Wasit Journal of Engineering Sciences

Journal homepage: https://ejuow.uowasit.edu.iq

Parallel Processing Distributed-Memory Approach Influences on Performance of Multicomputer-Multicore Systems Using Single-Process Single-Thread

Dildar Masood Abdulqader[1] | Subhi R. M. Zeebaree[2] | Rizgar R. Zebari[3] | Mohammed A. M.Sadeeq[2] | Umed H. Jader[4] | Mohammed Mahmood Delzy[4]

Affiliations
[1]Information Technology Dept., Akre University for Applied Science, Duhok, Iraq
[2] Energy Eng. Dept., Duhok Polytechnic University Duhok, Iraq
[3]Computer Science Dept., Knowledge University, Duhok, Iraq
[4]Information Technology Dept., Erbil Polytechnic University, Erbil, Iraq

Correspondence
Subhi R. M. Zeebaree,
Energy Eng. Dept., Duhok Polytechnic University
Duhok, Iraq
Email:subhi.rafeeq@dpu.edu.krd

Accepted
5-January-2024
Doi: 10.31185/ejuow.Vol12.Iss1.533

Abstract
Based on client/server architecture concepts, this research suggests a method for creating a multicomputer-multicore distributed memory system that can be implemented on distributed-shared memory systems. Both of number of the participated computers and number of existed processors for each of these computers, this research was depended with the specific design and its implementation. The suggested system has two primary phases: monitoring and managing the programmes that may be executed on multiple distributed-multi-core architectures with (2, 4, and 8) CPUs to perform a certain job. There might be a single client and unlimited servers in the network. The implementation phase relies on three separate scenarios covering most of the design space. The suggested system can determine the start time, duration, CPU use, kernel time, user time, waiting time, and end time for each server in the system. Single-Process Single-Thread (SPST) is considered a possible situation while developing User Programmes (UPs). The findings confirmed that more processing power (more servers and more processors on each server) increases the speed at which tasks can be solved. There was a 2.877-fold gain in task processing speed after considering three different possible SPST UPs situations. The C# programming language is used to create this system.

Keywords: distributed system, parallel processing, process/threads monitoring, process/threads controlling

الخلاصة: يقترح هذا البحث منهجًا لإنشاء نظام ذاكرة موزعة متعدد الحواسيب متعدد النوى يعتمد على مفاهيم بنية عميل/خادم، والذي يمكن تنفيذه على أنظمة الذاكرة المشتركة الموزعة. يعتمد تصميم وتنفيذ هذا البحث على عدد أجهزة الكمبيوتر المشاركة وعدد المعالجات الموجودة في كل منها. يتكون النظام المقترح من مرحلتين رئيسيتين : مراقبة وإدارة البرامج التي يمكن تنفيذها على بنيات متعددة النوى موزعة متعددة (بـ 2 و 4 و 8 معالجات) لأداء مهمة معينة .يمكن أن يكون هناك عميل واحد وعدد غير محدود من الخوادم في الشبكة. تعتمد مرحلة التنفيذ على ثلاثة سيناريوهات منفصلة تغطي معظم مجال التصميم. يمكن للنظام المقترح تحديد وقت البدء والمدة واستخدام وحدة المعالجة المركزية ووقت النواة ووقت المستخدم ووقت الانتظار ووقت الانتهاء لكل خادم في النظام .يعتبر Single-Process Single-Thread (SPST) حالة ممكنة عند تطوير برامج المستخدم (UPs) . أكدت النتائج أن قوة المعالجة الأكبر (مزيد من الخوادم والمزيد من المعالجات على كل خادم) تزيد من سرعة حل المهام . كان هناك ربح بمقدار (2.877) ضعفًا في سرعة معالجة المهام بعد النظر في ثلاثة مواقف مختلفة محتملة لبرامج UP SPST . تم استخدام لغة البرمجة C# لإنشاء هذا النظام.

1. INTRODUCTION

The computer component, which we shall refer to as a "node," may either be a hardware device or a software process, necessitating some kind of cooperation between the independent nodes [1-3]. High-performance computing workloads are other important types of distributed systems. In cluster computing, central machines

are made up of a collection of workstations or other comparable computers that are fast-connected to a local area network. Moreover, the same operating system is used by each node [4], [5]. Since it is used virtually every day for a range of applications, client-server systems have gained a lot of popularity. A software architecture consisting of a client and a server, in which clients always issue requests when the server replies to the requests provided, is referred to as a client-server system. Due to the data interchange between the client and server, each doing a distinct job, the client server facilitates inter-process communication. [6], [7].

Multi-core CPUs are already giving PCs, laptops, and business servers a new level of performance. Several of these systems must also adhere to strict guidelines for low weight, low power use, and low heat dissipation [8], [9]. These needs are directly met by multi-core processors, which provide much greater computing power per ounce, per watt, and per square inch than traditional processors [10], [11]. The performance and fairness of the system are significantly impacted by the efficient distribution of resources across the various accounts that fight for shared resources in multi-core systems [12], [13].

One CPU is insufficient for various applications, which leads to several issues with these systems, including poor operation and lengthy reaction times. We efficiently spread the computational load among the current CPUs, but sometimes a multiprocessor system is necessary. Thus, it is essential to break the overall work down into smaller tasks and properly arrange the execution sequence of these smaller activities [14], [15]. Multiprocessor computers, on the other hand, provide a supportive environment and are more potent than single processor systems for running particular programs. As a result, research into planning in a multiprocessor system is ongoing. Similar to single processor layout, multiprocessor central scheduling, and distributed scheduling, each job in such systems executes on a multiprocessor system. Real-time planning involves intricate load calculations. This implies that compared to single processor algorithms, planning algorithms in multiprocessor systems are substantially more difficult [16], [17]. A technique called multithreading enables a program or process to carry out many tasks concurrently. It enables the process to run in parallel mode on a system with only one CPU [18], [19]. Multithreading expands the idea of multi-tasking in applications, where you can split specific processes into individual threads within a single application. The operating system allocates processing time between different applications and between each thread within the application. Using proprietary hardware that provides multiple threads increases CPU usage and thus reduces overall program execution time [20], [21].

By gathering information on the performance of an application or a system, performance monitoring may be used to identify bottlenecks. The single-core or multi-core design of the processor unit determines how thread (process) monitoring systems operate [22]. Data placement, process affinity, or load balancing are a few performance challenges where this knowledge is crucial. The analysis is crucial for performance improvement. A utility is presented to execute thread (process) migrations and monitor program performance. This tool gathers data about all system threads and processes in order to optimize a shared memory program with parallel aims [23]. The need to executing complex problems with as minimum as possible of processing time benefiting from the system's resources, was depended as a motivation for this research.

2. LITERATURE REVIEW

In 2022, Z. N. Rashid et al. [24] presented a system to assist users in doing composite tasks interactively with the least processing time. Distributed-Parallel-Processing and Cloud Computing are the two most great technologies, can process and answer the user problem quickly. Hash codes are generated on the client-side and sent towards the webserver. The webserver delivers these does to cracking servers that have been specified. It has been verified that while employing light load (single hash-code) with multi-servers and multiprocessors, the suggested system gives improved efficiency (in terms of Kernel-burst, User-bust, and Total-execution) timings. Although it has been demonstrated that employing large loads (many under-testing codes) with many computing machines using multiprocessors improves the system's performance.

The CPU taking too long to employ computational marine hydrodynamics tokens with restricted cloud resources through private clouds in containers was a brand-new and significant problem that was solved by Y. Xu, et. al. [25] in 2019. A fresh match with the task resource is suggested by the proposed scheduling algorithm to promote efficient communication between service providers and end users. The simulation results demonstrate that, in terms of quick scheduling, resource use, and overall performance, our technique provides advantages over the underlying algorithms.

In 2020, Bianco et al. [26] presented the architecture and how it was implemented using open-source operating systems (OS) and Ethernet network interface cards (NICs). As this design can provide high-speed packet transmission, it has drawn growing attention from the research community. They suggested that by expanding

the number of potential pathways and priority classes for each path, their system might be adjusted. In order to implement a more sophisticated justice policy, it was also feasible to change the unit that distributes resources in the NIC's output during the scheduling algorithm's response phase.

integrated link and attribute, lexicographic breadth first search, approximation algorithm for shortest route length based on center-division zone distance, and axis-vertex for area and principal highway are some of the methods used by Z. Lv, D. Chen, and A. K. Singh [27] in 2021. The findings demonstrate that the calculation time for the framework is the fastest with 4 CPU threads (514.63 ms). The total computation time of the framework continuously lowers as the number of CPU cores rises. The number of jobs rises by one for every extra two CPU cores. The algorithm's arithmetic scale factor may be adjusted to 0.06 for less complicated networks and 0.2 for more common networks. The processing time, average query time, and total query time of the method are the shortest when compared to other algorithms in various datasets; they are 49.67 ms and 5.12 ms, respectively.

L. M. Haji,et. al. [28] in 2021, focused on developing a shared-memory parallel processing system that provided the abilities to monitor and control the processes and threads of their system. Also, they proved that their system could be run on number of multi-core system architectures. The algorithms associated with their work were designed for providing the ability to: provide information from the server computer system, check the status of all current processes with relevant information, and execute all possible threads/processes states that made up a user program.

3. METHODOLOGY

The proposed Process/Threads Monitoring and controlling distributed system (PTMCDS) consists of two main sides (client and servers), and working in three proposed scenarios to process the created load (by client-side) at the servers-side. The control part of the proposed system illustrated in Figure 1. Also, the data message is shown in Figure 2 and the mechanism of the proposed system operation is demonstrated in Figure 2. There are two main types of the messages transferred between client-side and servers-side witch are (control and data) messages. The complete system is shown in figure 3.

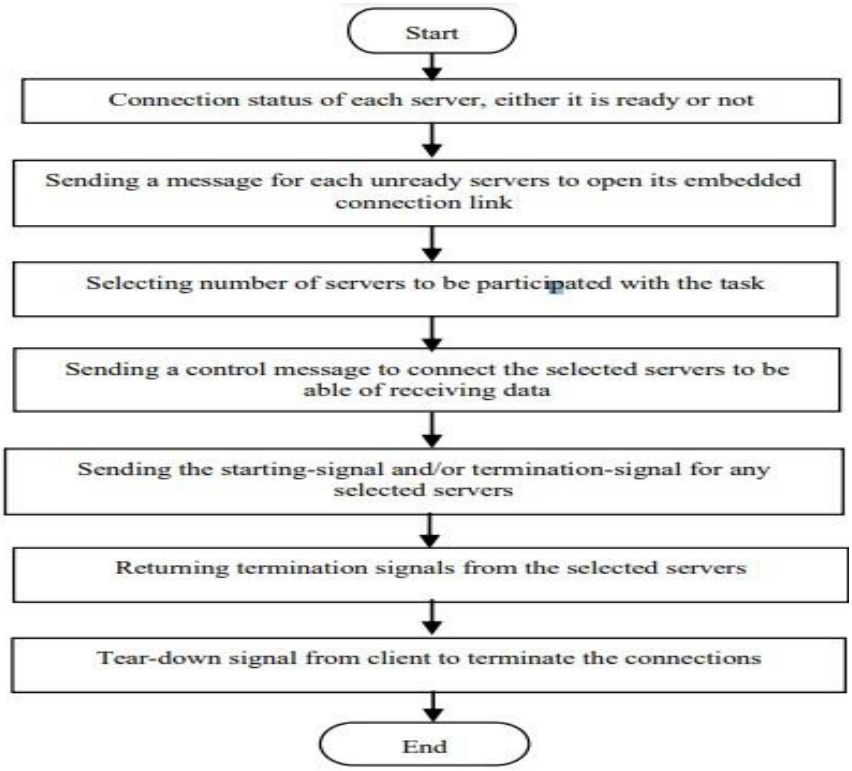

Figure 1 Block Diagram of control message.

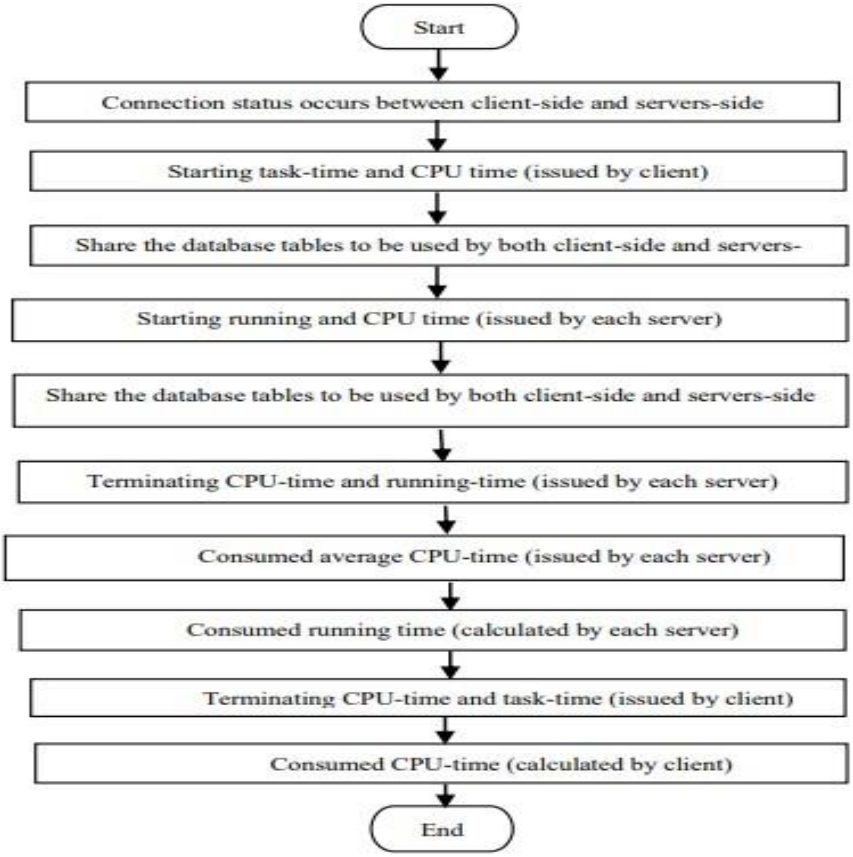

Figure 2 Block Diagram of data-messages.

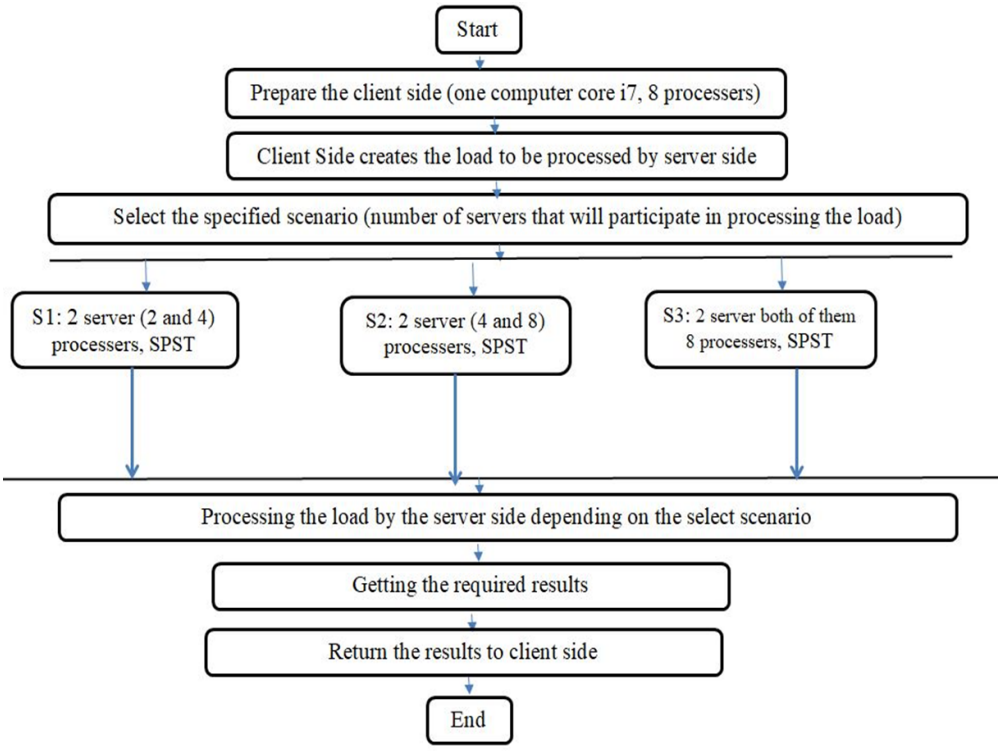

Figure 2 General diagram of proposed mechanism.

3. 1. Scenarios of Proposed System

Researchers often use one or more scenarios that explain these algorithms and get answers faster than a single processing technology to demonstrate the value of parallel processing approaches. Also, the techniques of numbers-sorting can be a suitable scenario study as an application to display the effects of Parallel Processing on the time consuming of execution and efficiency of processing approach. Four techniques of numbers-sorting are depended in this paper, which need multi server-computers to explain the advantages of Parallel Processing approach upon single processing approach. These techniques are: Selection Sort, Insertion Sort, Bubble Sort and Shakar Sort.

The sorting operations will be repeated for different orders to illustrate the effects of the mixed parallel processing approach as the load increases. In this study, the same client-host been depended with core i7 (8 processer). So, only features of the server-hosts is explained:

S1: Two servers core two duo (2 processors) and core i3 (4 processors) used for 2 UP, SPST.

S2: Two servers core i3 (4 processors) and core i7 (8 processors) used for 2 UP, SPST.

S3: Two servers both of them core i7 (8 processors) used for 2 UP, SPST.

3.2. Monitoring Implementation and Results

The proposed system was implemented using one UP until a hundred UPs, with single or multi processes, and each process considered to be single or multi threads. The MI component is applicable to cover all of the essential aspects of system. There are three different scenarios that are put into action, and their outcomes are tracked and analyzed before being presented in proper formats (Tables and Plots), such as those shown in Tables (1 to 3) and Figures 4, 5, and 6.

Scenario-1: Two servers core 2 duo (2 processors) and core i3 (4 processors) used for (SPST).

TABLE 1 Results of Scenario-1, two servers core two duo and core i3

	Client	Server1	Server2
IP	192.168.1.1	192.168.1.2	192.168.1.3
N_core	8	2	4
Proccess_Name	startproccess	SPST11	SPST12
Start_time	17:46:33:359	16:44:44:623	15:19:31:169
Elapsed_Time (ms)	1222923	1222813	1141714
Kernel_Time (ms)	116031	31	171
User_Time (ms)	360125	1220489	1120165
Total_CPU Time (ms)	476156	1220520	1120336
Waiting_Time (ms)	746767	2293	21378
End_time	18:06:56:280	17:05:07:474	15:38:32:895

The server's properties are (CPU-Type= 2Duo, RAM=4 GB, No. of Core=2, CPU-Frequency=2.2 GHz) and (CPU-Type= i3-2350, RAM=4 GB, No. of Core=4, CPU-Frequency=2.3 GHz). Table I illustrates the obtained

results related to monitoring total running of one UP, single process, and single thread. While elapsed, user, kernel, total CPU and waiting times are plotted as shown in Figure 4. Where 11 in SPST11 means that it is process-1 in UP-1.

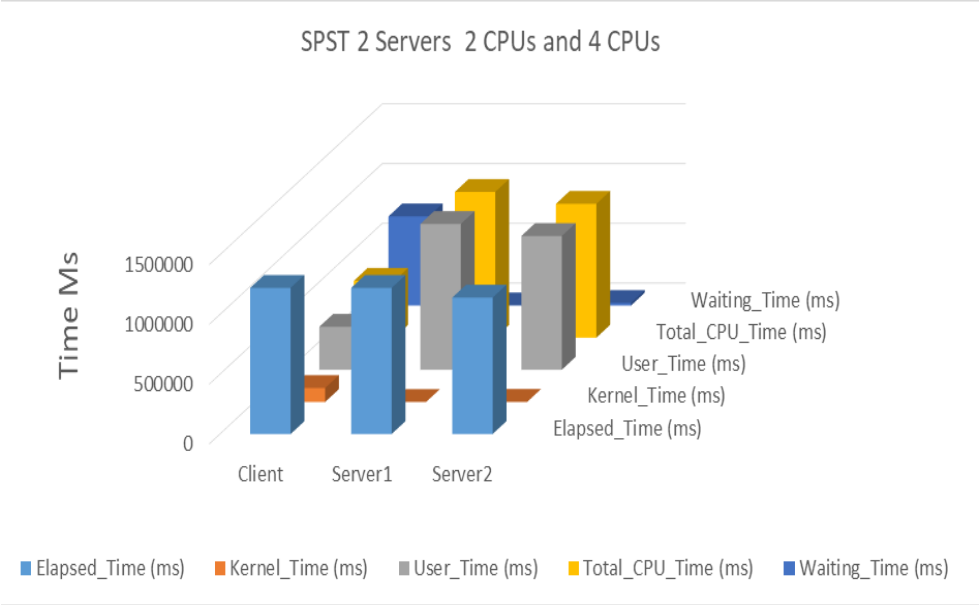

Figure 4 Plotted results of scenario-1 (SPST).

Scenario-2: Two servers core i3 (4 processors) and Corei7 (8 processors) used for (SPST).

The server's properties are (CPU-Type= i3-2350, RAM=4 GB, No. of Core=4, CPU-Frequency=2.3 GHz) and (CPU-Type= i7-6600, RAM=8 GB, No. of Core=8, CPU-Frequency=2.7 GHz).

Table 2 illustrates the obtained results related to monitoring total running of one UP, single process, and single thread. While elapsed, user, kernel, total CPU and waiting times are plotted as shown in Figure 5.

Table 2 Results of Scenario-2, two servers' core i3 and core i7 (SPST)

	Client	Server1	Server2
IP	192.168.1.1	192.168.1.3	192.168.1.4
N_core	8	4	8
Proccess_Name	startproccess	SPST11	SPST21
Start_time	01:19:31:232	14:19:31:169	09:34:44:998
Elapsed_Time (ms)	1141846	1141714	425203
Kernel_Time (ms)	88734	171	62
User_Time (ms)	279359	1120165	414447
Total_CPU_Time (ms)	368093	1120336	414509
Waiting_Time (ms)	773753	21378	10694
End_time	01:38:33:083	14:38:32:895	09:41:50:339

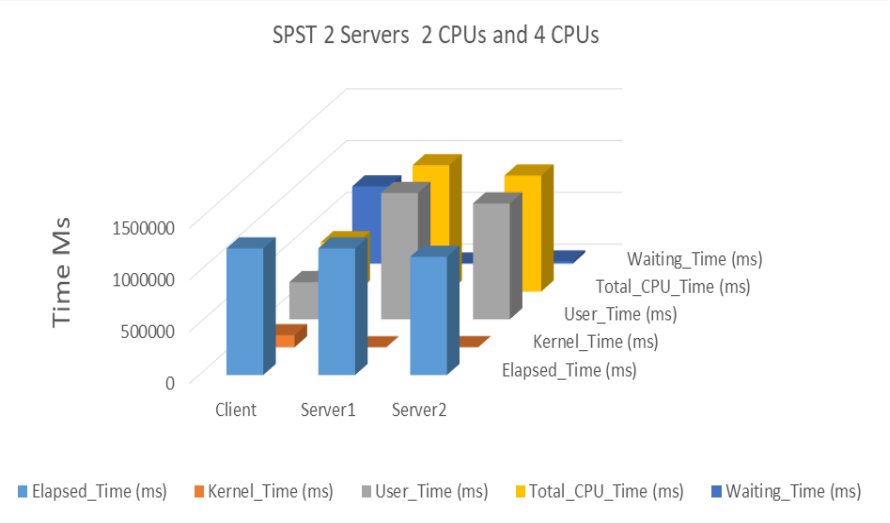

Figure 5 Plotted results of scenario-2 (SPST).

Scenario-3: Two servers both of them core i7 (8 processors) used for (SPST). The server's properties are (CPU-Type= i7-10700, RAM=8 GB, No. of Core=8, CPU-Frequency=3.8 GHz) and (CPU-Type= i7-6600, RAM=8 GB, No. of Core=8, CPU-Frequency=2.7 GHz). Table 3 illustrates the obtained results related to monitoring total running of one UP, single process, and single thread. While elapsed, user, kernel, total CPU and waiting times are plotted as shown in Figure 6.

Table 3 Results of Scenario-3, two servers both of them core i7 (SPST)

	Client	Server 1	Server 2
IP	192.168.1.1	192.168.1.4	192.168.1.5
N_core	8	8	8
Proccess_Name	startproccess	SPST11	SPST21
Start_time	09:24:53:652	09:34:44:998	01:50:20:998
Elapsed_Time (ms)	425062	425203	421341
Kernel_Time (ms)	42390	62	46
User_Time (ms)	138234	410447	402342
Total_CPU_Time (ms)	180625	410509	402388
Waiting_Time (ms)	244437	14694	18953
End_time	09:31:58:718	09:41:50:339	01:57:22:339

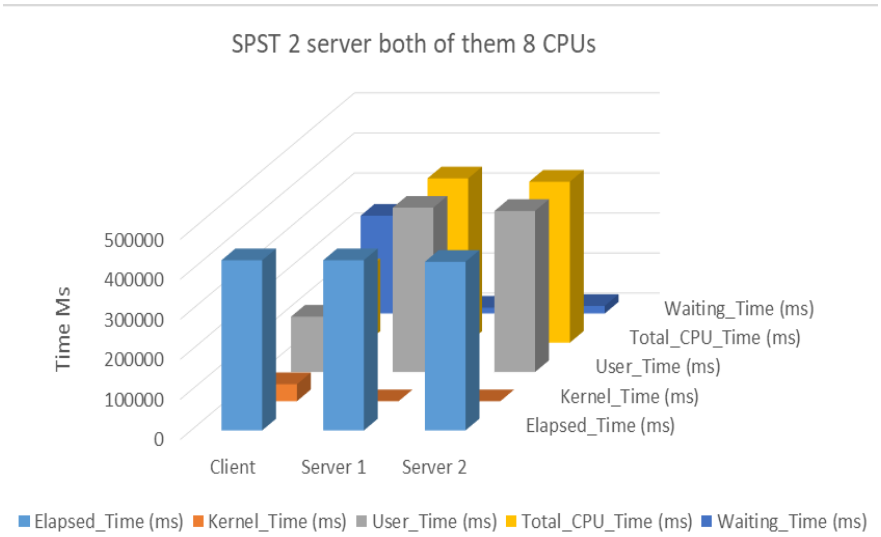

Figure 6 Plotted results of scenario-3 (SPST).

4. DISCUSSION OF OBTAINED RESULTS FROM PTMCDS SCENARIOS

Tables (1, 2, and 3) show that when using one server the Total-Task-Time (TTT) of single UP for SPST is reduced with increasing number of cores. For Scenario-1, when using a server of core two due type, the TTT was 1,222,923 ms. While, for Scenario-2, when using a server of core i3 type, the TTT was 1,141,846 ms. Finally, for Scenario-3, when using a server of core i7 type, the TTT was 425,062 ms. Hence, the TTT (from Scenario-1 to Scenario-3) is decreased by 797,861 ms (i.e. the processing speed increased by 2.877 times) as shown in Figure 5.

5. COMPARISON WITH PREVIOUS WORKS

Table 4 Comparison with Previous Works

Ref.	Technique	Operating System And Cores	Programing Language	Significant Results
[24]	sorting Algorithm and Hash-Codes	Windows (server, 7 ,10)	Visual C#	The result Consumed less time than other expected scenarios (minimal breaking-time besides cost effective usage of computer resources).
[25]	scheduling algorithm	Windows server2016, Core (4 processor)	java	The result is a 6 times improvement, a 44% volume reduction, and an average defined satisfaction rate, respectively
[26]	scheduling algorithm	Linux, PCI-X core	LOGIC design	It is processed and managed by the operating system in software, reducing load performance.
[27]	N-SPFA algorithm	Linux, PCI-X core	C# language	The processing time, average query time, and total query time of the algorithm are the shortest, being 49.67 ms, 5.12 ms, and 94.720 ms, respectively.
[28]	sorting Algorithm	Windows 10, core i5, core i7	C# language	Controlling and monitoring Single/Multiple Processes/Threads in multicomputer parallel processing system. Reduce the execution time for CPU.

Proposed System	**sorting Algorithm**	**Windows 10,** **Core 2 Duo, core i3, core i7**	**C# language**	**Controlling and monitoring Single/Multiple Processes/Threads in multicomputer parallel processing distributed system.** **Reduce the Client Total-Task-Time, server (elapsed and CPU) times.**

Depending on the properties shown in Table 4, the following important distinctions can be extracted:

- Load Type: The proposed PTMCDS deals with four sorting algorithms (which can be extendable to any number of sorting algorithms), giving it unlimited workload possibilities.
- Operating System: Microsoft Windows is not an open-source operating system and this was the biggest challenge for the proposed PTMCDS system. However, this problem was fixed and the system was designed to run under this operating system, while many of previous works mentioned above were done with open-source operating systems.
- Core Types: PTMCDS worked with four types of cores (i7, i5, i3 and core 2due), while others worked with one or two cores. In addition, PTMCDS can automatically work with any number of servers to participate in the task-processing automatically, and any core types without restrictions.
  Another important point of the proposed PTMCDS that differs from all the previous works addressed is that PTMCDS uses the power of the distributed systems for parallel processing, and at the same time utilizing the efficiency of shared memory systems for each server individually. While none of the previous works treated in this style, which gives PTMCDS more performance power. Remember that parallel processing is a very important technology today to speed up the solution of very complex problems, so it is very important to address the performance measurement of this technology.

6. CONCLUSION

In this research, the use of a Process/Threads Monitoring Controlling Distributed System (PTMCDS) is recommended as a means of providing such monitoring and controlling for UPs. The following paragraphs provide a summary of the most important findings of this article:

1) First: The CPU impact and real-time event recording are included into the integrated control choices for all procedures. Because of the restriction of using just one flavour of operating system and one version of that operating system, the problem was solved. It is anticipated that this technique would work well with all Windows OS variants.

2) To facilitate efficient parallel processing: The proposed system made use of OpenMP Communication as (shared memory) and MPI Communication as (distributed memory), two popular parallel processing methodologies.

3) According to the results: There is an increase in client-side task processing performance of a factor of 2.877 between Scenario 1 and Scenario 3.

As a future work, it is suggested to depend on Multicomputer-Multicore Systems Using Multi-Process Multi-Thread.

REFERENCES

[1] Z. N. Rashid, S. R. Zeebaree, R. R. Zebari, S. H. Ahmed, H. M. Shukur, and A. Alkhayyat, “Distributed and Parallel Computing System Using Single-Client Multi-Hash Multi-Server Multi-Thread,” in 2021 1st Babylon International Conference on Information Technology and Science (BICITS), 2021, pp. 222–227.

[2] Z. N. Rashid, S. R. Zeebaree, M. A. Sadeeq, R. R. Zebari, H. M. Shukur, and A. Alkhayyat, “Cloud-based Parallel Computing System Via Single-Client Multi-Hash Single-Server Multi-Thread,” in 2021 International Conference on Advance of Sustainable Engineering and its Application (ICASEA), 2021, pp. 59–64.

[3] H. Shukur, S. Zeebaree, R. Zebari, D. Zeebaree, O. Ahmed, and A. Salih, “Cloud Computing Virtualization of Resources Allocation for Distributed Systems,” Journal of Applied Science and Technology Trends, vol. 1, no. 3, pp. 98–105, 2020.

[4] Z. N. Rashid, S. R. Zeebaree, and A. Shengul, “Design and Analysis of Proposed Remote Controlling Distributed Parallel Computing System Over the Cloud,” in 2019 International Conference on Advanced Science and Engineering (ICOASE), 2019, pp. 118–123.

[5] S. R. Zeebaree, K. Jacksi, and R. R. Zebari, “Impact analysis of SYN flood DDoS attack on HAProxy and NLB cluster-based web servers,” Indonesian Journal of Electrical Engineering and Computer Science, vol. 19, no. 1, pp. 510–517, 2020.

[6] O. Alzakholi, L. Haji, H. Shukur, R. Zebari, S. Abas, and M. Sadeeq, “Comparison Among Cloud Technologies and Cloud Performance,” Journal of Applied Science and Technology Trends, vol. 1, no. 2, Art. no. 2, Apr. 2020, doi: 10.38094/jastt1219.

[7] H. S. Oluwatosin, “Client-server model,” IOSR Journal of Computer Engineering, vol. 16, no. 1, pp. 67–71, 2014.

[8] H. Shukur et al., “A State of Art Survey for Concurrent Computation and Clustering of Parallel Computing for Distributed Systems,” Journal of Applied Science and Technology Trends, vol. 1, no. 4, pp. 148–154, 2020.

[9] H. Shukur, S. Zeebaree, R. Zebari, O. Ahmed, L. Haji, and D. Abdulqader, “Cache Coherence Protocols in Distributed Systems,” Journal of Applied Science and Technology Trends, vol. 1, no. 3, pp. 92–97, 2020.

[10] R. Craig and P. N. Leroux, “Case study-making a successful transition to multi-core processors,” QNX Software Systems GmbH & Co, 2006.

[11] Z. N. Rashid, K. H. Sharif, and S. Zeebaree, “Client/Servers Clustering Effects on CPU Execution-Time, CPU Usage and CPU Idle Depending on Activities of Parallel-Processing-Technique Operations “,” INTERNATIONAL JOURNAL OF SCIENTIFIC & TECHNOLOGY RESEARCH, vol. 7, no. 8, pp. 106–111, 2018.

[12] J. Chang and G. S. Sohi, “Cooperative cache partitioning for chip multiprocessors,” in ACM International Conference on Supercomputing 25th Anniversary Volume, 2007, pp. 402–412.

[13] S. Zeebaree and I. M. Zebari, “Multilevel Client/Server Peer-to-Peer Video Broadcasting System,” International Journal of Scientific & Engineering Research, vol. 5, no. 8, Art. no. 8, 2014.

[14] G. P. Acharya and M. A. Rani, “FPGA Prototyping of Micro-Blaze soft-processor based Multi-core System on Chip,” International Journal of Engineering & Technology, vol. 7, no. 2.16, pp. 57–60, 2018.

[15] A. A. Yazdeen, S. R. Zeebaree, M. M. Sadeeq, S. F. Kak, O. M. Ahmed, and R. R. Zebari, “FPGA implementations for data encryption and decryption via concurrent and parallel computation: A review,” Qubahan Academic Journal, vol. 1, no. 2, pp. 8–16, 2021.

[16] M. Ababneh, S. Hassan, and S. Bani-Ahmad, “On Static Scheduling of Tasks in Real Time Multiprocessor Systems: An Improved GA-Based Approach.,” International Arab Journal of Information Technology (IAJIT), vol. 11, no. 6, 2014.

[17] S. R. M. Zeebaree et al., “Multicomputer Multicore System Influence on Maximum Multi-Processes Execution Time,” TEST Engineering & Management, vol. 83, no. May-June 2020, pp. 14921–14931, May 2020.

[18] A. S. Y. Subhi Rafeeq Mohammed Zebari, “Improved Approach for Unbalanced Load-Division Operations Implementation on Hybrid Parallel Processing Systems,” Journal of University of Zakho, vol. 1, no. (A) No.2, Art. no. (A) No.2, 2013.

[19] D. M. Abdulqader and S. R. Zeebaree, “Impact of Distributed-Memory Parallel Processing Approach on Performance Enhancing of Multicomputer-Multicore Systems: A Review,” QALAAI ZANIST JOURNAL, vol. 6, no. 4, pp. 1137–1140, 2021.

[20] N. Goel, V. Laxmi, and A. Saxena, “Handling multithreading approach using java,” International Journal of Computer Science Trends and Technology (IJCST), vol. 3, no. 2, pp. 24–31, 2015.

[21] L. Haji, R. R. Zebari, S. R. M. Zeebaree, W. M. Abduallah, H. M. Shukur, and O. Ahmed, "GPUs Impact on Parallel Shared Memory Systems Performance," International Journal of Psychosocial Rehabilitation, vol. 24, no. 08, pp. 8030–8038, 21, May, doi: 10.37200/IJPR/V2418/PR280814.

[22] O. H. Jader et al., "Ultra-Dense Request Impact on Cluster-Based Web Server Performance," in 2021 4th International Iraqi Conference on Engineering Technology and Their Applications (IICETA), 2021, pp. 252–257.

[23] O. G. Lorenzo, T. F. Pena, J. C. Cabaleiro, J. C. Pichel, and F. F. Rivera, "Multiobjective optimization technique based on monitoring information to increase the performance of thread migration on multicores," in 2014 IEEE International Conference on Cluster Computing (CLUSTER), 2014, pp. 416–423.

[24] Z. N. Rashid, S. R. M. Zeebaree, M. A. M. Sadeeq, R. R. Zebari, H. M. Shukur, and A. Alkhayyat, "Cloud-based Parallel Computing System Via Single-Client Multi-Hash Single-Server Multi-Thread," in 2021 International Conference on Advance of Sustainable Engineering and its Application (ICASEA), 2022, pp. 59–64.

[25] Y. Xu, P. Liu, I. Penesis, and G. He, "A task-resource mapping algorithm for large-scale batch-mode computational marine hydrodynamics codes on containerized private cloud," IEEE Access, vol. 7, pp. 127943–127955, 2019.

[26] M. P. R. B. A. Bianco, "HERO: High-speed Enhanced Routing Operation in Ethernet NICs for Software Routers⋆," 2020.

[27] Z. Lv, D. Chen, and A. K. Singh, "Big data processing on volunteer computing," ACM Transactions on Internet Technology, vol. 21, no. 4, pp. 1–20, 2021.

[28] L. M. Haji, S. R. M. Zeebaree, O. M. Ahmed, M. A. M. Sadeeq, H. M. Shukur, and A. Alkhavvat, "Performance Monitoring for Processes and Threads Execution-Controlling," in 2021 International Conference on Communication & Information Technology (ICICT), 2021, pp. 161–166.