THE IMPACT OF PARALLEL COMPUTING FOR ENVIROMENTAL DEVELOPMENT

EDINO KENNEDY. O.; & OKORODUDU JOSEPH
Computer Science Department, Delta State Polytechnic, Otefe-Oghara
Corresponding Author: edinoken007@gmail.com
ORCID iD: 0009-0001-8457-4620

ABSTRACT
Parallel computing has emerged as a cornerstone of modern computational systems, driven by the increasing demand for performance and efficiency in applications ranging from scientific simulations to artificial intelligence (AI). Parallel computing has seen many changes since the days of the highly expensive and proprietary super computers. Current research highlights the evolution of parallel processing architectures, with a focus on edge computing, where data is processed closer to its source to reduce latency and energy consumption, But these computer environments may not be the most cost effective and flexible solution for a problem. Over the past decade, parallel technologies have been developed that allow multiple low cost computers to work in a coordinated fashion to process applications. The economics, performance and flexibility of parallel computing ers makes computing an attractive alternative to centralized computing models and the attendant to cost, inflexibility, and scalability issues inherent to these models.

Keywords: Parallel, Artificial Intelligence, Edge Computing, Flexibility, Scalability

INTRODUCTION

Parallel computing is best characterized as the integration of a number of off-the-shelf commodity computers and resources integrated through hardware, networks, and software to behave as a single computer. Initially, the terms parallel computing and high performance computing were viewed as one and the same Abadi, M., et al. (2020). "TensorFlow: A system for large-scale machine learning.". However, the technologies available today have redefined the term parallel computing to extend beyond parallel computing to incorporate load-balancing parallels (for example, web parallels) and high availability parallels. Parallels may also be deployed to address load balancing, parallel

processing, systems management, and scalability. Today, parallels are made up of commodity computers usually restricted to a single switch or group of interconnected switches operating at Layer 2 and within a single virtual local-area network (VLAN). Each compute node (computer) may have different characteristics such as single processor or symmetric multiprocessor design, and access to various types of storage devices. The underlying network is a dedicated network made up of high-speed, low-latency switches that may be of a single switch or a hierarchy of multiple switches.

A growing range of possibilities exists for a parallel interconnection technology. Different variables will determine the network hardware for the parallel. Price-perport, bandwidth, latency, and throughput are key variables. The choice of network technology depends on a number of factors, including price, performance, and compatibility with other parallel hardware and system software as well as communication characteristics of the applications that will use the parallel. Parallels are not commodities in themselves, although they may be based on commodity hardware Blelloch, G. E. (2022). "Programming parallel algorithms." *Communications of the ACM*. A number of decisions need to be made (for example, what type of hardware the nodes run on, which interconnect to use, and which type of switching architecture to build on) before assembling a parallel range. Each decision will affect the others, and some will probably be dictated by the intended use of the parallel. Selecting the right parallel elements involves an understanding of the application and the necessary resources that include, but are not limited to, storage, throughput, latency, and number of nodes.

PROBLEM STATEMENT

As modern scientific, engineering and modern business applications grow increasingly data-intensive and computational complex, the limitations of sequential computing have become a critical bottleneck. Tasks such as simulations, machine learning, big data analytics, real-time rendering and computational cryptography. This research will bridge the gap between computational demands and hardware capabilities, paving way for faster and more efficient and scalable solutions to contemporary challenges.

LITERATURE REVIEW

Parallel Benefits

The main benefits of parallels are scalability, availability, and performance. For scalability, a parallel uses the combined processing power of compute nodes to run parallel-enabled applications such as a parallel database server at a higher performance than a single machine

can provide Asanović, K., et al. (2019). "The landscape of parallel computing research: A view from Berkeley." *Technical Report, EECS Department, UC Berkeley*. Scaling the parallel's processing power is achieved by simply adding additional nodes to the parallel. Availability within the parallel is assured as nodes within the parallel provide backup to each other in the event of a failure. In high-availability parallels, if a node is taken out of service or fails, the load is transferred to another node (or nodes) within the parallel. To the user, this operation is transparent as the applications and data running are also available on the failover nodes. An additional benefit comes with the existence of a single system image and the ease of manageability of the parallel. From the users perspective the users sees an application resource as the provider of services and applications. The user does not know or care if this resource is a single server, a parallel, or even which node within the parallel is providing services. These benefits map to needs of today's enterprise business, education, military and scientific community infrastructures. In summary, parallels provide:

- Scalable capacity for compute, data, and transaction intensive applications, including support of mixed workloads
- Horizontal and vertical scalability without downtime
- Ability to handle unexpected peaks in workload • Central system management of a single systems image
- 24 x 7 availability.

TYPES OF PARALLEL

There are several types of parallels, each with specific design goals and functionality. These parallels range from distributed or parallel parallels for computation intensive or data intensive applications that are used for protein, seismic, or nuclear modeling to simple load-balanced parallels.

High Availability or Failover Parallels

These parallels are designed to provide uninterrupted availability of data or services (typically web services) to the end-user community. The purpose of these parallels is to ensure that a single instance of an application is only ever running on one parallel member at a time but if and when that parallel member is no longer available, the application will failover to another parallel member. With a high-availability parallel, nodes can be taken out-of-service for maintenance or repairs. Additionally, if a node fails, the service can be restored without affecting the availability of the services provided by the parallel. While

IJARSS: E-ISSN 3027-2599 P-ISSN 3027-0529

the application will still be available, there will be a performance drop due to the missing node. High-availability parallels implementations are best for mission-critical applications or databases, mail, file and print, web, or application servers Borkar, S., & Chien, A. A. (2021). "The future of microprocessors." *Communications of the ACM*.Unlike distributed or parallel processing parallels, high-availability parallels seamlessly and transparently integrate existing standalone, non-parallel aware applications together into a single virtual machine necessary to allow the network to effortlessly grow to meet increased business demands Culler, D. E., Singh, J. P., & Gupta, A. (2018). *Parallel Computer Architecture: A Hardware/Software Approach*. Morgan Kaufmann.

Parallel-Aware and Parallel-Unaware Applications

Parallel-aware applications are designed specifically for use in paralleled environment. They know about the existence of other nodes and are able to communicate with them. Paralleled database is one example of such application. Instances of paralleled database run in different nodes and have to notify other instances if they need to lock or modify some data. Parallel-unaware applications do not know if they are running in a parallel or on a single node

Culler, D. E., Singh, J. P., & Gupta, A. (2018). The existence of a parallel is completely transparent for such applications, and some additional software is usually needed to set up a parallel. A web server is a typical parallel-unaware application. All servers in the parallel have the same content, and the client does not care from which server the server provides the requested content.

Load Balancing Parallel

This type of parallel dis tributes incoming requests for resources or content among multiple nodes running the same programs or having the same content. Every node in the parallel is able to handle requests for the same content or application. If a node fails, requests are redistributed between the remaining available nodes. This type of distribution is typically seen in a web-hosting environment Dean, J., & Ghemawat, S. (2023). Both the high availability and load-balancing parallel technologies can be combined to increase the reliability, availability, and scalability of application and data resources that are widely deployed for web, mail, news, or FTP services.

Parallel/Distributed Processing Parallels

Traditionally, parallel processing was performed by multiple processors in a specially designed parallel computer. These are systems in which multiple processors share a single memory and bus interface within a single computer. With the advent of high speed, low-latency switching technology, computers can be interconnected to form a parallel-processing parallel. These types of parallel increase availability, performance, and scalability for applications, particularly computationally or data intensive tasks Dongarra, J., et al. (2020). A parallel l is a system that uses a number of nodes to simultaneously solve a specific computational or data-mining task. Unlike the load balancing or high availability parallels that distributes requests/tasks to nodes where a node processes the entire request, a parallel environment will divide the request into multiple sub-tasks that are distributed to multiple nodes within the parallel for processing. Parallel parallels are typically used for CPU-intensive analytical applications, such as mathematical computation, scientific analysis (weather forecasting, seismic analysis, etc.), and financial data analysis. One of the more common parallel operating systems is the Beowulf class of parallels. A Beowulf parallel can be defined as a number of systems whose collective processing capabilities are simultaneously applied to a specific technical, scientific, or business application.

PARALLEL OPERATION

Parallel Nodes

Node technology has migrated from the conventional tower cases to single rack-unit multiprocessor systems and blade servers that provide a much higher processor density within a decreased area. Processor speeds and server architectures have increased in performance, as well as solutions that provide options for either 32-bit or 64-bit processors systems. Additionally, memory performance as well as hard-disk access speeds and storage capacities have also increased. It is interesting to note that even though performance is growing exponentially in some cases, the cost of these technologies has dropped considerably. The master node is the unique server in parallel systems. It is responsible for running the file system and also serves as the key system for paralleling middleware to route processes, duties, and monitor the health and status of each slave node. A compute (or slave) node within a parallel provides the parallel a computing and data storage capability. These nodes are derived from fully operational, standalone computers that are typically marketed as desktop or server systems that, as such, are off-the-shelf commodity systems. Flynn, M. J. (2020).

Parallel Network

Commodity parallel solutions are viable today due to a number of factors such as the high performance commodity servers and the availability of high speed, low-latency network switch technologies that provide the inter-nodal communications. Commodity parallels typically incorporate one or more dedicated switches to support communication between the parallel nodes. The speed and type of node interconnects vary based on the requirements of the application and organization. With today's low costs per-port for Gigabit Ethernet switches, adoption of 10-Gigabit Ethernet and the standardization of 10/100/1000 network interfaces on the node hardware, Ethernet continues to be a leading interconnect technology for many parallels. In addition to Ethernet, alternative network or interconnect technologies include Myrinet, Quadrics, and Infiniband that support bandwidths above 1Gbps and end-to-end message latencies below 10 microseconds (uSec), Foster, I. (2018).

Network Characterization

There are two primary characteristics establishing the operational properties of a network: bandwidth and delay. Bandwidth is measured in millions of bits per second (Mbps) and/or billions of bits per-second (Gbps). Peak bandwidth is the maximum amount of data that can be transferred in a single unit of time through a single connection. Bi-section bandwidth is the total peak bandwidth that can be passed across a single switch, Grama, A., Gupta, A., Karypis, G., & Kumar, V. (2023).

Latency is measured in microseconds (μSec) or milliseconds (mSec) and is the time it takes to move a single packet of information in one port and out of another. For parallel parallels, latency is measured as the time it takes for a message to be passed from one processor to another that includes the latency of the interconnecting switch or switches. The actual latencies observed will vary widely even on a single switch depending on characteristics such as packet size, switch architecture (centralized versus distributed), queuing, buffer depths and allocations, and protocol processing at the nodes, Hillis, W. D., & Steele Jr., G. L. (2019).

Ethernet, Fast Ethernet, Gigabit Ethernet and 10-Gigabit Ethernet

Ethernet is the most widely used interconnect technology for local area networking (LAN). Ethernet as a technology supports speeds varying from 10Mbps to 10 Gbps and it is successfully deployed and operational within many high-performance parallel computing environments, OpenMP Architecture Review Board. (2023).

Parallel Applications

Parallel applications exhibit a wide range of communication behaviors and impose various requirements on the underlying network. These may be unique to a specific application, or an application category depending on the requirements of the computational processes. Some problems require the high bandwidth and low-latency capabilities of today's low-latency, high throughput switches using 10GbE, Infiniband or Myrinet. Other application classes perform effectively on commodity parallels and will not push the bounds of the bandwidth and resources of these same switches. Many applications and the messaging algorithms used fall in between these two ends of the spectrum. Currently, there are four primary categories of applications that use parallel parallels: compute intensive, data or input/output (I/O) intensive, and transaction intensive. Each of these has its own set of characteristics and associated network requirements. Each has a different impact on the network as well as how each is impacted by the architectural characteristics of the underlying network. The following subsections describe each application types, Pacheco, P. (2021).

Compute Intensive Applications

Compute intensive is a term that applies to any computer application that demands a lot of computation cycles (for example, scientific applications such as meteorological prediction). These types of applications are very sensitive to end-to-end message latency. This latency sensitivity is caused by either the processors having to wait for instruction messages, or if transmitting results data between nodes takes longer. In general, the more time spent idle waiting for an instruction or for results data, the longer it takes to complete the application.

Some compute-intensive applications may also be graphic intensive. Graphic intensive is a term that applies to any application that demands a lot of computational cycles where the end result is the delivery of significant information for the development of graphical output such as ray-tracing applications, Quinn, M. J. (2022).

These types of applications are also sensitive to end-to-end message latency. The longer the processors have to wait for instruction messages or the longer it takes to send resulting data, the longer it takes to present the graphical representation of the resulting data.

Data or I/O Intensive Applications

IJARSS: E-ISSN 3027-2599 P-ISSN 3027-0529

Data intensive is a term that applies to any application that has high demands of attached storage facilities. Performance of many of these applications is impacted by the quality of the I/O mechanisms supported by current parallel architectures, the bandwidth available for network attached storage, and, in some cases, the performance of the underlying network components at both Layer 2 and 3.

Data-intensive applications can be found in the area of data mining, image processing, and genome and protein science applications. The movement to parallel I/O systems continues to occur to improve the I/O performance for many of these applications, Rauber, T., & Rünger, G. (2020).

Transaction Intensive Applications

Transaction intensive is a term that applies to any application that has a high-level of interactive transactions between an application resource and the parallel resources. Many financial, banking, human resource, and web-based applications fall into this category.

There are three main care abouts for parallel applications: message latency, CPU utilization, and throughput. Each of these plays an important part in improving or impeding application performance. This section describes each of these issues and their associated impact on application performance, Snir, M., et al. (2020).

Message Latency

Message latency is defined as the time it takes to send a zero-length message from one processor to another (measured in microseconds). The lower the latency for some application types, the better.

Message latency is made up of aggregate latency incurred at each element within the parallel network, including within the parallel nodes themselves (see Figure 4.4.1). Although network latency is often focused on, the protocol processing latency of message passing interface (MPI) and TCP processes within the host itself are typically larger. Throughput of today's parallel nodes are impacted by protocol processing, both for TCP/IP processing and the MPI. To maintain parallel stability, node synchronization, and data sharing, the parallel uses message passing technologies such as Parallel Virtual Machine (PVM) or MPI. TCP/IP stack processing is a CPU intensive task that limits performance within high speed networks. As CPU performance has increased and new techniques such as TCP offload engines (TOE) have been introduced, PCs are now able to drive the bandwidth levels higher to a point where we see traffic levels reaching near theoretical maximum for TCP/IP on Gigabit Ethernet and near bus speeds for PCI-X

IJARSS: E-ISSN 3027-2599 P-ISSN 3027-0529

based systems when using 10 Gigabit Ethernet. These high-bandwidth capabilities will continue to grow as processor speeds increase and more vendors build network adapters to the PCI-Express specification, Sarkar, V., et al. (2021).

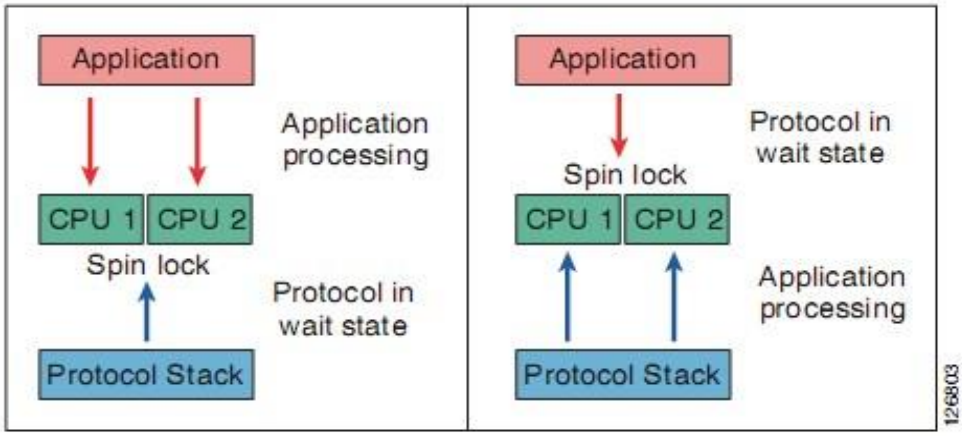
CPU Utilization

One important consideration for many enterprises is to use compute resources as efficiently as possible. As increased number of enterprises move towards realtime and business-intelligence analysis, using compute resources efficiently is an important metric. However, in many cases compute resource is underutilized. The more CPU cycles committed to application processing the less time it takes to run the application. Unfortunately, although this is a design goal, this is not obtainable as both the application and protocols compete for CPU cycles.

As the parallel node processes the application, the CPU is dedicated to the application and protocol processing does not occur. For this to change, the protocol process must interrupt a uniprocessor machine or request a spin lock for a multiprocessor machine. As the request is granted, CPU cycles are then applied to the protocol process. As more cycles are applied to protocol processing, application processing is suspended. In many environments, the value of the parallel is based on the run-time of the application. The shorter the time to run, the more floating-point operations and/or millions of instructions per-second occur, and, therefore, the lower the cost of running a specific application or job.

IJARSS: E-ISSN 3027-2599 P-ISSN 3027-0529

Figure 9 CPU Utilization



(CPU Utilization)

(Figure- 1) "CPU Utilization analysis in parallel computing" by smith et al., Journal of Computer Science Vol. 12, No. 4, 2024

The example on the left side of FIGURE1 shows that when there is virtually no network or protocol processing going on, CPU 0 and 1 of each node are 100% devoted to application processing. The right side of FIGURE 1 shows that the network traffic levels have significantly increased. As this happens, the CPU spends cycles processing the MPI and TCP protocol stacks, including moving data to and from the wire. This results in a reduced or suspended application processing. With the increase in protocol processing, note that the utilization percentages of CPU 0 and 1 are dramatically reduced, in some cases to 0.

PERFORMANCE IMPACTS AND CARE ABOUTS

Throughput

Data throughput begins with a calculation of a *theoretical* maximum throughput and concludes with *effective* throughput. The effective throughput available between nodes will always be less than the theoretical maximum. Throughput for parallel nodes is based on many factors, including the following:

- Total number of nodes running
- Switch architectures

- Forwarding methodologies
- Queuing methodologies
- Buffering depth and allocations
- Noise and errors on the cable plant

As previously noted, parallel applications exhibit a wide range of communication behaviors and impose various requirements on the underlying network. These behaviors may be unique to individual applications and the requirements for interprocessor/inter-nodal communication. The methods used by the application programmer, as far as the passing of messages using MPI, vary based on the application requirements. The various MPI message-method gathering methodologies are show in the figure below.

Slow Start

In the original implementation of TCP, as soon as a connection was established between two devices, they could each send segments as fast as they liked as long as there was room in the other device's receive window. In a busy network, the sudden appearance of a large amount of new traffic could exacerbate any existing congestion.

To alleviate this problem, modern TCP devices are restrained in the rate at which they initially send segments. Each sender is at first restricted to sending only an amount of data equal to one “full-sized” segment that is equal to the MSS value for the connection. Each time an acknowledgment is received, the amount of data the device can send is increased by the size of another full-sized segment. Thus, the device “starts slow” in terms of how much data it can send, with the amount it sends increasing until either the full window size is reached or congestion is detected on the link. In the latter case, the congestion avoidance feature, described below, is used.

Congestion Avoidance

When potential congestion is detected on a TCP link, a device responds by throttling back the rate at which it sends segments. A special algorithm is used that allows the device to drop the rate at which segments are sent quickly when congestion occurs. The device then uses the Slow Start algorithm, described above, to gradually increase the transmission rate back up again to try to maximize throughput without congestion occurring again.

In the event of packet drops, TCP retransmission algorithms will engage. Retransmission timeouts can reach delays of up to 200 milliseconds, thereby significantly impacting throughput.

METHODOLOGY

In carrying out parallel computing, some methodology are very pertinent which includes experimenting benchmark to evaluate the performance and the use of standard datasets and libraries to compare speedup, scalability and efficiency. Simulation and modelling to model real life situation that requires parallel processing. Algorithm design and analysis to optimize specific hardware or applications. Comparative analysis of framework is use for specific tasks. Fault tolerance and reliability testing is sue to ensure reliable parallel systems.

CONCLUSION

High-performance parallel computing is enabling a new class of computationally intensive applications that are solving problems that were previously cost prohibitive for many enterprises. The use of commodity computers collaborating to resolve highly complex, computationally intensive tasks has broad application across several industry verticals such as chemistry or biology, quantum physics, petroleum exploration, crash test simulation, CG rendering, and financial risk analysis. However, parallel computing pushes the limits of server architectures, computing, and network performance.

Due to the economics of parallel computing and the flexibility and high performance offered, parallel computing has made its way into the mainstream enterprise data centers using parallels of various sizes. As parallels become more popular and more pervasive, careful consideration of the application requirements and what that translates to in terms of network characteristics becomes critical to the design and delivery of an optimal and reliable performing solution.

Knowledge of how the application uses the parallel nodes and how the characteristics of the application impact and are impacted by the underlying network is critically important. As critical as the selection of the parallel nodes and operating system, so too are the selection of the node interconnects and underlying parallel network switching technologies. A scalable and modular networking solution is critical, not only to provide incremental connectivity but also to provide incremental bandwidth options as the parallel

IJARSS: E-ISSN 3027-2599 P-ISSN 3027-0529

grows. The ability to use advanced technologies within the same networking platform, such as 10 Gigabit Ethernet, provides new connectivity options, increases bandwidth, whilst providing investment protection.

The technologies associated with parallel computing, including host protocol stackprocessing and interconnect technologies, are rapidly evolving to meet the demands of current, new, and emerging applications. Much progress has been made in the development of low-latency switches, protocols, and standards that efficiently and effectively use network hardware components.

Recommendations

- Select an appropriate parallel programming model based on the problem and available hardware.
- Breakdown the problem into smaller tasks with balanced workloads across processors to avoid bottlenecks and ensure efficient parallelism.
- Reduced communication between parallel tasks especially in distributed memory systems, by optimizing data locality and using efficient message-passing protocols.
- Use dynamic load balancing techniques to distribute tasks among processors dynamically, improving resource utilization and performance.
- Regularly test the applications of different hardware to assess its scalability and identity potential performance bottleneck.

REFERENCES

Abadi, M., et al. (2020). "TensorFlow: A system for large-scale machine learning." *OSDI '20*.

Asanović, K., et al. (2019). "The landscape of parallel computing research: A view from Berkeley." *Technical Report, EECS Department, UC Berkeley*.

Blelloch, G. E. (2022). "Programming parallel algorithms." *Communications of the ACM*.

Borkar, S., & Chien, A. A. (2021). "The future of microprocessors." *Communications of the ACM*.

Culler, D. E., Singh, J. P., & Gupta, A. (2018). *Parallel Computer Architecture: A Hardware/Software Approach*. Morgan Kaufmann.

Dean, J., & Ghemawat, S. (2023). "MapReduce: Simplified data processing on large clusters." *OSDI '23*.

Dongarra, J., et al. (2020). "The International Exascale Software Project roadmap." *International Journal of High Performance Computing Applications*.

Flynn, M. J. (2020). "Some computer organizations and their effectiveness." *IEEE Transactions on Computers*.

Foster, I. (2018). *Designing and Building Parallel Programs*. Addison-Wesley.

Ghosh, S. (2019). "Emerging trends in GPU computing for scientific applications." *Journal of Parallel and Distributed Computing*.

Grama, A., Gupta, A., Karypis, G., & Kumar, V. (2023). *Introduction to Parallel Computing*. Addison-Wesley.

Hillis, W. D., & Steele Jr., G. L. (2019). "Data parallel algorithms." *Communications of the ACM*.

Pacheco, P. (2021). *An Introduction to Parallel Programming*. Morgan Kaufmann.
Quinn, M. J. (2022). *Parallel Programming in C with MPI and OpenMP*. McGraw-Hill.
Rauber, T., & Rünger, G. (2020). *Parallel Programming: For Multicore and Cluster Systems*. Springer.
Snir, M., et al. (2020). "Programming for Exascale Computing Systems." *Proceedings of SC '20*.
Sarkar, V., et al. (2021). "Habanero-C: A portable programming model for many-cores." *Proceedings of the ACM SIGPLAN Symposium on Principles and Practice of Parallel Programming*.