

Adaptive Graph-Based Framework for Secure and Efficient Distributed Data Processing

1st Sasi Kumar Kolla
 KLU University
 Computer Science
 sasikkolla@gmail.com

Abstract—Distributed data processing enables computation by distributing workload among multiple workers. By outsourcing data operations to distributed infrastructures such as Edge and Cloud, users enjoy advantages in privacy, fault tolerance, availability, latency, and resource management. However, research on the adaptivity of data processing is still in its infancy; as a result, existing distributed systems either require the data to be processed only once or are heavily restrictive on dynamic workloads. Graph-based representations of workloads offer potential support for designing adaptive distributive data processing systems.

Existing adaptive systems mainly focus either on ensuring privacy or on reducing latency across Edge-Cloud boundaries. Moreover, they do not consider the problem of throughput maximization under a given latency—an important factor in optimizing resource consumption. By employing the data processing task as a directed graph and expanding the task into two layers, a novel adaptive graph-based architecture is proposed for secure and efficient distributed data processing over dynamic workloads. Security constraints are taken into account during the adaptive scheduling, partitioning, replication, and routing of the graph.

Keywords—Adaptive Graph Processing, Distributed Data Processing, Graph-Based Framework, Secure Data Computation, Dynamic Graph Partitioning, Privacy-Preserving Data Analytics, Scalable Distributed Systems, Graph Neural Networks, Fault-Tolerant Computing, Resource-Aware Optimization.

1. INTRODUCTION

The evolution of cloud computing and distributed systems has seen dramatic increases in data generation and processing demands. Distributed data-processing systems have been developed to address these. Such systems rely on a large number of machines partitioned into groups and use a distributed file system to allow data to be shared during computation. The flexible nature of cloud services promotes cost-effective deployment, and advertising services can easily access large volumes of consumer data for advertisement recommendation and promotion. However, the centralized structure used for such systems limits reliability, leading to security concerns. Users place sensitive data in the cloud without considering the potential risks in management. Data protection techniques must, therefore, be applied. A recent technological trend for protecting sensitive data is Homomorphic Encryption (HE). HE enables ciphertexts to be processed directly while still maintaining security up to the results.

Data Encryption is performed using some shared key, and the corresponding decryption can be performed using the same key. A secure data-management scheme, the Data Encryption and Storage System (DESS), allows users to

encrypt data before storage and by using homogeneous encryption-based Public Key Cryptography to eliminate the need for private key sharing. Security and data protection, however, introduce additional costs and overhead for both processing and storage. The disadvantages of the above system can be overcome by speeding up the data-transmission process and improving security throughout the process.

Mathematical Formulas:

$$G = \langle V, E \rangle, (u, v) \in E \Leftrightarrow \text{data/control dependency } u \rightarrow v \quad (1)$$

$$t_{\text{finish}}(u) \leq t_{\text{start}}(v) \quad \forall (u, v) \in E \quad (\text{precedence constraint}) \quad (2)$$

$$L = \max_{v \in V} \{t_{\text{finish}}(v)\} \quad (\text{total system latency}) \quad (3)$$

$$T = \min_{i \in \text{Nodes}} \{r_i + w_i + d_i\} / L \quad (\text{throughput; } r, w, d = \text{read/write/dispose ops}) \quad (4)$$

$$\text{mode}^*(f) = \text{ASS}(f, S) \quad (\text{compute-mode chosen per function } f \text{ by security level } S) \quad (5)$$

$$C(S) = \alpha \cdot L(S) + \beta / S, \quad \alpha, \beta \geq 0 \quad (\text{illustrative security-latency cost objective}) \quad (6)$$

A. Motivation and Problem Statement

Distributed environments, such as cloud and edge computing, are increasingly used to store and process data generated by large-scale systems. In such a scenario, data processing involves multiple data owners who usually don't trust one another. Consequently, any distributed data processing system should provide security, efficiency, and scalability for these environments. Security means that the privacy of the data owned by each data owner should not be violated. Efficiency is related to minimizing the overall data communication and computation overhead incurred within the distributed environment. The communication overhead should be as low as possible to reduce the time required for data processing. The architecture should scale well with the increasing amount of data to be processed in the system. Current distributed data processing architectures do not provide adaptability according to the underlying data-processing environment or security in these environments for the distributed data processing frameworks.

Graph-based abstraction models are powerful and flexible for the representation of various systems. By formalizing the data and processing computation in processing, the data can be modeled as a directed acyclic graph (DAG) for secure processing on a distributed environment. A distributed data processing adaptive system on a DAG consists of both the data and the related set of computations that need to be performed on that data.

Table I. Notation Used in the Framework

Symbol	Meaning
$G = \langle V, E \rangle$	Directed graph modeling the data-processing task
V, E	Node set (data items / tasks) and directed-edge set (dependencies)
$G_{dep}(V_{dep}, E_{dep})$	Directed, labeled data-dependency graph
$G_{comp}(V_{comp}, E_{comp})$	Directed, labeled computation graph
ASS	Adaptive Secret Sharing scheme (per-function compute-mode selection)
L	Total latency of the distributed process
T	System throughput (min. required ops across nodes)
S	Security / protection level applied by ASS

2. Background and Related Work

Cloud computing and big data are two of the most alluring technologies of the 21st century that have attracted global attention. Cloud computing provides numerous advantages, such as low cost and high availability, while big data propels the inexorable trend of mass data generation in many fields, including social media, the Internet of Things, and video surveillance. Diverse applications generate massive amounts of data, posing significant technical challenges for data storage and processing. To meet these challenges, existing methods for distributed computing and storage of huge volumes of data employ two models: a distributed processing model with a storage system located elsewhere and a distributed cloud processing-model in which processing and storage are located in the same area. However, the way users organize data still remains largely unchanged. For centralized data, data processing becomes the bottleneck of development. A vastly parallel data processing model must be designed.

Suggestions that many users may wish to provide independent data processing services can be derived from the data processing ideas innovated for third-party services. User privacy is an important consideration when submitting user data to a third-party service provider. Users store their private

information on the cloud and only publish data that are not of their interest either privately or commercially. However, data are still accessible by the service provider, raising concerns among data owners about privacy leakage and potential exploitation. The need for more reliable support services through full decentralization cannot therefore be ignored. With these considerations in mind, a novel architecture for distributed data processing that is fully user-centric and supports mutual collaboration among user nodes is proposed.

Table II. Comparative Summary of Related Distributed Processing Approaches

System / Approach	Primary Focus	Security Mechanism	Adaptive?	Throughput-Aware?
MapReduce	Batch parallel processing	None (external)	No	No
Apache Spark	In-memory distributed processing	None (external)	Partial	No
DESS (HE-based)	Encrypted storage & retrieval	Homomorphic Encryption	No	No
Existing adaptive systems	Privacy OR latency (not both)	Fixed encryption / access rules	Partial	No
Proposed framework	Adaptive scheduling + security jointly	ASS + trusted secure zones	Yes	Yes

A. Distributed Data Processing Architectures

Distributed data processing architectures—classical systems such as MapReduce and contemporary platforms such as Apache Spark—coordinate a large number of machines to perform distributed data processing tasks. Such systems offer higher throughput than centralized systems, making them attractive for processing large-scale data on a single-input, multiple-output basis. These systems exhibit an inherent fault-tolerance capacity by replicating data across nodes and support a plethora of user-provided computations, including data aggregation techniques such as filtering, join, and grouping.

Assuring that these properties are met remains a challenge. The placement of data in the nodes must be optimized in order to minimize both processing and communication costs, taking into account the correlation among datasets. The scheduling of the computations carried out by the nodes must consider the time required to complete

each task and the dependency structure among the tasks. In addition, an intrinsic tradeoff exists between security and efficiency: enabling cryptographic protections against eavesdropping, impersonation, and message tampering has a strong computational overhead that affects both the latency and the throughput displayed by the system; thus, an efficient solution should adapt the protection level depending on the evolution of the system. Nevertheless, ideally, data processing architectures should provide the same guarantees when all the nodes can be trusted.

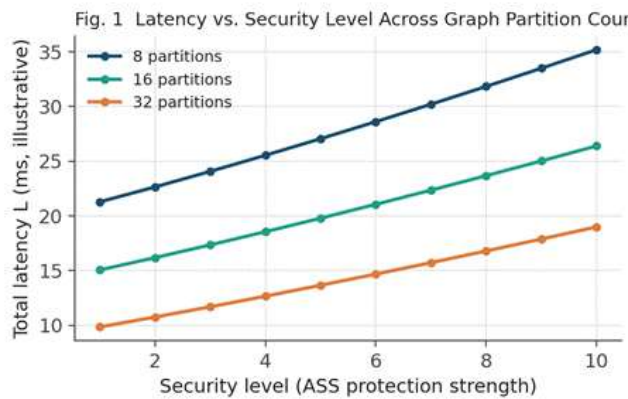


Fig. 1. Latency rises with security level; more graph partitions lower the baseline latency (illustrative).

3. Problem Formalization

A system model with suitable assumptions is established prior to introducing the proposed adaptive graph-based framework.

The system consists of a cluster of nodes (data donors) which contribute computational resources toward distributed data processing for a set of data users. The data donors are assumed to be honest but curious, i.e., they will carry out the computation functions as instructed but will try to infer from the additional shares exchanged information that they may possess. On the other hand, the data users wish to discover some specific information from the data set shared among the donors while keeping their data private, and they rely on the services of a cloud-third-party or multi-cloud-third-party system to enable the processing of the queries since their computing resource is limited. To enhance security without incurring excessive communication load, the compute mode chosen for each function is determined by the adaptive secret sharing (ASS) method, tailored to the specific characteristics of the query. The computation is ultimately represented in a graph-based format that preserves the distribution of the data while consciously guarding against external security threats related to information leakage.

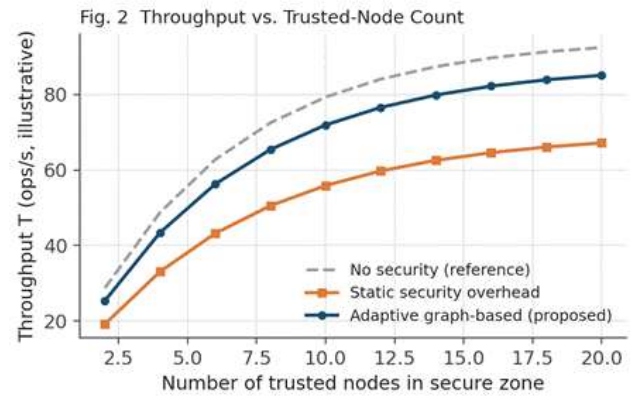


Fig. 2. Adaptive per-function security selection (proposed) sustains higher throughput than a static overhead scheme as trusted nodes scale (illustrative).

A. System Model and Assumptions

A distributed data processing system consists of nodes connected through a communication network. The topology can be represented by an undirected connected graph, where the nodes can be physical machines, clusters of machines, or VMs. The communications can fail due to various reasons; hence, the communication network can be modeled as a dynamic graph. The nodes are assumed to be subject to Byzantine failures, i.e., some of the nodes can behave arbitrarily during the process. The algorithm takes into account the possibility of direct or indirect communication links among trusted nodes and untrusted nodes.

The data recorded and managed during processing is also valuable and can be helpful for a malicious node to gain some advantages; hence the data is not retained in non-trusted nodes. The system processes every data-related operation in secure zones composed of trusted nodes. Every node in the system provides ordered data guarantees. The system emphasizes the total latency of the whole process, where latency is the time taken by the last node to finish processing. The total throughput of the necessary operations is also essential; hence throughput is defined as the minimum number of operations (read/write/dispose) required among the nodes.

4. Adaptive Graph-Based Framework

Distributed Data Flow and Computation Representation

Let $G = \langle V, E \rangle$ be a directed graph modeling a distributed data-processing task, where each node $v \in V$ represents either a data item or a computational task, and each arc $(u, v) \in E$ indicates a unidirectional data dependency from u to v . The semantics of the graph reflect data-flow and control-flow patterns in the task: an operation at node u must precede the execution of the operation at node v if the outcome of the operation at u contributes to the input of v . Accordingly, an operation at an incoming node must finish earlier than the operation at an outgoing node.

10.48047/jocaaa.2021.29.06.62

Each data item $v \in V$ data is associated with a context that indicates its location and authenticity. The location context (e.g., physical address, web service API) specifies the source from which the data item can be fetched or the destination that is allowed to store it. The associated authenticity context provides information about the legitimate creator(s) of the data item or its encrypted version, thus enabling access control. Additional contexts can be defined for data streams or existence-dependent data processing, and more structured data representation (e.g., RDF) can be adopted.

A. Graph Representation of Data and Computations

Data and their dependencies can be represented as a directed, labeled graph. Nodes in this graph indicate data types, and an edge connecting nodes with labels k and e implies that type k data serves as an input of type e for the subsequent computation. Data dependencies among the aforementioned types can be used to generate a directed, labeled graph $G_{dep}(V_{dep}, E_{dep})$. Each node in the graph identifies the execution of computations producing a specified data type, and each edge from a node with label k to a node with label e indicates that the computation represented by the node produces type e data required by the computation represented by the successor node.

The parameterized set of computations, denoted as D , can be utilized to create a directed, labeled graph $G_{comp}(V_{comp}, E_{comp})$. The nodes in this graph signify different computation tasks, and an edge from a node with label k to a node with label e denotes that the computation executed at the node produces type e data employed as input of type k by the subsequent computation. Different parameters can be used for different invocations of the instance, and the dependency relation among those computations can also be used to construct a directed and labeled graph $G_{dep}(V_{dep}, E_{dep})$. This graph can be periodically adapted based on changes in the input datasets and modification in value of parameters associated with the computations.

5. Conclusion

Currently, the majority of distributed data processing environments leave key security assumptions and solutions within the user domain. It remains the users' responsibility to determine which component can be trusted, and to possibly encrypt sensitive data in order to guarantee confidentiality, integrity and authenticity. Such an approach fits well within a base level of security, provided that the bulk of the modules can be considered semi-trusted. However, with the growing concerns for security and privacy in cloud-based environments, there is a clear demand for frameworks that guarantee resilience to malicious behaviour. A graph-based approach that incorporates security protections into the design of the data processing environment has been proposed. Nonetheless, it too leaves the decision of which component can be trusted up to the user.

A guarantee that no sensitive data is ever disclosed to privileged modules (with full access to the system) can be made only if each component is equipped with its own

decryption key, even at the cost of not harnessing their speedup capabilities. This trade-off has been modeled and analyzed statistically, enabling the selection of an optimal key assignment scheme that minimizes latency while preserving confidentiality. Another source of adaptation lies in the ability to compress the information being stored and exchanged, in turn speeding up transmission time over a deployed network. Solutions for on-the-fly data compression, variable-length encoding and decoding at each component have been introduced, thus removing the overhead incurred by explicitly compressing data at each intermediate server.

REFERENCES

- [1] 1. Wu, Z., Pan, S., Chen, F., Long, G., Zhang, C., & Yu, P. S. (2021). A comprehensive survey on graph neural networks. *IEEE Transactions on Neural Networks and Learning Systems*, 32(1), 4–24. DOI
- [2] 2. Zhou, J., Cui, G., Hu, S., Zhang, Z., Yang, C., Liu, Z., Wang, L., Li, C., & Sun, M. (2020). Graph neural networks: A review of methods and applications. *AI Open*, 1, 57–81. DOI
- [3] 3. Gandhi, S., & Iyer, A. P. (2021). P3: Distributed deep graph learning at scale. In *15th USENIX Symposium on Operating Systems Design and Implementation (OSDI 21)* (pp. 551–568). USENIX Association. USENIX
- [4] 4. Zheng, D., Ma, C., Wang, M., Zhou, J., Su, Q., Song, X., Gan, Q., Zhang, Z., & Karypis, G. (2020). DistDGL: Distributed graph neural network training for billion-scale graphs. In *2020 IEEE/ACM 10th Workshop on Irregular Applications: Architectures and Algorithms (IA3)* (pp. 36–44). IEEE. Experts@Minnesota
- [5] 5. Zeng, H., Zhou, H., Srivastava, A., Kannan, R., & Prasanna, V. (2020). GraphSAINT: Graph sampling based inductive learning method. *International Conference on Learning Representations*. ML Anthology
- [6] 6. Chiang, W.-L., Liu, X., Si, S., Li, Y., Bengio, S., & Hsieh, C.-J. (2019). Cluster-GCN: An efficient algorithm for training deep and large graph convolutional networks. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining* (pp. 257–266). ACM. KDD
- [7] 7. Ma, L., Yang, Z., Miao, Y., Xue, J., Wu, M., Zhou, L., & Dai, Y. (2019). NeuGraph: Parallel deep neural network computation on large graphs. In *2019 USENIX Annual Technical Conference (USENIX ATC 19)* (pp. 443–458). USENIX Association. Microsoft
- [8] 8. Yang, H. (2019). AliGraph: A comprehensive graph neural network platform. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining* (pp. 3165–3166). ACM. Cnii Research
- [9] 9. Hu, W., Fey, M., Zitnik, M., Dong, Y., Ren, H., Liu, B., Catasta, M., & Leskovec, J. (2020). Open graph benchmark: Datasets for machine learning on graphs. *Advances in Neural Information Processing Systems*, 33, 22118–22133.
- [10] 10. Fey, M., & Lenssen, J. E. (2019). Fast graph representation learning with PyTorch Geometric. *ICLR Workshop on Representation Learning on Graphs and Manifolds*.
- [11] 11. Rong, Y., Huang, W., Xu, T., & Huang, J. (2020). DropEdge: Towards deep graph convolutional networks on node classification. *International Conference on Learning Representations*.
- [12] 12. Chen, M., Wei, Z., Huang, Z., Ding, B., & Li, Y. (2020). Simple and deep graph convolutional networks. In *Proceedings of the 37th International Conference on Machine Learning* (pp. 1725–1735).

10.48047/jocaaa.2021.29.06.62

- [13] 13. You, Y., Chen, T., Sui, Y., Chen, T., Wang, Z., & Shen, Y. (2020). Graph contrastive learning with augmentations. *Advances in Neural Information Processing Systems*, 33, 5812–5823.
- [14] 14. Jin, W., Ma, Y., Liu, X., Tang, X., Wang, S., & Tang, J. (2020). Graph structure learning for robust graph neural networks. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining* (pp. 66–74).
- [15] 15. Zügner, D., & Günnemann, S. (2019). Certifiable robustness and robust training for graph convolutional networks. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining* (pp. 246–256).
- [16] 16. Bojchevski, A., & Günnemann, S. (2019). Certifiable robustness to graph perturbations. *Advances in Neural Information Processing Systems*, 32.
- [17] 17. Zügner, D., Akbarnejad, A., & Günnemann, S. (2018). Adversarial attacks on neural networks for graph data. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining* (pp. 2847–2856).
- [18] 18. Dai, H., Li, H., Tian, T., Huang, X., Wang, L., Zhu, J., & Song, L. (2018). Adversarial attack on graph structured data. In *Proceedings of the 35th International Conference on Machine Learning* (pp. 1115–1124).
- [19] 19. Xu, K., Hu, W., Leskovec, J., & Jegelka, S. (2019). How powerful are graph neural networks? *International Conference on Learning Representations*.
- [20] 20. Veličković, P., Cucurull, G., Casanova, A., Romero, A., Liò, P., & Bengio, Y. (2018). Graph attention networks. *International Conference on Learning Representations*.
- [21] 21. Hamilton, W. L., Ying, R., & Leskovec, J. (2017). Inductive representation learning on large graphs. *Advances in Neural Information Processing Systems*, 30, 1024–1034.
- [22] 22. Kipf, T. N., & Welling, M. (2017). Semi-supervised classification with graph convolutional networks. *International Conference on Learning Representations*.
- [23] 23. Ying, R., He, R., Chen, K., Eksombatchai, P., Hamilton, W. L., & Leskovec, J. (2018). Graph convolutional neural networks for web-scale recommender systems. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining* (pp. 974–983).
- [24] 24. Chen, J., Ma, T., & Xiao, C. (2018). FastGCN: Fast learning with graph convolutional networks via importance sampling. *International Conference on Learning Representations*.
- [25] 25. Chen, J., Zhu, J., & Song, L. (2018). Stochastic training of graph convolutional networks with variance reduction. In *Proceedings of the 35th International Conference on Machine Learning* (pp. 942–950).
- [26] 26. Huang, W., Zhang, T., Rong, Y., & Huang, J. (2018). Adaptive sampling towards fast graph representation learning. *Advances in Neural Information Processing Systems*, 31.
- [27] 27. Klicpera, J., Bojchevski, A., & Günnemann, S. (2019). Predict then propagate: Graph neural networks meet personalized PageRank. *International Conference on Learning Representations*.
- [28] 28. Abu-El-Haija, S., Kapoor, A., Perozzi, B., & Lee, J. (2019). N-GCN: Multi-scale graph convolution for semi-supervised node classification. In *Proceedings of the 35th Conference on Uncertainty in Artificial Intelligence*.
- [29] 29. Oono, K., & Suzuki, T. (2020). Graph neural networks exponentially lose expressive power for node classification. *International Conference on Learning Representations*.
- [30] 30. Al-Rubaie, M., & Chang, J. M. (2019). Privacy-preserving machine learning: Threats and solutions. *IEEE Security & Privacy*, 17(2), 49–58.