THE IMPACT OF MULTIPROCESSOR DISTRIBUTED MEMORY SYSTEMS ON PARALLEL COMPUTING APPLICATIONS

[1]G. O. Asoronye, [2]J.O. Nwachi-Ikpor, [3]C.O. Onyibe
[1]Department of Computer Engineering,
[2]Information Communication Technology Department,
[3]Department of Electrical and Electronic Engineering,
Akanu Ibiam Federal Polytechnic, Unwana, Ebonyi State, Nigeria.

Abstract
Compute-intense calculations and parallel computing concepts on Central Processing Units (CPU) require long execution times. Hence, to overcome this challenge which is as a result of limited memory for processes of high matrix orders, developers nowadays use parallel computing approach to develop applications. The aim of the study is to show the impact of multiprocessor distributed memory systems in executing these applications built using parallel computing approach with shorter execution times. The parallel computing approach allows for the breaking of high matrix order processes into smaller independent parts that will run concurrently or in parallel. In this study the client-server model was used. The algorithm used was able to compute the client and server CPU time, execution time, server CPU's percentage usage on high matrix orders. It was seen that the average CPU usage for the 4096 matrix order was almost 50% less the time that it usually took a uniprocessor to execute. While for a large matrix order of 5032 it took a little over the same time by the multiprocessors to execute. The study revealed that parallel computing applications execution time can be effectively reduced by the use of multiprocessor distributed memory systems.

Keywords:
Multiprocessor, Parallel Computing, Distributed Memory Systems, Matrix Order.

Introduction

Increase in human-machine interaction has given rise to more demands for faster computations, that is to say shorter processing or execution time. The invention of multiprocessor distributed memory systems is considered a giant stride towards high performance computers as it has made room for shorter processing time unable to be attained by uniprocessor systems (Deeb et al., 2021). Another added advantage is the ease of implementing parallel computing paradigms (Gurhem et al., 2019). The concept of parallel computing is the type of computation where many executions of processes are carried out simultaneously or concurrently (Azad & Buluç, 2016; Zafari et al., 2019).

The applications for parallel computing comes from two areas; one area is high performance systems for the speed up of calculations that are compute-intense (Nelson & Palmieri, 2020). The other area is in embedded control systems which require parallel computing concepts to control either internal processes or external actuators (Brown et al., 2019).

Nowadays, complex processes are broken down into independent parts and treated individually by separate processors in a multiprocessor distributed memory system concurrently and then recombined thus, complex applications can have their executable instructions programmed into smaller independent parts that will run concurrently or in parallel(Benoit et al., 2009). This approach now makes it possible to measure the time consumed in the execution of processes by each processor in a multiprocessor distributed memory system (Rein & Karabtsev, 2020).

There is no standing rule that all applications will leverage on the gains of parallel computing approaches. The structure of an application and its programming design style in addition to the available resources will affect how it maximizes parallel computing approaches (Jones, 2012). To unlock the power of parallel computing, more than one computer (multiprocessor) with high configurations that are capable of utilizing the related computer resources must be used (Mehrabi et al., 2019).

The challenge of limited matrix order (i.e. number of elements to be processed at a time) due to the restrictions posed by the size of available memory in computer systems used in executing these parallel computing applications has been the major cause of long computation execution time (Karn & Kumar, 2021). Developers are confined to the matrix order of single processor systems, hence underutilization of multiprocessor distributed memory systems.

With the above assertion in mind, the study points out the impact of multiprocessor distributed memory systems on parallel computing applications that makes reduction in the total execution time for computation possible. This is in a bid to enhance parallel computing approach on multiprocessor distributed memory systems by relying on active subroutines to alleviate the shortcomings of memory heap experienced in single processor systems.

Materials and Methods

Hardware Materials

Two computer systems were used in the setup for this study, one as a server and the other as a client. The configurations of both systems were Dual Core Central Processing Units (CPU) with processor speeds of 2.6GHz, 4GB of RAM, and 250GB hard disk drive respectively.

Software Materials

The software part of the study similar to the hardware has two aspects, server-side and client-side software. The putty software on the server was responsible for listening to the client connection, receiving and sending all controls and responses from and to the client, calculating the various server processor time variants and CPU usage was carried out using the inbuilt Task Manager App. The putty software on the client was responsible for making socket connection with the server, receiving and sending all controls and responses from and to the server, and calculating the various client processor time variants was done with the inbuilt Task Manager App.

Procedures

A single processor on the server was first used to execute a load (an application programmed with a parallel computing approach) and accessed by the client. This was done with a matrix order of 4096. The execution and CPU times of both client and server were measured and recorded. The same process was done distributing the processing between CPU 1 and CPU 2 on a 50% ratio and the server CPU usages in percentages measured. The entire procedures were repeated for matrix orders of 5000, 5016 and 5032. In each case the average server CPU usage was calculated.

Results

Table 1: performance comparison with varied matrix orders

Matrix order	Client		Server				
	CPU time (µs)	Execution time (µs)	CPU time (µs)	Execution time (µs)	CPU 1 usage %	CPU 2 usage %	Average CPU usage%
4096	813.41	848.72	641.24	751.82	83	0	41.5
5000	1025.73	1060.76	853.83	972.67	88	6	47.0
5016	1046.32	1081.23	867.54	968.87	86	21	53.5
5032	1078.39	1110.48	904.63	1020.21	85	31	58.0

Discussion

The results from Table 1 showed that both CPU and execution times of the client and server systems are at very acceptable intervals. This minimal loss in time between the client and server systems was achieved by implementing an algorithm with routines for very active programming.

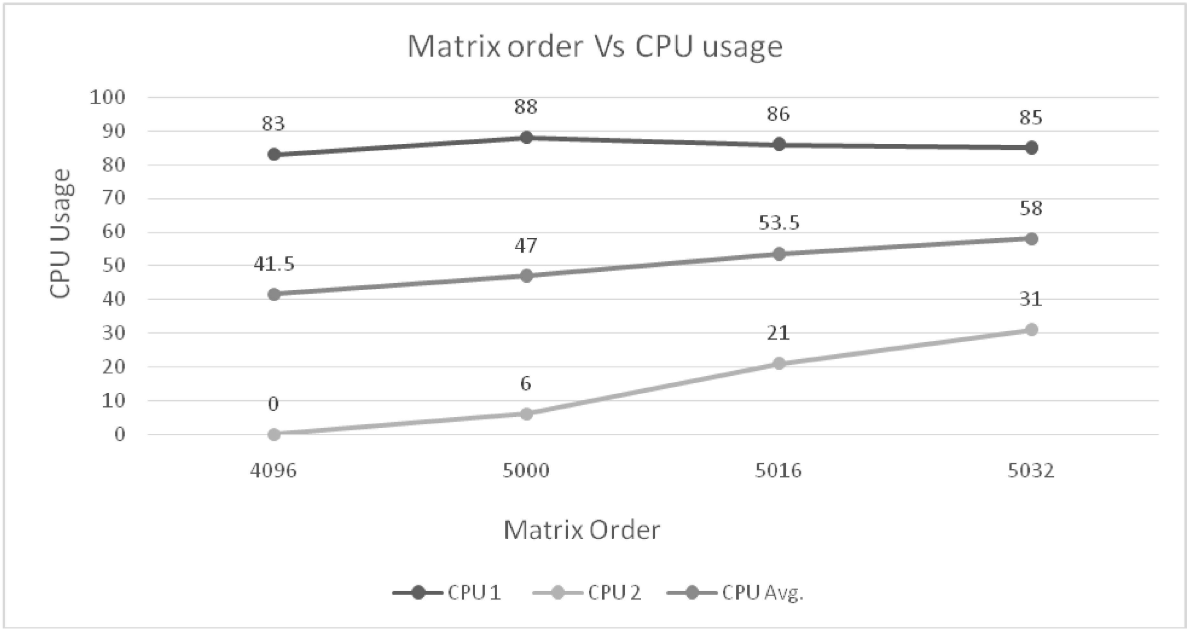

Figure 1: A comparison of matrix order and CPU usage.

The impact of load overflow is clearly depicted, revealing a constant load balancing between CPU 1 and CPU 2 thus maintaining low execution time on the server system. From the graph of Figure 1 above, it is seen that the average CPU usage for the 4096 matrix order was almost 50% less the time that it usually took a uniprocessor to execute. While for a very large matrix order of 5032 it took a little over the same time by the multiprocessors to execute.

Conclusion

The findings of this study showed that the challenge of limited matrix order due to the restrictions posed by the size of memory available in uniprocessors can be effectively taken care of by multiprocessor distributed memory systems. Hence, developers can leverage on the higher matrix orders provided by multiprocessors to develop applications that support parallel computing using either agile software development or plan driven software development models.

References

Azad, A., & Buluç, A. (2016). A matrix-algebraic formulation of distributed-memory maximal cardinality matching algorithms in bipartite graphs. *Parallel Computing*, *58*. https://doi.org/10.1016/j.parco.2016.05.007

Benoit, A., Hakem, M., & Robert, Y. (2009). Parallel Computing. *Network*, *35*(1), 83–108. https://doi.org/10.1016/j.parco.2008.11.001

Brown, G., Reyes, R., & Wong, M. (2019). Towards heterogeneous and distributed computing in C++. *ACM International Conference Proceeding Series*. https://doi.org/10.1145/3318170.3318196

Deeb, H., Sarangi, A., & Sarangi, S. K. (2021). Improvement of load balancing in shared-memory multiprocessor systems. *Smart Innovation, Systems and Technologies*, *153*. https://doi.org/10.1007/978-981-15-6202-0_8

Gurhem, J., Petiton, S. G., Tsuji, M., & Sato, M. (2019). Distributed and parallel programming paradigms on the K computer and a cluster. *ACM International Conference Proceeding Series*. https://doi.org/10.1145/3293320.3293330

Jones, T. (2012). Linux kernel co-scheduling and bulk synchronous parallelism. *International Journal of High Performance Computing Applications*, *26*(2), 136–145. https://doi.org/10.1177/1094342011433523

Karn, A. K., & Kumar, A. (2021). K -group of absolute matrix order unit spaces. *Advances in Operator Theory*, *6*(2). https://doi.org/10.1007/s43036-021-00134-5

Mehrabi, M., Giacaman, N., & Sinnen, O. (2019). @PT: Unobtrusive parallel programming with Java annotations. *Concurrency Computation* , *31*(1). https://doi.org/10.1002/cpe.4831

Nelson, J., & Palmieri, R. (2020). Performance Evaluation of the Impact of NUMA on One-sided RDMA Interactions. *Proceedings of the IEEE Symposium on Reliable Distributed Systems*, *2020-September*. https://doi.org/10.1109/SRDS51746.2020.00036

Rein, T. S., & Karabtsev, S. N. (2020). Software implementation of the conjugate gradient method for shared and distributed memory multiprocessor systems. *IOP Conference Series: Materials Science and Engineering*, *862*(5). https://doi.org/10.1088/1757-899X/862/5/052038

Zafari, A., Larsson, E., & Tillenius, M. (2019). DuctTeip: An efficient programming model for distributed task-based parallel computing. *Parallel Computing*, *90*. https://doi.org/10.1016/j.parco.2019.102582

CONTRIBUTORY EFFECT OF POST BROODING AGE ON GROWTH PARAMETERS OF ABOR ACRE BROILER CHICKEN

*Nwaodu, O.B.[1]; Abdullahi, J[1]; Eziuloh, N.E.[1]; Abe, O.S.[2]
[1]Department of Agricultural Technology, Akanu Ibiam Federal Polytechnic, Uwanna, Nigeria
[2]Department of Animal Science, Adekunle Ajasin University, Akungba-Akoko, Nigeria
*Corresponding author: email oziomaonumajuru@gmail.com

ABSTRACT

Growths in broiler are mostly in two phases of starter and finisher. The most profound developmental changes, both qualitative and quantitative, occur at starter phase which are sometimes relatively short when compared with the finisher phase. Knowing the contributory effect of the finisher phase on growth will guide farmers to optimized profitability. The study analyzed the growth performance and the correlation between weight and age of abor acre commercial broiler strains, post brooding. A total of one hundred and fifty day old chicks were used for the study. The chicks were fed commercial starter mash between day one and 28 days and then changed to finisher crumble for the period of the study. The initial weight of the birds were taken individually at day 28 and were subsequently weighed at 5 days interval for the duration of the study. All data collected were analyzed using the general linear model procedure of SAS (1999) to estimate the heritability, repeatability and the correlation coefficient. The result of the study showed that percentage weight gain increased at a reducing rate as age increases. The standard deviation (SD) for the mean weight ranged between 25.17 and 35.54, showing little genetic by environmental variations between the individual body weight records and the overall mean. The coefficient of variation (CV) were low and ranged between 1.34 and 3.07. The values were relatively similar, suggesting a more precise estimate of the body weight. The study also recorded an average total weight gain of 1115.51g in body weight for the duration of the study which translated to 55.78g average body weight on daily basis and 5.00% in growth and development. The repeatability and heritability estimates obtained in this study were between 0.62 and 0.77, and 0.27 and 0.50 respectively. This was suggestive of reduced impact of environment as the bird increases in age. The correlations between the ages were positive, high and shows significance ($P<0.05$) as the bird increases in age with phenotypic correlation coefficient ranging between 0.63 and 0.92, the genetic correlation coefficient ranged between 0.51 and 0.89 while the environmental correlation ranged between 0.88 and 0.99. The study therefore concludes that environmental variation fades off as the bird advances in age and as age increases the weight was increasing at a reducing manner.

Keywords
Arbor Acre, Broiler Chicken, Post Brooding Age, Growth Parameters

INTRODUCTION

Growth involves increase in size and changes in functional capabilities of the various tissues and organs of animals (Ojedapo et al.,2012) and has consistently been the prime selection trait since the 1950s, with more recent emphasis placed on meat yield, liveability and feed use efficiency (Muir and Aggrey, 2003; Laughlin, 2007; Renema et al., 2007). Growth in farm animals is however, very complex and not a straight forward affair as it entails different phenomenal of increase in increasing rate or increase at a decreasing rate. The performance of broiler birds is determined by its genotype and environmental factors (Boukwamp et al.1973; Edward and Denman, 1975). This is because there are combinations of factors which may favour or disfavour it.