ISSN 2661-2666 (Online) International Scientific Journal Monte (ISJM) DOI: 10.33807/monte.20233018 Volume 8, (No).1 (2023): December

THE ROLE OF PARALLEL COMPUTING IN DATA SCIENCE

Mefail Tahiri[1*] *Ejup Rustemi*[2]

[1]*University of Tetova, North Macedonia*

[2]*University of Tetova, North Macedonia*

[*] *mefailt@gmail.com*

ABSTRACT

The machine intelligence of algorithms is now dispersed in a cloud computing environment, which will help businesses in the future find insightful information and carry out various tasks using APIs. Due to the fact that it meets economies of scale in a distributed setting, organizations are mass producing algorithms. With machine intelligence platforms that use machine learning to quickly prototype and deploy in production from sandboxes, artificial intelligence is the next fire for igniting the AI cold (which lasted from the 1990s through the 2010s). Recent times have seen the release of a number of open-source machine learning and deep learning platforms, including TensorFlow by Google, Caffe by the University of Berkeley, NLTK (Natural Language Tookit) by the University of Pennsylvania for natural language processing, Scikit-learn for Python, a number of R packages for deep learning and machine learning, Theano, a Python library for numerical computation, and Torch, a platform for developing machine learning algorithms. Big data has been greatly impacted by artificial intelligence, which has changed numerous global sectors by resolving previously intractable problems. This paper aims to evaluate the need for powerful processing infrastrucure for successful application of data science.

Keywords: *data science, parallel computing, big data, machine learning, multiprocessing.*

INTRODUCTION

Massive amounts of big data have been produced as a result of new capabilities and sources that have been provided as part of the web's progression from Web 1.0 to Web 5.0 over the past few decades. The terms Internet and World Wide Web have the same meanings. However, the functioning of the World Wide Web differs because it uses technology supported by Web-scale complex networks to build a social ecosystem. The World Wide Web offers the network infrastructure that enables social interaction and cooperative communication among people. In 1989, Tim Burners-Lee had the idea for the World Wide Web. The World Wide Web has changed significantly from Web 1.0 to Web 5.0 during the last few decades.

The goal of building Web 1.0 was to provide humans cognitive ability, particularly so that businesses could use the new network of technology to disseminate their information. In contrast to the traditional method of information transmission via paper, this was a catalyst-broadcasting media. The majority of the HTML material on the Web was static, and no changes or interactions were made to the data that was being transmitted across it. It included a few search features. Distribution and involvement for the content contribution were modest. Building a single UDI (Universal Document Identifier) that can hold a lot of data in a single document was the goal of creating Web 1.0. Web 1.0 mainly had read-only capabilities and was static. Businesses pioneered the Web 1.0 revolution as a platform for displaying all of their company features to a wider audience. The websites generally resembled newspaper advertisements or informational brochures because they lacked responsive design to offer any interactive elements. The web's information was consistent and kept around for a very long time without being updated. The HTTP (Hypertext Transfer Protocol), HTML (Hypertext Markup Language), and URI protocols served as the foundation for Web 1.0. (Universal Resource Identifier). The newest protocols, including XHTML, XML, ASP, CSS, CGI, JSP, PERL client-side scripting, VBScript, and JavaScript, were also included in the Web 1.0 (Pulipaka, 2018). The architecture was built on a client-server model, with the web client acting as the web browser on the Internet and sending requests to web services that would then provide a response to the web client. 1989 through 2005 was the extended lifespan of the Web 1.0. The bulk manufacturing of publications on the internet was not possible. It was primarily for a specific corporate entity with a special display of content pooling URIs (URIs). The Webmaster connects all of the hyperlinks to the published information and serves as Web 1.0's administrator. Web 1.0 has made extensive use of frames. Web 1.0 had no machine intelligence, machine-to-machine, or human-to-machine communication protocols, therefore only humans could understand the data content.

The idea behind Web 2.0 was to use communication networks to power the World Wide Web. Dale Dougherty came up with the idea for Web 2.0 in 2004. Web 2.0 brought the write functionality of the data with the read functionality for the first time. People could engage with one other based on their common interests thanks to the data aggregation, which was drawn from a huge number of populations. A conference about the web's future was held by O'Reilly. A novel idea for read-write functionality on the web was unveiled at the conference. In contrast to static HTML websites that only acted as information portals, Web 2.0 offered collaborative networks and human interactions that provided scattered contributions over vast networks. A number of media organizations expressed interest in using the conference to disseminate their data and enable public participation in social technology networks. Web 2.0 platforms served as public knowledge centers (Pulipaka, 2018). The internet was no longer stagnant. Web 2.0 replaced the static web content display methods with collective intelligence and cooperation, enabling users to generate enormous volumes of big data online. The Web 2.0 era has brought about new innovative technologies that power the web and combine data from various networks. A variety of social networking tools, including Twitter, Facebook, the video-sharing website YouTube, and knowledge-based websites like Wikipedia, have emerged with the emergence of Web 2.0. These websites have quickly gathered the data using HTTP services with URIs and HTML. Building social groups that act as content aggregators has taken precedence over corporate firms disseminating their data through static HTML content displays in the form of information brochures. Peer-to-peer networks replaced client-server networks as the dominant form of technology. With the help of new RSS feeds (Really Simple Syndication), anyone can create their own content aggregator. The owners of the commercial websites were no longer the webmasters. The general public might participate, work together, and share the content. The public was able to produce enormous amounts of data because to the networks' speedy transition from dialup to broadband. From a static information portal, the web has transformed into a dynamic platform. Blogs and articles that included text, video, and other media greatly increased the production of new types of data. The popularity of wikis has led to the creation of knowledge hubs that collect enormous amounts of data online.

Web 3.0 was developed with the idea of human collaboration in mind. The emphasis was on automating human-performed jobs using artificial intelligence and semantics that computers can understand. The Web 3.0 revolution for human-machine collaboration is partly responsible for the growth of social networking groups. Around 2006, John Markoff first used the term Web 3.0. Semantic web was a well-known moniker for Web 3.0. The web was created with clever and sophisticated programs that let users add to and read content. Encyclopedias, online novels, health care research data, and geospatial GIS information are a few examples of the new forms of data artifacts that Web 3.0 has produced. The material might be executed thanks to Web 3.0. A unique international character set that could be written in any language was represented by the union of Unicode and URI as distinct data identifiers. Information may now be transmitted between different platforms thanks to the development of XML (Extensible Markup Language). With framework elements like URI, Unicode, XML, RDF schema, RDF, ontology, proof, and unifying logic, Web 4.0 is a multi-layered semantic web (Pulipaka, 2018).

In order to integrate networks and transmit data with intelligent interactions between a coalition of human and machine intelligence, Web 4.0 was built. Massive amounts of machine-to-machine and human-to-machine data were produced by the Web 4.0, which also introduced machine intelligence and operational intelligence. Websites like Amazon have figured out their visitors' buying patterns and offered suggestions. The web has migrated to a variety of gadgets, including smart phones and tablets. The creation of billions of transistors on a single silicon chip has pushed the limits of technology, enabling the web to power nanoscale devices for improved performance and healthcare. The volume of data exploded to petabytes.

The World Wide Web's next version, Web 5.0, is also known as Brain-Net. It is a work-in-progress that aims to bring about an interaction revolution for all eras where people can share feelings, memories, and experiences. With the ideas of communication through SmartCommunicators and brain, it is known as the Symbionet Web. It's thought of as a 3D World Wide Web.

THE WEB OF BUSINESS

The web is where most large data comes from. Businesses using web technology, like Google, process over 3.5 billion queries daily and currently have 10 exabytes of data stored. To process the large amounts of big data streaming from the web, Google and Microsoft have a combined million servers. Every day, Facebook processes about 500 gigabytes of data. The globe will produce a staggering 40 zettabytes of data by 2020, according to a prediction from the research juggernaut IDC. By 2012, the World Wide Web alone has generated enormous volumes of data totaling 500 exabytes. Big data's projected growth trend indicates that it will double every two years and increase 50-fold between 2010 and 2020, exhibiting characteristics of the coming technological singularity and the law of exponential returns. There are a number of effects on the big data processing engines that come from the web when the volume of big data increases dramatically. The following elements may be the cause of performance bottlenecks:

Huge workloads are exceeding the servers' DRAM allotment. The time required to run the SQL and NoSQL queries Deadlocks on the database when writing data in parallel using distributed clusters of computers challenges with scaling when connecting numerous databases to obtain records from massive volumes of data.

A shared hard drive's numerous read and write operations could cause power outages or heat death. Big data tools like Apache Hadoop, Cloudera, HortonWorks, and MapR technologies through programming languages like Java are required for the complex processing of unstructured big data from the web. Due to parallel computing, distributed computing, and multithreaded processing, the programming may have performance bottlenecks. processing of numerous relationships between complex events (Pulipaka, 2018). Because the software isn't scalable, the performance bottlenecks in the programming can also result from a lack of best practices. the intricate, computationally intensive processing of algorithmic programming. Using traditional disk-based databases to access Operating system buffer overflows on Linux, Unix, macOS, and other platforms TCP protocols' allocation of non-scalable buffer sizes Memcached techniques are not used.

Lack of web cache optimization I/O bottlenecks, network bandwidth, and L1 and L2 caches localized data overloads on the CPUs lack of bandwidth in the network's packet sizes.

Due to the large volume of big data in recent years, issues with data transfer over the web may have worsened as it continues to grow across the web. The biological research labs saw a continuation of the pattern. These Boston and Seattle research labs chose to physically ship the data hard disks because doing so is very time- and money-consuming and runs the risk of performance bottlenecks. Due to web-based network bandwidth and latency restrictions, a new necessity for data in transit has arisen. Tumor photos ranging in size from 3 gigabytes to 10 gigabytes are included in the data from the medical research facilities. The size of the big data increases dramatically as there are thousands of malignant tumor sample (Pulipaka, 2018). Congestion avoidance is one of the most prevalent speed obstacles encountered when moving huge data over the internet. The ratio of data given to network demands determines the network's data transfer rate and efficiency. The network begins to experience missed packets and develops a delayed network path, which can cause the huge data packets to travel through the network in very slow motion if the data supply outpaces the demand. Although the congestion avoidance method may perform well in local area network settings, it requires a completely different approach when used in global computer networks or wide area networks. The network's data speed is directly inversely related to the distance between the source of the data and its destination. In order to address performance issues, IBM recently bought Aspera, a company that uses the fasp protocol to transfer data over networks and clouds of clouds at lightning-fast speeds (Pulipaka, 2018).

USING THE POWER OF PARALLEL COMPUTING

Today, big data is emerging from both traditional big scientific and engineering applications and the networked world. The rise of big data is also aided by new sensor and communication technology as well as new applications like cloud computing and the internet of things. Today's big data era is centered on the study and use of information technologies. However, the majority of big data research currently conducted focuses on system technology and applications, with little work done on the theoretical underpinnings. In addition, the first two of the three fundamental qualities or difficulties of big data—volume, velocity, and variety—have received a lot of attention while "variety" has received relatively less research (Chen, Mao, Lu, 2016).

In parallel programming, there are two primary performance problems:

Cost of communications: Normally, information has to be passed back and forth between processes. This requires time, which might have a negative impact on output. Additionally, if multiple programs attempt to access the same data at once, they may interfere with one another. When attempting to access the same communication channel, memory module, etc., they may run into one other. Another speed sapper, this.

In general, the term "granularity" refers to the proportion of computation to overhead. Large enough computing chunks are involved in large-grained or coarse-grained algorithms that overhead isn't a major issue. We absolutely need to avoid overhead as much as possible in fine-grained algorithms.

Load balance: As mentioned in the previous chapter, if we are not careful in how we distribute work to processes, we run the risk of giving certain processes significantly more work than others. Performance is compromised because some processes become ineffective at the conclusion of the run when work is yet to be done.

A multiprocessor system, as the name suggests, has two or more processors, also known as two or more CPUs, allowing for the simultaneous calculation of multiple programs or segments of a single program.

As we will see later, a multicore system, which is popular in homes, is essentially a low-end multiprocessor. Since they do share the same physical RAM, multiprocessors are sometimes known as shared-memory systems (Cespa, 2020).

Nowadays, practically all laptops and household computers have at least two cores. Congratulations, you own a multiprocessor system if you have one of these machines!

A cluster is made up of several independent computers that are networked together to allow them to work together in concert to solve a challenging numerical problem.

The terms shared-memory and networked above provide hints as to the difficulties that occur with computational speed, which are crucial.

One Ethernet, let's say inside a building, is referred to as a network. The interconnection of millions of different networks is all that the Internet is. If you are in San Francisco and tell your computer to open the Cable Network News (CNN) homepage, for example. The packets of information will travel from San Francisco to Atlanta because CNN's headquarters are there. Actually, since Internet service providers (ISPs) frequently cache Web sites, they might not go quite that far. However, let's assume it doesn't. The voyage of a parcel will actually be quite difficult.

Your web request will be written to a socket by your browser application. The latter is a software interface that connects your program to the network rather than a real thing. Your request will be converted into a packet by the socket software, which will then pass it through your OS's network protocol stack. The packet will expand when more information is added, but it will also divide into several smaller packets along the way. The packets eventually get to your computer's network interface hardware, where they are sent out onto the network. The packets will be sent to another network that the gateway is also connected to when a gateway on the network realizes that the final destination is outside of this network. Your packets will go from network to network as they are sent across the nation. Your packets will now progress via the OS layers until they reach the Web server program on a CNN computer.

Physical execution of your software takes place on various processors. Your software and data are stored in memory banks during execution. A bus, which is a group of parallel wires used for communication between various computer parts, is connected to the processors and memory banks. Although there may be more than one bus, our focus will primarily be on the processors and memory (Singh, 2022). Your input/output gear, like as disk drives, keyboards, and so on, are also connected to the bus. A threaded program will contain multiple instances of itself, known as threads, that cooperate to accomplish parallelism. With the exception of sharing the program's data, they each execute independently. If your application is threaded, it will execute on many processors simultaneously, with each thread executing on a different core.

THE IMPORTANCE OF MULTIPROCESSING

Over the past few decades, the computer community has become increasingly interested in the topic of parallel computing. Even today's ultra-fast single processor computers are under a lot of strain due to the constantly expanding quantity of databases and the increasingly complicated nature of new challenges. The global computer science community is currently searching for a computing environment that can multiply the existing processing power by a factor of thousands. The emergence of parallel computing, or the use of many processors operating simultaneously, is the most obvious solution.

Many calculations are performed simultaneously in a process known as parallel computing, which is based on the idea that big problems may frequently be broken down into smaller ones and tackled simultaneously ("in parallel").

In parallel computing, the same operation is simultaneously carried out on numerous processors after being divided into smaller tasks to provide results more quickly. The concept is based on the observation that most problems may be broken down into smaller jobs that can be completed simultaneously with certain coordination techniques.

Software for computers has often been created for serial computation. An algorithm is created and implemented as a serial sequence of instructions to solve a problem. One computer's central processor unit carries out these instructions. One instruction may only run at a time; after that instruction is complete, the next may run (Cespa, 2020).

On the other hand, parallel computing uses several processing units at once to address an issue. In order for each processing element to execute its portion of the algorithm concurrently with the others, the problem must be divided into independent portions. A single computer with several processors, a number of networked computers, customized hardware, or any combination of the aforementioned can be used as one of the processing elements.

Multicore computing

A CPU with many execution units, or "cores," on the same chip is referred to as a multicore processor. In contrast to superscalar processors, which may execute multiple instructions per cycle from a single instruction stream (thread), multicore processors can execute multiple instructions per cycle from a number of instruction streams (Singh, 2022) . In a multicore processor, each core has the potential to be superscalar, which means that each core can issue multiple instructions from a single instruction stream throughout each cycle. Pseudo-earliest multicoreism's iteration was simultaneous multithreading, best known as Intel's HyperThreading. A CPU that supports simultaneous multithreading only has one execution unit (or "core"), but when that core is idle (such when there is a cache miss), it processes a second thread.

Symmetric multiprocessing

A computer system having numerous identical processors that share memory and are connected by a bus is referred to as a symmetric multiprocessor (SMP). Scalability of bus designs is hampered by bus contention. SMPs typically have no more than 32 processors as a result. Such symmetric multiprocessors are incredibly cost-effective, given that there is sufficient memory bandwidth, due to the tiny size of the processors and the huge reduction in the requirements for bus bandwidth gained by big caches (Singh, 2022).

.

CONCLUSION

The processing components of a distributed computer system, commonly referred to as a distributed memory multiprocessor, are linked by a network. Highly scalable distributed computing systems exist.

The most dispersed type of parallel computing is distributed computing. To solve a specific issue, it uses computers talking to each other online. Due of the extremely high latency and poor capacity of the Internet, distributed computing typically only addresses problems with embarrassingly parallel patterns. The most well-known distributed computing programs are SETI@home and Folding@home, which are among the many that have been developed.Distributed systems are collections of networked computers working toward the same objective. There is a lot of overlap and no unique meanings for the phrases "concurrent computing," "parallel computing," and "distributed computing."

The same system can be described as both "parallel" and "distributed"; in a typical distributed system, processors operate simultaneously in parallel.Distributed computing can be considered as a particular form of tightly connected parallel computing, whereas parallel computing can be seen as a sort of loosely coupled distributed computing.However, using the following standards, concurrent systems can be loosely categorized as "parallel" or "distributed":All processors may have access to a shared memory during parallel computing in order to communicate with one another.Each CPU in distributed computing has its own private memory (distributed memory).

REFERENCES AND BIBLIOGRAPHY

- Cespa, G., & Vives, X. (2017). High frequency trading and fragility (Working Papers Series). European Central Bank (2020).
- Chen, G., Mao, R., Lu, K. (2016). A Parallel Computing Framework for Big Data. Higher Education Press and Springer-Verlag Berlin Heidelberg.
- Fan W, Geerts F, Neven F. Making queries tractable on big data with preprocessing: through the eyes of complexity theory. Proceedings of the VLDB Endowment, 2013, 6(9): 685–696
- Fox, J. (2019). Regression Diagnostics: An Introduction. 2nd ed. Quantitative Applications in the Social Sciences Book. Vol. 79. New York: SAGE Publications Inc.
- Mingers, J. and C. Standing. 2018. What is information? toward a theory of information as objective and veridical. Journal of Information Technology 33, 85–104.
- Navarro G. Searching in metric spaces by spatial approximation. The VLDB Journal, 2002, 11(1): 28–46reateSpace Independent Publishing Platform.
- Navarro G. Searching in metric spaces by spatial approximation. The VLDB Journal, 2002, 11(1): 28–46.
- Pulipaka, G. (2018). The Future of Data Science and Parallel Computing – A Road to Technological Singularity. C
- Singh, A. (2022). Parallel and Distrubuted Computing – Featuring MPI/PVM Implementation. Open HPI.
- Zezula P, Amato G, Dohnal V, Batko M. Similarity Search: the Metric Space Approach. Springer Science & Business Media, 2006