The Distributed Computing Model Based On The Capabilities Of The Internet

Ms.S. AbikayilAarthi, AP/CSE[1], Ms.G.Rasika[2], Ms.S.Nesika[3], Ms.A.Joshica[4]
[1, 2, 3, 4]Kings College of Engineering, Punalkulam, Pudhukottai.

Abstract*- This paper explores both the theoretical foundation and practical implementation of a proposed model that leverages distributed computing within the expansive framework of a global Internet communication network. Distributed computing has emerged as a cornerstone of contemporary technological solutions, particularly in domains requiring substantial computational power that exceeds the capacity of centralized systems. Examples include high-performance research endeavors such as simulations, data modeling, and real-time analytics.*

The proposed model capitalizes on distributed architectures, emphasizing scalability, efficiency, and resilience. By utilizing open-source technologies, the solution demonstrates adaptability and accessibility, ensuring a cost-effective deployment.

The computing resources are crowdsourced from volunteer Internet users, fostering a community-driven approach to computation while promoting shared innovation and resource optimization. This paradigm not only addresses challenges like scalability and resource limitations but also offers insights into eco-friendly computation practices by reducing reliance on large-scale data centers.

Through extensive experimentation and analysis, the paper outlines the architecture, data flow, and computational strategies underlying the system. A comprehensive evaluation is conducted to assess performance metrics such as speedup, reliability, fault tolerance, and security. The findings highlight the potential of distributed computing to democratize computational resources, enabling breakthroughs in research and innovation across multiple sectors.

Keywords- Distributed computing, architecture, design systems, volunteer computing, scalability, global network.

I. INTRODUCTION

distributed system is fundamentally defined as a collection of independent computing devices interconnected to function as a single, seamless logical entity. While the term "equipment" traditionally refers to desktop computers, advancements in technology have expanded this definition to include modern devices such as media tablets and smartphones. These devices contribute significantly to the versatility and scalability of distributed systems, enabling their deployment across diverse platforms and use cases.

- The solution discussed in this article relies primarily on Internet-based communication, leveraging its global reach and ubiquity. However, distributed systems are not inherently dependent on sophisticated communication infrastructures and can operate effectively using alternative methods. By design, distributed systems offer a unified and centralized appearance to users, a property known as *transparency*. Transparency masks the complexities of distributed components, presenting a single cohesive system—a hallmark feature that defines this class of architectures.
- The concept of distributed computing was first introduced in the late 1970s, emerging as a response to the growing need for scalable computational resources that could address complex and resource-intensive problems. Over the decades, the exponential growth of computing power, coupled with the proliferation of the Internet, has transformed this concept into a practical reality. Today, distributed systems harness the power of millions of devices worldwide, with resources often voluntarily contributed by users. This democratization of computational capabilities has opened the door to groundbreaking innovations across various domains, from scientific research to real-time data processing.
- This paper delves into the architecture, design, and implementation of a distributed system model that exemplifies the potential of leveraging volunteer-provided resources. It aims to explore the efficiency, scalability, and challenges of such systems, particularly in an Internet-driven era. By understanding the principles and advantages of distributed computing, this study underscores its significance as a solution for addressing the growing demand for computational power in an increasingly data-driven world.

GENERAL:

Distributed systems come in various architectural models, each catering to specific requirements and use cases. Among the most recognized and widely implemented models are **Peer-to-Peer (P2P)**, **Cloud Computing**, **Grid Computing**, and **Client/Server** architectures. These models play a crucial role in shaping modern distributed computing solutions,offering different paradigms to meet the diverse computational needs of users and organizations.

- **Peer-to-Peer (P2P):**

The Peer-to-Peer model emphasizes decentralized control, where each participating device, or "peer," operates as both a client and a server. This architecture is particularly suited for applications requiring resource sharing among a distributed network without reliance on a central authority. P2P systems are highly resilient and scalable, making them ideal for file-sharing platforms and distributed ledgers such as blockchain.

- **Cloud Computing**

Cloud computing has revolutionized the way resources are accessed and managed. By centralizing computing resources in large-scale data centers, cloud models provide on-demand services such as Infrastructure as a Service (IaaS), Platform as a Service (PaaS), and Software as a Service (SaaS). Cloud computing is known for its scalability, elasticity, and cost-effectiveness, enabling businesses to scale their operations with minimal upfront investment.

- **Grid Computing**

Grid computing involves pooling together resources from multiple distributed systems to tackle complex computational problems. Unlike the centralized model of cloud computing, grid computing focuses on collaborative use of diverse resources across organizations or geographies. This model is frequently employed in scientific research for tasks like climate modeling, protein folding simulations, and high-energy physics experiments.

- **Client/Server Model**

The Client/Server model remains one of the most prevalent and widely adopted architectures in distributed systems, particularly in open-source solutions. This architecture is based on a central server that provides services or resources to multiple clients, ensuring efficient management and coordination. The Client/Server model excels in applications requiring centralized control and structured resource management, making it a cornerstone of many distributed computing frameworks.

One of the most exemplary implementations of the Client/Server model is **BOINC** (Berkeley Open Infrastructure for Network Computing). Initiallydeveloped as a resource pooling solution for the SETI@Home project, BOINC has since evolved into a versatile platform for distributed computing. The SETI@Home project, which began at the University of California, Berkeley, aimed to analyze radio signals from space to search for extraterrestrial intelligence. Through BOINC, millions of volunteers worldwide contributed their unused computational resources to process vast amounts of data, showcasing the power of distributed collaboration.

BOINC has since expanded beyond SETI@Home, supporting diverse projects in areas such as molecular biology, climate change modeling, and astrophysics. The platform leverages the Client/Server architecture to distribute workloads efficiently, with central servers managing the allocation of tasks and aggregation of results. Its open-source nature has made it a benchmark for similar distributed computing initiatives, highlighting the transformative potential of volunteer-based resource pooling in solving large-scale computational problems.

By understanding and leveraging these distributed system models, organizations and researchers can design robust solutions tailored to specific challenges, unlocking new possibilities in scientific discovery, industrial applications, and beyond.Computing), that started as a resource pooling solution for SETI@Home project and is constantly being developed at University of California, Berkeley.Each of these distributed system models offers unique advantages depending on the specific needs of an application. While **Peer-to-Peer** excels in decentralization and fault tolerance, **Cloud Computing** provides flexibility and scalability for enterprises, and **Grid Computing** supports high-performance tasks that require vast computational resources. The **Client/Server** model, as exemplified by **BOINC**, remains a reliable solutionfor managing large-scale distributed networks, especially when centralized coordination and resource management are essential. Together, these models demonstrate the vast potential of distributed computing in addressing diverse challenges across various domains.

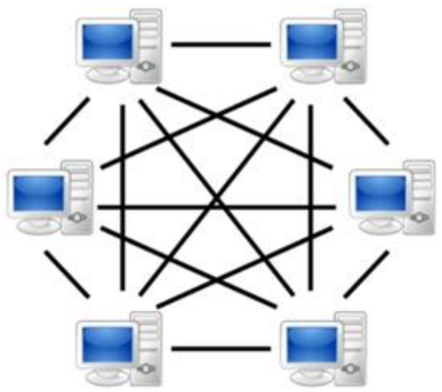


Figure 1.Peer-To-Peer Architecture Diagram

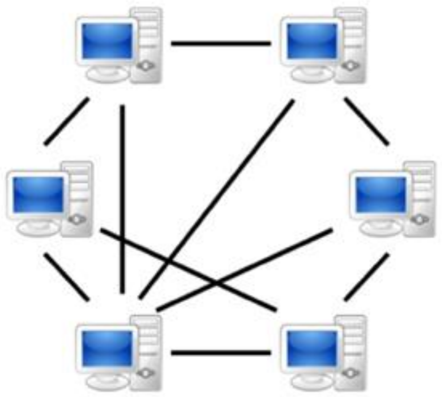


Figure 2. Client/Server Architecture Diagram

ADVANTAGES AND DISADVANTAGES OF DISTRIBUTED SYSTEMS

While distributed systems offer numerous benefits, they also come with several limitations. Unlike supercomputers, which are typically structured in computing clusters and commonly used in academic research, distributed systems cannot handle every problem that supercomputers can solve. One of the most significant limitations is the data transfer rate. Distributed systems rely on internet connections, which are slower compared to high-speed connections like Infiniband used in computing clusters. As a result, tasks involving large data transfers may not be ideal for distributed systems, which are better suited for long computations with relatively small data inputs. Another key challenge is the uncertainty of calculations. Since data is processed on third-party computers, there is no guarantee of accuracy, and there is a risk that results might not be successfully returned to the server. Users could uninstall the software or reinstall their operating system at any time, further complicating the process.

However, the advantages of distributed systems are noteworthy. One of the most significant benefits is their relatively low operating costs. In the Client/Server architecture, only the server requires maintenance, often involving just one physical server. This server is responsible for assigning and managing tasks, while the actual computation is carried out by other computers. The affordability of such systems has made them attractive to amateur programmers, individual researchers, and scientific institutions.

A comparison between distributed systems and some of the world’s top supercomputers, as shown in Fig. 3, highlights that distributed systems, while not always on par with cutting-edge technology, are often capable of competing with the performance of systems used by organizations like NASA and the military, which require significant financial investment. For the comparison, the LINPACK performance benchmark was used to measure supercomputing performance, while the performance of distributed systems was assessed based on the number of results processed in a given period.

A simple comparison between supercomputers and distributed systems is presented in Table I.

	Distributed system	***Supercomputer***
Reliability	Low	High
Independence	Low	High
Scalability	High	Mean
Cost	Very low	Very high

POSSIBLE SERVER DAEMONS

In a distributed system, various services need to be provided by the server, including task generation, distribution of tasks to clients, and the analysis of results returned from clients. The number of daemons and the scheduling of work do not necessarily have to be evenly distributed across the system; however, in practice, the structure is often similar to that used by BOINC.

a) **Assimilator**: The Assimilator daemon handles tasks that have been completed and whose results are already known. It is responsible for saving the relevant data to the central database. When necessary, it may also delete temporary data from the system to maintain efficiency.

b) **Transitioner**: The Transitioner daemon monitors the status of tasks. It ensures that tasks are reassigned to other computers if the original one fails to return results within a given timeframe, while the previous task is canceled. Additionally, if the same task is processed by multiple computers and returns conflicting results, the Transitioner daemon sends the task to more computers for verification and correction of the results.

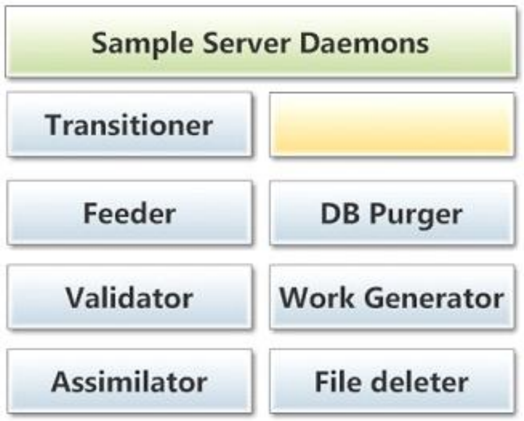
c) **Validator**: The Validator daemon is one of the final stages in task-related operations. It verifies the results uploaded by the clients, ensuring their correctness. In many systems, such as BOINC, the Validator daemon also awards credits or points to users as a recognition for their contribution to the computation.

d) **Work Generator**: The Work Generator daemon is responsible for creating tasks in an automated manner. These tasks are then distributed to other computers in the network. The task generation process can take into account various factors such as the computational capabilities of different systems, including available RAM, free disk space, and the type of CPU.

e) **Other Daemons**: In addition to the main daemons, there may be other auxiliary daemons in the server structure. These can perform tasks such as cleaning up files or managing database records. The role of these daemons is crucial in ensuring that the system runs efficiently, preventing unnecessary data accumulation over time and ensuring that obsolete information is removed.

Many distributed systems also feature a browser-based interface that allows users to view tasks and track system-related statistics. In addition, other services may be integrated with the interface to facilitate interaction and enhance the overall user experience.

a) ed in the server structure. These daemons can be responsible for other operations (e.g. cleaning) on files and data base records. Due to their work, despite long operating times of the system, the amount of stored data does not increase indefinitely and waste information are not stored.



SCALABILITY:

Within the term scalability in distributed systems there are clearly a few key aspects that can be mentioned. Figure 5 presents these aspects.

Size scalability can pose a significant challenge. There might be a large number of users joining the system that is available on the Internet. The amount of clients and computers can not increase indefinitely, therefore at a certain point, the server will become the bottleneck. Furthermore, computers can be located in different parts of the world, sometimes even in regions where downloading larger amounts of data can result in long waiting times. Due to the large physical separation, issues related with Internet backbone can also be visible. The term scalability also defines system administration and maintenance. Despite the fact that the system is being distributed, it should be perceived by the users as one logically consistent system.

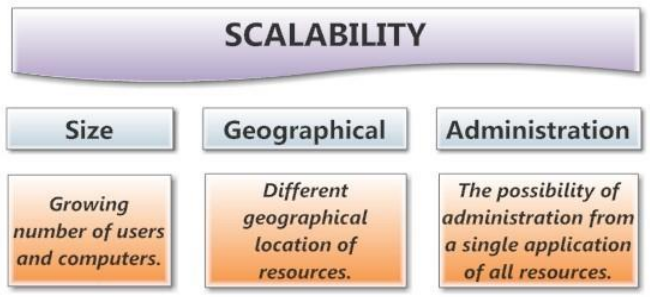

SCALABILITY
Size
Geographical
Administration
Growing number of users and computers.
Different geographical location of resources.
The possibility of administration from a single application of all resources.

Figure 5. Main problems concerning the scalability of distributed systems.

SECURITY

Similarly to the scalability issue, during the security analysis one can denote smaller, distinct components.

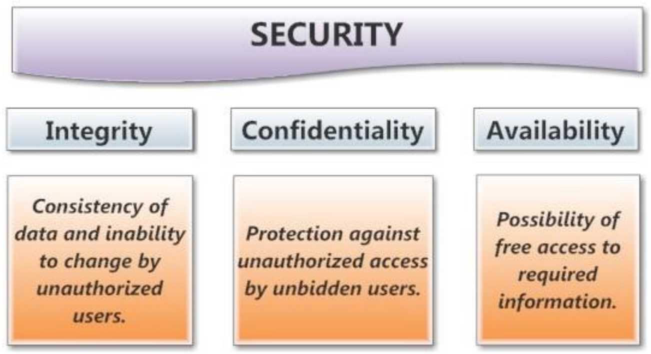

SECURITY
Integrity
Confidentiality
Availability
Consistency of data and inability to change by unauthorized users.
Protection against unauthorized access by unbidden users.
Possibility of free access to required information.

Figure 6. Main problems regarding distributed systems security

Data processed using distributed systems are more vulnerable to illegal intrusions. The platform should maximize security, however it will always be lower than with supercomputers that are operated only by certain entities.

APPLICATIONS AND ALGORITHMS

Distributed systems have a broad range of applications beyond traditional computing tasks. They are also used in network routing, distributed databases, and industrial control systems (ICS). For example, in aviation, distributed systems are crucial for aircraft flight control, where communication between heterogeneous systems is common.

The algorithms implemented in high-performance computing systems can be categorized as follows:

1. **Parallel Algorithms in Shared-Memory Model** (e.g., SMP architecture computers),
2. **Parallel Algorithms in Message-Passing Model** (e.g., performance clusters),
3. **Distributed Algorithms**.

The key distinction between these algorithms lies in the communication methods. In the shared-memory model, the programmer has access to shared memory, which simplifies the implementation of many problems (e.g., using the OpenMP library). In the message-passing model, where the Message Passing Interface (MPI) library is used, the programmer has less flexibility and must define a custom logical network structure for data transfer. For distributed algorithms, however, the designer must recognize that the network infrastructure could become a potential bottleneck, which makes it a critical factor in performance.

GOLDBACH CONJECTURE PROJECT

In this research, a self-developed platform operating under BOINC was used to test the validity of the Goldbach conjecture. This well-known problem in number theory proposes that every even natural number greater than 2 is the sum of two prime numbers. Despite being unproven or disproven for over 250 years, theconjecture is ideal for distributed computing due to its ability to easily divide tasks that can be executed independently. The project was supported by approximately 15,000 computers, connected through nearly 1,000 users. Remarkably, the project did not rely on any promotional efforts, highlighting the organic interest users had in contributing to this scientific endeavor.

One significant challenge encountered with the BOINC server was the MySQL database, which became heavily utilized. Without removing old results from the database for several days, the accumulation of historical data led to inefficient query processing. The problem could have been alleviated with modern SSD arrays and distributed table storage, but due to resource constraints, only a standard hard drive was used. RAM is another critical component for the server, with a minimum of 4 GB required for larger projects. The monthly maintenance cost for a dedicated server at a hosting company is approximately $50 (excluding activation fees). Running a distributed data processing server from home requires a high-throughput internet connection, as many residential connections have low upload speeds and cannot handle a large number of simultaneous connections. This limitation can deter volunteers from participating, ultimately affecting the overall system performance.

Interestingly, the CPU was not heavily utilized during the Goldbach Conjecture Project. Despite this, the official BOINC requirements for the CPU are high, especially for demanding tasks like validating returned results. The average load on the CPU (Intel Celeron 1200 MHz) during the Goldbach Conjecture Project is shown in Fig. 7. The average system performance reached about 10 TeraFLOPS, a remarkable result given the minimal operational costs. Performance measurements of the platform are provided in Fig. 8.

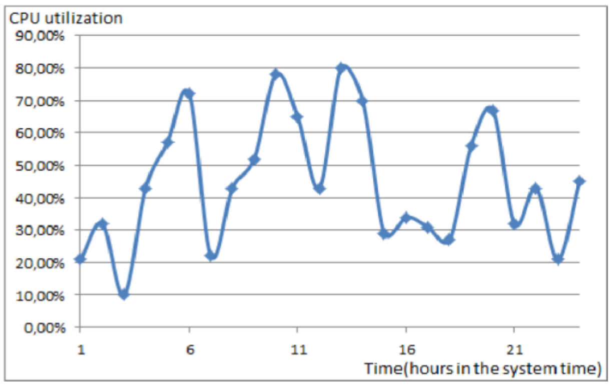


Figure 7. Avarage server CPU utilization (Intel Celeron 1200 MHz)

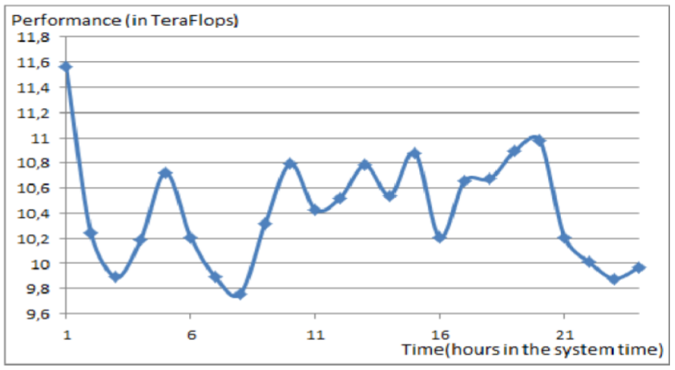


Figure 8. Average performance platform Goldbach Conjecture Project

It is easy to realize, that performance remains fairly constant regardless of the time. Intuition suggests, that during night-time, data processing speed should drop, because many users would turn off their computers. However, one should note, that the users and their computers are scattered around the globe, and thus are located in different time zones. It is the very reason why performance level is steady. Most severe drop in performance is always caused by server problems. The whole system is dependent on its reliability. In addition, one needs to keep in mind that users participating in our project, collect points, which are stored in the database. Therefore, one needs to particularly care about not losing their achievements, since they are often the key to volunteers motivation. On average, 1.5 users per day joined the Goldbach Conjecture. From another perspective, number of computers increased every day by approximately 2. It is shown on Fig. 9 and Fig. 10 respectively. These values were measured for a system already running longer than 6 months. In case of a new BOINC project, they are initially much higher. GoldbachConjeture Project did not solve the Goldbach's conjecture. However, working with server infrastructure developed by the University of California illustrated the great potential that resides in this solution.

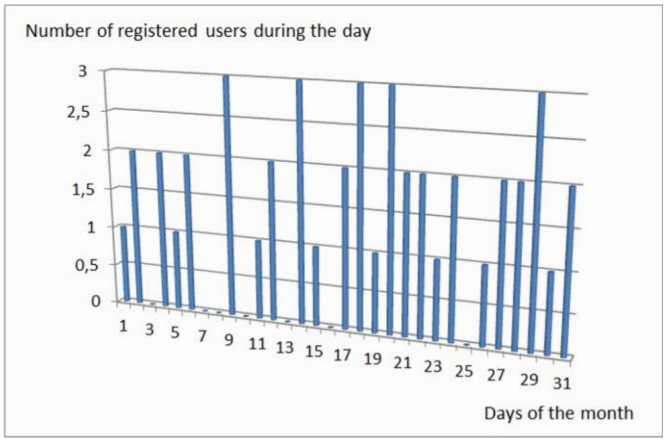


Figure 9. Number of registered users during certain days

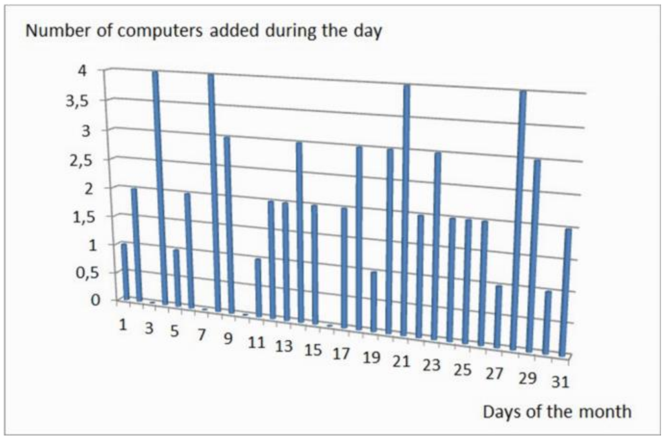


Figure 10. Number of computers added during certain days

II. CONCLUSION

In conclusion, this paper emphasizes the growing significance of distributed systems as a practical alternative to traditional centralized systems in specific applications. Unlike supercomputers, which demand extensive financial investments for both capital and operational expenses, distributed systems present a more accessible and cost-effective solution. These systems, leveraging the power of numerous individual devices connected via the internet, enable the distribution of computational tasks, effectively harnessing the collective processing power of a vast number of volunteers. This trend has gained momentum, as more internet users contribute their computing resources to support research and computational efforts at academic institutions, scientific projects, and various other fields globally.

With the continued growth of the internet and advancements in related technologies, the role of distributed systems will likely expand. However, this growth comes with the need to address new challenges. As the scale and complexity of distributed systems increase, issues such as security, data privacy, network bandwidth, and system reliability will become more prominent. The development of more efficient algorithms, robust fault-tolerant mechanisms, and scalable infrastructure will be crucial to overcoming these challenges. As the technology matures, distributed systems will continue to evolve and play a significant role in fields ranging from academic research to industrial applications, offering a promising solution to many computational problems that require high processing power.

REFERENCES

[1] G. Coulouris, J. Dollimore and T. Kindberg, Distributed Systems - Concepts and Design. U.K.: Addison-Wesley, Fourth Edition, 2005.

[2] B. C. Neuman, Scale in Distributed Systems, Readings in Distributed Computing Systems. IEEE Computer Society Press, 1994.

[3] B. Jacob, M. Brown, K. Fukui, N. Trivedi, Introduction to Grid Computing. International Business Machines Corporation, U.S.A.: IBM, International Technical Support Organization, First Edition, 2005.

[4] D. P. Anderson, BOINC: A System for Public-Resource Computing and Storage. Proceedings of the Fifth IEEE/ACM International Workshop on Grid Computing, 2004.

[5] C. U. Sottrup, J. G. Pedersen, Developing Distrubited Computing Solutions: Combining Grid Computing and Public Computing. M. Sc. Thesis, Department of Computer Science, University of Copenhagen, 2005.

[6] R. J. Al-Ali, K. Amin, G. von Laszewski, O. F. Rana, D. W. Walker, M. Hategan, N. Zaluzec, Analysis and Provision of QoS for Distributed Grid Applications. Kluwer Academic Publishers, 2004. W.Wang,J.Gao,M.Zhang,S.Wang,G.Chen,T.K.Ng,B.C.Ooi,J.Shao,andM. Reyad. Rafiki: Machine Learning as an Analytics Service System. *Proc. VLDBEndow.*,12(2):128–140,2018.

[7] W. Wang, X. Yang, B. C. Ooi, D. Zhang, and Y. Zhuang. Effective deep learning-basedmulti-modalretrieval.*VLDBJ.*,25(1):79–101,2016.

[8] P. Watcharapichat, V. L. Morales, R. C. Fernandez, and P. Pietzuch. Ako: De-centralised deep learning with partial gradient exchange. In *Proceedings of theSeventhACMSymposiumonCloudComputing*,SoCC'16, page84–97,2016.

[9] J. Weston, F. Ratle, and R. Collobert. Deep learning via semi-supervised embed-ding. In *ICML*, volume 307 of *ACM International Conference Proceeding Series*,pages1168–1175.ACM,2008.

[10] D. Xin, S. Macke, L. Ma, J. Liu, S. Song, and A. G. Parameswaran. Helix: HolisticOptimizationforAcceleratingIterativeMachineLearning.*Proc.VLDBEndow.*,12(4):446–460,2018.

[11] A.YoshitakaandT.Ichikawa.ASurveyonContent-BasedRetrievalforMulti-mediaDatabases.*IEEETrans.Knowl.DataEng.*,11(1):81–93,1999.

[12] B. Yuan, D. Jankov, J. Zou, Y. Tang, D. Bourgeois, and C. Jermaine.TensorRelational Algebra for Machine Learning System Design. *CoRR*, abs/2009.00524,2020. Y. Bengio.Rmsprop and equilibrated adaptive learning rates for nonconvexoptimization.*corrabs/1502.04390*,2015.

[13] J. Bergstra, D. Yamins, and D. D. Cox. Making a Science of Model Search: Hyper-parameterOptimizationinHundredsofDimensionsforVision Architectures.In*ICML(1)*,volume28of*JMLRWorkshopand ConferenceProceedings*,pages115–123.JMLR.org,2013.

[14] M.Boehm,M.Dusenberry,D.Eriksson,A.V.Evfimievski,F. M.Manshadi,

[15] M.Boehm,S.Tatikonda,B.Reinwald,P.Sen,Y.Tian,D.R.Burdick,andS. Vaithyanathan.Hybrid Parallelization Strategies for Large-Scale Machine