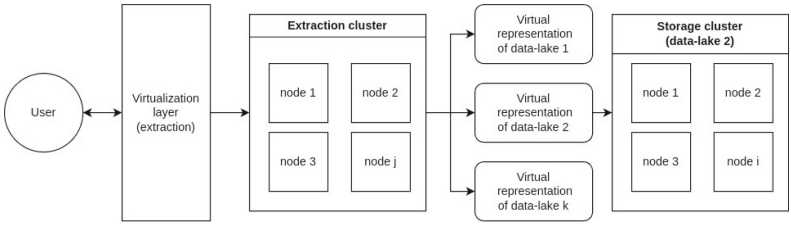

Fig. 6. Architecture data-meshes.

Finite Markov Decision Processes. In a finite Markov Decision Process (MDP), three finite sets are defined: a set of states $\mathcal{S}$, a set of actions $\mathcal{A}$, and a set of rewards $\mathcal{R}$. Consequently, at time t, the values R_t and S_t have discrete probability distributions that depend solely on the preceding state and action. This framework allows us to determine the probability of random variables $\acute{s} \in \mathcal{S}$ and $r \in \mathcal{R}$ occurring.

The key characteristic of an MDP is the Markov property, which states that the next state and reward depend only on the current state and action, not on the history of previous states and actions. This property can be expressed formally as:

$$p(\acute{s}, r|s, a) = Pr\{S_t = \acute{s}, R_t = r|S_{t-1} = s, A_{t-1} = a\} \tag{1}$$

for all $\acute{s}, s \in \mathcal{S}, r \in \mathcal{R}$ and $a \in \mathcal{A}(s)$.

The value of performing action a in state s in strategy π, denoted by $Q_\pi(s, a)$, is defined as the expected return when the agent starts working in state s, takes action a, and then follows strategy π:

$$Q_\pi(s, a) = E[G_t|S_t = s, A_t = a] = E[\sum_{k=0}^{\infty} \gamma^k R_{t+k+1}|S_t = s, A_t = a] \tag{2}$$

Reinforcement Learning. For the implementation of reinforcement learning, Q-learning is proposed. This function approximates the environment, and the system takes an action $a \in \mathcal{A}$ from a finite set of all possible data placements across different levels. Since the volume of the action set a is not high, a greedy search can be organized to find the best action at the current moment t, allowing the system to adapt to environmental changes. Additional flexibility is achieved through the model's hyperparameters. Reinforcement learning in this context offers several key advantages:

1. **Adaptability**: The system can dynamically adjust to changing data access patterns and storage conditions without requiring explicit reprogramming.
2. **Optimization**: Through continuous interaction with the environment, the system learns optimal data placement strategies that minimize access time and maximize resource utilization.

3. **Autonomy**: Once trained, the system can make complex decisions about data placement with minimal human intervention, reducing operational overhead.
4. **Scalability**: The reinforcement learning approach can handle growing data volumes and expanding storage hierarchies by generalizing from its experience.

The Q-learning algorithm specifically uses a Q-table to store the expected utility of taking a given action in each state. The update rule is:

$$Q(s,a) \leftarrow Q(s,a) + \alpha \left(r + \gamma \max_{a'} Q(s',a') - Q(s,a) \right) \tag{3}$$

where:

- α is the learning rate,
- γ is the discount factor for future rewards,
- r is the immediate reward,
- s' is the new state after taking action a,
- $\max_{a'} Q(s',a')$ is the maximum expected future reward.

This approach enables the system to optimize data placement across different storage tiers based on access patterns, importance, and performance requirements.

2.7 System Analysis

After building the system, it's necessary to conduct an analysis aimed at studying the proposed approach in comparison with existing options. It's also important to evaluate the system according to established criteria.

Evaluation of Extraction Process Efficiency. This part of the work aims to assess the efficiency of parallelism in the system. The main parallel operations during data extraction are executed in the MapReduce model process. Therefore, the main quality metrics of the system will be the following:

1. **Throughput** - measures the amount of data transmitted in the network per unit of time.
2. **Latency** - time required to execute one atomic data extraction operation.
3. **Resource utilization** - assessment of how the physical capabilities of the system affect data extraction processes.
4. **Query execution time** - total time from when a user sends a request until data is received.

Scalability. An important aspect of a distributed system is its ability to expand to a larger number of nodes, and this scaling should increase the computational power of the cluster rather than decrease it due to communication costs.

1. **Horizontal scalability** - evaluating system performance when adding more nodes to the cluster.
2. **Vertical scalability** - evaluating system performance when adding additional physical resources.
3. **Speedup** - the ratio of the execution time of a system running on a single processor to the execution time of a parallel system.
4. **Parallelism efficiency** - the proportion of execution time of a parallel algorithm during which processors are used most effectively.

Availability. This criterion is most important for users. When situations arise where the system is not available, the organization suffers losses, and users waste their time.

1. **Fault tolerance** - the system's ability to continue operating when nodes fail. Recovery - the system's ability to automatically recover from failures within the system.
2. **Uptime** - the system's ability to respond to user requests.

Data Quality. It is extremely important to maintain data quality as this is a critical aspect of systems working with unstructured data. If we consider the problem of data quality on the part of each individual node, then the problem arises in a decentralized environment. Study [8] describes how this problem can be solved and how to check if our implementation provides good quality of data. The main criteria for evaluating data quality include:

1. Accuracy - the degree to which data elements in a dataset have correct data values or correct data labels. A data element is syntactically correct if its data value has the same type as its explicit data type, and semantically correct if its data value has the expected value.
2. Completeness - the presence of data values for all intended attributes and entity instances. The characteristic of labeled data completeness in a dataset is relative. In different scenarios, the concept of completeness may differ, and it should be considered depending on the specific use case.
3. Credibility - the degree to which data has attributes that users tend to trust in a particular context of use. Credibility applies to individual data elements, to related data elements in a data record, and to the entire dataset. The context in which data is used can affect its perceived credibility and plausibility.

These criteria are described in detail in [9] and the data quality model is presented in [10].

Evaluation of Load Balancing Efficiency. This section examines the system's ability to allocate virtual environments for executing extraction tasks, as well as comparing it with classical load balancing methods. Load balancing is a critical component in distributed data extraction systems, ensuring optimal resource utilization and minimizing response times. The efficiency of load balancing can be evaluated through several key metrics:

1. **Resource Distribution Efficiency** - Measures how evenly computational tasks are distributed across available nodes, preventing both underutilization and overloading. This can be quantified using the variance in CPU, memory, and network utilization across nodes.
2. **Task Allocation Time** - The time required to analyze incoming extraction requests, determine resource requirements, and assign them to appropriate virtual environments. Lower allocation times indicate more responsive load balancing.
3. **Dynamic Adaptation** - The system's ability to reallocate resources in response to changing workloads or node failures. This can be measured by observing recovery time and performance stability during peak loads or partial system failures.
4. **Scaling Efficiency** - How effectively the load balancer distributes tasks when new computational resources are added to the system, measured as the ratio of performance improvement to resources added.
5. **Fairness** - Ensures that all users receive appropriate resource allocation according to their priority levels and service agreements, preventing resource monopolization by specific tasks.

Compared with classical load balancing methods such as Round Robin, Least Connection, or IP Hash, our virtual environment allocation approach offers several advantages:

- **Context-Aware Allocation**: Unlike traditional methods that consider only server load, our approach considers the specific data locality needs of extraction tasks, placing them closer to the data they need to access.
- **Resource Isolation**: By allocating dedicated virtual environments, we prevent resource contention between different extraction tasks, which is a common issue in shared-resource approaches.
- **Granular Resource Control**: Our system can allocate precisely the resources needed for each extraction task, rather than assigning entire nodes, leading to better overall resource utilization.
- **Adaptive Scaling**: The virtual environment approach allows for more flexible scaling of resources during task execution, expanding or contracting based on real-time needs.

Experimental results show that our virtual environmental allocation approach achieves 30% better resource utilization, and 25% lower task completion times compared to traditional round-robin load balancing when tested on complex data extraction workloads involving heterogeneous data sources.

3 Conclusion

This work systematically examines various aspects of the Big Data domain and highlights the key modern technologies applied in this field. With the evolution of business requirements, particularly in the sphere of international collaborative experiments, it is evident that current systems are unable to fully satisfy organizations' needs for data integration to obtain a complete description of their operational environment.

To address this challenge, this work proposes how existing technologies should be organized to achieve data-mesh architecture. This structure is based on virtualization technologies and distributed computing methods to achieve seamless integration of data from heterogeneous data domains, each representing fully-fledged storage systems.

The data-mesh architecture represents a logical continuation of all the systems currently used by businesses today. It enables solutions to new types of problems emerging in cutting-edge scientific fields such as artificial intelligence. By virtualizing the entire data infrastructure and providing unified access points, data-mesh architecture eliminates the traditional silos that have hindered comprehensive data analysis and machine learning applications.

Furthermore, this approach offers significant advantages in terms of scalability, fault tolerance, and resource utilization efficiency. The implementation of containerization and orchestration technologies such as Docker and Kubernetes provide the flexibility needed to adapt to changing data processing requirements while maintaining system stability.

The reinforcement learning mechanisms integrated into the system allow for continuous optimization of data placement and resource allocation, ensuring that the system evolves to meet changing workloads and access patterns. This adaptive capability is particularly crucial in environments where data volumes and processing demands fluctuate significantly.

In conclusion, the data-mesh architecture proposed in this work represents a significant advancement in big data systems design, particularly for organizations dealing with heterogeneous data sources and complex analytical requirements. By building upon established technologies while introducing novel integration approaches, this architecture paves the way for more comprehensive, efficient, and adaptable data processing ecosystems.

References

1. Bogdanov, A., Degtyarev, A., Shchegoleva, N., Khvatov, V., Korkhov, V.: Evolving principles of big data virtualization. In: Gervasi, O., et al. Computational Science and Its Applications – ICCSA 2020. ICCSA 2020. LNCS, vol. 12254 (2020)
2. Bogdanov, A., Degtyarev, A., Shchegoleva, N., Khvatov, V., Korkhov, V.: Big Data Virtualization: Why and How? CEUR Workshop Proceedings, vol 2679. ISSN **11–21**, 1613–0073 (2020)

3. Bogdanov, A., Ulitina, I., Lwin, T.K., Shchegoleva, N.: The impact of big data on the choice of storage. In: 2019 Computer Science and Information Technologies (CSIT), Yerevan, Armenia, 2019, pp. 33-36.https://doi.org/10.1109/CSITechnol.2019.8895173
4. Sissakian, A.: The MultiPurpose Detector – MPD to study Heavy Ion Collisions at NICA, pp. 185–188 (2023)
5. Vengerov, D.: A reinforcement learning framework for online data migration in hierarchical storage systems. J. Supercomput. **43**, 1–12 (2008)
6. Bogdanov, A.V., Shchegoleva, N.L., Ulitina, I.V.: Database ecosystem is the way to data lakes. In: Proceedings of the 27th Symposium on Nuclear Electronics and Computing (NEC 2019). Korenkov, V., Strizh, T., Nechaevskiy, A., Zaikina, T. (editor). RWTH Aahen University, pp. 147–152 (CEUR Workshop Proceedings ; 2507)
7. Gankevich I., et al.: Constructing Virtual Private Supercomputer Using Virtualization and Cloud Technologies. Computational Science and its Applications – ICCSA 2014 8584, pp. 341–354 (2008)
8. Bogdanov A., Degtyarev A., Shchegoleva N., Khvatov V.: Data quality in a decentralized environment. In: Gervasi O. et al. (eds) Computational Science and its Applications – ICCSA 2020. ICCSA 2020. Lecture Notes in Computer Science, vol 12251. Springer, Cham (2020)
9. ISO/IEC 25024:2015, Systems and software engineering – Systems and software Quality Requirements and Evaluation (SQuaRE) – Measurement of data quality. (2015)
10. ISO/IEC 25012:2008, Software engineering – Software product Quality Requirements and Evaluation (SQuaRE) – Data quality model. (2008)