https://doi.org/10.3311/PPee.37547

Periodica Polytechnica Electrical Engineering and Computer Science, 69(1), pp. 33–42, 2025

Implementation of Parallel Applications on Linear Systolic and Star Topologies by Using Multistage Omega Network

Qusay Samir Alsaffar[1*], Leila Ben Ayed[2]

[1] National School of Electronics and Telecommunications of Sfax, University of Sfax, 3029 Sfax, P.O.B. 1169, Tunisia
[2] National School of Computer Science Tunisia, University of Manouba, Campus Universitaire de La Manouba, 2010 Manouba, Tunisia
[*] Corresponding author, e-mail: qusay_saffar@mohesr.gov.iq

Received: 27 May 2024, Accepted: 18 October 2024, Published online: 28 October 2024

Abstract

Contemporary computer architecture comprises multicomputer settings. Multiple computers provide the facility of high performance to implement multiple threads that are faster and concurrently with different processors that can execute tasks simultaneously. The challenges are as follows: cost and high performance. Certain firms have to provide funding to establish data centers that need locations, hardware, and technicians. After the passage of a period of time, such equipment requires maintenance (updating and upgrading) to take place on it. This paper tackles these challenges by passing the drawbacks of data centers. It provides an algorithm that simulates a virtual machine, as one client is to be connected with eight servers to implement two applications namely convolution and mathematical operations. To represent different topologies, the algorithm simulates supercomputer systems by applying multistage omega network topology and parallel processing. Two types of topologies execute linear systolic and star, and is implemented on transmission control protocol (TCP) and user datagram protocol (UDP) protocols. The goal of this paper is to provide a kernel that draws near a cloud computing system by exploiting the JAVA programming language with its techniques (threading, socket, socket server, Datagram socket, and datagram packet). The results offer the connection between the client and servers to be successfully simulated by using multistage omega network. The linear systolic and star topologies are simulated in virtual parallel processing.

Keywords

multistage omega network, linear systolic topology, star topology, parallel processing, JAVA, threading, cloud system

1 Introduction

Multistage interconnection networks (MINs) are constructed to be a path to implement several missions simultaneously, and they make the rate of cost a lessor one. It allows a variety of applications to be processed by using parallel processing in real-time applications which need many processors as the processing power of a single processor which isn't enough for meet the requests of these applications [1, 2].

The P size of MINs is composed of $\log_2 P$ stages, each stage of which consists of $P/2$ nodes of size 2×2 and provides fascinating utilities such as a small number of switches, an effective algorithm of routing, and partition ability. Some models of topologies are represented by $\log_2 P$ stages and these are multistage omega network, Butterfly, and Baseline [3].

Omega relates to banyan-type MINs [4], and has the primary properties of banyan such as self-routing devices, where one input port is connected to one output port through a path. The routing algorithm (perfect shuffle) is provided in Omega networks. A switch receives a packet in its in-port and redirects the packet to a destination through it's out-porting according to packet address. This operation is called (self-routing). This routing algorithm works by rotating to the left destination of the packets. The connection permutation continues as it goes through the stages of the Omega network [5].

The main benefit of utilizing multiprocessors is to provide sturdy processing through connecting the multi-processors. The fastest single-processor can't reach the power of a multiprocessor [6]. The construction of a powerful single processor is with some costs, so the constructing of a multiprocessor includes many single processors which is more of cost-effectiveness. Moreover, fault tolerance is regarded as another advantage of a multiprocessor system.

Cite this article as: Alsaffar, Q. S., Ben Ayed, L. "Implementation of Parallel Applications on Linear Systolic and Star Topologies by Using Multistage Omega Network", Periodica Polytechnica Electrical Engineering and Computer Science, 69(1), pp. 33–42, 2025. https://doi.org/10.3311/PPee.37547

If a processor is down, then it can convert the task of this processor into another processor [7].

Parallel processing consists of different types, the most common of which are single instruction multiple data (SIMD) and multiple instructions multiple data (MIMD) [8]. SIMD is a type of parallel processing where the system consists of more than one processor and each of which executes diverse types of data and corresponds to the same group of instructions. MIMD is a system consisting of more than one processor, and each one of which processes its routine and gets data from various flows [9].

Transmission control protocol (TCP) transfers the data in a two-directional network and guarantees a successful delivery, and it can control congestion facility. The user datagram protocol (UDP) transfers the data from a certain source to its destination in a one-directional network, yet still there is no acknowledgment whether it can be obtained when data is received [10, 11].

This research presents two models that are proposed to overcome the data center problems (cost saving, hardware, etc.), namely the simulates multistage omega network and parallel processing. The first model is convolution and is implemented by using linear systolic topology and is executed on TCP protocol. The second model, however, is 8 mathematical operations and is implemented by the use of star topology and is executed on UDP protocol. The two models use JAVA (threading, socket server, socket, datagram socket, and datagram packet). The application data is split into threads and then is sent from a specific client to 8 computers(servers) to achieve the task. The connection between a selected client and servers is enabled by the use of switching of multistage omega network and parallel processing, and the communication is then implemented by using liner systolic (a convolution application) and star (mathematical operations) topology simultaneously.

This research attempts to reach the cloud computing systems by introducing a small sample to represent the multiple topologies.

The research is organized as follows: the Related works are stated in Section 2, while the convolution and mathematical operations are illustrated in Section 3. The parallel connection using the multistage omega network is explained in details in Section 4, leaving the space for Section 5 to describe the distributed system topologies. In Section 6 the cloud system is briefly defined. The proposed system and algorithms are detailed in Section 7. The results that are obtained are illustrated in details in Section 8, and Section 9 of the research reveals the conclusions.

2 Related works

Zheng et al. [12], developed a simulator that depends on the tracing parallel discrete events. It simulates a completed paradigm for processing and communication (processors and network) with rational detailing. The simulator permits the valuation of a variety of levels of details in the IN: from small latency to the detailed paradigms consisting of d-ary m-trees, d-ary, and m-cubes of the network. The best characteristics of the simulator are the utmost modularity that is to be at work with a simple technique to model specific modern routing algorithms and topologies. The system permits the processing of big simulations of existing and future models in parallel implementation form, and surveys the actions of applications that are built for those same models. The system is formatted at the time of processing, which means that any update in the paradigm is obligated to reprocess the destination units.

Keshk [13], used RMI based on JAVA threads to multiply the matrix in parallel. He used four computers; that were divided into one RMI for the server and three ones as clients. 2-matrices of size $n \times n$ are created by the server and every client gets the first matrix that is connected *via* a particular row (relying on client ID). The output rows are sent by the clients to the server. The output of multiplying two rows is gathered by the server to give the result. The output demonstrates that using matrices of size (256×256) causes a speedup of up to 2.2 (reducing the amount of multiplication time to 52.5). Such work can be developed by providing more than one server.

Argollo et al. simulated an entire system environment [14]. This system uses a network-switching simulator and it is an open source which can simulate a group of systems multi-core. The method is very simple for network modelling and cannot simulate a bigger network.

Denzel et al. developed a simulation of parallel systems [15], the main function of which is to evaluate the parallel applications. The system consists of detailed paradigms of nodes and communication. The system provides different types of routing and switching functions with many multistage topologies. There are multi-processing cores that are supported by this system and each core of which has a MPI stack. The system is executed in parallel and is regarded as powerful in corresponding with MPI, yet it is not scalable and can support a few thousand nodes.

Badia et al. [16] designed a simulator to evaluate the applications behavior. It utilizes the parallel application for tracing in any architecture for remodelling the running of that application. The model is designed to compute the

interconnection network in a simple method with accuracy in computing element models. The payloads that are used with the simulator are designed in details, with many important states that are provided for the application thread. The problem with this simulator is that it is sophisticated to get the traces with a convenient level of detail and that needs a provided kernel. The model is designed to seek unbalances that can affect the execution of parallel applications, and this means that the bottlenecks that happen in the network cannot be identified [16].

Zhou et al. developed a simulator as a supercomputer. The simulator is precision cycle-driven and is accurate in its setting as a real system. The system is not scalable and is caused by restricted to a 32-destination node [17].

3 Convolution and mathematical operations

This research applies the convolution and mathematical (addition, multiplication, subtraction, division, mod, power, max, and min) operations in a multistage omega network. In convolution, there are two arrays: the F is called filter and the A is called input, and consisting of N and M integers respectively. The task is to construct a convolution array C of size $(N + M - 1)$. The convolution of two arrays is defined as in the following Eq. (1). [18]:

$$C[i+j] = \sum_{0}^{k-1} \left(a[i] * b[j] \right), \tag{1}$$

where $a[]$ and $b[]$ are two arrays and C is the result, i, j are the elements of arrays.

4 Parallel connection using multistage omega network

To execute the algorithm, the researchers chose the omega network which is a type of MIN. The XOR between the source and destination binary number nodes represents the path. The Omega network is constructed by multiple stages. Each stage consists of S/2 switching elements [19]. A single switch consists of two ports (crossover and passthrough) for connection status. In the passthrough state, input is sent straight path to the output, in the crossover state, the exchange of the path from input to output [20] as it is shown in Fig. 1 [21].

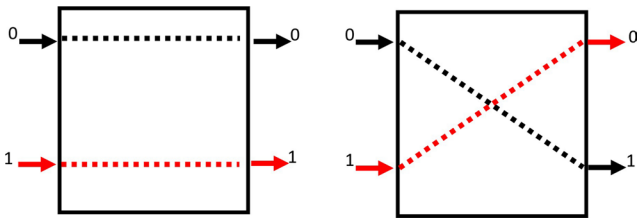

Fig. 1 2 × 2 switching connection [21]

The $\log_2 P$ represents the stages of the Omega network. The P inputs are connected to the outputs through an interconnection mode for every stage. The connection between input i and output j can be explained in Eq. (2), [22], as illustrated in Fig. 2:

$$j = \begin{cases} 2i, & 0 \le i \le \dfrac{P}{2} - 1 \\ 2i + 1 - P, & \dfrac{P}{2} \le i \le P - 1 \end{cases}, \tag{2}$$

where j input the ID to the next stage, i is the node ID to input to switch and P is the number of clients and servers.

5 Distributed system topologies

In this research, the linear systolic topology is chosen to execute the communication on the convolution operation. The filter array elements are sent from the selected client to the servers where each element is sent to each server, while all the input array elements are sent to each server as it will be explained in algorithm section. The filter and input array elements are sent as threads. This mechanism represents the SIMD parallel processing. The transmission threads between the client and servers and between servers themselves are achieved simultaneously. Fig. 3 demonstrates this mechanism. The TCP protocol is exploited to execute the communication.

The star topology is chosen to execute the communication on the mathematical operations. The selected client sends a request to a specific server to do the operation (sending two numbers as threads), and the threads are then transmitted simultaneously. Fig. 4 demonstrates the star topology. The UDP protocol is applied to execute the communication.

6 Cloud system

Cloud computing introduces cloud services to different users, and these services are provided by remote locations from third parties [23].

These services are classified as IaaS (Infrastructure as a Service), PaaS (Platform as a Service), and SaaS (Software as a Service) [24]. These three technologies can offer cost-effective and optimal use of resources to a great extent [25]. In this research, the SaaS cloud computing system for parallel processing network topology is designed.

7 Proposed system

This research classifies the proposed algorithm into three parts: (connection between client and server algorithm, convolution algorithm, and mathematical operations algorithm).

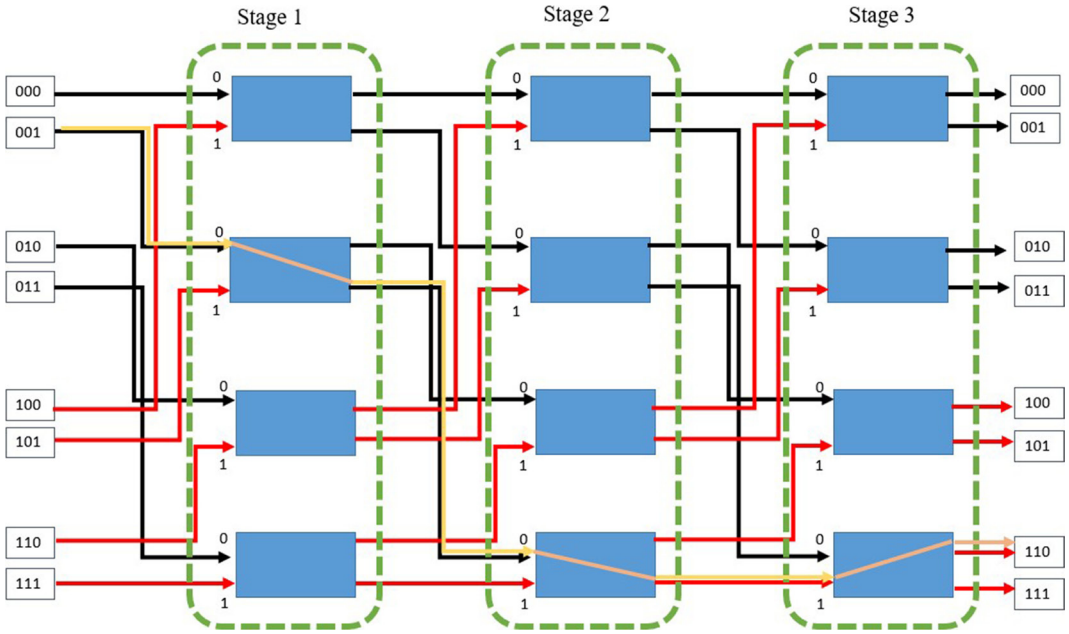

Fig. 2 The multistage omega network [22]

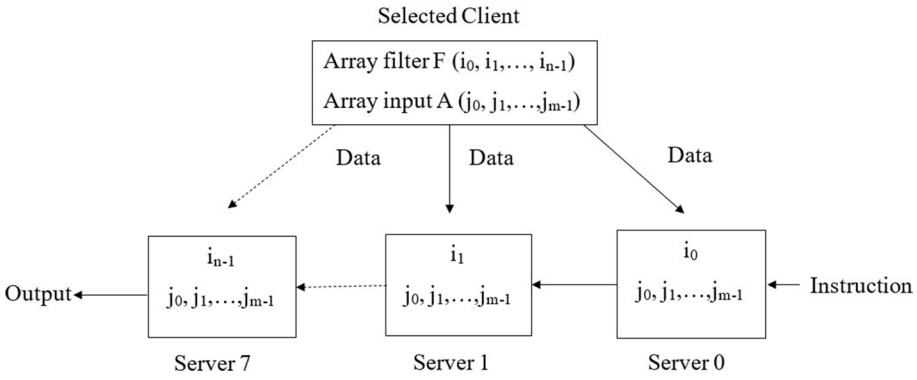

Fig. 3 Communication using linear systolic topology

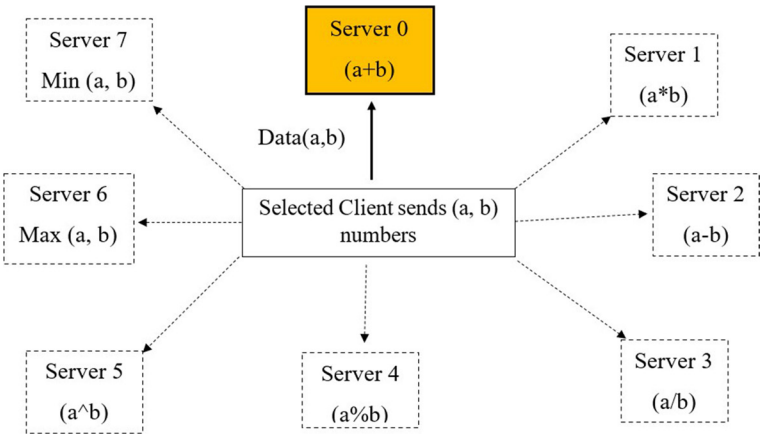

Fig. 4 Communication using star topology

7.1 Connection between client and server algorithm

To attain the connection between a client and server in the multistage omega network, the path between the source and destination nodes has to be determined. The path passes through three stages and each stage consists of four switches. At first of four clients, the entrance into each switch is port 0, while in the second four clients, the entrance of which is port 1, as follows in Algorithm 1.

Algorithm 1 The connection between client and server

1. **Input:** i, P, client C_{id}, server S_{id}
2. **Output:** Destination server
3. Calculate the number of stages $M = \log_2(P)$
4. Convert C_{id} to 3-bit binary form $(C[\,]_{id})$ and S_{id} to 3-bit binary form $(S[\,]_{id})$
5. $L[\,]_{id} = (C[\,]_{id})$ XOR $(S[\,]_{id})$, the most significant bit represents the first stage, the least significant bit represents the M stage
6. Determine port IDs of switches from 0 to $P-1$ according to
   If $(0 \le i \le P/2-1)$ **Then** ID = $2 \times i$
   Else If $(P/2 \le i \le P-1)$ **Then** ID = $2 \times i + 1 - P$
7. **If** L [the significant bit] = 0 **Then** the switch operates in Straight mode
8. **If** L [the significant bit] = 1 **Then** the switch operates in Exchange mode
9. The next stage starts returning from step 6 until you reach the final stage M

   End

7.2 The convolution algorithm

Algorithm 2 selects one client and 8 servers. There are two arrays: the first one is filter F. It convolutes on the second array A to get the results. The client sends one value from F elements to each server as a thread and then sends all elements of A to each server as threads. There is a queue in each server. The results are accumulated at the last server.

The process of sending the threads (F and A) between the client and servers and (S_{new}) between servers themselves is performed concurrently in parallel processing that is applied on linear systolic topology through the use of socket, socket server and TCP protocol.

7.3 The mathematical operations algorithm

There are eight mathematical operations that are implemented in Algorithm 3, namely: (addition, multiplication, subtraction, division, mod, power, maximal and minimal value). By using the parallel processing to simulate the star topology, the servers work simultaneously to execute the operation and send the results back to the client through the use of the datagram socket, datagram packet and UDP protocol.

Fig. 5 demonstrates the activity diagram of the connection process between the client and server.

Fig. 6 (activity diagram) reflects the parallel processing of the convolution by transmitting threads by using the socket and socket server.

Fig. 7 activity diagram explains the parallel processing of the mathematical operations by transmitting threads using the Datagram packet and datagram socket.

Algorithm 2 The convolution

1. **Input:** $F(i_0, i_1, \ldots, i_7)$, $A(j_0, j_1, \ldots, j_7)$, 1 client and 8 servers
2. **Output:** Convolution array C
3. **Foreach** (x = 0 to 7) **do**
   $S_{current} = F \times A[x]$
   Put $S_{current}$ into q
   Take $S_{current}$ from the q
   $S_{new} = S_{current} + S_{previous}$ //in server 0 the $S_{previous}$ is 0
   Send S_{new} to the next server and it will be the $S_{previous}$

End

Algorithm 3 The mathematical operation

1. **Input:** first value, second value, 1 client and the server that is exclusive for the operation
2. **Output:** a result of the operation
3. The client sends 2 numbers to the specific server as a threads
4. The server processes the mathematical operation and sends back the result to the connected client as a thread.

End

8 Illustration and experimental results

Section 8 explains the results that were obtained, the first part related to the connection process, the second part explains the convolution process, and the third part explains the mathematical operations.

8.1 The connection between client and server process

The enabling of a successful connection from client no. (6) that will send a thread to the server no. (7) that will provide a specific task, as it is illustrated in Fig. 8:

1. The number of nodes(P) = 8, Client no. (C_{id}) = 6, server no. (S_{id}) = 7.
2. Convert C_{id} and S_{id} numbers from decimal to binary form where C_{id} = 110 and S_{id} = 111.
3. The number of stages $M = \log_2(P) \rightarrow M \log_2(8) = 3$.
4. (110) XOR (111) = (001).
5. $(P/2 \le i \le P-1) \rightarrow (8/2) \le 6 \le 8-1 \rightarrow$ ID = $2 \times i + 1 - P$ $\rightarrow$ ID = $2 \times 6 + 1 - 8 \rightarrow$ ID = 5 (the first stage entrance is 5 as illustrated in Fig. 8).
6. The significant bit of the XOR result is 0 port, the switch operates in straight mode (Fig. 8).
7. $(P/2 \le i \le P-1) \rightarrow (8/2) \le 5 \le 8-1 \rightarrow$ ID = $2 \times i + 1 - P$ $\rightarrow$ ID = $2 \times 5 + 1 - 8 \rightarrow$ ID = 3 (the second stage entrance is 3 as illustrated in (Fig. 8).
8. The second bit of the XOR result is 0 port, the switch operates in a straight.
9. $(0 \le i \le P/2-1) \rightarrow (0 \le 3 \le 8/2-1) \rightarrow$ ID = $2 \times i \rightarrow$ ID = $2 \times 3 = 6$ (the third stage entrance is 6).
10. The least significant bit of the XOR result is 1 port, the switch operates in exchange mode.
11. The destination server is (111) and the connection is done.

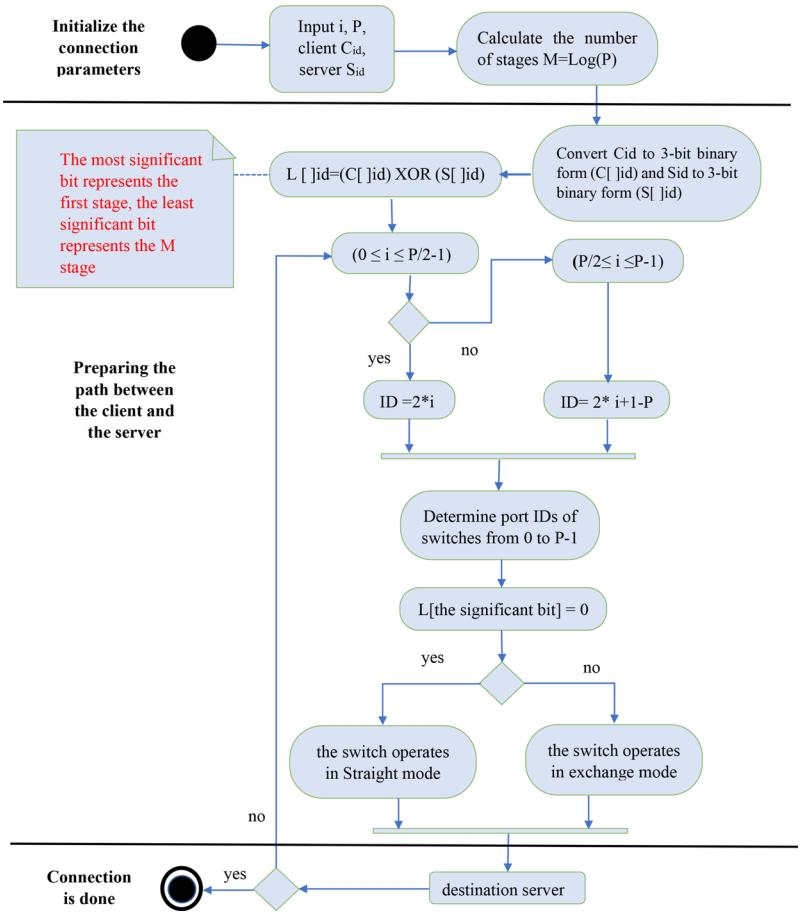

Fig. 5 The connection process between the client and the server

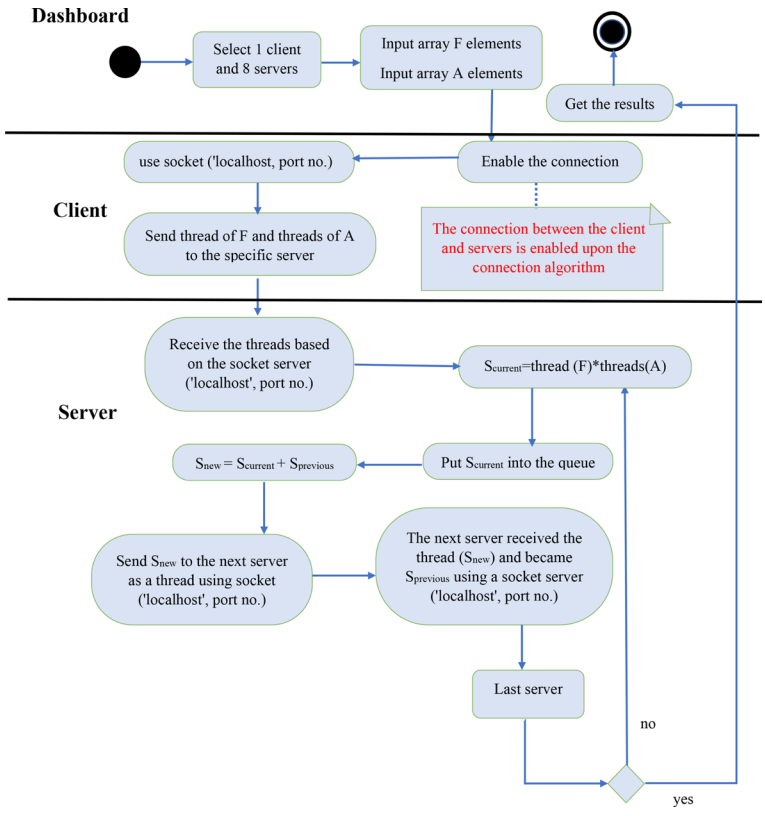

Fig. 6 The parallel processing of the convolution

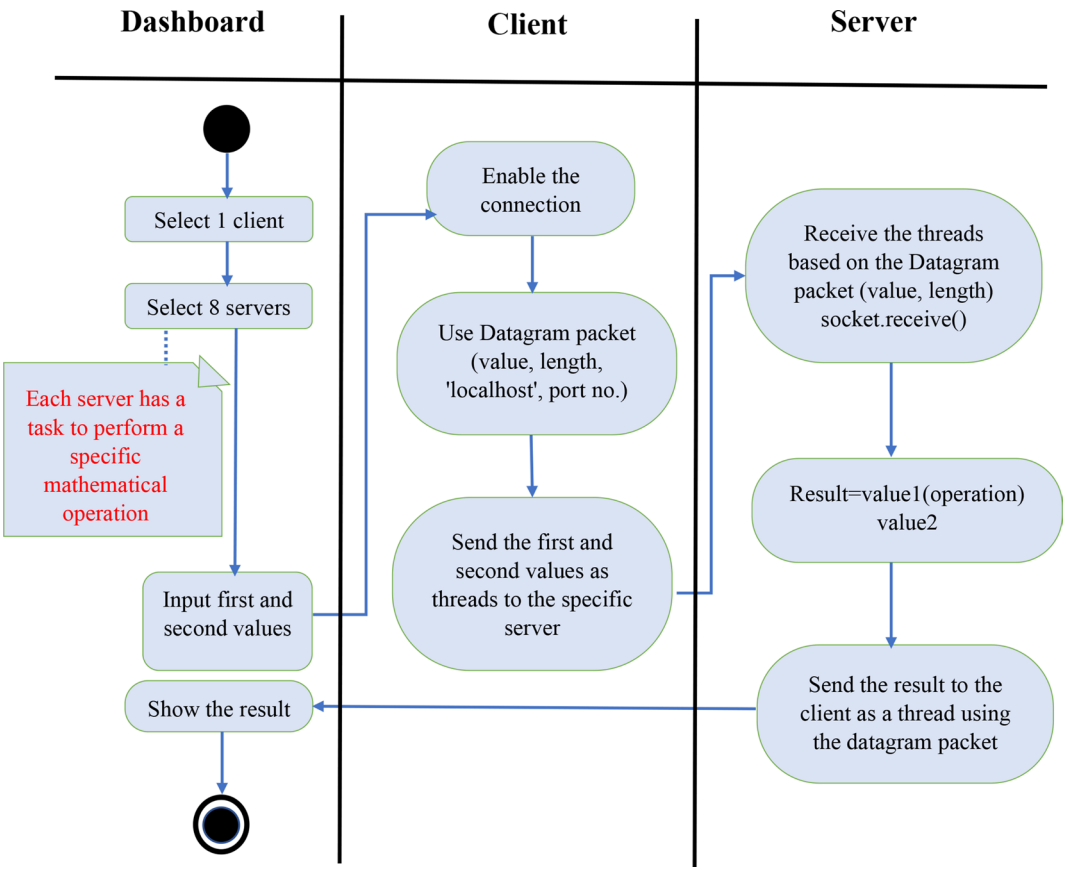

Fig. 7 The parallel processing of the mathematical operations

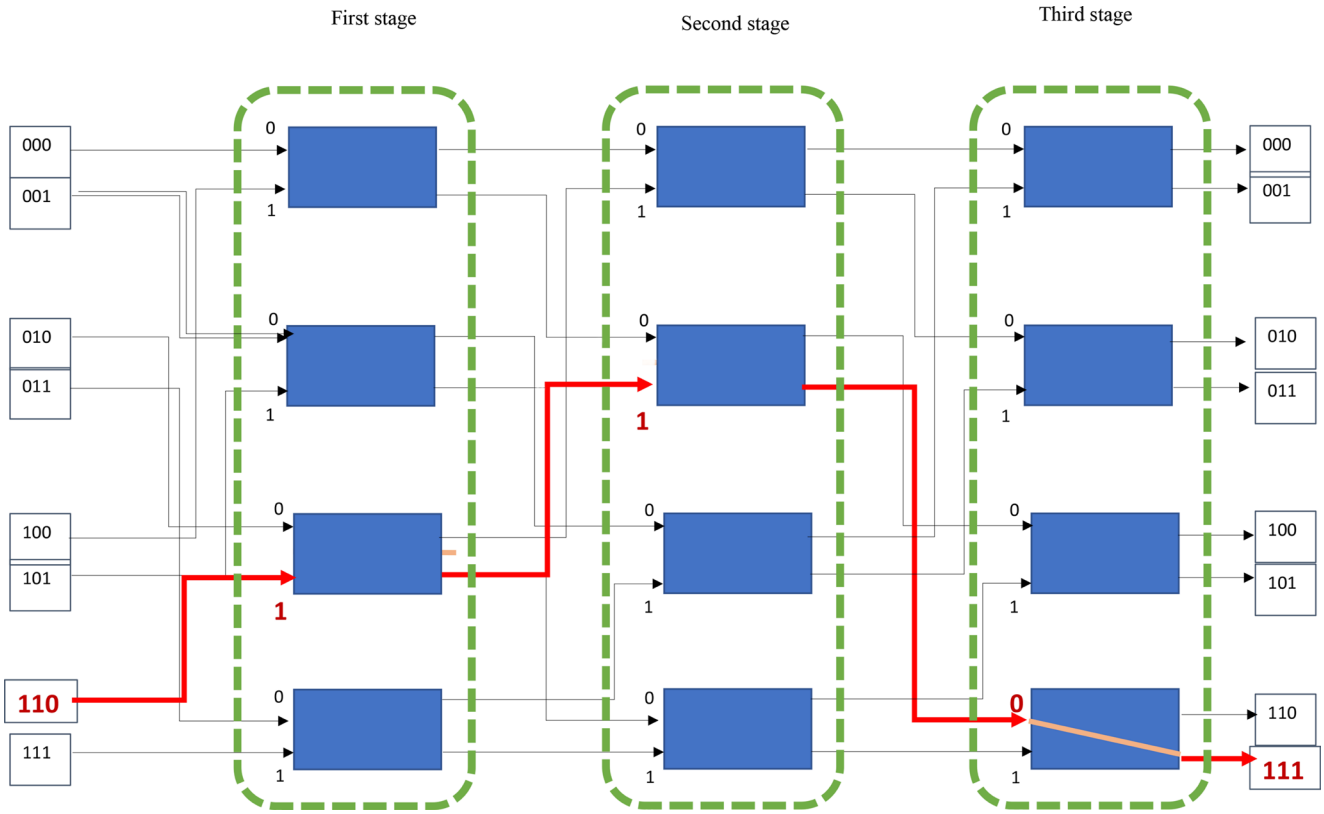

Fig. 8 Connection between client no. 6 and server no. 7

8.2 Linear systolic topology and parallel processing for convolution

Section 8.2 explains the use of linear systolic topology to achieve the parallel processing and simultaneous communication between processors to calculate the results through sending and processing the threads, as it is illustrated in Fig. 9:

1. Select 1 client and 8 servers.
2. Input *F* (1, 2, 3, …, 8), *A* (9, 10, 11, …, 16).
3. Apply the connection algorithm.
4. The selected client sends the first value of the filter (1) to server_0 the second value (2) to server 2, last value (8) to server 7 as a thread, by using socket ('localhost', port no.).
5. The selected client sends the array values (9, 10, 11, …, 16) to each server as threads, using a socket ('localhost', port no.).
6. The servers receive all threads of (*F* and *A*) by using the Socket server ('localhost', port no.).
7. Server_0 processes the threads by multiplying the thread (1 × 9 = 9) and then putting the result in the queue (9).
8. Server_0 takes (9) from the queue, then add 9 + 0 = 9.
9. Then repeat the process by multiplying *F* (1) with the rest of the (*A*) threads and putting them into the queue, then take them from the queue and add to 0.
10. Server_0 sends the results of summation to the next server as threads using a socket('localhost', port no.).
11. Server_1 receives the threads from the selected client and server_0, using the socket server ('localhost', port no.).
12. The next server repeats the same process from point (7), but *F* will be (2), and the queue has two indexes then put the thread that received from server_0 in the first queue index, then put the multiplication result in the second index (9 × 2 = 18), then it repeats the process.
13. The final results of the summation are aggregated at server_7.

8.3 Star topology and parallel processing for the mathematical operations

Section 8.3 explains the using of star topology to achieve the parallel processing of the 8 mathematical operations, where each server processes a specific operation, and Fig. 10 illustrates the communication between the client and servers:

1. Select 1 client and 1 server.
2. Input 6 and 5 values.
3. Apply the connection algorithm.
4. The client sends the values (6 and 5) to the servers as threads by using DatagramPacket (value, length, 'host', port).

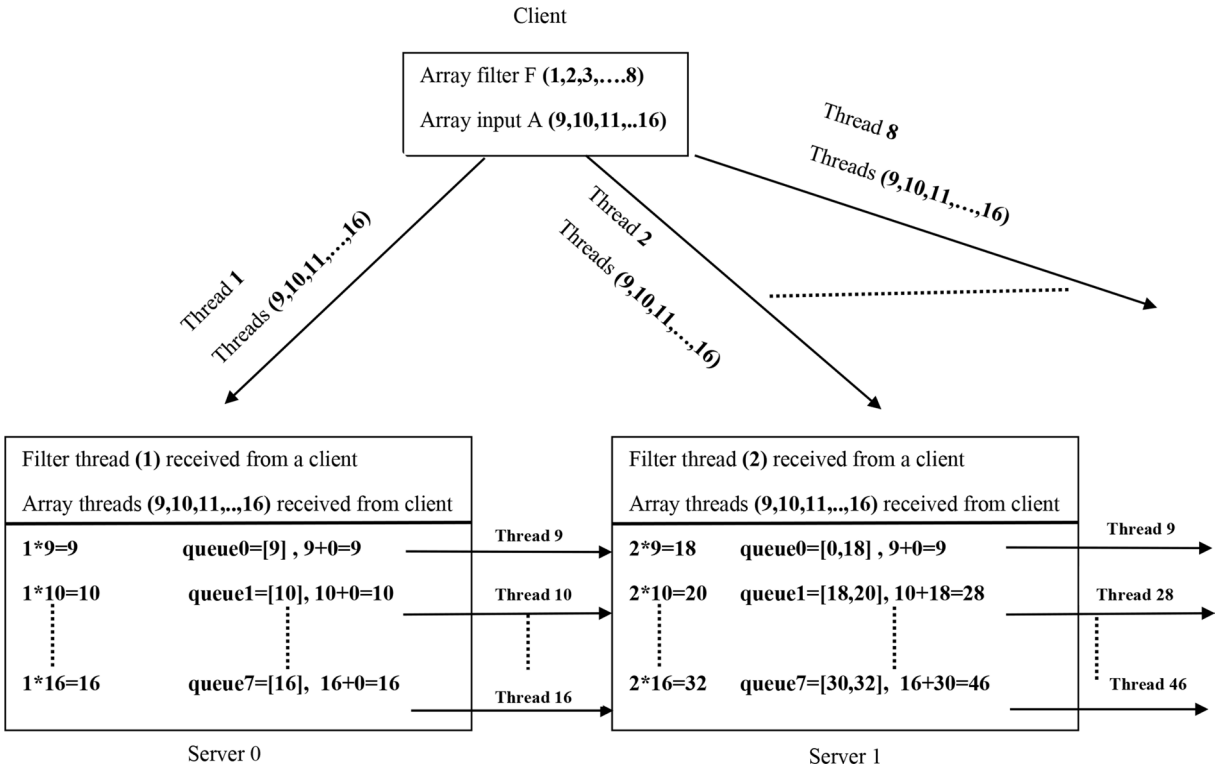

Fig. 9 The linear systolic topology of parallel processing to execute the convolution

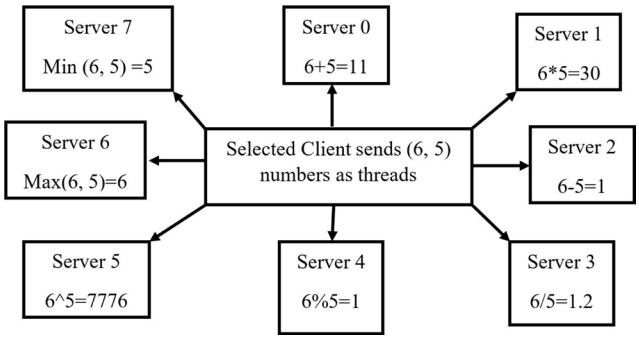

Fig. 10 The star topology of parallel processing to execute the mathematical operation

5. The servers receive the threads by using DatagramPacket (value, length) and socket.receive().
6. Each server achieves the specific task and gives the result.

9 Conclusions

This research makes a simulation of the parallel processing of a virtual network as a physical network. The connection between the client and servers is successfully enabled through the use of a multistage omega network. The JAVA programming language and its techniques (threading, socket, socket server, datagram packet, and datagram socket) are used to execute the system. The linear systolic topology is applied to execute the convolution. Star topology is applied to execute the mathematical operations. In these two topologies, the connection between the client and servers, in addition to the communication between servers are successfully applied through transmitting the threads and all servers process them concurrently. The TCP and UDP protocols are used to achieve the communication. The virtualization and variety of topologies and protocols that are applied in this research give a sample of cloud computing systems. The trend towards virtual machines and the cloud system provides the most recent resources and is considered the best choice to be done. It is also by passing the drawbacks of data centers since it saves costs. Later on, the data centers become outdate. The amount of resources that is provided in the cloud system is increased when the users become in need for more and it decreases when the users need less. The system uses 3 stages omega network and this can extend to more stages, and thus the system is scalable. This research discussed the construction of a supercomputer that can be done by applying 8 virtual clients and 8 virtual servers in a single physical machine.

The algorithm can be developed by further future work in providing a special mechanism by converting the task of a server to another one when the first server is down (fault tolerance) and by the use of another type of interconnection network topology.

References

[1] Amodu, O. A., Othman, M., Yunus, N. A. M., Hanapi, Z. M. "A Primer on Design Aspects and Recent Advances in Shuffle Exchange Multistage Interconnection Networks", Symmetry, 13(3), 378, 2021. https://doi.org/10.3390/sym13030378

[2] Prakash, A., Yadav, D. K., Choubey, A. "A Survey of Multistage Interconnection Networks", Recent Advances in Electrical & Electronic Engineering, 13(2), pp. 165–183, 2020. https://doi.org/10.2174/1872212113666190215145815

[3] Izzi, D., Massini, A. "Realizing Optimal All-to-All Personalized Communication Using Butterfly-Based Networks", IEEE Access, 11, pp. 51064–51083, 2023. https://doi.org/10.1109/ACCESS.2023.3279494

[4] Stergiou, E., Garofalakis, J., Liarokapis, D., Margariti, S. V. "A Study of Multilayer Omega Networks Operating with Cut-Through Mechanism under Uniform Traffic", In: 2020 5th South-East Europe Design Automation, Computer Engineering, Computer Networks and Social Media Conference (SEEDA-CECNSM), Corfu, Greece, 2020, pp. 1–7. ISBN 978-1-7281-6446-5 https://doi.org/10.1109/SEEDA-CECNSM49515.2020.9221819

[5] Md Yunus, N. A., Othman, M., Hanapi, Z. M., Kweh, Y. L. "Evaluation of Replication Method in Shuffle-Exchange Network Reliability Performance", In: Advances in Data and Information Sciences: Proceedings of ICDIS 2017, Amarkantak, India, 2017, pp. 271–281. ISBN 978-981-13-0276-3 https://doi.org/10.1007/978-981-13-0277-0_22

[6] Shukla, S. K., Murthy, C. N. S., Chande, P. K. "A Survey of Approaches used in Parallel Architectures and Multi-core Processors, For Performance Improvement", In: Progress in Systems Engineering: Proceedings of the Twenty-Third International Conference on Systems Engineering, Las Vegas, NV, USA, 2015, pp. 537–545. ISBN 978-3-319-08421-3 https://doi.org/10.1007/978-3-319-08422-0_77

[7] Eijkhout, V. "Parallel Programming in MPI and OpenMP", Lulu.com, 2017. ISBN 1387400282

[8] Tino, A., Collange, C., Seznec, A. "SIMT-X: Extending Single-Instruction Multi-Threading to Out-of-Order Cores", ACM Transactions on Architecture and Code Optimization, 17(2), 15, 2020. https://doi.org/10.1145/3392032

[9] Xiao, F., Zhan, C., Lai, H., Tao, L., Qu, Z. "New parallel processing strategies in complex event processing systems with data streams", International Journal of Distributed Sensor Networks, 13(8), 1550147717728626, 2017. https://doi.org/10.1177/1550147717728626

[10] Soomro, M. A., Channa, M. I., Zardari, B. A., Nizamani, S. Z., Arain, A. A., Ahmed, M. "Performance Evaluation of TCP and UDP Protocols under EPLAODV Routing Protocol in Emergency Situations", Quaid-E-Awam University Research Journal of Engineering, Science & Technology, 20(1), pp. 29–36, 2022. https://doi.org/10.52584/QRJ.2001.05

[11] Kanellopoulos, D. N., Wheeb, A. H. "Simulated Performance of TFRC, DCCP, SCTP, and UDP Protocols Over Wired Networks", International Journal of Interdisciplinary Telecommunications and Networking (IJITN), 12(4), pp. 88–103, 2020. https://doi.org/10.4018/IJITN.2020100107

[12] Zheng, G., Wilmarth, T., Jagadishprasad, P., Kalé, L. V. "Simulation-Based Performance Prediction for Large Parallel Machines", International Journal of Parallel Programming, 33(2), pp. 183–207, 2005. https://doi.org/10.1007/s10766-005-3582-6

[13] Keshk, A. E. "Implementation of Distributed Application using RMI Java threads", In: 2007 IEEE International Symposium on Signal Processing and Information Technology, Giza, Egypt, 2007, pp. 1017–1022. ISBN 978-1-4244-1834-3 https://doi.org/10.1109/ISSPIT.2007.4458214

[14] Argollo, E., Falcón, A., Faraboschi, P., Monchiero, M., Ortega, D. "COTSon: Infrastructure for full system simulation", ACM SIGOPS Operating Systems Review, 43(1), pp. 52–61, 2009. https://doi.org/10.1145/1496909.1496921

[15] Denzel, W. E., Li, J., Walker, P., Jin, Y. "A Framework for End-to-End Simulation of High-performance Computing Systems", Simulation, 86(5–6), pp. 331–350, 2010. https://doi.org/10.1177/0037549709340840

[16] Badia, R. M., Labarta, J., Gimenez, J., Escalé, F. "DIMEMAS: Predicting MPI Applications Behavior in Grid Environments", Workshop on Grid Applications and Programming Tools (GGF8), 86, pp. 52–62, 2003. [online] Available at: https://www.researchgate.net/publication/242562299_DIMEMAS_Predicting_MPI_applications_behavior_in_Grid_environments [Accessed: 05 November 2023]

[17] Zhou, W., Chen, J., Cui, C., Wang, Q., Dong, D., Tang, Y. "Detailed and clock-driven simulation for HPC interconnection network", Frontiers of Computer Science, 10(5), pp. 797–811, 2016. https://doi.org/10.1007/s11704-016-5035-3

[18] Gonzalez, R. C., Woods, R. E. "Digital Image Processing", Pearson, 2018. ISBN 0-201-18075-8

[19] Stergiou, E., Garofalakis, J., Liarokapis, D., Margariti, S. "Investigating Multilayer Omega-Type Networks Operating with the Cut-Through Technique under Uniform or Hotspot Traffic Conditions", International Journal of Computer Networks & Communications (IJCNC), 13(5), pp. 89–109, 2021. [online] Available at: https://ssrn.com/abstract=3960712 [Accessed: 05 November 2023]

[20] Stergiou, E., Liarokapis, D., Margariti, S., Bobotsaris, I. "Omega Multistage Interconnection Network with High-speed Forwarding Technique Handling a Double-pattern Load", In: 2022 International Conference on Information Technologies (InfoTech), Varna, Bulgaria, 2022, pp. 1–4. ISBN 978-1-6654-6871-8 https://doi.org/10.1109/InfoTech55606.2022.9897116

[21] Md Yunus, N. A., Othman, M., Mohd Hanapi, Z., Lun, K. Y. "Reliability Review of Interconnection Networks", IETE Technical Review, 33(6), pp. 596–606, 2016. https://doi.org/10.1080/02564602.2015.1130595

[22] Hamarsheh A. "Exploiting Omega Networks to Hide Text-in-Text Messages", International Journal of Computer Science and Network Security (IJCSNS), 15(5), pp. 39–43, 2015. [online] Available at: https://www.researchgate.net/publication/277547179_Exploiting_Omega_Networks_to_Hide_Text-in-Text_Messages#fullTextFileContent [Accessed: 05 November 2023]

[23] Rashid, A., Chaturvedi, A. "Cloud Computing Characteristics and Services: A Brief Review", International Journal of Computer Sciences and Engineering, 7(2), pp. 421–426, 2019. https://doi.org/10.26438/ijcse/v7i2.421426

[24] Wulf, F., Lindner, T., Westner, M., Strahringer, S. "IaaS, PaaS, or SaaS? The Why of Cloud Computing Delivery Model Selection – Vignettes on The Post-Adoption of Cloud Computing", In: Proceedings of the 54th Hawaii International Conference on System Sciences, Honolulu, HI, USA, 2021, pp. 6285–6294. ISBN 978-0-9981331-4-0 https://doi.org/10.24251/HICSS.2021.758

[25] Bani Baker, Q., Hammad, M., Al-Rashdan, W., Jararweh, Y., AL-Smadi, M., Al-Zinati, M. "Comprehensive comparison of cloud-based NGS data analysis and alignment tools", Informatics in Medicine Unlocked, 18, 100296, 2020. https://doi.org/10.1016/j.imu.2020.100296