British Journal of Mathematics & Computer Science
13(1): 1-15, 2016, Article no.BJMCS.21744
ISSN: 2231-0851

SCIENCEDOMAIN *international*
www.sciencedomain.org

Empirical Performance Analysis of Parallel Programs on Wired and Wireless Local Area Networks Using Beowulf Clusters

C. I. Saidu[1*], A. A. Obiniyi[2], S. B. Junaidu[2] and Y. J. Gambo[3]

[1]Department of Computer Science, Bingham University, Nasarawa, Nigeria.
[2]Department of Mathematics, Ahmadu Bello University, Zaria, Nigeria.
[3]Department of Computer Science, Federal University, Wukari, Nigeria.

Article Information

DOI: 10.9734/BJMCS/2016/21744
Editor(s):
(1) Dariusz Jacek Jakóbczak, Chair of Computer Science and Management in this department, Technical University of Koszalin, Poland.
Reviewers:
(1) Ahmet Sayar, Kocaeli University, Turkey.
(2) Anand Nayyar, KCL Institute of Management and Technology, India.
(3) Xinwei Niu, SASTRA University, Thanjavur, India.
Complete Peer review History: http://sciencedomain.org/review-history/12383

Original Research Article

Received: 01 September 2015
Accepted: 06 October 2015
Published: 21 November 2015

Abstract

The basic idea about parallel computing is about putting independent processing units together to collectively solve a task. However, the amount of speedup attained by this collection of processing units is a function of several factors, one of which is the interconnection network.
This paper focuses on measuring performance of parallel programs deployed on wired and wireless networks. Our experiments were conducted on Beowulf clusters; a parallel computer built using a collection of everyday personal computers. This paper shows empirically that distributed memory parallel programs (MPI) written for Beowulf clusters on wireless LAN (IEEE 802.11 g) do not gain appreciable speedup as the number of processing nodes increases compared to the same parallel programs written for the same Beowulf clusters but on wired LAN. It further shows the impact the kind of network has in the overall performances of parallel programs when a multiprogramming approach is used to achieve speedup.

Keywords: MPI; Beowulf; parallel programming; processing units or processing nodes; wired network; wireless network.

1 Introduction

The need for faster systems has brought about various innovations in the way computers are built. These innovations in the quest for faster computers brought about concept like increasing the clock rate of

**Corresponding author: E-mail: charlessaidu@binghamuni.edu.ng;*

processors, improving instruction set architecture, pipelining and parallel computing with parallel computers quickly gaining grounds [1,2].

Parallel computing is a form of computation in which many calculations are carried out simultaneously. This idea is based on the fact that it will theoretically take less time for two or more processing unit to work on a piece of computation than a single processing unit working on the same amount of work. Therefore, neglecting other variables such as network speed and race conditions for memory access, for a given computational problem, doubling the number of processing units will in effect reduce the amount of time it will take to compute the problem set. However, in practice, networks and race conditions [3,4] are one of the pivotal factors that can contribute negatively against the speedup attained by an increase in the number of processing unit. This necessitated the study as to how much one of these limitations (networks) to speedup could possibly affect the overall speedup of computations as the number of computing nodes increases.

A special case of study in this paper is the parallel program based on the popular Message Passing Interface (MPI). MPI programs follow the distributed memory architecture parallel programming paradigm [5,6], hence the idea is that for two processing notes to communicate during execution of a parallel program, message is sent between the processing node via the link or the network connecting the node. According to [5], MPI is a library specification for message-passing, proposed as a standard by a broadly based committee of vendors, implementers, and users. This standard guides the format and defines certain procedures and protocols for communication among processing nodes. Machines with distributed memory or hybrid of both distributed and shared memory architectures apply MPI standards through MPI libraries to achieve parallelism in program execution.

2 Review of Related Works

Over the years, comparisons have been made on various subjects bordering parallel programming but as at the time of this research no empirical analysis have been conducted on the performance of parallel programs on both wired and wireless LAN.

Ravela, [4] compared the performance of shared memory parallel programs like TBB, Pthreads, Cilk++ and OpenMP. Though an empirical analysis was carried out, it was based on comparing shared memory parallel computing libraries.

William et al. [7] surveyed and analysed existing frameworks for network computing with a particular focus on computing systems for Network of Workstations (NOW).

Parallel Computing in Local Area Networks [8] compared performance of broadcast with point-to-point communication (both blocking and non-blocking) of the MPI-I standard library on a cluster computer in communicating the same data block among all processing node. They compared the performance in terms of delay by varying the number of processing nodes and the data block size.

Cameron and Tracey [9] analyzed problems and possible solutions taking into account the main factors of computing and communications in parallel programs installed on local area networks.

In all the related works found and sited, none of them discussed the empirical performance of parallel programs on wireless LANs as compared to programs on wired LANs. The viability and scalability of parallel programs on wireless clusters has never been measured empirically.

2.1 Beowulf Cluster

The Beowulf cluster is a category of computers that came about largely due to the popularity of personal computers. Accord to [6], Beowulf is an example of a system constructed out of commodity, off-the-shelf-(COTS) components [7]. Unlike some commercial high-end servers, commodity clusters typically are not

balance between compute speed and communication speed. The communication among Beowulf clusters is slow compared to the high speed and space locality of high speed parallel processors in servers [6,8]. Beowulf parallel computers designs were largely influenced by their low financial cost of acquisition. In a Beowulf cluster, the idea was that, putting together everyday personal computer connection via a network to solve a computational task is far less expensive compared to high end supercomputers involving many processors.

However, increasing the number of processing nodes for a computational problem is not a guarantee that the running time for such problem will decrease. In fact, the running time for a parallel program on a parallel computer is a function of several factors which includes, the data dependencies of the parallel program which relates to the serial portion of the computation that cannot be parallelised and the interconnection network. This serial portion of the program has been long shown by Amhdal (1960) and Gustafason to contribute to the overall speedup of parallel computations [6,9].

A typical point to note is the network that connects these computers. To this end this paper tries to find empirically the extent to which the interconnection network contributes to the overall performance gain as the number of computers (processing nodes) in a typical Beowulf cluster grows. Also, this paper tries to find out if there are specific programs that when fine-tuned would work better for wireless networks. A comparison is made between wired and wireless local area networks with benchmark programs involving high network overheads and high computational overheads. The benchmark programs used are the popular parallel matrix-vector multiplication algorithms using row-wise decomposition, column-wise decomposition, checkerboard decomposition and parallel matrix-matrix multiplication algorithm using row-wise decomposition. These benchmark programs were chosen because of their wide application in science and engineering problems.

2.2 Message Passing Interface Standard

The Message Passing Interface Standard is not a programming language but a reference guide for vendors writing message passing libraries. According to [5], the standard was proposed by a broadly based committee of vendors, implementers and user. These committee members were drawn from IBM, Intel, TMC, Meiko, Cray, Convex, Ncube, PVM writes and application specialists/consultants. Their coming together produced the first version of MPI in 1994 which was known as MPI Standard version 1.1. Versions like 1.2 and the current version 2.0 later followed suit.

The first widely known implementation of the MPI standard is MPICH (Message Passing Interface Chameleon) [10]. The MPICH is an open source implementation of MPI and it is available under the Linux, Mac, Solaris and Windows platforms. The most current version of MPICH is the MPICH version 3.0.

Another implementation of MPI standard interface is DeinoMPI used in this research (11). DeinoMPI is an implementation of MPI-2 for Microsoft Windows environment. Its current release version is 2.0.1 and comes in both Win32 and Win64 machines. DeinoMPI provides libraries necessary to write parallel programs in distinctively three sequential programming languages namely C, C++ and Fortran.

DeinoMPI is designed with a process manager which makes it possible for processes on multiple machines in a cluster to be securely started remotely [11].

In order to view the performance and behaviour of MPI primitive functions and user-defined functions, DeinoMPI implementation of the MPI comes bundled with a “Jumpshot” program.

Jumpshot is a java tool to view log files created when MPI jobs are profiled.

2.3 Wired and Wireless Local Area Networks

Networks constitute a pivotal part of any parallel system and A Beowulf cluster is not left out as it also requires interconnection which could be achieved through wired and wireless means.

2.4 Wired Local Area Network

Wired have been existing for decades now with technologies like Ethernet, Token Ring, Tokin Bus, Fibre Distributed Data Interface and Asynchronous Transfer Mode (ATM) LAN.

The need to standardize deployment of these networks across multiple vendors brought about the Institute of Electric and Electronics Engineering (IEEE) standard known as the project 802 standard in 1987 [12]. This standard divides the Data link layer of the OSI models into two sub-layers namely. The logical link control which provides one single data protocol for all IEEE LANs, and the protocol data unit (PDU) in the LLC takes care of framing, flow control and error control [12].

The media access control sub-layer on the other hand provides different protocols for Token Ring, Ethernet LAN, FDDI and Token Bus.

Ethernet LAN however, has evolved due to its wide acceptance and implementations by various vendors. It evolved from the standard Ethernet (10 Mbps) to fast Ethernet (100 Mbps), Gigabit Ethernet (1 Gbps) and now ten-gigabit Ethernet (10 Gbps).

The first set of Ethernet standard used a shared Ethernet cabling mechanism which gave raise to contention in the media and as such required a protocol to deal with collision in the media. This gave rise to the popular Carrier Sense Multiple Access with Collision Detection protocol CSMA/CD.

The introduction of fast Ethernet LAN protocol made it possible to drop the bus topology of the standard Ethernet. This provided the choice between a full-duplex and half duplex operation mode. With the full duplex Ethernet mode, there was no need for the implementation of CSMA/CD protocol since the possibility of a collision was eradicated with the non-implementation of the bus topology.

2.5 Wireles Local Area Networks

A wireless network is any type of network that is not connected by cables of any kind. Wireless telecommunications networks are generally implemented and administered using radio communication [13]. This implementation takes place at the physical layer of the OSI model network structure [14]. Types of wireless networks include, Wireless PAN, Wireless LAN, Wireless MAN, Wireless WAN and Cellular networks.

Wireless Wide Area Networks (Wireless WANs) are wireless networks that typically cover large areas, such as between neighbouring towns and cities, or city and suburb. These networks can be used to connect branch offices of businesses or as a public internet access system. The wireless connections between access points are usually point to point microwave links using parabolic dishes on the 2.4GHz band, rather than omnidirectional antennas used with small networks.

Over the years, the Institute of Electrical and Electronics Engineering (**IEEE**), European Telecommunication Standard Institute (**ETSI**) and Home Radio Frequency Working Group (Home RFWG), have been involved in developing standards for the WLAN [12]. These include the IEEE 802.11x, HiperL ANx, and Home RG. Of these, the IEEE 802.11 family of protocols has clearly become the dominant standard for WLAN in the world [13].

One of the categories of IEEE 802.11x is the IEEE 802.11g network.

The IEEE 802.11g network is the third modulation standard for WLAN. It operates on 2.4GHz like IEEE 802.11b. The physical layer can use either Direct-Sequence Spread Spectrum (**DSSS**) or Orthogonal Frequency Division Multiplexing (**OFDM)**. It can achieve higher throughput of up to 54 Mbps due to its heritage from the IEEE 802.11 g [13].

Wireless medium is a contentius based medium and as such gives room for collision of packets during data transmission. In order to solve this problem, the Carrier Sense Multiple Access with Collision Avoidance (CSMA/CA) and the Request to Send (RTS)/ Clear to Send (CTS) protocols were developed [15].

2.6 Carrier Sense Multiple Access with Collision Avoidance (CSMA/CA)

This is a scheme used in detecting and avoiding collision in wireless networks. In this CSMA/CA stations try to avoid collisions by only transmitting when medium is idle.

Packets delivery in IEEE 802.11 is asynchronous and connectionless therefore; it employs a best effort approach by default unless quality of service is turned on [13].

The Media Access layer of the IEEE 802.11 employs methods for transmission namely Distributed Coordination Function (**DCF**), Point Coordination Function (**PCF**), or Hybrid Coordination Function (**HCF**) [13,15].

DCF is carrier sense multiple access with collision avoidance and a random backoff interval is used for the determination of the defer-period after collision is sensed.

PCF is an option access method, which is only for configuration of the infrastructure network. In the operation of PCF, the infrastructure polls every station to determine the sequence of transmission. The information in PCF is distributed by beacon management frames. As the infrastructure controls the transmission, contention can be mitigated to some extent.

CSMA/CA can optionally be supplemented by the exchange of a **Request to Send** (RTS) packet sent by the sender S, and a **Clear to Send** (CTS) packet sent by the intended receiver R. Thus alerting all nodes within range of the sender, receiver or both, to not transmit for the duration of the main transmission. This is known as the IEEE 802.11 RTS/CTS exchange. Implementation of RTS/CTS helps to partially solve the hidden node problem that is often found in wireless networking [15].

3 Experimental Setup

3.1 System Configuration

In this experiment, a network is setup containing ten (10) laptops all with the same hardware configuration. Each of the laptops has a 32 bit Pentium® Dual-Core T4200 2.00GHZ 2.00 GHZ, 2.0 GB Random Access Memory, Realtek RTL8187B Wireless 802.11b/g 54mbps wireless adaptor and PIC-E Gigabit Ethernet Network Interface Card for wired communication.

3.2 Network Setup

The Topology for both wired and wireless local area networks is star. For the wired network, a fast Ethernet is setup composing of a 24 Port DLink switch and a category five Ethernet cable used to connect the laptops.

The wireless network was setup using a DLink wireless Access point. The wireless network setup conforms to the IEEE 802.11 g standard hence has the maximum throughput of 54 Mbps. A secured SSID was setup for connecting the laptops wirelessly. On each laptop is a wireless 802.11b/g adapter with a maximum speed of 54 mbps.

3.3 Operating System

All the laptops were installed with Microsoft Windows 7 Operating System. It should be noted that the Windows firewall and antivirus for each laptop was switched off in order to avoid communication bottlenecks among the laptops. The firewall and antivirus could have been configured to include exceptions for the intended programs but for avoidance of complexity they were switched off.

Also, each laptop was setup never to hibernate or sleep when idle. This was to make sure all laptops remain active during program execution.

3.4 Parallel Library Setup (Using DeinoMPI)

All the laptops were installed with the 32bit version of DeinoMPI. On each laptop, a user was setup at the operating system level. At the DeinoMPI level, each laptop is required to have a credential store that can be tied to the system's registry. And a DeinoMPI user and password was created. This user and password is similar to the username and password created at the operating system's level. Dev++ were used to compile the programs.

3.5 Execution Plan and Experiment Program

All programs were executed using Windows shell script. An execution plan was setup and for each case study (wired or wireless), Windows shell script was written to run the already compiled program.

3.6 Sequential Matrix – Vector Multiplication Algorithm

Given a Matrix A with dimension $m\ x\ n$ and a column Vector B with dimension $n\ x\ 1$, a sequential matrix-vector multiplication algorithm multiplies each row of matrix A with the column Vector B. Therefore, the algorithm requires two loop nested loops. The outer loop traverses the rows of A while the inner loop traverses the columns of A whose size equals the rows of the vector B. In general for any matrix A of dimension $m\ x\ n$, performs n multiplication m times, requiring a running time complexity of $\theta(mn)$. For a square matrix with dimension $n\ x\ n$, the running time complexity becomes $\theta(n^2)$.

3.7 Parallel Matrix-Vector Multiplication Algorithm

There are various decomposition techniques namely, **row-wise decomposition**, **column-wise decomposition** and **checkerboard decomposition [6]**.

Each of these decompositions come with its each communication and computation bottlenecks, hence each speed is likely to vary according to the decomposition method used.

In this analysis, all three decomposition techniques are executed on both wired and wireless networks to observe their performance differences as related to the network used. The sole reason for picking these algorithms is to have different bases for speedup comparison of performance of MPI programs on wired and wireless networks. The well-known sequential Matrix-Vector Multiplication algorithm runs at θ(n^2) time complexity for square matrix n x n and θ(mn) time complexity for matrix of dimension m x n, both with Vector B of dimension n x 1. The time complexity for the parallel matrix-vector multiplication algorithm is dependent on factors such as the communication complexity and the decomposition complexity.

3.8 Parallel Matrix-Vector Multiplication Algorithm (Row-wise Decomposition)

The goal of every parallel program is to find a way to speed up computation. Given an $n\ x\ n$ Matrix A and an $n\ x\ 1$ Vector B, row-based decomposition algorithm divides Matrix A such that, if there are p number of

processes then each process is allocated at most $\lceil n/p \rceil$ rows of the $n \; x \; n$ matrix A. The $n \; x \; 1$ Vector B is distributed across all processes in the communication grid.

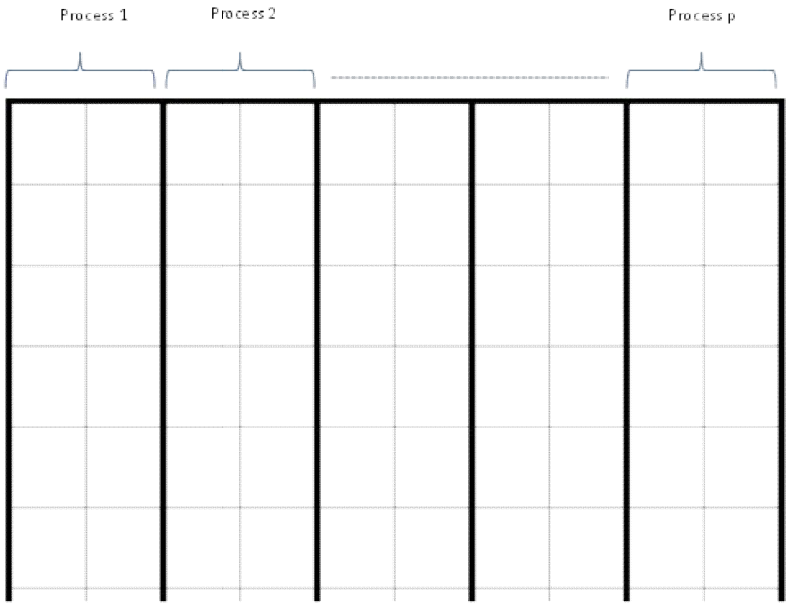

Fig. 1. Column-wise decomposition of matrix A

In the row-based matrix-vector multiplication, domain decomposition is done such that each row of the matrix is associated with the entire vector b. This is because it has only n elements hence decomposing it will lead to additional and unnecessary communication. The Row-wise decomposition is as shown in Fig. 2.

3.9 Parallel Matrix-Vector Multiplication Algorithm (Column-wise Decomposition)

The parallel Matrix-vector multiplication algorithm using Column-based decomposition has a similar concept with the row-based decomposition, except that matrix decomposition is column-wise.

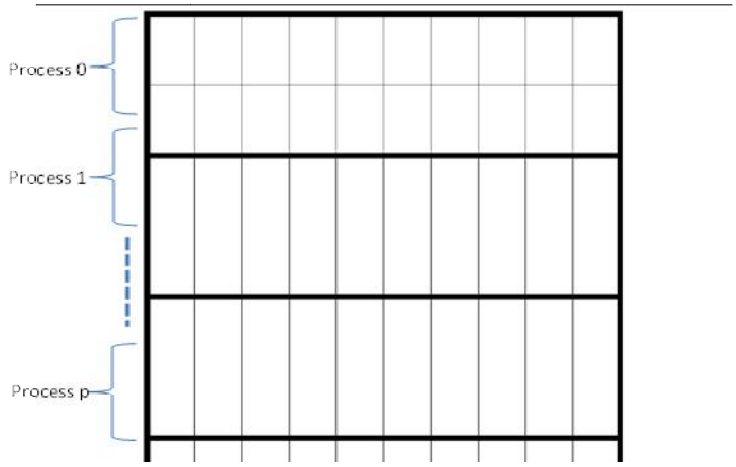

Fig. 2. Row-wise decomposition of matrix A

In this algorithm a domain deposition of the matrix is based on the columns, hence its columns are distributed among the various MPI processes participating in the communication as depicted in Fig. 1.

3.10 Parallel Matrix-Vector Multiplication Algorithm (Checkerboard Decomposition)

Unlike the row-based and column-based matrix decomposition, in checkerboard parallel multiplication algorithm, domain decomposition is done in a two dimensional grid-like form and each grid is associated with an MPI process. The matrix domain decomposition is depicted in Fig. 3.

3.11 Sequential Matrix-Matrix Multiplication Algorithm

Given a matrix A and B with dimensions A_{mn} and B_{nm} where m and n are integers ranging from 0 to $m-1$, $n-1$ respectively, the product matrix C_{mn} is defined as the product of $A_{mn}.B_{nm}$.. Therefore, $C[c_{ij}]=A.B$, $c_{ij} = \sum_k^n A_{ik}.B_{kj}$ where c_{ij} is an element of product matrix C and k are values ranging from 0 to $n-1$. The sequential naïve implementation of this algorithm requires three nested loops with the inner loop calculating the c_{ij} term of the resultant matrix C. Therefore, the running time for this algorithm is $\boldsymbol{\theta}(mn^2)$. In the case of a square matrix A with the same dimension as matrix B, then the running time for the sequential naïve algorithm will be $\boldsymbol{\theta}(n^3)$.

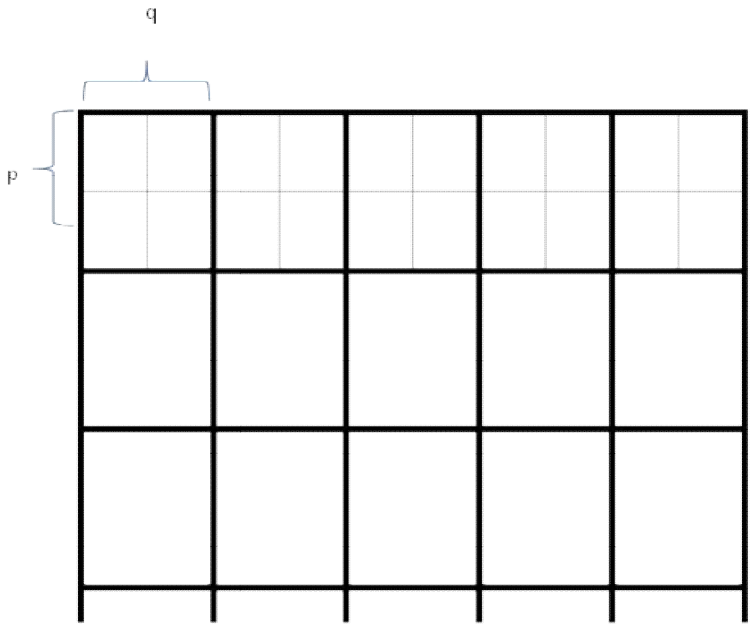

Fig. 3. Checkerboard decomposition of matrix A

3.12 Parallel Matrix-Matrix Multiplication Algorithm (Row-wise Decomposition)

Sequential Matrix-Matrix multiplication algorithm has a time complexity of $\boldsymbol{\theta}(n^3)$. Because of this high computation overhead, it lends itself as a very good algorithm to be parallelized.

In this work, a parallel matrix-matrix multiplication with row-wise decomposition is used. In this row-wise decomposition technique of matrix-matrix multiplication algorithm, domain decomposition is row-wise.

Each matrix A and B is decomposed (just like the row-wise matrix decomposition in the row-wise matrix-vector multiplication algorithm) and distributed among the processes partaking in the computation.

The steps in the parallel matrix-matrix multiplication algorithm is as follows

In this algorithm, the primitive task involves multiplying a row from Matrix A with a row from Matrix B producing a partial resultant sub-matrix C. Multiplication is done such that given a row i of matrix A and a row i of Matrix B, each computation produces $c_{i,1}$, $c_{i,2}$, …$c_{i,n}$ of the partial resultant sub-matrix C, where n is the number of columns of Matrix B. Thus elements $C_{i,j}$ of Matrix C form the partial resultant sub-matrix of C.

This algorithm is illustrated in Fig. 4.

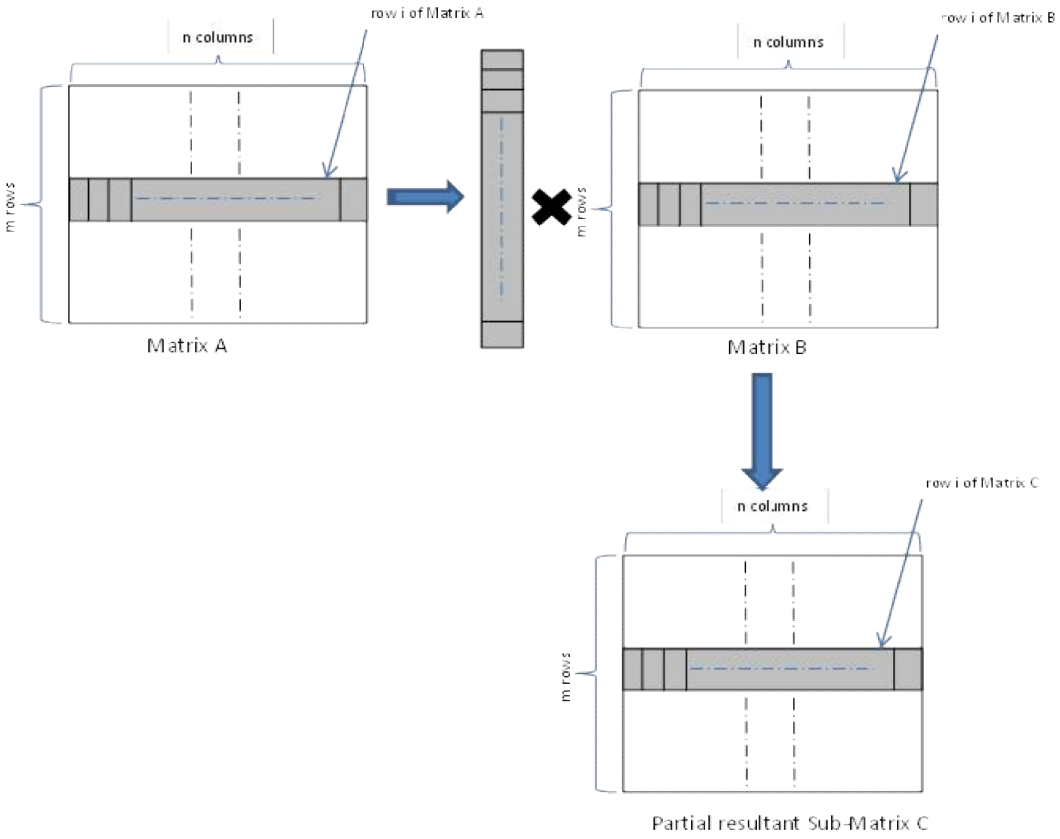

Fig. 4. Matrix- Matrix multiplication (Row-wise decomposition)

4 Experiment Results

Figs. 5, 6 and 7 are results obtained from the parallel matrix-vector row-wise decomposition, column-wise decomposition and checkerboard decomposition respectively. In each of the experiments, a 4000 x 4000 matrix whose elements are of data type double was multiplied with a 4000 x 1 vector whose elements are also of data type double. A total of 100 runs were made. Ten (10) runs were made for each number of clusters. After each cluster runs, the average execution time was obtained. The experiment was carried out on two (2) to ten (10) clusters.

Fig. 8 shows the result for the parallel matrix-matrix multiplication algorithm using row-wise decomposition. In the experiment, a 2000 x 200 matrix is multiplied by a 2000 x 2000 matrix. There are a total of 100 runs with 10 runs each number of clusters observed.

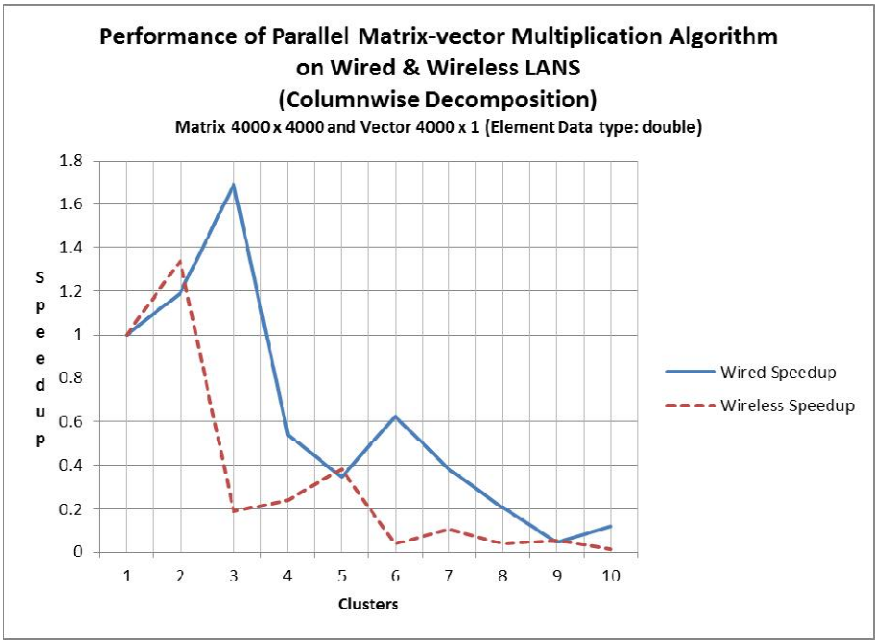

Fig. 5. Matrix-vector multiplication (Column-wise decomposition) speedup on wired and wireless

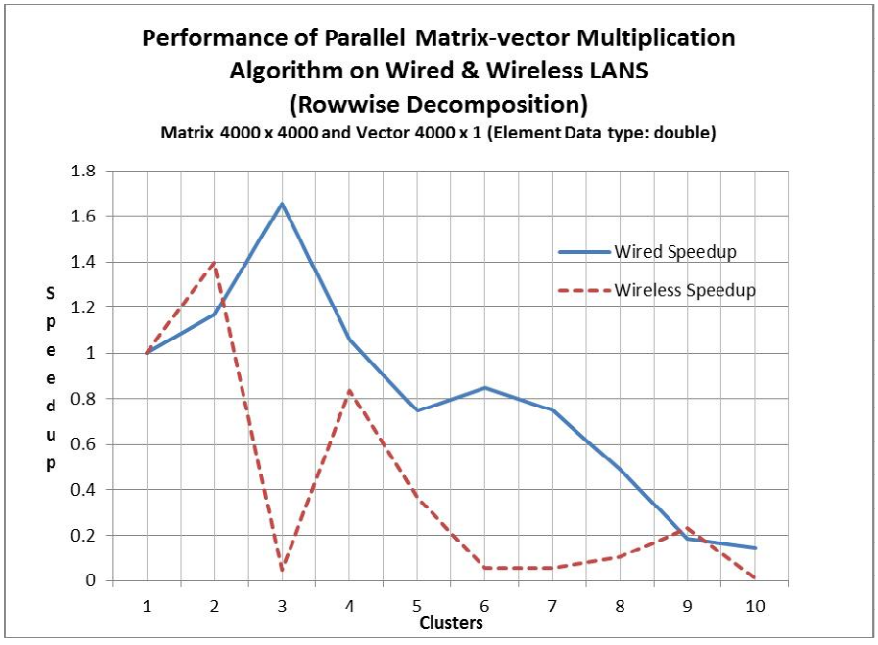

Fig. 6. Parallel matrix-vector multiplication: Row-wise decomposition speedup on wired and wireless

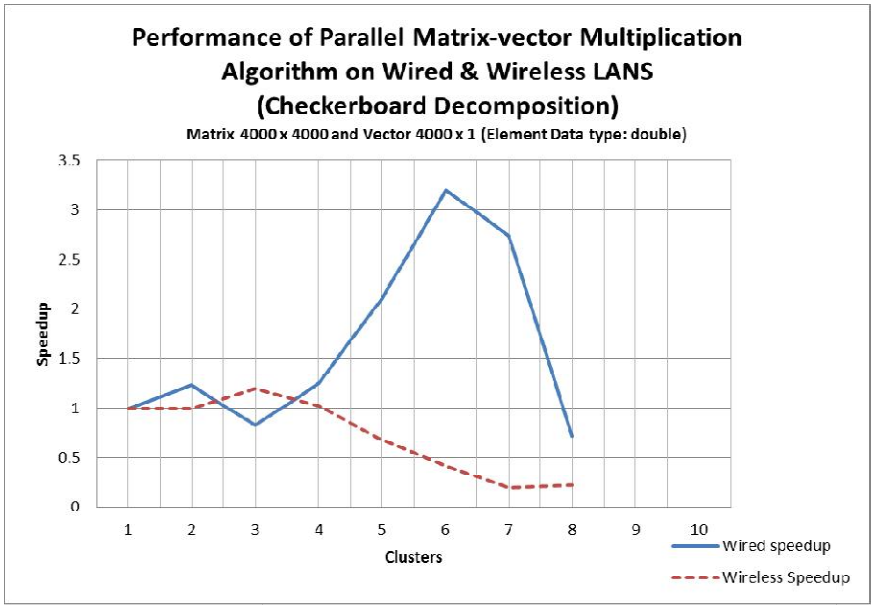

Fig. 7. Parallel matrix-vector multiplication: Checkerboard decomposition speedup on wired and wireless

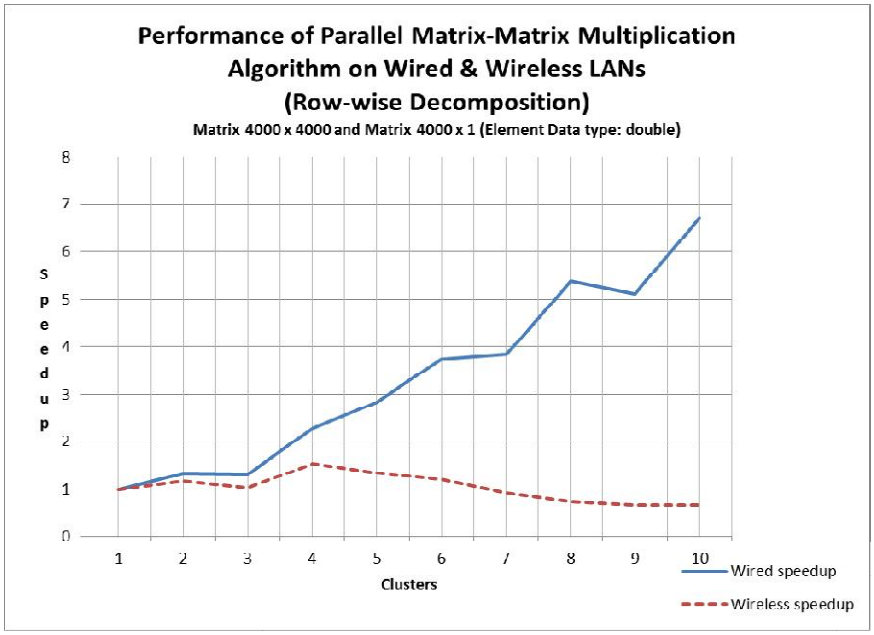

Fig. 8. Parallel matrix-matrix multiplication speedup on wired and wireless

In all the algorithms, it should be noted that each computer in the cluster has a 2.2Ghz processor and 2GB of RAM with a 1024KB cache. Therefore for matrix-vector multiplication involving a 4000 x 4000 matrix and a 4000 x 1 vector and algorithm time complexity of $\theta(n^2)$, computation is appreciably faster on each individual system. The processing speed of the individual computer contributes immensely to the speedup

observed for the parallel computation hence, the communication overhead forces a sharp decline in the overall performance as observed with the wireless networks speedup in Figs. 5, 6 and 7 respectively.

Also, all output shows the limit of parallelism as described by Gustafson's law. It shows that parallelism can only be exploited to a certain point where an additional processing nodes will cause overall parallel performance to degrade. The average point of degradation in this experiment was at cluster size three.

Fig. 7, showed a better performance for in the wired network speedup compared to Figs. 6 and 5. This shows that the parallel algorithm design plays a pivotal role in speedup as the checkerboard algorithm design outperforms the Row and Column-wise decomposition matric-vector multiplication, hence Algorithms can be fine-tuned for better performance however the improve performance cuts across the two types of networks observed.

In the parallel matrix-matrix multiplication algorithm in Fig. 8, computational and communication overhead is high hence parallelism is exploited more for wired/wireless network. But the wireless network degrades even with the high computational overhead inherent in the parallel matrix-matrix multiplication algorithm.

A further look in the data showed that the average blocking time for communication primitives like *All-Gatherv*, *All-Scatterv* exhibited longer execution time for wireless network (especially as the number of clusters increases) based on the nature of the medium of communication. The nature of the Fast Ethernet LAN (Full Duplex) makes it possible to eliminate collisions among the nodes or computers partaking in the computation. Connectivity in Ethernet LAN was made possible via a switch device. A typical switch device is a layer 2 device that breaks up collision domain among its interfaces, therefore the possibility nodes sharing communication medium is eliminated thereby eliminating collision and in effect, there is no need for CSMA/CD protocol.

However, the same cannot be said of the wireless LAN (IEEE 802.11g) setup in this experiment. Connectivity is achieved through an access point thereby creating a Star topology with all the computers contending for the medium. Therefore, CSMA/CA's random backoff timer or the RTS/CTS overhead within the network increases the overall network overhead thereby quickly nullifying the effect parallel design would have had on the overall speedup. The speed of wireless LAN network also plays a negative role as it slowed down the overall speed of the algorithm because speed of communication also accounts for the general speed of the parallel algorithm.

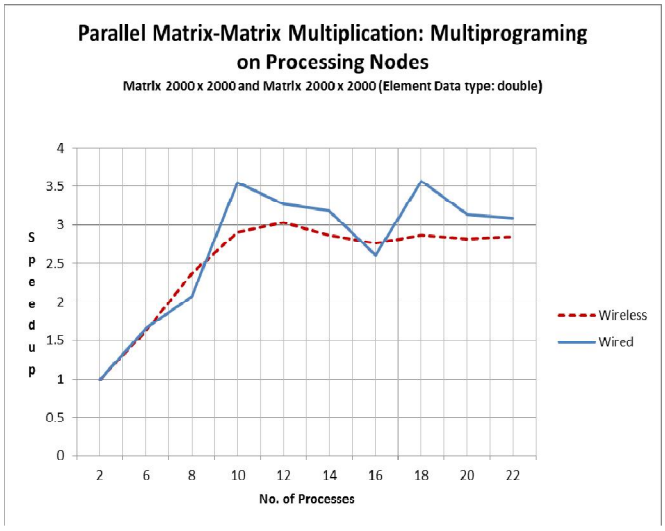

Fig. 9. Matrix-matrix multiplication-multiprogramming approach on two machines

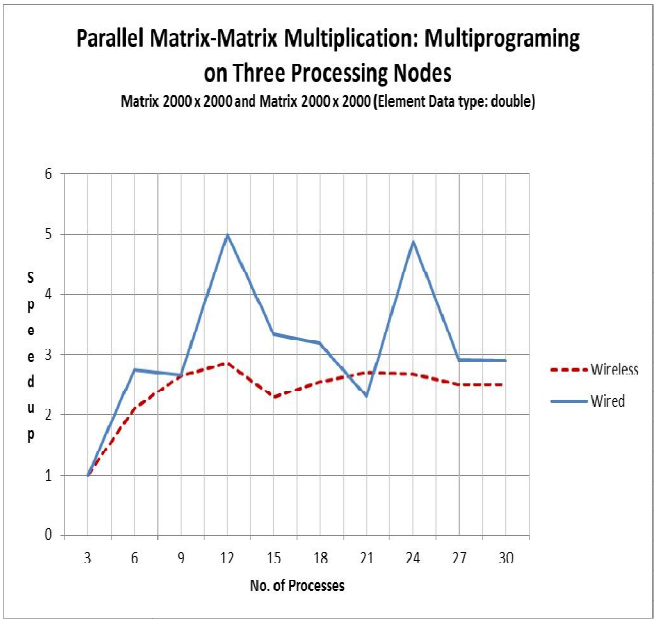

Fig. 10. Matri-matrix multiplication-multiprogramming approach on three machines

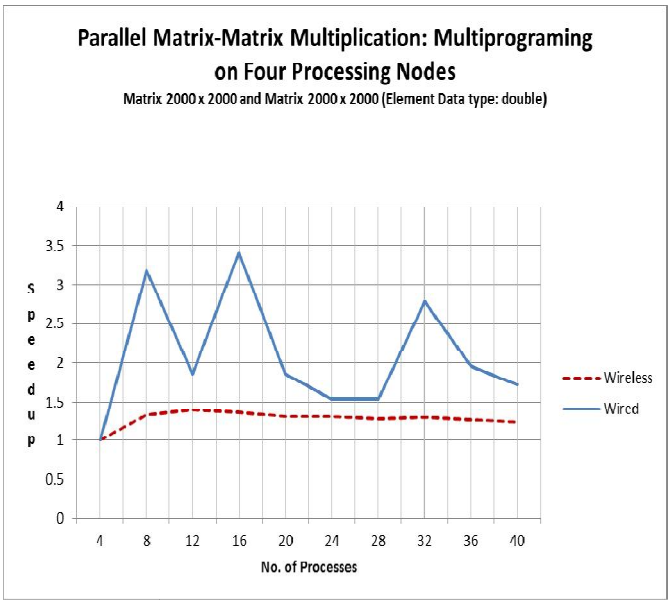

Fig. 11. Matrix-matrix multiplication - multiprogramming approach on Four machines

These fact is evident in above Figs. 9, 10 and 11.**Fig. Fig.** In the multiprogramming approach, multiple processes were executed per machine. This was to localise as much as possible process communication and therefore reducing the overall communications across machines. It was observed that both wired and wireless networks exhibited appreciable speedup for two machines. However, as the number of machines increases so also was communication hence appreciable speedup was no longer observed for the wireless network. This further buttresses the argument about how the nature of the wireless communication network affects speedup.

Hence, the greatest impact on the performance is the availability of contention for media resource.

5 Conclusion

This experiment results showed that parallel programs written for Beowulf clusters on wireless LAN (IEEE 802.11) do not gain appreciable speedup as the number of processing nodes increases compared to the same parallel programs written for the same Beowulf clusters but on wired LAN.

This is due to the nature and protocols of both networks. While the wired network (fast Ethernet) running on full-duplex mode suffers no collision and as such requires no CSMA/CD protocol with random back-off time, the wireless LAN (IEEE 802.11g) is half-duplex with an Access point and each station shares the same medium of communication (i.e, wave). The blocking nature of the send and receive primitive couple with the random back-off time contribute to the increase in delay experienced by the send/receive primitives of the parallel programs hence nullifying the positive effect an increase in the number of processing node in a Beowulf cluster –deployed on a wireless LAN -would have on the overall speedup of parallel program.

It also shows that any improvement or performance gain observed from redesigning an algorithm for wired networks will also be seen in the wireless environment albeit not so obvious due to the nature of the wireless network.

Competing Interests

Authors have declared that no competing interests exist.

References

[1] Linda Null, Lobur Julia. The essentials of computer organization and Archictecture. London: Jones and Bartlett Publishers, Inc.; 2006.

[2] John L. Hennessy, David A. Patterson. Computer architecture, a quantitative approach 4th edition. Amsterdam: Morgan Kaufmann Publisher; 2007.

[3] Ananth Grama, et al. Introduction to parallel computing. s.l.: Addison Wesley, January 16; 2003.

[4] Ravela Srikar Chowdar. Comparison of shared memory based parallel programming models: Thesis Report. Blekinge Institute of Technology, Ronneby, Sweden: s.n.; 2010.

[5] MPI, website. Standard Message Passing interface MPI. [Online] 2013. [Cited: 01 28, 2013]. Available: http://www.mcs.anl.gov/research/projects/mpi/

[6] Michael J. Quinn. Parallel programming in C with MPI and OpenMP. New York: McGraw-Hill; 2003.

[7] Gropp William, Ewing Lusk, Thomas Sterling (eds). Beowulf cluster computing with Linux. Massachusetts London: The MIT Press; 2003.

[8] Parallel Computing in Local Area Networks. Fernando, Gustavo, Tinetti. Journal of Computer Science; 2004.

[9] Cameron Hughes, Tracey Hughes. Parallel and distributed programming in C++. s.l.: Addison Wesley, August; 2003.

[10] Website, MPICH. MPICH. [Online] 2013. [Cited: 01 23, 2013.]
Available: http://www.mpich.org

[11] DeinoMPI. DeinoMPI Documenation; 2009.

[12] Behrouz A. Forouzan, Sophia Chung Fegan. Data communication and networking. New York: McGraw Hill; 2007.

[13] Dharma Prakash Agrawal, Zeng Quin-An. Introduction to wireless & mobile systems. Stamford: Cengage Learning; 2011.

[14] Wikipedia. Wireless Networks. [Online] 2013. [Cited: January 31, 2013].
Available: http://en.wikipedia.org/wiki/Wireless_network

[15] Wikipedia. Carrier Sense Multiple Access with Collision Avoidance. [Online] 2013. [Cited: February 1, 2013.]
Available: http://en.wikipedia.org/wiki/Carrier_sense_multiple_access_with_collision_avoidance

Peer-review history:
The peer review history for this paper can be accessed here (Please copy paste the total link in your browser address bar)
http://sciencedomain.org/review-history/12383