Implicit in the development of parallel programming language using Visual Basic

Zainab fahad mhawes al-naseri
Department of mathematic
College of Education/University of Qadisiyah
Al-Qadisiyah, Iraq
Zainab.alnaseri@qu.edu.iq

Received july 27, 2017. Accepted for publication january 16, 2018
DOI: http://dx.doi.org/10.31642/JoKMC/2018/050101

ABSTRACT*:Several studies have been performed to investigate the benefits of parallel programming models, and compared the models with same language family. The parallel technique is applicable only when the programmer has a single computer with multiple processors to perform various tasks individually or the programmer number of different computers connected by a single network. Parallel programming model in computing is an idea of parallel computer architecture. This research introduced the concept of parallel programming models and languages. The present research mainly focused on the different parallel programming models proposed by previous researchers using distinct programming languages. Through analyzing the strengths and weaknesses of various programming models used for parallel computing, a new model is developed. Along with this, the concept of OOP (Object-oriented programming) in the context of parallel computing is also discussed. This research proposed a model named as Asynchronous Dynamic load Balancing (ADLB). ADLB model used MPI as avehicle for library implementation rather as an end-user programming system.*

keyword—Parallel programming, Parallel computing, MPI, Xcalable or XMP, Programming language, OOP.

I.INTRODUCTION

Parallel computing is a type of communication in which large calculations or the execution of programs are carried out simultaneously. Large problems could be divided into smaller parts, which then can be further solved in the same time. Parallel computing is the process utilized for parallel programming. In parallel computing, large computations are broken into small subtasks that can be computed independently, and the results are combined after completion.

II.1 Parallel Programming and Parallel Computing

Parallel computing is a type of system in which a number of calculations or processes are executed at the same in stance of time. In case there are a large number of problems in a process or

function, they can be divided into parts and can be solved separately at the same time. Parallel computing uses elements, which have multi-processing on abilities to work simultaneously. It means breaking the problem into parts and working on each of them at the same time. [7]

Under parallel programming, more than one processors is used to complete the task as each processing unit has its own qualities and limits. Under this type of programming, the tasks are independent, and the order of execution does not matter. The processors work separately on functional and data parallelism. The performance of single CPU is limited due to available memory and its performance therefore parallel programming can help solve problems at a faster rate and with more ease.Even a single server system can hold parallel programming with the help of multi-level processors. The art of using computer systems, which have multiple processing elements is known as parallel computing. [7] The operation of parallel computing is shown in the figure.

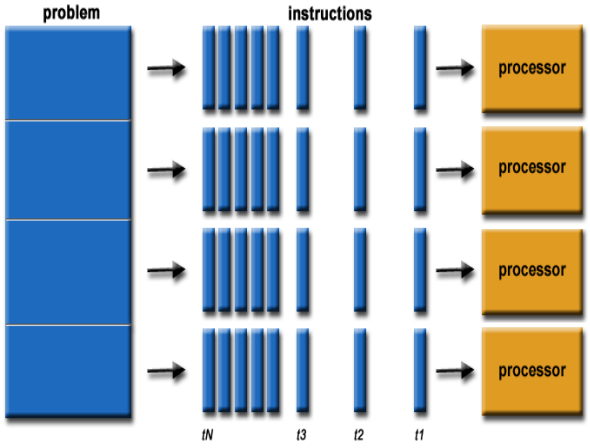

Figure 1:Parallel Programming operation

II.2 Different Models of Parallel Programming Platforms

Programming languages emerge as a source that one can program parallel computers whiche become larger and complicated. As computers become significant and have vast memories,to achieve higher performance, a programming language is necessary. In programming, we use an inter-node interface such as Message Passing Interface for coordinating nodes and reducing the complexity X calableMP extension is used. An example of Partitioned Global Address Space language is XcalableMP or (XMP).

II.2.1 XcalableMP or (XMP)

XcalableMP (XMP) is a directory based language extension and allow users to compute programs for distributed memory systems efficiently. XMP tackles typical parallelization methods based on the data parallel paradigm and makes the parallelization of the original sequential code possible with minimum amendments. Although MPI is a parallel programming on distributed memory systems , writing MPI programs is a complicated and time-consuming process.XcalableMP is a language extension registered in the library of C and Fortran for distributed memory systems that help users to tackle with programming efforts. XcalableMP includes two programming models. The first one is the global model, which aids parallelization based on the data and parallel paradigm and makes possible the parallelizing of original sequential code using minimal modification with Open MP- like directives. [6]

II.2.2 CUDA

CUDA, Stands for Computer Unified Device Architecture.CUDA, is a platform for parallel computing and Application Processing Interface, which was developed by NVIDIA. CUDA is used with programming languages such as C, C++, and Fortran. CUDA makes it easier to us GPU resources in parallel programming. NVIDIA coupled the intuitive software and hardware tools and hailed Ian Buck to join the company and start inventing a solution to run C language programs on the GPU gradually. Putting the software and hardware together, NVIDIA disclosed CUDA in 2006, which emerged as world's first ever solution for general computing on GPUs. Today, the CUDA platform is growing as other companies are providing world-class tools,and services. If the user wants to write his own code, the easiest way to increase the performance of GPUs is through CUDA toolk it which provides a convenient way for C and C++ developers. [1]

II.2.3 Message Passing Interface(MPI)

Message Passing Interface (MPI) is a parallel computing communication protocol. It is a highly dominant model of modern high-performance computer systems. The MPI is designed to provide the essential synchronization and functionality of communication between a different set of functions assigned by the server to different processors. For best performance, each processor is assigned to the single process.The assignment is created by the agent who starts the MPI program called mpiexec or mpirun. MPI provides the facility of message passing to the programs written in Fortran or C which results in the easy interchange of programs without losing information. Using MPI, the user tackles with the capital cost occurring in the program. MPI was the first message passing technique across the whole parallel processing communication. Nowadays, MPI becomes the basis for most communication interfaces employed by parallel computing programmers. [12]

II.3 Parallel Programming Languages

Parallel Programming Languages are the languages designed to support parallel computing rather than sequential languages, Though some of them are based on traditional languages ,The hope is that they comprise code bases that can be reused. This dissertation classifies parallel languages as being either Global-view or Local-view.

Global-view languages are those languages in which the programmer describes the behavior of their algorithm once as a whole, avoiding the fact that multiple processors will be used to implement the program. The compiler is therefore responsible for managing the parallel implementation data, including data communication and Distribution. Many global-view languages are providing language-level concepts that are made specifically for parallel computing. The ZPL language is one such example. [6]

Local-view languages are those languages in which the program implementer is responsible for specifying the program’s behavior on a multi-processor basis. Thus, details such as data distribution, communication, and load balancing must be handled by the programmer. The primary advantage of local-view type languages is that users have complete control of the parallel implementation of their programs allowing him to implement any parallel algorithm that he wishs to do. [6]

II.4 Object Oriented Programming Language

The parallel programming language is a kind of program which is used for designing algorithms and apps using different class and objects as parts of the same program when working on parallel

computers. To ensure the easy functioning of work with lesser errors, reduces the complexity of work, Parallel programming languages are also known as concurrent languages as they can process mapped concurrent actions to design parallel algorithms of types of languages help which easy execution and reduction of design time. Unlike conventional programming languages, the programming syntax of object-oriented programming language (OOPL) supports one or more objects, whereas another procedural language follows logical procedures. In OOPL, objects interact with one another; have their own methods, functions, and procedures; are part of a class and may be again used in one or more programs. [7]Most programming languages today are object-oriented or support the OOP model some extent. Some of the Popular OOPLs are Java, C++, Python, and Smalltalk. One of the major benefits of object-oriented programming languages over procedural programming language is that they enable programmers to create such type of modules , which do not need to be changed when a new object is added.[7]

II.5 High-Performance Computing with Parallel Programming

Parallel programming is a big achievement in the field of computer programming and application designing industry. It enables a group to work on a problem more effectively by working on separate elements which have reduced the work load as well has increased the ease of working. Two or more than two systems can communicate and send and receive information through the use of Message Passing Interface. The complex task now can be assigned to single processor or multiprocessors for problem-solving and task performing. The use of techniques such as MPI and high-level programming language have made the performance of computers far better than the earlier times.The main application of parallel machines can be found in scientific and engineering computing, where high-performance computing (HPC) systems are employed today with hundreds or even thousands of

III-RELATED WORKS

Karlin, et al.compared different implementations of LULESH (Livermore Unstructured Lagrange Explicit Shock Hydrodynamics), a proxy application for parallel computing, to determine the best programming model for use in parallel computations. The author expressed a problem associated with parallel programming models like C,C++ and MPI in which it is difficult to write source code which is unique for many results. The study focused on traditional languages like Open MPI,MPI,CUDA and four recently developed programming models (Charm++, Chapel, Liszt, and Loci). By evaluating all these programming models, the results observed the better productivity and ease of optimization.[1]

N. Vanneschi proposed some new approaches in ASSIST, a software development system based upon integrated skeleton technology to overcome the limitations of traditional skeletons. The author wish to overcome the limitations of ASSIST that came in his previous experience. A new paradigm called “parallel module” was defined, which is able to express more external data structures. The motivation of this research was the requirement of development of High-performance software component technology and overcoming the limitations of structured programming language.[2]

J. Mcguiness and C. Egan defined an orthogonal DSEL (Domain Specific Embedded Language) to improve parallel programming because it gives

correct and efficient schedules and guarantees algorithmic run-time. He said that the increase in development of multiple computers has made programming possible in parallel computing but it is yet hard to program parallel algorithms correctly. An implementation of DSEL in C++ is shown in the document. With the help of DSEL, one can add effective schedules to program algorithm , which so assists the user in debugging.[3]

A.C. Chozas, et al. expressed the challenges faced while using OpenMPin parallel programming and described the use of IBM Watson Cognitive system to know about novice parallel programmers. Using the dialogue service IBM Watson, the author has introduced a solution which will help to avoid common OpenMP mistakes. The author conducted a survey with novice parallel programs and proved that results are better with use of IBM solution. [4]

M. Perovsek, et al. presented a web-based natural language processing and text mining platform. The research explained the text mining and language processing module.The research presented the Text Flows which is a natural language and text mining programming platform based on web. These types of processing platforms support the execution, sharing, and workflow construction. The processing platform used to enable the visual construction of the text mining workflows ,through the web browser, and the execution was done in the processing cloud of the constructed workflows. [5]

K. Tsugane, et al. implemented GTC-P, which is a nuclear simulation code used in local-view and global view programming models for parallel programming languages in XMP. In the first version, they used the XMP-local view programming model that uses the array communication. It replaces the MPI point-to-point communication except MPI All reduce. The second version uses the XMP-hybrid view that distributes the calculation domain and directs the global view programming mode and coarray communication in local-view programming for particle motion. The results showed that XMP-localview implementation has the same performance as MPI and the performance of XMP-hybrid view is degraded by 20%. The high productivity was obtained from the XMP implementation [6]

E. G. Pinho, et al. Described object-oriented programming language, which has the ability to encapsulate the software. The parallel computations cannot encapsulate the individual object that resides in the single address space. Object-Oriented Parallel Programming used for reconciliation of the distributed-memory and object-oriented parallelism. The result showed that implementation, design, and the performance of Pob C++ prototype. The proposed approach reconciles the style of the object-oriented and parallel programmers with the help of OOP and MPI. [7]

E. Lusk, et al. evaluated the minimal parallel programming model with the help of self-scheduling task parallelism , which is a programming model. In this research, the authors demonstrated how the extremely simple task model used in the asynchronous dynamic load balancing system. Results showed that the scalable implementations are capable of supporting the sophisticated applications of supercomputers. The research also explained the asynchronous dynamic load balancing used in the applications of Green's function Monte Carlo. The results also observed that the extreme scalability can be achieved in minimal programming model without introducing complexity. [8]

C. Hundt, et al. created a platform for the graduate engineers to learn parallel programming challenges and memory access patterns. The author proposed a

software SAUCE (System for Automated Code Evaluation) in which parallel algorithms based on MPI, OpenMPI, C++ are embedded in parallel computing lecture. The prevalent hardware trends towards the parallel algorithm and architecture which creates the growing demands for the students which are familiar with the programming software. [9]

M. Martineau, et al. evaluated various emerging programming models RAJA, OpenMP against the traditional Open CL and CUDA. The authors found tha tthe best performance is achieved with architecture specific modules. From the result, it was concluded that the best performances achieved by the architecture-specific implementations is obtained by the portable performance models which are capable of solving the problems with 5 to 30 percent performance penalty. [10]

E. Calore, et al. successfully ported, benchmarked, and tested the whole multi-node LB code with the help of Open ACC and then characterized its performance with the help of the accurate timing model. In this research, the authors also addressed precisely the issue with the help of test-bench , which is a massively parallel lattice Boltzmann algorithm. For this, they described the optimization and implementation of the multimode with the help of MPI and OpenACC. After that, the codeis benchmarked on different processors which include GPUs and CUDA provided the accurate performance than another GPU implementation. [11]

K. Tsugane, et al. implemented the two versions of GTC-P which are the large-scale nuclear fusion simulation code with the help of the local-view programming and global view models in XMP for parallel programming languages. The research also evaluated their productivity and performance. In the first version of GTC-P, XMP-local view is used for the coarray communication in the local-view programming model that replaces the MPI point-to-point communication in which it is excepted for the collective communication like MPI All reduce. In the second version, XMP-hybrid view used the distribution of the calculation domain , which reflects the directive in the programming model of aglobal view and coarray communication. Both are used for the particle motion in the local-view programming model Experiment evaluation , which shows the implementation of the XMP-local view same as MPI. Results showed that XMP-hybrid view implementation minimized the performance by 5 to 25%. [12]

By X. Liao, et al. provided the overview of the MilkyWay-2 project and also described the design of software and hardware system of theMilkyWay-2 project. MilkyWay-2 (Tianhe-2) are the super computers which are crowned as the fastest super computer in the world and on the 41st in top 500 lists. This architecture employs the proprietary interconnection chips in order to support the massively parallel message-passing communications, efficient software stacks, core processor for computing scientifically, intelligent system administration and emerging programming model for the heterogeneous systems. The research also performed the extensive evaluation with the help of the wide-ranging applications such as Graph 500 benchmarks and LINPACK for the massively parallel software deployed in the system. [13]

By H. Murai and M. Satoimplemented the three approaches for the stencil communication used in the Omni XMP complier. The first method is related to the derived data type messaging which is general and simple. The second methods is related to the packing or unpacking and has the advantage of multi

threading in the multicore environment. The third method is an experiment and related to the extended interface of RDMA of the K computers which are capable of achieving the higher performance. The results showed that the first and second method are effective in various conditions and selecting at the runtime and the third method is promising, but it creates the problem in the higher performance. [14]

By M. Nakao, et al. introduced the XcalableACC (XACC) which is a programming model and hybrid model of the XcalableMP (XMP) Partitioned Global Address Space (PGAS) language and OpenACC. This research also evaluated the performance and productivity of the XACC with the help of the implementations of the HIMENO benchmark. Results showed that the improvement of XACC which requires the less than half source lines of the code as acomparison to the Open ACC and Message Passing Interface (MPI) which were used together as the programming model. It was found that XACC achieved the 2.7 times faster performance as compared to the combination of MPI and OpenACC programming model. [15]

K. Z. Ibrahim, et al. described a coupled-cluster technique, which provided the accurate models of molecular structure using explicit numerical calculations of tensors that represent the correlations between the electrons. The proposed algorithm was irregular and parallelization achieved with the help of specialized data decomposition and dynamic scheduling. Liberation framework is extended in the distributed memory environment in energy-efficient and scalable manner and get 240 x speeds up than the optimized memory implementation of Libtensor. [16]

IV.PROPOSED MODEL

The proposed model in this research is named as Asynchronous Dynamic Load Balancing (ADLB). In this model, every worker communicates with the server using put and get.The number of the servers are specified when ADLB is initialised.The percentage of the MPI process that becomes server varies with the application is about 10% of the total numberof MPI process.The figure below shows the basic structure of ADLB.

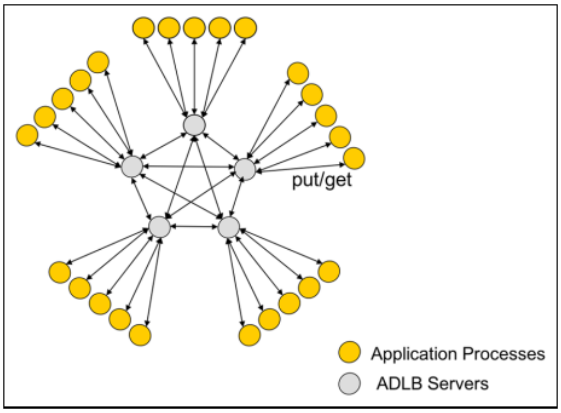

Figure 2:An implementation of ADLB

In above figure, application process goes to their assigned server and carried out MPI. If there is no room on the server then work is distributed among the servers. The number and types of work units stored on each server are maintained in a ‘status vector.'The status vector also records processes in waiting of work and of which type, so that new task can be routed to the application process requesting it. The application process may be multithreading and communicate between them using MPI. In this system, ADLB is using MPI as avehicle for library implementation rather than as an end-user programming system. Servers used MPI that sends and receives by which they maintain a queue of active MPI request objects.

V.METHODOLOGY

The parallel programming involves how the processes are executed parallel and access the data for read or write operation. In the designed program, a thread is initialized when the buttonis pressed, and that will go through a process of reading or writing operation.

There are four different ways for parallel programming:

1.Parallel programming (auto) – used to read or write buffer from any location automatically.

2.Parallel Programming (manual) – specify the inputs location to read or write manually.

3.Read or Write Parallel Programming (auto) – In this part, a thread performs a single operation , which is selected by user and access or write data automatically to any selected location.

4.Read or Write Message Parallel Programming – In this part, a thread writes a message to a buffer which can be read from any thread and the thread specifies read or write input by the user. The message is necessary for a write operation performed by the thread.

Each of them has its unique features and perform their operations with multiple threads, and message passing using read or write operations.

- A thread can read or write the data to any location or any length of data.
- A single thread in auto parallel programming can do five operations which involve maximum 3 write operations.
- If one thread is performing a write operation, then another thread has to wait for the buffer to release from another.
- Many threads can access the data from buffer any time in parallel even after it is writing by another thread.
- But more than one threads cannot allow writing the data at a single time.

VI.RESULTS

The following is the results of the designed programme:

Main Window

The figure below shows the user main window interface to call another window. In this window, we have four buttons which are used to go to the next window for parallel programming. The four buttons will enable the user to select one of the four parallel programming options.

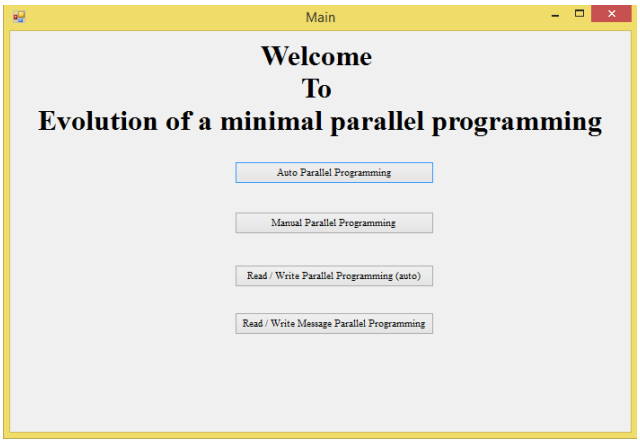

Figure 3:Main Window

Auto Parallel Programming

It is used for creating threads by pressing the start button and then go for further operation of read-write access. It allows the system to automatically do all

the operations of buffer memory location, and decide how much the data is to read or write.

All the process details are shown on process window, and log panel describes the log details of accessing the data. Buffer panel is showing the buffer details of location that a location has how much data. The figure below shows the process window of AutoParallel Programming.

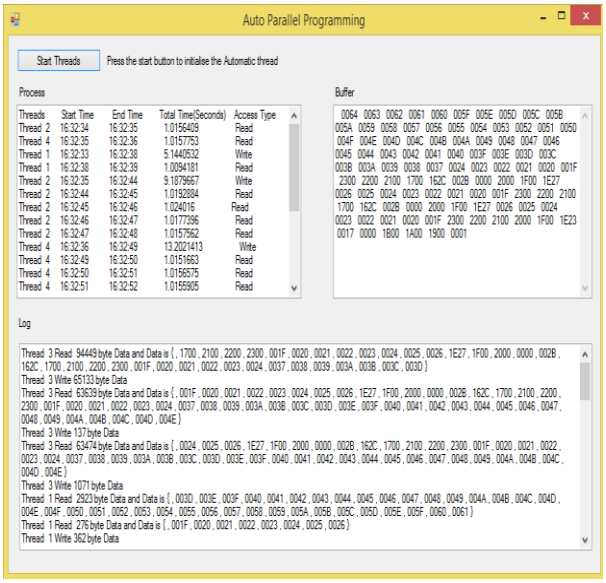

Figure 4:Process window of Auto Parallel Programming

Manual Parallel Programming

It is used to input the user defined value for a start, and end location to read the buffer by a thread and write the location in the buffer and how much locations are written to buffer. In this step, all the inputs are manually entered by the user to perform the manual parallel programming. After that, it will show the result by user inputs, and the result will be displayed as in Auto Parallel Programming,but they would be based on the user's choice

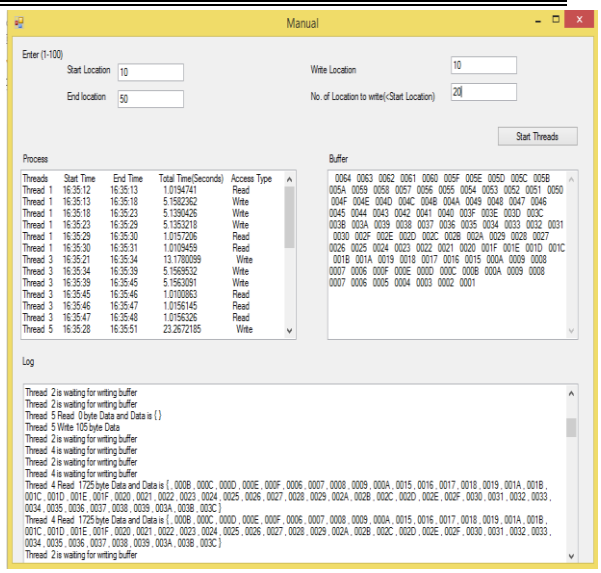

Figure 5:Process window of Manual Parallel Programming

Read / Write parallel programming

In this step, the user decides the thread to read or write the buffer then it will automatically decide the location to read or write the data. It will automatically use all the values like how much data is read and written. In this step, after pressing the start button, a single thread will be initialized which will work for a single operation, and it can be of any type, i.e. read or write.

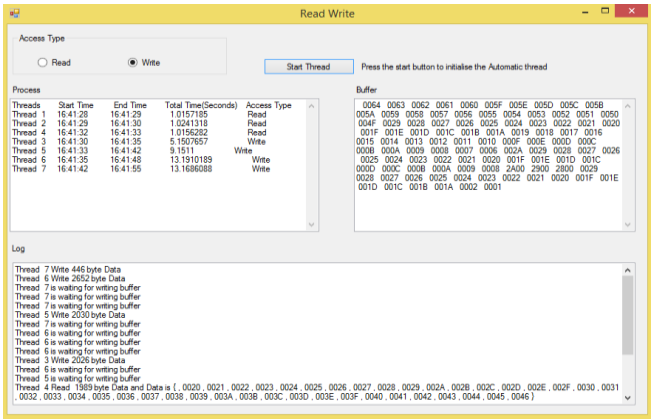

Figure 6:Process window of Read/Write Parallel Programming

Read / Write Message Parallel Programming

In this step, the user decides the message to store in the buffer which further can be read by any thread. If the threadis of read type, then it will read the buffer which is changed by the previous write type thread and by which a thread communication will be setup in parallel programming.A single thread can perform the single operation of read or write, but in this step, the message is of user's choice that the user can enter it and program store message in the form of bytes in the buffer to the locations.

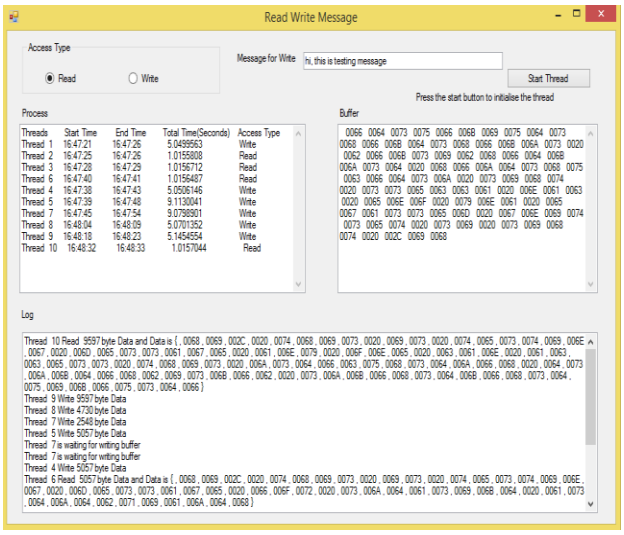

Figure 7:Process window of Read/Write Message Parallel Programming

VII.CONCLUSION AND FUTURE SCOPE

High-performance parallel computing is accompanied by splitting up complex and large tasks among multiple processors. The main application of parallel machines can be found in scientific and engineering computing, where high-performance computing systems are employed today thousands of computer cores . Many devices today provide CPUs that contain multiple cores in the CPU. In each case, the potential of these large devices can be realized through the process of parallel programming. Some of the problems require large working memory , which cannot be available on a single machine.This research provided an insight into the use of parallel programming and its different models to solve such problems. The findings of this study can help in future to design new parallel programming architecture with high productivity and performance.

REFERENCES

[1] Karlin, A. Bhatele, J. Keasler, B. L. Chamberlain, J. Cohen, Z. Devito, R. Haque, D. Laney, E. Luke, F. Wang, D. Richards, M. Schulz, and C. H. Still, "Exploring Traditional and Emerging Parallel Programming Models Using a Proxy Application," 2013 IEEE 27th International Symposium on Parallel and Distributed Processing, 2013.

[2] M. Vanneschi, "The programming model of ASSIST, an environment for parallel and distributed portable applications," Parallel Computing, vol. 28, no. 12, pp. 1709–1732, 2002.

[3] J. Mcguiness and C. Egan, "A Domain-Specific Embedded Language for Programming Parallel Architectures," 2013 12th International Symposium on Distributed Computing and Applications to Business, Engineering & Science, 2013.

[4] A. C. A. C. Chozas, S. Memeti, and S. Pllana, "Using Cognitive Computing for Learning Parallel Programming: An IBM Watson Solution," Procedia Computer Science, vol. 108, pp. 2121–2130, 2017.

[5] M. Perovšek, J. Kranjc, T. Erjavec, B. Cestnik, and N. Lavrač, "TextFlows: A visual programming platform for text mining and natural language processing," Science of Computer Programming, vol. 121, pp. 128–152, 2016.

[6] K. Tsugane, T. Boku, H. Murai, M. Sato, W. Tang, and B. Wang, "Hybrid-view programming of

nuclear fusion simulation code in the PGAS parallel programming language XcalableMP," Parallel Computing, vol. 57, pp. 37–51, 2016.

[7] E. G. Pinho and F. H. D. Carvalho, "An object-oriented parallel programming language for distributed-memory parallel computing platforms," Science of Computer Programming, vol. 80, pp. 65–90, 2014.

[8]E. Lusk, R. Butler, and S. C. Pieper, "Evolution of a minimal parallel programming model," The International Journal of High-Performance Computing Applications, p. 109434201770344, 2017.

[9] C. Hundt, M. Schlarb, and B. Schmidt, "SAUCE: A web application for interactive teaching and learning of parallel programming," Journal of Parallel and Distributed Computing, vol. 105, pp. 163–173, 2017.

[10] M. Martineau, S. Mcintosh-Smith, and W. Gaudin, "Assessing the performance portability of modern parallel programming models using TeaLeaf," Concurrency and Computation: Practice and Experience, 2017.

[11] E. Calore, A. Gabbana, J. Kraus, S. F. Schifano, and R. Tripiccione, "Performance and portability of accelerated lattice Boltzmann applications with OpenACC," Concurrency and Computation: Practice and Experience, vol. 28, no. 12, pp. 3485–3502, 2016.

[12] K. Tsugane, T. Boku, H. Murai, M. Sato, W. Tang, and B. Wang, "Hybrid-view programming of nuclear fusion simulation code in the PGAS parallel programming language XcalableMP," Parallel Computing, vol. 57, pp. 37–51, 2016.

[13] X. Liao, L. Xiao, C. Yang, and Y. Lu, "MilkyWay-2 supercomputer: system and application," Frontiers of Computer Science, vol. 8, no. 3, pp. 345–356, 2014.

[14] H. Murai and M. Sato, "An Efficient Implementation of Stencil Communication for the XcalableMP PGAS Parallel Programming Language," 7th International Conference on PGAS Programming Models, pp. 1–15, 2013.

[15] M. Nakao, H. Murai, T. Shimosaka, A. Tabuchi, T. Hanawa, Y. Kodama, T. Boku, and M. Sato, "XcalableACC: Extension of XcalableMP PGAS Language Using OpenACC for Accelerator Clusters," 2014 First Workshop on Accelerator Programming using Directives, 2014.

[16] K. Z. Ibrahim, E. Epifanovsky, S. W. Williams, and A. I. Krylov, "Cross-scale Efficient Tensor Contractions for Coupled Cluster Computations Through Multiple Programming Model Backends," 2016.