ISSN: 2350-0328

International Journal of Advanced Research in Science, Engineering and Technology

Vol. 6, Issue 9 , September 2019

Algorithms and Programs for the Distributed Processing of Two-Dimensional Signals on a MPI Computing Cluster based on the Haar Wavelet

Utkir Khamdamov, Khakimjon Zaynidinov

Associate professor, Department of Hardware and software of control systems in telecommunications, Tashkent University of Information Technologies, Tashkent, Uzbekistan
Professor, Department of Information Technologies, Tashkent University of Information Technologies, Tashkent, Uzbekistan

ABSTRACT: In this paper, algorithms and software for the distributed processing of two-dimensional signals in a computing cluster based on the Haar wavelet transform have been investigated and developed. The research was conducted on the basis of the MPI computing cluster, which consists of one head node and four computing nodes based on homogeneous computers using HPC technologies to implement distributed processing algorithms for two-dimensional signals. The Windows HPC Server 2008 R2 operating system and the HPC Pack 2008 software package with cluster operation utilities were used as the cluster management software. As the results of the study and computational experiments, the results of evaluating the performance of the distributed signal processing algorithm on a computing cluster and the level of utilization of the data transfer network between cluster nodes have been presented. The corresponding evaluations of computational acceleration based on parallel algorithms based on MPI technologies have been proven.

KEYWORDS: wavelet transform, Haar wavelet, MPI cluster, two-dimensional signal, image processing, SPMD, compute node, parallel processing algorithm.

I.INTRODUCTION

MPI (Message Passing Interface) is a specification of a message passing library interface. First of all, the MPI is oriented towards a parallel programming model with messaging in which the data is moved from the address space of one process to the space of another process through joint operations on each process. The main purpose of the messaging interface is to develop a widely used standard to support the development of parallel programs. The MPI interface [4] is intended for use by the developers of parallel computer programs based on the messaging in C. This includes developers of individual applications, developers of software which dedicated for working on parallel machines, as well as developers of environments and tools. The interface provides a simple, easy-to-use user interface with support for the high-performance messaging operations available on modern computers.

MPI technology supports the creation of parallel programs based on MIMD (Multiple Instruction Multiple Data), which involves combining processes with different source codes of programs. The development and debugging of such programs are very difficult, therefore, in practice, the SPMD (Single Program Multiple Data) models [8] is used for parallel programming. In such programs, all parallel processes utilize the same source code of the program. Programs written on the ground of MPI demonstrate the aggregation of simultaneously interacting processes. All processes are generated once by forming a parallel part of the program. The creation of additional or destruction of existing processes is not allowed during the execution of the MPI program. Each process operates in its own address space and common variables or data do not exist in MPI. In this case, the main way of interaction between processes is the transmission of messages.

ISSN: 2350-0328

International Journal of Advanced Research in Science, Engineering and Technology

Vol. 6, Issue 9 , September 2019

In order to localize the interaction of parallel processes of the program, the set of processes can be created by providing them with a separate environment for communication that is a communicator [3]. Processes interact only within the communicator, while the messages sent in different communicators do not overlap and do not affect each other. Communicators in C / C ++ have the type of MPI_Comm.

During the launching of the program, all processes work with a common communicator, so-called MPI_COMM_WORLD [1]. This communicator will always exist and will serve for the interaction of all running processes of the program. In addition, when starting the program, there may be a communicator MPI_COMM_SELF containing only one current process, as well as a communicator MPI_COM_NULL that does not contain any processes. Each process of the MPI program has a unique identifier for the process number in each group into which it belongs, which is a non-negative integer. A significant part of the interaction between processes occurs using this identifier. Therefore, all processes in one communicator have different numbers, but if a process simultaneously enters into different communicators, then its number in one communicator may differ from its number in another communicator. According to this, each process has two attributes: the communicator and the number in the communicator.

The main method that processes communicate with each other is by sending messages. A message is a data set that has several attributes, such as the number of the sending process, the number of the receiving process, the identifier of the message, and others. One of the important attributes of a message is its identifier (tag) [7]. The process identifier distinguishes between messages that came to it from the same process. The message identifier is a non-negative integer within the range from 0 to MPI_TAG_UP.

Almost all programs developed using MPI communication technology contain tools not only for generating and completing parallel processes but also for the interaction of running processes with each other. Such interaction in MPI technology is carried out by sending messages [2]. Message passing procedures are divided into two groups. The first group includes procedures for the interaction of only two program processes. Such procedures are called point-to-point type. The procedures of another group include the interaction of all the processes of a communicator and these procedures are called collective.

II. RELATED WORK

Parallel programming frameworks such as MPI, OpenMP and OpenCL have been widely used in many scientific domains to implement distributed applications. While they have the same purpose, these frameworks differ in terms of programmability, performance, and scalability under different applications and cluster types. In the work [10] considered several popular parallel programming frameworks for distributed applications. Analyzed memory model, execution model, synchronization model and GPU support. Compared programmability, performance, scalability, and load-balancing capability on homogeneous computing cluster equipped with GPUs. In the work [11] presented an implementation of a heterogeneous programming model which combines OpenCL and MPI. The model is applied to solving a Markov decision process with value iteration method. In the work [12] presented fast-mRMR-MPI, a novel hybrid parallel implementation that uses MPI and OpenMP to accelerate feature selection on distributed-memory clusters. The performance evaluation on two different systems using five representative input datasets shows that fast-mRMR-MPI is significantly faster than fast-mRMR while providing the same results.

III.SCOPE OF RESEARCH AND PROPOSED METHODOLOGY

A. MATHEMATICAL MODEL OF THE WAVELET TRANSFORM OF SIGNALS

From a mathematical point of view, the wavelet functions are localized functions in frequency and time. The wavelet transform of the signals is based on the use of two functions: the wavelet function and the scaling function, i.e. they are constructed from the same mother wavelet ψ (t) by temporarily displacing the signal b and changing the time scale$a$ [6]:

$$\psi_{ab}(t) = \frac{1}{\sqrt{(|a|)}}\psi\left(\frac{t-b}{a}\right), \quad (a,b) \in R, \quad \psi(t) \in L^2(R) \qquad (1)$$

In digital signal processing, the wavelet function is used to extract the details of the signal and its local features, and the scaling function is used to approximations of the signal. In practice, depending on the class of signals, continuous and discrete wavelet transformsare used to the signal processing. In the current study, the discrete wavelet transform methods have been chosen as the mathematical model, since the signals used in the experiments are digital.

ISSN: 2350-0328

International Journal of Advanced Research in Science, Engineering and Technology

Vol. 6, Issue 9 , September 2019

As a result of a discrete wavelet transform, the processed signal is divided into two components of equal size. One of them is the averaged signal a_n or approximation and the other is a differential signal d_n or details. In studies for the wavelet transform of signals, the Haar wavelet function is used as a mathematical model for signal filtering. The Haar wavelet transform is based on the following equation:

$$a_n = \frac{f_{2n-1}+f_{2n}}{\sqrt{2}},\ d_n = \frac{f_{2n-1}-f_{2n}}{\sqrt{2}},\ n = 1,2,3,\dots,N/2 \quad (2)$$

The use of wavelet transform methods based on the lifting scheme and bandpass filters provides the possibility of developing effective and fast algorithms for digital processing of one-dimensional and two-dimensional signals.

B. MESSAGE EXCHANGE PROCEDURES BETWEEN THE PROCESSES OF THE CLUSTER

In the computing cluster [5], point-to-point operations are assumed for the parallelization of the processes of image processing based on the Haar wavelet. In this case, the interaction of the program processes occurs between two processes. The first process is the sender of the message, and the other is the recipient. The sending process calls one of the data transfer procedures and indicates the number of the receiving process in some communicator. Therefore, the recipient process must call one of the procedures for getting data indicating the same communicator. The transfer of an array of data between processes is carried out in a synchronous manner using the MPI_SEND command.

MPI_Send (Sender, n * m, MPI_FLOAT, i, TagA, COMM);
Sender - data transfer buffer;
n*m is the number of transmitted elements;
MPI_FLOAT - a type of transmitted data;
i is the number of destination process;
TagA - message tag;
COMM - communicator for the interaction of processes;

The present line of code of the program does the transfer of the Sender data array with TagA identifierconsisting of *n*m* elements of MPI_FLOAT type, to the process number *i* in the COMM communicator. All elements of the message that to be sent must be located in a row in the Sender buffer.A message is received, that is, an array of data from the sending process is performed using the MPI_RECV command.

MPI_Recv (Receiver, n*m, MPI_FLOAT, root, TagA, COMM, & status);
Receiver - data reception buffer;
n*m is the number of received elements;
MPI_FLOAT - a type of received data;
root - source process number;
TagA - message tag;
COMM - communicator for the interaction of processes;
Status - message status.

Using this line of code, the program receives a data array into the Receiver buffer of no more than n*m message elements of the MPI_FLOAT type with TagA identifier from the process with root number in the COMM communicator with filling in the attributes array of the incoming Status message. If the number of actually received elements is less than n*m, it is guaranteed that only the elements matching the elements of the received message are changed in the Receiver buffer. If the number of elements in the received message is greater than n*m, an overflow error occurs. To avoid this, the structure of the incoming message using the MPI_PROBE procedure can be determined. To find out the exact number of elements in a received message, the MPI_GET_COUNT proceduremust be used.

In the ideal case, each transmission operation will be synchronized with its corresponding reception. But in the real case, it happens that the data transfer operation occurs earlier than the readiness to receive, or several messages arrive at the same recipient process, which is able to receive only one message at a time. This problem associated with storing messages until the recipient process is ready and it is solved with the implementation of the buffer in MPI applications. In this case, the transmitted data array with a certain number of elements goes to the receiver's system buffer, accordingly, it is accepted by the process.When the synchronous transmission or reception is called, the system suspends the execution of the user program until the message transmission/reception buffer becomes available for use.

ISSN: 2350-0328
International Journal of Advanced Research in Science, Engineering and Technology
Vol. 6, Issue 9 , September 2019

When transmitting messages, the transmitted data will be copied from the transfer buffer, and when receiving messages, the received data will be located in the receive buffer.
In the experiment related to the signal transform based on the Haar wavelet, the values of two-dimensional signal (image) with a size of 1024x1024 are usedas an array of data. In order to parallelize the calculations, the MPI program using the attribute of MPI_Comm_size determines the number of processes. Based on the number of processes, the data array of the processed two-dimensional signal is distributed according to the principle of "one data block to one processor", as shown in Fig. 1.

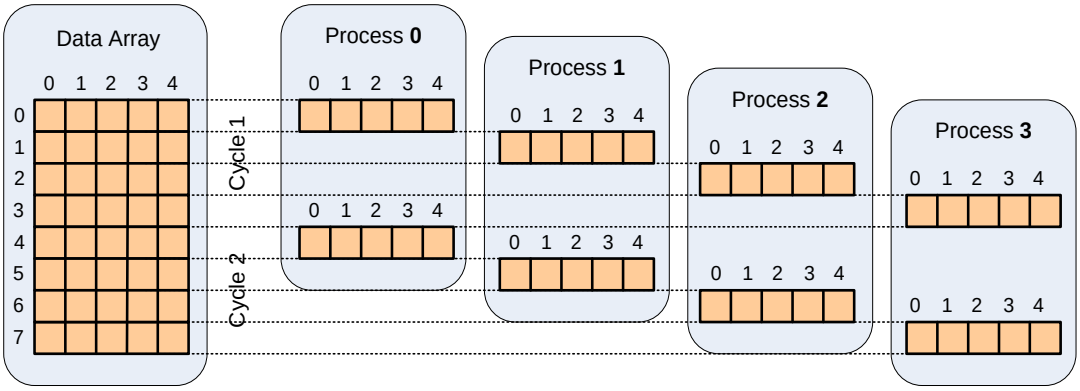

Fig.1 The procedure for the distribution of the data between processes.

In this case, the data block includes either one row of elements of the data array of the reduced signal, consisting of 1024 values, or either part of a two-dimensional array with a size of 256x256, 512x512. Distribution of string data of an array between processors occurs in stages. The given data array will be distributed completely in 256 cycles. If necessary, one line of the array can also be distributed evenly between the four processors. In this case, 1024/4 = 256 values are distributed for each processor, therefore, the complete array transform takes place in 1024 cycles.

C. ALGORITHM OF THE DISTRIBUTED SIGNAL PROCESSING

After each data distribution cycle, the transform process is performed in parallel on 4 processors. Parallel processing of the values of a two-dimensional signal using the Haar wavelet transform [9] is carried out according to the SPMD principle -single program multiple data, as shown in Fig. 2.In order to perform a parallel signal transform, the control program loads an instance of the Haar wavelet transform program (ParallelHWT) on each processor creating separate processes. The MPI program determines the process numbers that are running on the processors by using the MPI_Comm_rank attribute. The program will begin the data distribution to the processes for parallel computation as the process identifiers have been determined. Process data is delivered using MPI_SEND and MPI_RECV procedures.In this case, the dt [x] array with the size oflen = 256 is used to distribute the data. In the first processing cycle, each processor receives the following data:

CPU 1: dt[0:255] with the size oflen = 256.
CPU 2: dt[256:511] with the size oflen = 256.
CPU 3: dt[512:767] with the size oflen = 256.
CPU 4: dt[768:1023] with the size oflen = 256.

Each processor receives an array of data with the same number of elementswhile processing two-dimensional signals. At the same time, the main MPI program creates a control process (P0) for creating other processes, distributing data and collecting results (Fig. 3).While in the control process, the main body of the program loads copies of parallel sections of the programs on individual processors, creating processes P1, P2, P3, P4 in it. Consequently, it distributes the rows of data array elements to the corresponding processes. As a result, it collects the results of parallel computations from processes.
The process of parallel image processing in a computing cluster occurs in the following order:

- ProgramLoading: - debugging and loading the program to a shared folder on the head node of the cluster;

ISSN: 2350-0328
International Journal of Advanced Research in Science, Engineering and Technology
Vol. 6, Issue 9 , September 2019

- Program Deployment: - deployment and launch of the program on computing nodes;
- Data distribution: - distribution of vector data by 256 values for each process running in computing nodes;
- Data Computation: - reception and calculation of values of a part of a vector in computational nodes and reverse transmission of results;
- ResultsCombination: - receiving the results in the main process in the same manner as they were distributed and combining them;
- Results Output: - representation of processing results to the user.

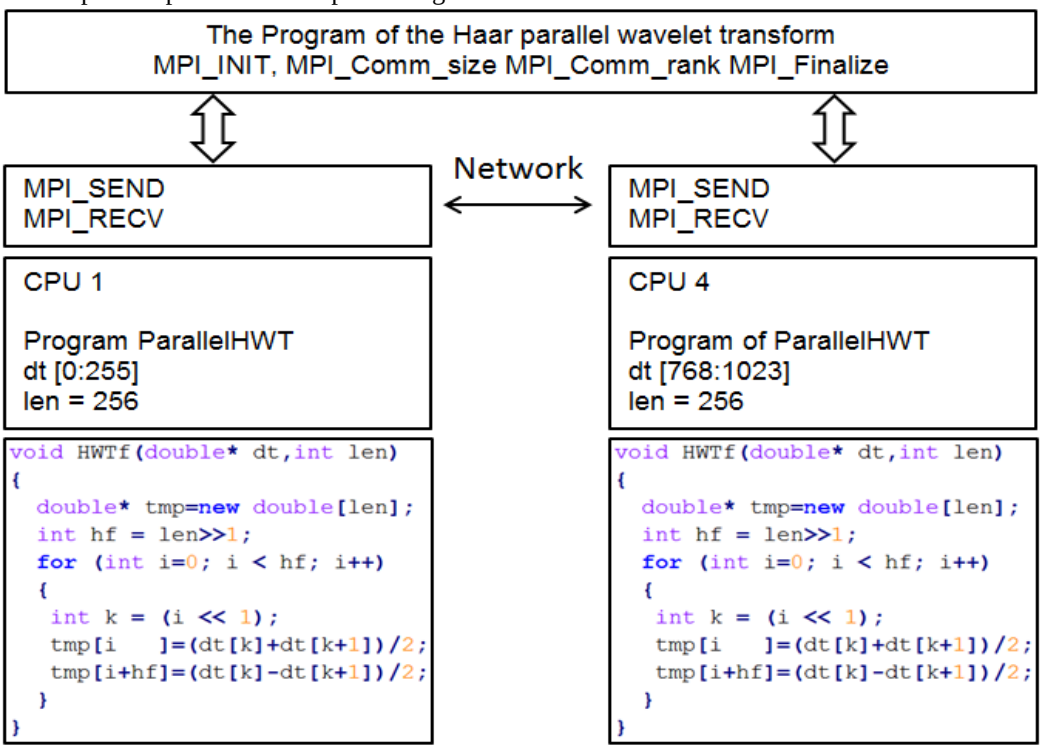

Fig.2 The procedure for the data exchange between processes based on MPI

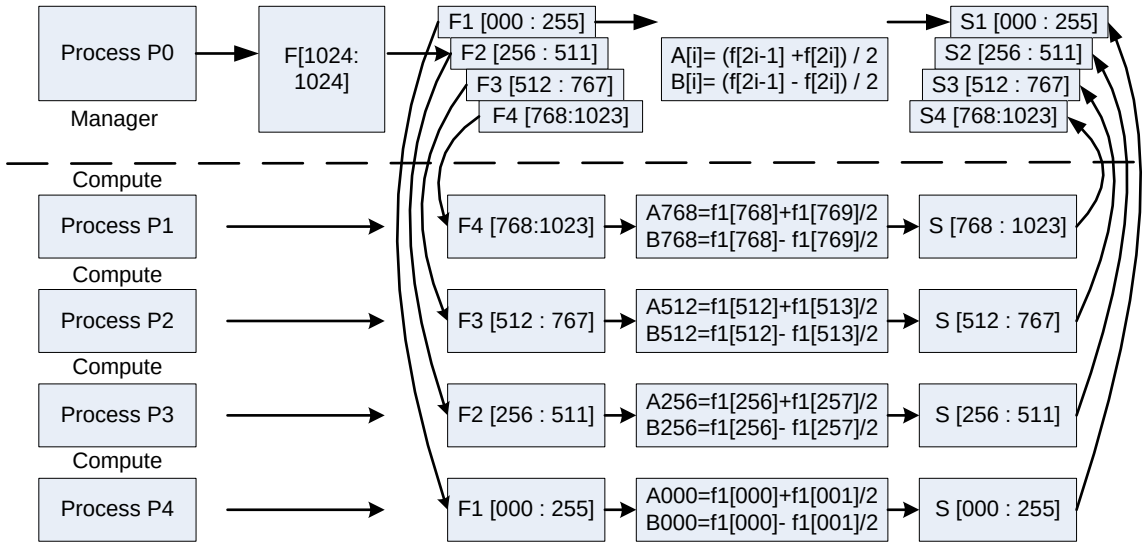

Fig.3 Managing Parallel Processes and Data in MPI

ISSN: 2350-0328
International Journal of Advanced Research in Science, Engineering and Technology
Vol. 6, Issue 9 , September 2019

To implement and evaluate the acceleration of the parallel wavelet transform algorithm for the image signals based on the Haar wavelet in multiprocessor systems, the studies were conducted on debugging the application for one MPI process on the local computer and several MPI processes on the local computer, as well as debugging several MPI processes in the cluster.

IV.EXPERIMENTAL RESULTS

To conduct computational experiments and evaluation of the capabilities of MPI applications, the developed algorithm uses three options for distributing data between MPI processes:

1) block-wise distribution of rows of an image array: - each row of a two-dimensional array is distributed evenly between 4 processes, and the number of distribution cycles is equal to the number of rows;
2) line-by-line distribution of the image array: - each row of a two-dimensional array is completely distributed between 4 processes in turn, and the number of distribution cycles becomes four times less than the number of lines;
3) distribution of the entire image array: - in this case, the option of processing several images from the video stream is considered simultaneously, which in each distribution cycle for each process a two-dimensional array of the entire image is assigned, and the number of cycles is reduced by four times than the number of frames in the video stream.

The main results obtained in the implementation of the parallel image processing algorithm based on the MPI cluster and the cluster characteristics are presented in Fig. 4-6. Fig. 4 shows the results of accelerating the wavelet image transform process based on MPI multiprocessing. The time taken to calculate the image processing algorithm on one process of a serial computer and on four MPI processes of a local computer is presented. In this case, the block-wise distribution of the rows of the image array is carried out. The results show that with the increasing the image sizes, especially with the size of 2048x2048, the processing time of sequential algorithm increases sharply than a parallel algorithm based on MPI processes. As a result, the parallelization of processes using MPI technologies received 3.2 times acceleration of the algorithm.

Fig. 5 shows the processing time of images from a video stream, thathas 24 image frames, which is the duration of the first second of a video stream in MPI processes of a local computer based on the proposed algorithm. In this case, the algorithm performs a line-by-line distribution of the image array between MPI processes. As can be seen from the graphs while processing images of a video stream of 24 frames with a size of 2048x2048, the processing time based on the MPI algorithm is 3.6 times less than a sequential algorithm. From the results, we can say that the parallelization of the processes of image processing based on the proposed algorithm can be applied in real-time for images with sizes up to 2048x2048.

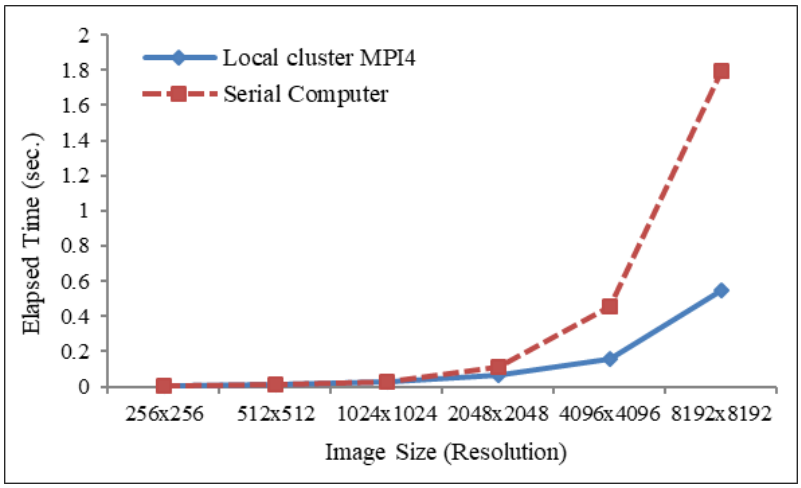

Fig.4 Image processing time in MPI processes of the local computer

ISSN: 2350-0328
International Journal of Advanced Research in Science, Engineering and Technology
Vol. 6, Issue 9 , September 2019

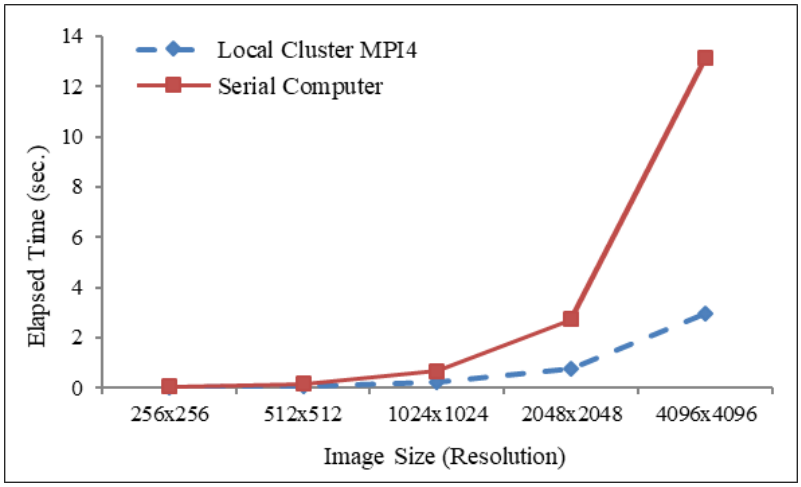

Fig.5 Processing time of 24 video frames in MPI processes of the local computer

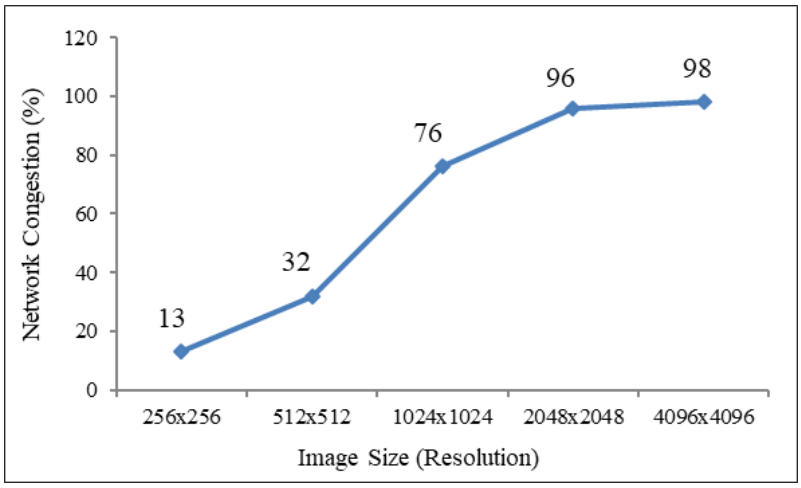

Fig.6 Network Usage in a Distributed Cluster

Since the computing cluster is built on the basis of a local area network with a bandwidth of 100 Mbps, its parameters and congestion are important in speeding up the calculations. Fig. 6 shows the results of the network usageby the cluster. The figure also represents that, with the increasing of the image size, the network load also increases, because in this case an entire array of image data is allocated for each cluster process. While, exchanging the images with sizes of 2048x2048 and more the network congestion is achieved at 96-98%.

V. CONCLUSION

The results of experiments on parallelizing calculations based on the developed algorithm show that with increasing the image size, the processing time of a sequential algorithm increases sharply than that of a parallel algorithm based on MPI. As a result, parallelization of signal processing using MPI technologies, acceleration of the algorithm is 3.2 times faster than a sequential algorithm.

In addition, from the results of image processing of frames of a video stream with a resolution of 2048x2048, it can be highlighted that the signal processing time based on the MPI algorithm is 3.6 times less than the sequential algorithm. It worth to emphasize that the parallelization of the processes of image processing based on the proposed algorithm can be applied in real-time for the images with sizes up to 2048x2048.

ISSN: 2350-0328

International Journal of Advanced Research in Science, Engineering and Technology

Vol. 6, Issue 9 , September 2019

Thus, we can say that network bandwidth also affects the overall performance of the cluster, since the network connects all the computing nodes of a distributed cluster. As the image size increases, network congestion also increases. From the starting the exchange of images with dimensions of 2048x2048, the network congestion has been achieved at 96-98%.

REFERENCES

[1]. Frank Nielsen,"Introduction to HPC with MPI for Data Science", Springer International Publishing Switzerland, 2016, – 282 p.
[2]. ShenHua, Zhang Yang, "Comparison and Analysis of Parallel Computing Performance Using OpenMP and MPI",TheOpenAutomationandControlSystemsJournal, Vol. 5, pp.38-44, 2013
[3]. Antonov A.S.,"Parallel Programming Using MPI Technology: Textbook", M.:MSU, 2004. – 71 p.
[4]. Khamdamov U.R.,"Efficiency analysis of computational MPI cluster with limited buffer based on queuing model", Problems of computational and applied mathematics, Vol. № 4 (22), pз. 37-46, 2019
[5]. Klochkov M.A.,"High-Performance Computing Basics in Windows HPC Server 2008", Ijevsk: «Udmurt University», 2010,- 62 p.
[6]. SundararajanD,"DiscreteWaveletTransform: ASignalProcessingApproach",JohnWiley&SonsSingaporePte. Ltd, 2016. - 344 p.
[7]. EnsarAjkunic, Hana Fatkic, EminaOmerovic, Kristina Talic and NovicaNosovic,"A Comparison of Five Parallel Programming Models for C++", MIPRO 2012/SP, - p. 2203-2207, 2012
[8]. Fayez Gebali,"Algorithms and parallel computing", John Wiley & Sons,2011,- 365p.
[9]. Khamdamov U.R.,"Algorithms for parallel Bitmap image processing based on the Haar Wavelet", International Conference on Information Science and Communications Technologies ICISCT 2017,Tashkent, Uzbekistan, November 2-4, 2017
[10]. Gu R., Becchi M.,"A Comparative Study of Parallel Programming Frameworks for Distributed GPU Applications", ACM International Conference on Computing Frontiers (CF 2019), Alghero, Italy, 30 April 2019, pp. 268-273
[11]. Li M., Hawrylak P., Hale J.,"Combining OpenCL and MPI to support heterogeneous computing on a cluster", 2019 Conference on Practice and Experience in Advanced Research Computing: Rise of the Machines (Learning), PEARC 2019, Chicago, United States, 28 July 2019
[12]. González-Domínguez J., Bolón-Canedo V., Freire B., Touriño J. "Parallel feature selection for distributed-memory clusters", Information Sciences, Vol. 496, pp. 399-409, 2019

AUTHORS' BIOGRAPHY

Utkir Rakhmatillayevich Khamdamov – PhD in Technical Sciences, Associate Professor, Department of Hardware and Software of control systems in Telecommunication of the Tashkent University of Information Technologies named after Muhammad al-Khwarizmi.
100200, Tashkent, 108, Amir Temur str.

HakimjonNasriddinovichZayniddinov– Doctor of Technical Sciences, Professor, Head of the Department of Information Technologies of the Tashkent University of Information Technologies named after Muhammad al-Khwarizmi.
100200, Tashkent, 108, Amir Temur str.