Effects of Multicore Distributed Memory Systems on Parallel Processing Applications

Mohammed J. Mohammed

Department of Computer and Communication Engineering, College of Engineering, Nawroz Univesity,

Kurdistan Region – F.R. Iraq

ABSTRACT

Complex problems need a longtime to be solved, with low efficiency and performance. Hence, to overcome these drawbacks, the approach of breaking the problem into independent parts and treating each part individually in the way that each processing element can execute its part of the problem simultaneously with the others. The systems that contain many computing elements combined. Parallel processing (PP) is divided into three types; shared, distributed, and hybrid memory systems are usually adopted. The aim of this research is to point out the effects of multicore distributed memory systems on PP applications that can reduce the total execution time of the programs. In this work, distributed- and shared-memory systems addressed depends on client/servers principles. However, to get the exact evaluation of our aim, just one client and one server have been depended. The algorithm used here is capable of calculating: The started, consumed, and terminated for CPU and total execution times, CPU usage of servers, and CPU and Total execution times for the client. The results compared with previous works depending on distributed memory systems, to overcome the previous drawbacks taking in the consideration the effects of multi-core processor. All of these algorithms are implemented using Java Language.

KEY WORDS: Central Processing Unit, Multicore Distributed Memory, Parallel Processing, Client-server Principles, Hardware and Software Parts

1. INTRODUCTIOSWN

To provide a high speed of programming - running the new style of programming depends on the problem division into small parts. In general, time-consumed of programming execution can be measured depending on number of processors that participated with the processing. A parallel system is a combination of a parallel algorithm and a machine on which it operates (Dietz, 2004; Wilkinson and Allen, 2004; Naiouf, 2004). Preparing high-performance computers are considered a great step of moving toward multiprocessors (MPs). The MP computer system will provide higher speed than that of uniprocessor one. The uses or applications for parallel processing (PP) come from two different areas; on the one hand, there are high-performance systems for speeding up compute-intense calculations. These can be executed on traditional supercomputer systems or large clusters of workstations. On the other hand, there are embedded control systems on sequential hardware, which require parallel programming (PProg) concepts to control concurrent external actuators or internal processes (Naiouf, 2004; El-Rewini and Abd-El-Barr, 2005; Braunl, 2010). The distributed systems are constructed to overcome the drawbacks of single-processor systems to be considered as one system that has more than one processing unit (Loosley and Douglas, 1998). In 2006, El Saifi and Midorikawa (2006) described parallel message passing interface. The main idea of this work is treatment with more than one platform and more than one program language. Depending on the principles of message passing, this system has the ability of combining different parties from different domains. In 2006, Kessler (2006) addressed the advantages of investing PProg features for browsing the power of PProg through students' courses. This teaching provided very acceptable results by clarifying the real capabilities of PProg applications, especially those related to the shared memory systems. In 2010, Sola (2010) studied the usefulness of PP approaches for addressing the problems of optimizing many aims dynamically. This has been done using motivated methods related with a multi-

Academic Journal of Nawroz University (AJNU)
Volume 6, No 3(2017), 3 pages
Received 1 August 2016; Accepted 3 October 2016
Regular research paper: Published 18 July 2017
Corresponding author's e-mail: mjm562000@nawroz.edu.krd

aim approach. The depended algorithm is an economic time, cost, and fast speed with wide production. In 2010, Yaseen introduced a diagnostic approach for improving the implementation of parallel processing (PP) operations. His work based on two case studies, namely, matrix algebra and sorting algorithms. For each case study, there are many general algorithms and other related algorithm (i.e., Network - Connection - Checking, Load - Division, Messages - Sending/Delivering, Timings - Calculating, Results - Checking, and Results - Receiving/Storing).

2. PP AND PPROG

PP is certainly not a new concept. There are three distinct areas of PP: Server-side functions, server process client-side functions, and client process object rendering (Frachtenberg, 2007). Although using parallelism is used for many applications, it is not a standard that all applications will get the advantages of the PP approaches. The structure of application and design of the programming style added to the available resources will affect the scope of benefiting from PP approaches (Dietz, 2004). To get powerful utilization from PProg, it prefers to combine more than one computer with high features that are capable of using the related computer system resources. A structured programming style will push the programmers to deal with big problems efficiently. The efficiency of PProg came from subdividing big problems into many pieces each piece will be specified to one processing unit within the multiprogramming system. Hence, the overall system will get high production through little time with high processing speed (Carriero and Gelernter, 1992; Funga, et al., 2003).

3. THE PROPOSED SYSTEM AND ALGORITHMS

In a distributed memory system, the processes have individual private spaces of addresses and communicate by MPIs. Each processor has its own local memory. The processors are connected to each other. In distributed memory system, there is no limitation on number of processors and memory modules because servers are connected as cluster network and it can be extended to any required number. The proposed algorithm has been designed for the software that guides the necessary hardware components and manages the passing of messages between client and servers with different cases. This work is focused on improving the programming style of distributed memory system PP approach by depending on active subroutines to overcome the drawbacks of memory heap in compare to the previous works (for example 10). For the simplicity, this case deals with one client and one server and depends exactly on the same computers used by Yaseen (2010). Adding to the previous aims, there is additional aim for this case study related to the central processing unit (CPU) usage of the server side. Due to that the previous work faces the problem of limiting the maximum order of used matrices, this case study is built to overcome this problem. This is done by increasing the heap size of the related OS and the special functions within the application package, also monitoring the available RAM to provide as possible as large amount of free space of memory. All matrix orders that depend on by Yaseen (2010) are applied then continued farther by several steps. This case study shows the effects of multi-core systems on the distributed memory system PP approach.

3.1. Hardware Part

The hardware part is constructed of client side and server side. In such works, the properties of computers are important, in this work both computers are fully identical completely, and have the following properties (CPU: Core 2 Due, Speed 2.6 MHz, RAM: 2 GByte, and HD: 120 GByte).

3.2. Software Part

As the hardware part consists of two sides, the software part also consists of two sides, which are client-side software and server-side software. Client-side software represents the main program, which is responsible of: Detecting the connected server socket at other side, sending/receiving the control/response messages to/from the client side (as message text or as data files), and receiving the calculated results by server socket and accumulating them to get the final results. Server-side software represents the program that service the commands issued from the main program (i.e., client program). The software at server host is responsible for: Detecting the connection status of the client-host, receiving/sending the response/control messages from/to client-host and guide the execution of the server program to apply the client requirements, and run the appropriate subroutines according to the requirements of client-host and calculate the correct results.

4. IMPLEMENTATION AND RESULTS

First, the load will be sent to the server side using just one processor and monitoring its effects on the CPU usage. Then, the procedure will be repeated taking into consideration distributing the load between the two processors of the server side and remonitoring the effect on CPU usage. The previous test does not exceed 4096 order as a high order of the matrices because of the available RAM problems. Hence, in this work, this problem has been overcome and four additional matrices orders are depended which are: 5000, 5016, 5032, and 5048. The reason behind the starting from these orders is to browse the effects of multi-core systems on PP. The results of this case study are shown in Table 1.

The practical measuring of CPU usage for a multiprocessor system, CPU usage can be calculated by letting all processors to have equal ratio of usage. Hence,

TABLE 1
Average Values of Matrix Algebra - Case for One Server

Server case	Number of elements				
Time types (seconds)	4096	5000	5016	5032	5048
Client CPU time	1813.3907	2025.6322	2046.2263	2078.2725	2128.7123
Client execution time	1848.6799	2060.9314	2081.1493	2110.3745	2170.1938
Server average consumed CPU time	1641.1406	1853.7533	1874.3375	1904.8782	1976.8119
Server total execution time	1751.7007	1963.8849	1984.9488	2020.1114	2099.0017
Server CPU-0 usage %	93	98	96	95	94
Server CPU-1 usage %	0	7	22	32	37
Server system CPU usage %=(CPU-0 usage+CPU-1 usage)/2	46.5	57.5	59	63.5	65.5

CPU: Central processing unit

for this case, we have just two CPUs (0 and 1), each will get a ratio of 50% of overall system usage.

5. DISCUSSION AND CONCLUSIONS

The obtained results illustrate that the problem of limited matrix order due to the restrictions of benefit from the available system's RAM has been overcome in this work and the effects of these added orders appeared, especially with certain states. Furthermore, the good efficiency of distributed PP with high load, the results of this figure for both of (consumed CPU and total execution) times are very acceptable. The important thing here is the ability of using an order higher than that used by Yaseen (2010) and this is done successively depending on the orders (5000, 5016, 5032, and 5048) where the maximum order used by Yaseen (2010) was (4096). The most important point that can be concluded here is the effect of multi-core systems on treating with high loads. This point was not discussed by Yaseen (2010) because he did not benefited from the facility of dual processor of his computer system. By chance, this problem did not appear (depending on the environment and organization of the installed applications in his computer system) which produced no such problems with increasing the matrix order to (4096). However, this work shows that for each system when reaching a certain point (may be before 4096 matrix order) of high load, the system will automatically distribute it among all its logical processors. Hence, both processors will participate with the processing operations and the CPU usage for each one will be less than that of one of them (i.e., CPU-0). This condition appeared at (order 5000 and above).To illustrate the organization of load distribution and its effect on CPU usage for each processor, the heavy load as much as possible at the maximum capability has been specified for first processor (i.e., CPU-0). Hence, the effect of overflow loading to the CPU-0 is illustrated on CPU-1 usage that has been started from a little value and grows with increasing its load. The main points arise from the research employed in this thesis can be summarized as follows:

1. Distributing the memory systems addressed depending on client/servers principles.
2. The algorithm used here is capable of calculating: The started, consumed, and terminated for CPU and total execution times, CPU usage of servers, and CPU and total execution times for the client.
3. The algorithm is designed in very active programming routines to get a minimum loss of time spent through the running state (at both client and server sides).
4. The capability of using multi-core systems within distributed PP systems has been addressed here in details side by side with the results affected by these processors. Added to that this work has addressed the shared memory system PP and its effects on multi-processes/multi-threads and their execution timing values.

REFERENCES

Braunl, T. (2010). *Parallel Processing: Parallel Computer Architecture and Parallel Software Design Book*. University of Western Australia.

Carriero, N & Gelernter, D. (1992). *How to Write Parallel Programs Book*. Cambridge, MA: Massachusetts Institute of Technology.

Dietz, H. (2004). Linux Parallel Processing HOWTO. v2.0, 28-06. Available from: http://www.aggregate.org/LDP. [Last accessed on 2017 May 26].

El Saifi, M.M & Midorikawa, E.T. (2006). *PMPI: A multi-platform, multi-programming language MPI using NET*. Sao Paulo, SP, Brazil: Polytechnic School-University of São Paulo.

El-Rewini, H & Abd-El-Barr, M. (2005). *Advanced Computer Architecture and Parallel Processing*. New York: John Wiley & Sons, Inc.

Frachtenberg, E. (2007). Job Scheduling Strategies for Parallel Processing, JSSPP, June 17; 2007.

Funga, Y.F., Ercanb, M.F., Chonga, Y.S M., Hoa, T.K., Cheunga, W.L. & Singha, G. (2003). *Teaching Parallel Computing Concepts with a Desktop Computer*. Hong Kong: The Hong Kong Polytechnic University.

Kessler, C.W. (2006). *Teaching Parallel Programming Early*. Sweden: Linköping University.

Loosley, C & Douglas, F. (1998). *High-Performance Client/Server*. New York, NY: John Wiley & Sons.

Naiouf, M.R. (2004). Parallel processing. *Dynamic Load Balance in Sorting Algorithms*. University Nacional de La Plata, Facultad de Ciencias Exactas.

Sola, M.C. (2010). Parallel Processing for Dynamic Multi-objective Optimization, Ph.D. Thesis, University of GRANADA, April; 2010.

Wilkinson, B & Allen, M. (2004). *Parallel Computers*. Boston, MA: Pearson Education Inc.

Yaseen, N.O. (2010). Diagnostic Approach for Improving the Implementation of Parallel Processing Operations, Thesis Zakho University; 2010.