Indonesian Journal of Computer Science
ISSN 2549-7286 (*online*)
Jln. Khatib Sulaiman Dalam No. 1, Padang, Indonesia
Website: ijcs.stmikindonesia.ac.id | E-mail: ijcs@stmikindonesia.ac.id

Distributed Graph Processing in Cloud Computing: A Review of Large-Scale Graph Analytics

Diler Atrushi[1,3], Subhi R. M. Zeebaree[2]
diler.ahmed@auas.edu.krd, subhi.rafeeq@dpu.edu.krd
[1] Information Technology Department, Technical College of Informatics-Akre, Akre University for Applied Sciences, Duhok, Iraq.
[2] Energy Eng. Department, Technical College of Engineering, Duhok Polytechnic University, Duhok, Iraq.
[3] Department of Computer Science, University of Duhok, Duhok, Kurdistan Region, Iraq.

Article Information

Submitted : 7 Mar 2024
Reviewed: 20 Mar 2024
Accepted : 1 Apr 2024

Keywords

Distributed Graph, Graph Processing, Cloud Computing, Distributed Systems, Large-Scale Graph.

Abstract

The rapid growth of graph data in various domains has propelled the need for efficient distributed graph processing techniques in cloud computing environments. This paper presents a comprehensive review of distributed graph processing for graph analytics of massive size in the context of cloud computing. The paper begins by highlighting the challenges associated with distributed graph processing, including load balancing, communication overhead, scalability, and partitioning strategies. It provides an overview of existing frameworks and tools specifically designed for distributed graph processing in cloud environments. Furthermore, the review encompasses various techniques and algorithms employed in distributed graph processing. The paper also reviews recent research advancements in optimizing distributed graph processing in cloud computing. To provide practical insights, the paper presents a comparative analysis of representative large-scale graph analytics applications implemented on different cloud computing platforms. Performance, scalability, and efficiency metrics are evaluated under varying workload sizes and graph characteristics. Overall, this comprehensive review paper serves as a highly prized asset for researchers and large-scale graph analytics professionals who are practitioners in the field. It provides a holistic understanding of the state-of-the-art distributed graph processing techniques in cloud computing and guides future research efforts towards more efficient and scalable graph processing in cloud environments.

A. Introduction

Cloud computing is a web-based platform that allows for the utilization of software, data, and resources from any location on the Internet. The recent evolution from cloud computing to hosting and delivering internet applications is a modern example[1], [2], [3]. Big data analytics and visualization have become integral in response to the exponential growth of data from computers, social media, and mobile devices. This transformation is acknowledged in the literature on big data challenges and applications[4]. Visualization, the graphical representation of facts, is crucial for interpreting and gaining deeper insights from large datasets. Scholars emphasize the formal interpretation of data visualization as a necessity in assessing and extracting meaningful insights from complex data. The role of data visualization extends to facilitating the consolidation of diverse data points, enhancing comprehension of data relationships, enabling real-time problem discussion, and identifying key analysis focal points[5]. Distributed Graph Processing in Cloud Computing is a pivotal paradigm for analyzing complex relationships and patterns within massive datasets. In the era of big data, the large size of graphs, such as billions of edges, and the complexity of graph computing provide substantial obstacles to computer systems and architecture. graph analytics has become a significant method for comprehending the connections between diverse forms of data. This enables data analysts to extract key insights from the patterns, benefiting various real-world applications including fraud detection[6], Tasks related to the field of machine learning[7], signal processing[8], social media content processing [9] Natural Language Processing (NLP) [10], [11], large-scale graph visualizations[12] and many other increasing fields.

Processing large graphs in a distributed manner is generally difficult because of their size and the inherent irregular structure of graph computations[13]. Enormous graphs may exceed the memory limit of a solo system; and even if they can be accommodated, the performance will be limited by the quantity of processor cores. Furthermore, real-world graphs often exhibit sparsity and are stored in compressed formats, which presents difficulties for traditional memory hierarchies. Graph algorithms sometimes suffer from poor locality as a result of random accesses when updating neighboring nodes, and they often require high memory bandwidth due to the little amount of computation performed between these random accesses[14]. The fundamental elements of a distributed system are illustrated in Figure 1.

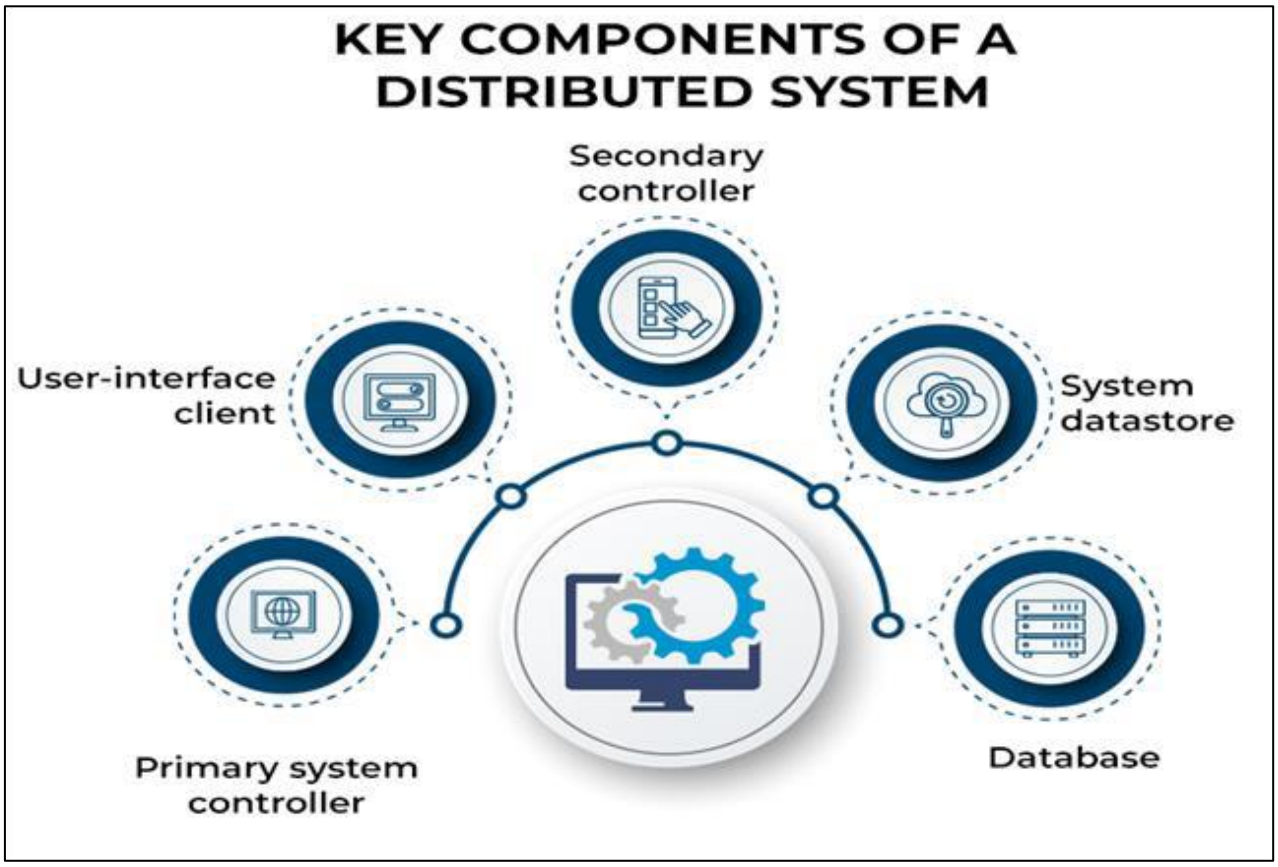

Figure 1. Distributed system components[15]

B. Background Theory

Distributed graph processing in cloud computing refers to the study and manipulation of extensive graphs by distributing computational activities among numerous machines inside a cloud environment. This section presents a theoretical foundation on the fundamental principles and methodologies employed in distributed graph processing.

Graph processing involves performing computations on graphs in order to uncover significant insights and patterns. Graphs are composed of vertices, also known as nodes, and edges, which reflect the links or connections between the vertices. Graph processing is able to be classified into two primary types: graph traversal and graph analytics. Graph traversal entails traveling the structure of a graph to uncover interactions and investigate associated components, whereas graph analytics concentrates on obtaining more advanced information from the graph, such as community detection, centrality analysis, and graph clustering[16].

Various methodologies have been devised to tackle the difficulties associated with distributed graph processing. The Bulk Synchronous Parallel (BSP) paradigm offers a conceptual foundation for creating algorithms that process graphs in a distributed manner. The computation is partitioned into supersteps, wherein each superstep comprises a calculation phase followed by a synchronization phase. The BSP approach facilitates fault tolerance, load balancing, and scalability in distributed graph processing by guaranteeing that all machines achieve synchronization points before advancing to the next superstep[17].

Graph processing frameworks offer conceptual models and programming interfaces that streamline the creation of scalable graph algorithms. These frameworks manage the fundamental aspects of distributed computation, fault tolerance, and data partitioning. Notable graph processing frameworks include Apache Giraph, Apache Flink, and GraphX. These frameworks offer advanced programming interfaces for defining graph algorithms and automatically manage the distribution of computations across the cluster[18], [19].

Cloud computing technologies, such as Amazon Web Services (AWS), Google Cloud Platform (GCP), and Microsoft Azure, play a crucial role in enabling distributed graph processing. They offer the essential framework, flexible scalability, and resilient resources needed for handling extensive graphs. Cloud platforms provide cost-effective and scalable services such as virtual computers, storage systems, and data processing frameworks that enable distributed graph processing[20], [21].

C. Distributed Systems

Nowadays, the amount of data to process is beyond of the capability to be processed on a multicore single system[22]. The internet and distributed systems are experiencing a growing redundancy. Typically, the combined servers contain approximately 4 petabytes of data. The technologies intricately handle this vast amount of data in an effective manner. The data is stored in multiple distributed devices and can be accessed using parallel processing[23], [24], [25]. Distributed systems enable multiple clients to access a shared computing resource, facilitating resource sharing. Examples of distributed computing include air traffic control, online railway reservation systems, and internet banking[26]. Distributed systems are crucial in the current technological environment, enabling smooth communication and collaboration across interconnected devices and services[27]. Distributed systems are extensively employed in modern applications for many objectives. According to Van Steen [28], they play a crucial role in cloud computing by facilitating the efficient and scalable distribution of resources among several computers. Online education has been improved by the use of distributed computing, which has enhanced eLearning experiences and made better use of resources[29]. The essential components of a distributed system are shown in Figure 1.

D. Graph Processing

Graph processing has become essential in the current technological environment, with applications spanning various disciplines. The authors in [30] emphasizes that social networks utilize graph processing for the purpose of modeling and analysis. Extracting information from graphs, such as those found in social networks or other contexts, typically requires global processing, which can be accomplished using several techniques [31]. Graphs are widely used in computer science as models for many structures found in nature and created by humans, highlighting the extensive range of applications for graph theory[32]. Sakr highlights the importance of massive graph processing in data centers, which is in line with the widely accepted reference architecture, as addressed by the community. Future systems are expected to offer highly scalable solutions for graph processing, recognizing graphs as a fundamental abstraction in contemporary data pipelines[33].

E. Cloud Computing

The theoretical implementation of cloud computing involves the integration of virtualization, service models, and deployment methods[34]. Cloud computing has become a fundamental aspect of contemporary technology, providing unparalleled adaptability and expandability[35], [36]. The discipline is characterized by its dynamic nature, as evidenced by recent advances. The projected significant changes in 2024, as described by in[37], highlight the way cloud computing is reforming company operations and IT strategy, emphasizing its crucial position in the digital future. The authors in[38], [39] examine pioneering themes, including the democratization of innovation through AI-as-a-service, environmentally friendly efforts, and the use of edge computing. These patterns highlight the continuous development of cloud computing, placing it at the forefront of technological progress. Cloud architecture refers to the integration of diverse technological elements that constitute a cloud system. Typically, this entails utilizing virtualization technology to consolidate several resources and distribute them across a network. The services provided by the cloud are controlled by the cloud operating system[40]. Figure 2 demonstrate the architecture of cloud computing.

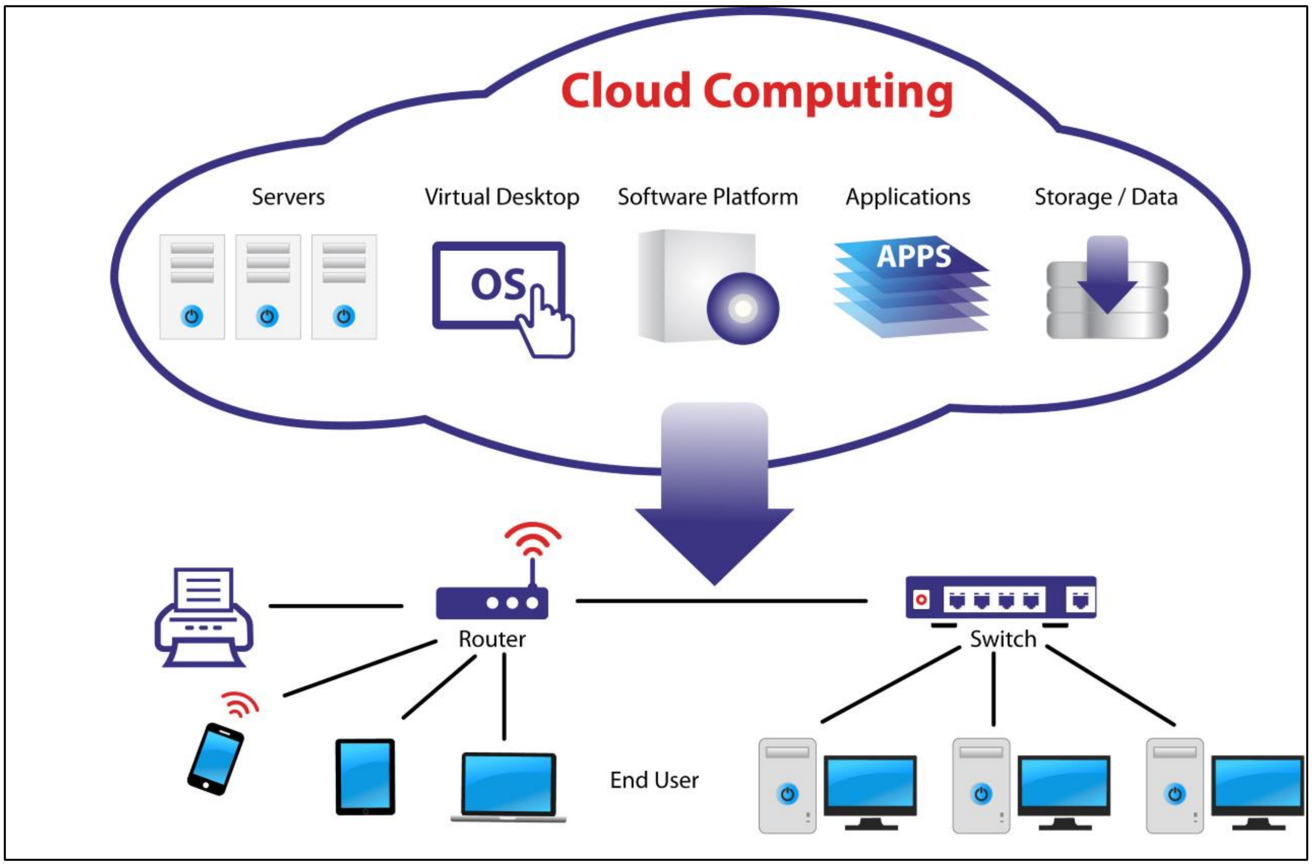

Figure 2. Cloud computing architecture[41]

F. Large-Scale Graph

The rapid growth of the Internet has led to a substantial increase in the quantity of electronic data. The act of categorizing these materials into coherent groups has become imperative. The vast amount of web pages across several domains poses a challenge for consumers to efficiently navigate and locate pertinent information[42]. From that view point, Large-scale graph analysis has proven crucial in extracting relevant insights across several areas. Coimbra et al.[43] examine the prospects and difficulties linked to large-scale data-intensive computing for social network analysis, genomics, and security[44], [45], [46] applications. This highlights the adaptability of extensive graphs in tackling intricate problems and extracting significant patterns. The importance of large-scale graphs is apparent in the domain of big data. Sakr in[33] underscores the prevalent framework of data centers, emphasizing the alignment of numerous graph processing ecosystems with this configuration. This insight is essential for the ability of massive graph processing to handle large amounts of data and be used in different scenarios that require a lot of data. Majeed's[30] review of graph theory highlights its potential applicability in computer science, providing distinct solutions throughout the subject. The study[47] offers a crucial analysis of graph data science, specifically highlighting the significance of graph visualization and its contribution to the examination of various graph categories. The ongoing significance of large-scale graphs in data-processing pipelines, offering extremely scalable solutions for modern applications[33].

G. Challenges

The task of processing graphs in a distributed environment inside a cloud computing has certain difficulties, as indicated by research investigations. The challenges include

- Scalability: As graph sizes continue to grow, ensuring scalability in distributed graph processing becomes crucial. Algorithms and techniques should be designed to handle graphs with billions or even trillions of vertices and edges, while efficiently utilizing the available computing resources [14], [48].
- Load balancing: Balancing the computational workload across distributed nodes in a cloud environment is a challenge in distributed graph processing. Unequal distribution of graph data and computations can lead to performance bottlenecks and inefficient resource utilization[49].
- Partitioning strategies: Effectively partitioning a graph across distributed nodes is essential for load balancing and minimizing communication overhead. Choosing appropriate partitioning strategies based on graph properties, such as vertex connectivity and data locality, is a challenge in distributed graph processing[50].
- Fault tolerance: Distributed graph processing systems should be resilient to node failures or network disruptions. Ensuring fault tolerance and fault recovery mechanisms to handle failures and maintain the consistency of graph computations is a challenge in cloud-based graph processing[51].
- Irregular Memory Access Patterns[52].
- Network Overheads and Bottlenecks[53].
- Performance Optimization Issues [54].
- Efficiency and Programming Flexibility [55].

- The Complexity in Distributed Visualization Algorithms [56].

H. Related works

In their study, [57] the author presents a distributed and parallel technique utilizing the MapReduce architecture to identify 2-Edge Connected Components (2-ECCs) in extensive graphs. The paper emphasizes the constraints of current single-node algorithms for graph analysis, which are inadequate for handling enormous graphs including billions of edges and vertices. The BiECCA algorithm tackles this difficulty by using the parallel and distributed capabilities of MapReduce, facilitating accelerated processing of large-scale graphs and facilitating the handling of stream data. The paper's contributions encompass the design and architecture of the proposed algorithm, the implementation of five distinct MapReduce tasks in a cascaded manner, and a comprehensive analysis of the algorithm's temporal complexity. The findings and assessments are offered in relation to the quantity of vertices and edges in comparison to the duration required for locating 2-ECCs. Furthermore, the study proposes innovative concepts for expanding upon the research.

The paper titled "Outsourced Analysis of Encrypted Graphs in the Cloud with Privacy Protection" addresses the challenge of securely analyzing large graphs in the cloud while maintaining privacy [58]. The authors propose cryptographic techniques for protecting the privacy of outsourced graph data and present two encryption algorithms: additional substance homomorphic encryption (ASHE) and some degree homomorphic encryption (SDHE). The primary aim of the study is to develop security-preserving methods for essential graph analysis tasks, specifically extraterrestrial examination of graphs outsourced to the cloud server. The authors focus on addressing the accountability and protection concerns associated with cloud-based graph storage and analysis. The paper highlights the importance of cloud computing in handling extensive graph data due to its processing capacity and cost-saving benefits. The results suggest that SDHE-based strategies perform well in reducing computing time, while ASHE-based methods are more efficient in reducing storage costs.

The study [59] introduces the concept of Graph Processing-as-a-Service (GPaaS) for large-scale graph processing in cloud computing environments. The aim of the paper is to develop a graph processing framework that takes into account quality of service (QoS) requirements and efficiently provisions resources to minimize monetary costs and execution time. The GPaaS framework considers service level agreements (SLAs) and QoS requirements to provision the appropriate combination of resources. The authors emphasize the importance of considering monetary costs and the heterogeneity of cloud resources in graph processing. They also address challenges specific to cloud environments, such as limited resources, time limitations, and dynamic network metrics.

In a study conducted by [60] This tackles the problem of communication bottleneck in distributed graph processing systems, which occurs when a significant amount of messages are exchanged between servers during calculations. The objective is to present a coded computing framework that utilizes computation redundancy to decrease the amount of communication required in the processing of large-scale graphs. The authors propose a new coding method that adds structured redundancy during the calculation phase. This allows for coded multicasting possibilities during message exchange and leads to a significant improvement in performance. The proposed framework expands on the graph-based MapReduce technique and presents a mathematical model for decomposing graph computation jobs in MapReduce. The calculation is partitioned into distinct Map and Reduce steps. During the Map phase, every server calculates intermediary values for the vertices within its assigned subgraph. During the Shuffle phase, servers share the intermediate values that are needed to execute the Reduce jobs. During the Reduce phase, each server performs the designated computations utilizing local and received intermediate data. The authors conduct actual experiments where they apply the PageRank algorithm on simulated and real-world datasets using Amazon EC2. The results exhibit substantial advancements, showcasing an improvement of up to 50.8% when compared to the usual application of PageRank.

The authors in [61] introduces the concept of edge computing, an extension of cloud computing, which aims to provide low-latency computing capabilities to users by deploying edge servers at base stations. The paper presents two approaches to tackle the CEDC problem. First, an optimal approach Constrained Edge Data Caching (CEDC) called CEDC-IP is introduced, which utilizes Integer Programming techniques to solve the problem exactly. Second, an approximation algorithm named CEDC-A is proposed to efficiently find approximate solutions for large-scale CEDC problems. The approximation ratio of CEDC-A is also proven. The results indicate that both CEDC-IP and CEDC-A beat the other approaches in terms of benefit per cache cost and serviced request ratio per cache cost.

As processing large graphs in the cloud environment is a challenging process. In a study [62] that proposed framework incorporates a network performance-aware partitioning the graph method. To capture the roughness of the network bandwidth, the machines selected for graph processing are modeled as a complete undirected graph. The framework recursively partitions both the data graph and the machine graph, ensuring that the number of cross-partition edges aligns with the aggregated bandwidth among machine graph partitions. Hierarchical combination techniques are employed to exploit data locality and improve network performance. The authors developed a system prototype called Surfer based on the Pregel graph processing engine. Experimental evaluations were conducted using a real-world social network and synthetic graphs exceeding 100GB each. The results on a local cluster demonstrated that the proposed partitioning scheme improved partitioning performance by 39-55% and graph processing by 6-71% under different network topologies. The optimizations reduced network traffic by 30-95% and total execution time by 30-85%. Furthermore, experiments on Amazon EC2 showed an average reduction of 49% in total execution time.

The paper [63] addresses the challenges of scaling Graph Neural Networks (GNNs) for large real-world graphs in a distributed setting. the scale of real-world graphs, often consisting of billions of nodes and edges, poses challenges for model training. The aim of the paper is to address the scalability challenges in training GNNs on large graphs in a distributed fashion. Existing frameworks for GNN training are limited to single machine multi-GPU setups and smaller graph sizes. The paper proposes P3 as a solution to enable efficient distributed training of GNNs on large input graphs. P3 aims to reduce network communication inefficiencies, utilize GPUs effectively, and outperform existing state-of-the-art distributed GNN frameworks.

Trinity, is a distributed graph engine that is introduced by [64]designed to address the challenges of large graph computation. Graph algorithms require random data access, which is not efficiently provided by disk technology. Memory-based methods are constrained by their lack of scalability. Trinity's objective is to enhance memory management and network connection in order to facilitate rapid graph exploration and effective graph parallel computing. Additionally, it offers a specialized specification language called TSL, which simplifies the maintenance and computation of graphs. Trinity's objective is to offer a versatile graph engine that facilitates both real-time query processing and offline graph analytics. It specifically targets the challenges of graph computation, such as the high ratio of data access to computation and the need for random data access. Trinity employs a distributed memory storage technology that enables globally accessible distributed memory for performing large-scale graph computations. It utilizes enhanced memory management and network connection to optimize performance. The system employs a graph parallel computing methodology and enhances access patterns for both online and offline computation. Experiments carried out on Trinity showcase its ability to perform well in both quick graph searches and efficient graph analysis on large-scale graphs with billions of nodes. Trinity has effectively been implemented in practical scenarios, including knowledge bases, knowledge graphs, and social networks. The report conducts a comparative analysis of Trinity in relation to other prominent graph systems, emphasizing its superior qualities and benefits. Current systems often prioritize either online transaction processing (OLTP) or offline analytics with high latency and high throughput. However, Trinity is designed to effectively handle both scenarios by utilizing scalable memory-based computation.

ShenTu is a graph processing framework that is designed general-purpose that is introduced by [65] and can effectively handle the challenges posed by large-scale graphs, such as the imbalanced load, lack of locality, and irregularity in access. ShenTu incorporates four key innovations to achieve its extraordinary performance and the ability to scale. First, it utilizes hardware specialization to select the best computational element and memory for individually task. Furthermore, super node routing adjusts global communication to the specific structure of the machine's topology. Third, on-chip sorting maps local message to manycore processors. Finally, degree-aware messaging selects the most suitable communication scheme based on vertex properties, such as degree. ShenTu is capable of efficiently dealing with a graph with 70 trillion edges and analyzing a 12 trillion-edge Internet graph for spam detection in a matter of seconds.

There are many challenges faced in graph processing, which involves understanding relationships in large datasets. Conventional architectures suffer from poor locality, high memory bandwidth requirements, and energy consumption. To overcome these challenges, the study [66]proposes a novel approach that leverages ReRAM (Resistive Random Access Memory) as a hardware building block for graph processing acceleration. The goal of the paper is to introduce GRAPHR, as the primary accelerator of graph processing that is ReRAM-based. The system adheres to the concept of near-data processing and aims to perform highly efficient parallel analog operations with minimal hardware and energy requirements. GRAPHR consists of two main components: memory ReRAM and a graph engine (GE). The core graph computations are performed using ReRAM crossbars, which enable efficient sparse matrix-vector multiplication (SpMV). This approach allows for a higher computation-to-data movement ratio, increased parallelism, and reduced energy waste due to sparsity. The authors demonstrate that ReRAM-based computation can be utilized in a broad variety of contexts of graph algorithms. The experimental results show that GRAPHR outperforms CPU and GPU baselines in terms of speedup and energy efficiency. GRAPHR outperforms a CPU baseline system by achieving a speedup of increase to 132.67 times and an energy saving of 33.82 times on the geometric mean. GRAPHR outperforms GPUs with a speedup ranging from 1.69 to 2.19 times and consumes significantly less energy, ranging from 4.77 to 8.91 times less. In addition, GRAPHR exhibits a performance improvement ranging from 1.16 to 4.12 times and is significantly more energy-efficient, with a range of 3.67 to 10.96 times, compared to a PIM-based design.

The authors in [67] addresses the need for performing real-time analytics on evolving graphs to extract value from big data. The purpose of the study is to design a unified graph data store that supports both batch and stream analytics on evolving graphs. The key objectives include efficient data access, concurrent execution of diverse real-time analytics, high ingestion rate, and data consistency. GraphOne combines edge list and adjacency list storage formats to leverage their respective advantages. It introduces a new data abstraction called GraphView, enabling data access at different granularities of data ingestion. The system employs dual versioning to decouple graph computations from updates, ensuring data consistency during concurrent processing. Experimental evaluations compare GraphOne with state-of-the-art graph systems, demonstrating its superior performance in ingestion rate, batch analytics (e.g., BFS and PageRank), and stream analytics (e.g., streaming BFS). The experimental results show that GraphOne outperforms existing dynamic graph systems in terms of ingestion rate, achieving an average speed up of 11.40× against LLAMA and 5.36× against Stinger.

The paper [68]addresses the challenge of graph partitioning in distributed graph processing applications. Graph partitioning involves dividing a large graph into subgraphs to be processed by distributed systems. The paper introduces CUTTANA, a streaming graph partitioner for the purpose of partition massive graphs with high quality compared to existing streaming solutions. It aims to reduce workload execution time, worker imbalance, and network overhead. The paper also focuses on evaluating the performance of CUTTANA in distributed graph analytics and databases, comparing it with other partitioning methods. It uses a scalable coarsening and refinement technique to improve the intermediate assignment made by a streaming partitioner. The buffering approach avoids storing the entire graph in memory while ensuring sufficient data for accurate partitioning decisions. The coarsening and refinement strategy efficiently identifies and implements the best moves to enhance partitioning quality. Additionally, a parallel implementation of CUTTANA is provided to achieve rapid partitioning speed. Experimental analysis demonstrates that CUTTANA consistently outperforms existing streaming vertex partitioners in terms of both edge-cut and communication volume metrics. It exhibits better partitioning quality, resulting in improved runtime performance in graph analytics applications (up to 59% compared to various streaming partitioners) and higher query throughput in graph databases (up to 23% improvement over the best existing partitioner). CUTTANA also addresses the worker imbalance issue observed in edge-cut partitioners.

The large-scale graphs are computationally expensive. K-Path centrality quantifies the transmission of information within a graph along direct channels having a maximum length of K. the researchers of[69] Presenting a novel technique, known as the random neighbor traversal graph (RaNT-Graph), for enhancing the calculation of K-Path centrality. The RaNT-Graph is a decentralized data structure for graphs that integrates vertex delegation splitting and rejection sampling algorithms. The objective is to facilitate the selection of a vast number of random walks and paths in extensive scale-free graphs. The RaNT-Graph technique employs vertex delegation partitioning to evenly distribute compute, communication, and storage across processors. Vertices with a high degree or hubs are divided, and their lists of adjacent vertices are disseminated to all processors. This aids in mitigating the disparity in computational capacity. In addition, rejection sampling is used to effectively sample random pathways. Rejection sampling is a method that chooses vertices that have not been visited yet, hence decreasing computing time by excluding vertices that have already been visited. The weak scaling trials showcase the effectiveness of RaNT-Graph on R-MAT graphs, but the strong scaling studies exhibit a substantial improvement in speed compared to the baseline 1D partitioned version. RaNT-Graph demonstrated a significant acceleration of 56,544 times when predicting K-Path centrality in an experiment conducted on a graph containing 89 million vertices and 1.9 billion edges.Efficient graph processing in various domains such as bioinformatics, social networking, and web analysis is very important. A study [70] that discuss the existing frameworks, including vertex programming and sparse linear algebra approaches, and identify the key building block operations, SpMSpV and SpGEMM, for expressing graph computations. The authors proposed the development of GraphPad, a high-performance framework for generalized SpMSpV and SpGEMM primitives, and evaluate its scalability and performance compared to existing frameworks. It also investigates partitioning strategies and communication optimizations that are crucial for efficient graph processing. The authors implement four graph applications using GraphPad, which offers flexibility in accommodating different data layouts, partitioning strategies, and communication optimizations. They study real-world graphs with over a billion edges and synthetic graphs with up to 8 billion edges. The paper explores load balancing, communication optimizations, and different partitioning schemes to optimize performance. In result, the study demonstrate that GraphPad outperforms CombBLAS, a high-performing graph analytics framework, by up to 40 times in terms of performance. The scalability of GraphPad is shown on a scale of up to 64 nodes, and its performance is within 2 times of GraphMat, a high-performance graph framework, for four out of five benchmarks on a single node.

I. Discussion and Comparison

The papers analyze diverse obstacles and suggest creative strategies for handling extensive graph processing in various computational settings. The examined research employs several methodology and strategies, which are categorized into Aims, Techniques, and Results as in Table 1.

Table 1. A summary of the reviewed articles.

References	Aims	Model	Results
[57] 2023	Enables efficient processing of large graphs and facilitates real-time applications	proposed an algorithm, called BiECCA that is builds upon existing “Star Algorithm”	Time of finding 2-ECCs increases with an increase in the graph size.
[58] 2023	The primary aim of the study is to develop security methods for essential graph analysis tasks, specifically graphs outsourced to the cloud server	Two algorithms: Additional Substance Homomorphic Encryption (ASHE) and Some Degree Homomorphic Encryption (SDHE)	SDHE-based strategies perform well in reducing computing time, while ASHE-based methods are more efficient in reducing storage costs
[59] 2019	To develop a graph processing framework that takes into account quality of service (QoS) requirements and efficiently provisions resources to minimize	Used optimization techniques called dynamic auto-scaling algorithm. In addition to dynamic repartitioning approach and mapping strategy	GPaaS reduces the execution time by 10-15% compared to Giraph and significantly reduces monetary costs by more than 40% compared to both Giraph and PowerGraph

	monetary costs and execution time		
[60] 2020	leverages computation redundancy to reduce the communication load in large-scale graph processing	Proposed framework that is built upon the graph-based MapReduce approach and introduces a mathematical model for MapReduce decomposition of graph computation tasks	up to 50.8% improvement
[61] 2022	Provide low-latency computing capabilities to users by deploying edge servers at base stations	Proposed CEDC-IP and approximation algorithm named CEDC-A	The average advantage of CEDC-IP compared to other methods is 3.44% to 39.29%
[62] 2012	Develop a novel graph partitioning framework that enhances the network performance of graph partitioning	use two models namely partition sketch (used a multi-level graph partitioning algorithm) and machine graph (developed a network bandwidth aware)	performance improvement of 30–85% and reducing network traffic by 30–95%
[63] 2021	To address the scalability challenges in training GNNs on large graphs in a distributed fashion	Proposes P3 optimization method that will reduce network communication inefficiencies, utilize GPUs effectively	P3 outperforms state-of-the-art frameworks by up to 7 times
[64] 2013	provide a general-purpose graph engine that supports both online query processing and offline graph analytics	Introduced a distributed graph engine called Trinity	Optimizing memory, communication and improved performance
[65] 2018	Process massive graphs on supercomputers, allowing for timely and efficient analysis of complex systems	Combine four techniques (Hardware specialization, Super node routing, on chip sorting, and degree aware messaging)	Using a super computer it could process 12 trillion edges in 8.5 seconds
[66] 2017	Overcome poor locality, high memory bandwidth requirements, and energy consumption	Introduce GRAPHR, the primary accelerator of graph processing that depends on ReRAM.	Compared to CPU, GPUs and in memory processing, GRAPHR is faster and more energy saving

[67] 2020	Graph with efficient data access, concurrent execution of diverse real-time analytics, high ingestion rate, and data consistency.	Propose a graph data store called GraphOne	GraphOne achieves high performance, concurrent execution of diverse analytics, and efficient data access.
[68] 2023	Dividing a large graph into subgraphs to be processed by distributed systems	Introduces CUTTANA, a streaming graph partitioner	improved runtime performance by 59% and 23% higher query throughput
[71] 2023	To optimize the estimating K-Path centrality in large-scale graphs	A new approach called the random neighbor traversal graph (RaNT-Graph)	RaNT-Graph achieved a 56,544x speedup when guessing K-Path significance on a graph with 89 million vertices and 1.9 billion edges
[69] 2016	Optimize the implementations of graph analytics	Developing GraphPad, a set of optimized graph primitives	GraphPad outperforms CombBLAS by 40x performance, and GraphMat by 2x and Scalability up to 64 nodes

J. Extracted Statistics

The article reviewed are tackling variety of graph areas, the table below list the theme and the research that investigating it.

Table 2. Viewed themes references

Graph Processing Area	References
Graph Processing Frameworks	[59], [60], [62], [64]
Graph Optimization	[63], [69], [71]
Real-time Graph applications	[57], [67]
Partitioning of Large Graphs	[61]
Hardware-Based Approaches in Graph Processing	[65]
Graph Security	[58]

Furthermore, the pie chart illustrates the number of and percentage of the viewed articles.

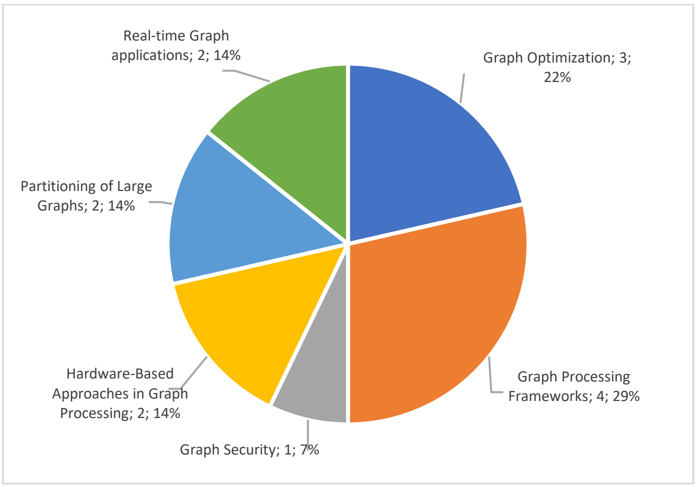

Figure 3: Viewed articles themes, number of and percentage

K. Recommendations

Based on a comprehensive review of the research studies cited in this article, the authors propose the following recommendations.

1. Development of Efficient Partitioning Strategies: Given the challenges associated with distributed graph processing, it is recommended to focus on the development of efficient partitioning strategies. These strategies should aim to balance the workload and minimize communication overhead, ensuring optimal performance in cloud computing environments.
2. Further Research on Load Balancing Techniques: Load balancing plays a crucial role in distributed graph processing. Future research efforts should focus on exploring and developing advanced load balancing techniques that can effectively distribute the computational workload across multiple machines, maximizing resource utilization and minimizing processing time.
3. Exploration of Optimization Techniques: To enhance the efficiency and scalability of distributed graph processing, it is recommended to explore and develop optimization techniques specifically tailored for cloud computing environments. These techniques may include algorithmic improvements, data compression methods, and memory management strategies to address the challenges posed by large-scale graphs.
4. Comparative Analysis of Graph Processing Frameworks: Conducting a comparative analysis of existing graph processing frameworks, such as Apache Giraph, Apache Flink, and GraphX, would provide valuable insights into their performance, scalability, and ease of use. This analysis can help researchers and practitioners in selecting the most suitable framework for their specific graph analytics applications.
5. Evaluation of Graph Processing on Different Cloud Computing Platforms: It is advisable to evaluate the performance, scalability, and efficiency metrics of representative large-scale graph analytics applications on different cloud computing platforms, such as Amazon Web Services (AWS), Google Cloud Platform (GCP), and Microsoft Azure. This evaluation can provide practical

insights into the strengths and weaknesses of each platform, enabling informed decision-making in terms of platform selection for distributed graph processing tasks.

6. Collaboration between Researchers and Practitioners: Given the complexity and evolving nature of distributed graph processing in cloud computing, collaboration between researchers and practitioners is essential. Close collaboration can facilitate the exchange of knowledge, ideas, and practical experiences, leading to the development of more efficient and scalable graph processing techniques.

L. Conclusion

This study has presented a comprehensive review of the challenges, techniques, frameworks, and advancements in distributed graph processing. The rapid growth of graph data in various domains has created a need for efficient graph processing techniques in cloud computing environments. The article highlights the challenges associated with distributed graph processing, including load balancing, communication overhead, scalability, and partitioning strategies. It provides an overview of existing frameworks and tools specifically designed for distributed graph processing in cloud environments. Various techniques and algorithms employed in distributed graph processing are discussed, and recent research advancements in optimizing distributed graph processing in cloud computing are reviewed. To provide practical insights, the article presents a comparative analysis of representative large-scale graph analytics applications implemented on different cloud computing platforms. Performance, scalability, and efficiency metrics are evaluated under varying workload sizes and graph characteristics. Overall, this comprehensive review serves as a valuable resource for researchers and practitioners in the field of large-scale graph analytics. It provides a holistic understanding of the state-of-the-art distributed graph processing techniques in cloud computing and guides future research efforts towards more efficient and scalable graph processing in cloud environments. The article emphasizes the importance of distributed graph processing in analyzing complex relationships and patterns within massive datasets. It highlights the significance of graph analytics in various real-world applications such as fraud detection, machine learning, signal processing, social media content processing, natural language processing, and large-scale graph visualizations. Furthermore, the article discusses the theoretical foundations and methodologies employed in distributed graph processing, including the Bulk Synchronous Parallel (BSP) paradigm and graph processing frameworks such as Apache Giraph, Apache Flink, and GraphX. It also emphasizes the role of cloud computing technologies in facilitating distributed graph processing by providing essential frameworks, scalability, and resilient resources. In summary, the article recognizes the challenges and opportunities in distributed graph processing for large-scale graph analytics in cloud computing. It provides a comprehensive overview of the current state-of-the-art, explores optimization techniques, and presents practical insights through comparative analysis. This review serves as a valuable guide for researchers and practitioners seeking to enhance the efficiency and scalability of graph processing in cloud environments.

M. References

[1] C. Mustafa Mohammed and S. R. M Zeebaree, “Sufficient Comparison Among Cloud Computing Services: IaaS, PaaS, and SaaS: A Review,” *International Journal of Science and Business*, vol. 5, no. 2, pp. 17–30, 2021, doi: 10.5281/zenodo.4450129.

[2] S. R. M. Zeebaree, A. B. Sallow, B. K. Hussan, and S. M. Ali, “Design and Simulation of High-Speed Parallel/Sequential Simplified DES Code Breaking Based on FPGA,” in *2019 International Conference on Advanced Science and Engineering (ICOASE)*, 2019, pp. 76–81. doi: 10.1109/ICOASE.2019.8723792.

[3] S. R. M. Zeebaree, “DES encryption and decryption algorithm implementation based on FPGA,” *Indonesian Journal of Electrical Engineering and Computer Science*, vol. 18, no. 2, pp. 774–781, 2020, doi: 10.11591/ijeecs.v18.i2.pp774-781.

[4] G. Andrienko *et al.*, “Big Data Visualization and Analytics: Future Research Challenges and Emerging Applications,” 2020. [Online]. Available: http://graphics.stanford.edu/projects/dataExtract

[5] A. and B. A. J. and M. A. and B.-M. A. Curry Edward and Metzger, “A Reference Model for Big Data Technologies,” in *The Elements of Big Data Value: Foundations of the Research and Innovation Ecosystem*, A. and Z. S. and P. J.-C. and G. R. A. Curry Edward and Metzger, Ed., Cham: Springer International Publishing, 2021, pp. 127–151. doi: 10.1007/978-3-030-68176-0_6.

[6] S. Srivastava and A. K. Singh, “Fraud detection in the distributed graph database,” *Cluster Comput*, vol. 26, no. 1, pp. 515–537, 2023, doi: 10.1007/s10586-022-03540-3.

[7] W. Xiao *et al.*, “Tux: Distributed Graph Computation for Machine Learning,” in *14th USENIX Symposium on Networked Systems Design and Implementation (NSDI 17)*, Boston, MA: USENIX Association, Mar. 2017, pp. 669–682. [Online]. Available: https://www.usenix.org/conference/nsdi17/technical-sessions/presentation/xiao

[8] E. Isufi, F. Gama, D. I. Shuman, and S. Segarra, “Graph Filters for Signal Processing and Machine Learning on Graphs,” Nov. 2022, [Online]. Available: http://arxiv.org/abs/2211.08854

[9] M. Ali, M. Hassan, K. Kifayat, J. Y. Kim, S. Hakak, and M. K. Khan, “Social media content classification and community detection using deep learning and graph analytics,” *Technol Forecast Soc Change*, vol. 188, p. 122252, 2023, doi: https://doi.org/10.1016/j.techfore.2022.122252.

[10] A. Goyal, H. D. Iii, and R. Guerra, “Fast Large-Scale Approximate Graph Construction for NLP,” Association for Computational Linguistics, 2012.

[11] A. Alexandrescu and K. Kirchhoff, “Data-Driven Graph Construction for Semi-Supervised Graph-Based Learning in NLP,” 2007.

[12] A. Arleo, W. Didimo, G. Liotta, and F. Montecchiani, “Large graph visualizations using a distributed computing platform,” *Inf Sci (N Y)*, vol. 381, pp. 124–141, 2017, doi: https://doi.org/10.1016/j.ins.2016.11.012.

[13] R. Elshawi, O. Batarfi, A. Fayoumi, A. Barnawi, and S. Sakr, “Big Graph Processing Systems: State-of-the-art and Open Challenges.” [Online]. Available: http://hama.apache.org/

[14] Y. Zhuo *et al.*, “Trinity: A Distributed Graph Engine on a Memory Cloud,” *ACM Transactions on Computer Systems*, vol. 37, no. 1–4, Jun. 2021, doi: 10.1145/3453681.

[15] Z. S. Ageed and S. R. M. Zeebaree, "Distributed Systems Meet Cloud Computing: A Review of Convergence and Integration," *Original Research Paper International Journal of Intelligent Systems and Applications in Engineering IJISAE*, vol. 2024, no. 11s, 2024, [Online]. Available: www.ijisae.org

[16] A. Rajaraman and J. D. Ullman, *Mining of massive datasets*. Cambridge University Press, 2011.

[17] L. G. Valiant, "A bridging model for parallel computation," *Commun ACM*, vol. 33, no. 8, pp. 103–111, 1990.

[18] Apache Software Foundation, "Apache Giraph", Accessed: Jan. 24, 2024. [Online]. Available: https://giraph.apache.org/intro.html

[19] J. E. Gonzalez, R. S. Xin, A. Dave, D. Crankshaw, M. J. Franklin, and I. Stoica, "{GraphX}: Graph processing in a distributed dataflow framework," in *11th USENIX symposium on operating systems design and implementation (OSDI 14)*, 2014, pp. 599–613.

[20] Google, "Google Cloud Platform." Accessed: Jan. 24, 2024. [Online]. Available: https://cloud.google.com/

[21] Amazon, "Amazon Web Services." Accessed: Jan. 24, 2024. [Online]. Available: https://aws.amazon.com/

[22] S. R. M. Zeebaree *et al.*, "Multicomputer Multicore System Influence on Maximum Multi-Processes Execution Time," *TEST Engineering & Management*, vol. 53, no. 03, pp. 14921–14931, 2020.

[23] S. R. M. Zeebaree, H. M. Shukur, L. M. Haji, R. R. Zebari, K. Jacksi, and S. M. Abas, "Characteristics and Analysis of Hadoop Distributed Systems," *Technology Reports of Kansai University*, vol. 62, no. 04, 2019.

[24] Z. N. Rashid, H. Sharif, S. Rafeeq, and M. Z. Sulaimani, "Client/Servers Clustering Effects on CPU Execution-Time, CPU Usage and CPU Idle Depending on Activities of Parallel-Processing-Technique Operations ","," *INTERNATIONAL JOURNAL OF SCIENTIFIC & TECHNOLOGY RESEARCH*, vol. 7, no. 8, 2018, [Online]. Available: www.ijstr.org

[25] L. M. Haji, S. R. M. Zeebaree, Z. S. Ageed, O. M. Ahmed, M. A. M. Sadeeq, and H. M. Shukur, "Performance Monitoring and Controlling of Multicore Shared-Memory Parallel Processing Systems," in *2022 3rd Information Technology To Enhance e-learning and Other Application (IT-ELA)*, IEEE, 2022, pp. 44–48. doi: 10.1109/IT-ELA57378.2022.10107953.

[26] H. Shukur, S. Zeebaree, R. Zebari, O. Ahmed, L. Haji, and D. Abdulqader, "Cache Coherence Protocols in Distributed Systems," *Journal of Applied Science and Technology Trends*, vol. 1, no. 3, pp. 92–97, Jun. 2020, doi: 10.38094/jastt1329.

[27] Y. S. Jghef, S. R. M. Zeebaree, Z. S. Ageed, and H. M. Shukur, "Performance Measurement of Distributed Systems via Single-Host Parallel Requesting using (Single, Multi and Pool) Threads," in *2022 3rd Information Technology To Enhance e-learning and Other Application (IT-ELA)*, 2022, pp. 38–43. doi: 10.1109/IT-ELA57378.2022.10107923.

[28] M. van Steen and A. S. Tanenbaum, "A brief introduction to distributed systems," *Computing*, vol. 98, no. 10, pp. 967–1009, 2016, doi: 10.1007/s00607-016-0508-7.

[29] S. Caballé, W. Li, R. Hoseiny, A. Zomaya, and F. Xhafa, "Applications of Distributed and High Performance Computing to Enhance Online Education," Jan. 2018, pp. 586–600. doi: 10.1007/978-3-319-69835-9_55.

[30] A. Majeed and I. Rauf, "Graph Theory: A Comprehensive Survey about Graph Theory Applications in Computer Science and Social Networks," *Inventions*, vol. 5, no. 1, 2020, doi: 10.3390/inventions5010010.

[31] M. E. Coimbra, A. P. Francisco, and L. Veiga, "An analysis of the graph processing landscape," *J Big Data*, vol. 8, no. 1, Dec. 2021, doi: 10.1186/s40537-021-00443-9.

[32] F. Riaz and K. Ali, "Applications of Graph Theory in Computer Science," Jan. 2011, pp. 142–145. doi: 10.1109/CICSyN.2011.40.

[33] S. Sakr *et al.*, "The future is big graphs: a community view on graph processing systems," *Commun. ACM*, vol. 64, no. 9, pp. 62–71, Aug. 2021, doi: 10.1145/3434642.

[34] H. M. Zangana and S. R. M. Zeebaree, "Distributed Systems for Artificial Intelligence in Cloud Computing: A Review of AI-Powered Applications and Services," *International Journal of Informatics, Information System and Computer Engineering (INJIISCOM)*, vol. 5, no. 1, pp. 1–20, Jan. 2024, [Online]. Available: https://ojs.unikom.ac.id/index.php/injiiscom/article/view/11883

[35] P. Y. Abdullah, S. R. M. Zeebaree, H. M. Shukur, and K. Jacksi, "HRM System using Cloud Computing for Small and Medium Enterprises (SMEs)," *Technology Reports of Kansai University*, vol. 62, no. 04, 2020.

[36] P. Y. Abdullah, S. R. M. Zeebaree, K. Jacksi, and R. R. Zeabri, "AN HRM SYSTEM FOR SMALL AND MEDIUM ENTERPRISES (SME)S BASED ON CLOUD COMPUTING TECHNOLOGY," *International Journal of Research -GRANTHAALAYAH*, vol. 8, no. 8, pp. 56–64, Aug. 2020, doi: 10.29121/granthaalayah.v8.i8.2020.926.

[37] Y. Liu, L. Wang, and X. Vincent Wang, "Cloud manufacturing: latest advancements and future trends," *Procedia Manuf*, vol. 25, pp. 62–73, 2018, doi: https://doi.org/10.1016/j.promfg.2018.06.058.

[38] M. Alam and K. Shakil, "Recent Developments in Cloud Based Systems: State of Art," Jan. 2015.

[39] Z. N. Rashid, S. R. M. Zeebaree, and A. Sengur, "Novel Remote Parallel Processing Code-Breaker System via Cloud Computing," *TRKU*, vol. 62, no. 04, 2020.

[40] H. Malallah *et al.*, "A Comprehensive Study of Kernel (Issues and Concepts) in Different Operating Systems," *Asian Journal of Research in Computer Science*, vol. 8, no. 3, pp. 16–31, May 2021, doi: 10.9734/ajrcos/2021/v8i330201.

[41] A. Maier, "Understanding the fundamentals of a Cloud Computing Architecture," Code Coda. Accessed: Jan. 24, 2024. [Online]. Available: https://codecoda.com/en/blog/entry/understanding-the-fundamentals-of-a-cloud-computing-architecture#

[42] R. K. Ibrahim *et al.*, "Clustering Document based on Semantic Similarity Using Graph Base Spectral Algorithm," in *2022 5th International Conference on Engineering Technology and its Applications (IICETA)*, 2022, pp. 254–259. doi: 10.1109/IICETA54559.2022.9888613.

[43] D. Bader, H. Meyerhenke, and J. Riedy, "Applications and Challenges in Large-scale Graph Analysis," Jan. 2013.

[44] S. R. M Zeebaree, R. R. Zebari, K. Jacksi, and D. Abas Hasan, "Security Approaches For Integrated Enterprise Systems Performance: A Review," *INTERNATIONAL JOURNAL OF SCIENTIFIC & TECHNOLOGY RESEARCH*, vol. 8, 2019, [Online]. Available: www.ijstr.org

[45] T. Mohammed, G. Sami, S. R. M. Zeebaree, and S. H. Ahmed, "Designing a New Hashing Algorithm for Enhancing IoT Devices Security and Energy Management,"

International Journal of Intelligent Systems and Applications in Engineering, vol. 12, no. 4s, pp. 202–215, 2024, [Online]. Available: www.ijisae.org

[46] T. Mohammed, G. Sami, S. R. M. Zeebaree, and S. H. Ahmed, "A Novel Multi-Level Hashing Algorithm to Enhance Internet of Things Devices' and Networks' Security," *Original Research Paper International Journal of Intelligent Systems and Applications in Engineering IJISAE*, vol. 2024, no. 1s, pp. 676–696, 2024, [Online]. Available: www.ijisae.org

[47] R. Das and M. Soylu, "A key review on graph data science: The power of graphs in scientific studies," *Chemometrics and Intelligent Laboratory Systems*, vol. 240, p. 104896, 2023, doi: https://doi.org/10.1016/j.chemolab.2023.104896.

[48] M. Li *et al.*, "Scaling Distributed Machine Learning with the Parameter Server," in *11th USENIX Symposium on Operating Systems Design and Implementation (OSDI 14)*, Broomfield, CO: USENIX Association, Oct. 2014, pp. 583–598. [Online]. Available: https://www.usenix.org/conference/osdi14/technical-sessions/presentation/li_mu

[49] A. C. Facebook, H. Lane, S. Edunov, M. Kabiljo, and S. Muthukrishnan, "One Trillion Edges: Graph Processing at Facebook-Scale," 2150.

[50] G. Malewicz *et al.*, "Pregel: a system for large-scale graph processing," in *Proceedings of the 2010 ACM SIGMOD International Conference on Management of Data*, in SIGMOD '10. New York, NY, USA: Association for Computing Machinery, 2010, pp. 135–146. doi: 10.1145/1807167.1807184.

[51] J. Ekanayake *et al.*, "Twister: a runtime for iterative MapReduce," in *Proceedings of the 19th ACM International Symposium on High Performance Distributed Computing*, in HPDC '10. New York, NY, USA: Association for Computing Machinery, 2010, pp. 810–818. doi: 10.1145/1851476.1851593.

[52] A. Sahebi, M. Barbone, M. Procaccini, W. Luk, G. Gaydadjiev, and R. Giorgi, "Distributed large-scale graph processing on FPGAs," *J Big Data*, vol. 10, no. 1, p. 95, 2023, doi: 10.1186/s40537-023-00756-x.

[53] J. Malicevic, A. Roy, and W. Zwaenepoel, "Scale-up graph processing in the cloud: Challenges and solutions," Jan. 2014. doi: 10.1145/2592784.2592789.

[54] V. Kalavri, *Performance Optimization Techniques and Tools for Distributed Graph Processing*. 2016.

[55] R. Chen, X. Weng, B. He, and M. Yang, "Large graph processing in the cloud," in *Proceedings of the ACM SIGMOD International Conference on Management of Data*, Jan. 2010, pp. 1123–1126. doi: 10.1145/1807167.1807297.

[56] A. Arleo, W. Didimo, G. Liotta, and F. Montecchiani, "Large graph visualizations using a distributed computing platform," *Inf Sci (N Y)*, vol. 381, pp. 124–141, 2017, doi: https://doi.org/10.1016/j.ins.2016.11.012.

[57] D. Dahiphale, "MapReduce for Graphs Processing: New Big Data Algorithm for 2-Edge Connected Components and Future Ideas," *IEEE Access*, vol. 11, pp. 54986–55001, 2023, doi: 10.1109/ACCESS.2023.3281266.

[58] D. Selvaraj, S. M. U. Sankar, D. Dhinakaran, and T. P. Anish, "Outsourced Analysis of Encrypted Graphs in the Cloud with Privacy Protection," *SSRG International Journal of Electrical and Electronics Engineering*, vol. 10, no. 1, pp. 53–62, Jan. 2023, doi: 10.14445/23488379/IJEEE-V10I1P105.

[59] S. Heidari and R. Buyya, "Quality of Service (QoS)-driven resource provisioning for large-scale graph processing in cloud computing environments: Graph Processing-

as-a-Service (GPaaS)," *Future Generation Computer Systems*, vol. 96, pp. 490–501, Jul. 2019, doi: 10.1016/j.future.2019.02.048.

[60] S. Prakash, A. Reisizadeh, R. Pedarsani, and A. S. Avestimehr, "Coded Computing for Distributed Graph Analytics," *IEEE Trans Inf Theory*, vol. 66, no. 10, pp. 6534–6554, Oct. 2020, doi: 10.1109/TIT.2020.2999675.

[61] X. Xia, F. Chen, J. Grundy, M. Abdelrazek, H. Jin, and Q. He, "Constrained App Data Caching Over Edge Server Graphs in Edge Computing Environment," *IEEE Trans Serv Comput*, vol. 15, no. 5, pp. 2635–2647, 2022, doi: 10.1109/TSC.2021.3062017.

[62] R. Chen, X. Weng, B. He, M. Yang, B. Choi, and X. Li, *Improving Large Graph Processing on Partitioned Graphs in the Cloud*. 2012.

[63] S. Gandhi, A. P. Iyer, and P. Iyer, *P3: Distributed Deep Graph Learning at Scale*. 2021. [Online]. Available: https://www.usenix.org/conference/osdi21/presentation/gandhi

[64] B. Shao, H. Wang, and Y. Li, *Trinity: A Distributed Graph Engine on a Memory Cloud*. ACM, 2013.

[65] H. Lin *et al.*, "Shentu: processing multi-trillion edge graphs on millions of cores in seconds," *SC18: International Conference for High Performance Computing, Networking, Storage and Analysis*, pp. 706–716, 2018.

[66] L. Song, Y. Zhuo, X. Qian, H. Li, and Y. Chen, "GraphR: Accelerating Graph Processing Using ReRAM," Aug. 2017, [Online]. Available: http://arxiv.org/abs/1708.06248

[67] P. Kumar and H. Howie Huang, "Graphone: A data store for real-time analytics on evolving graphs," *ACM Transactions on Storage*, vol. 15, no. 4, Jan. 2020, doi: 10.1145/3364180.

[68] M. R. Hajidehi, S. Sridhar, and M. Seltzer, "CUTTANA: Scalable Graph Partitioning for Faster Distributed Graph Databases and Analytics," Dec. 2023, [Online]. Available: http://arxiv.org/abs/2312.08356

[69] L. Fletcher, T. Steil, and R. Pearce, "Optimizing a Distributed Graph Data Structure for K-Path Centrality Estimation on HPC," in *2023 IEEE High Performance Extreme Computing Conference (HPEC)*, 2023, pp. 1–7.

[70] M. J. Anderson, N. Sundaram, N. Satish, M. M. A. Patwary, T. L. Willke, and P. Dubey, "GraphPad: Optimized Graph Primitives for Parallel and Distributed Platforms," in *Proceedings - 2016 IEEE 30th International Parallel and Distributed Processing Symposium, IPDPS 2016*, Institute of Electrical and Electronics Engineers Inc., Jul. 2016, pp. 313–322. doi: 10.1109/IPDPS.2016.86.

[71] L. Fletcher, T. Steil, and R. Pearce, "Optimizing a Distributed Graph Data Structure for K-Path Centrality Estimation on HPC," 2023.