An Evolutionary Technical & Conceptual Review on High Performance Computing Systems

Nilayam Kumar Kamila
Research Scholar, CSE Dept.
Shri Venkateshwara University,
Gajraula, India

Subhendu Kumar Pani
Principal
KCA, Biju Patnaik University of Technology
Bhubaneswar, India

Pawan Kumar Bharti
Hon'ble Vice Chancellor
Shri Venkateshwara University,
Gajraula, India

Sarbananda Sahoo
Professor & Dean(School of Commerce & Management)
Shri Venkateshwara University,
Gajraula, India

***Abstract*— Computing and Communications are two major distinct components in the field of Computer Science and Engineering. These two components with a vast technological improvement inherently affect the economic and financial development of the country. Many engineers and researchers made multiple successful attempts to make the communication faster. At the same time there are many major developments and progress are being made in the high-performance computing field. The evolutionary concept of High-Performance computing is yet a very long journey, this improvisation based on different parts of the computer and processing units. High Performance evolution still needs most researchers' complete attention so as to contribute for its advancements to solve the high complexity of today's world engineering and research application. In this article we mainly focus on the overview of the progressive work on high performance computation. This article mainly discusses the historical and evolutionary improvements on this topic, ultimately will be helpful to solve and attempt most advanced high-performance systems and extreme complex science and engineering research problems in the high-performance research fields.**

Keywords—high performance, HPC Survey, computation evolutions, HPC Importance

I. INTRODUCTION

The technological devolvement along with socio-economic development is completely based on the computing performance and hence the overall development of the society, community and the ultimately the nation. With the evolution of computer science and engineering technology, the performance of the systems is improved for the single node system in old age systems and then now a days its improved for the multi-cluster systems platforms. In single node processors the high-performance concepts applied onto different component of the system e.g., the speed of the processor, the I/O processing speed, instruction processors pipeline executions etc. A single node system could process up to a maximum of 10¬15 FLOPS (Floating Points Operations per Seconds). This opened a wide door towards the multi-core processors evolution in the field of the high-performance computing technology. The multi-processor system depends on a shared memory system and could operate upto a maximum operation of 40X10¬15 FLOPS.

The rise of Super Computers makes a major and significant difference in this high-performance computing field. The performance speed is 100 quadrillion FLOPS [2]. After Super Computer evolution, then several successful attempts made for the computer clustering concepts [3]. In a computing cluster, multiple multi-core systems configured to share the load of the processing and IO operations. So, the load to do similar or homogeneous tasks are distributed among the different nodes and the output is resulted. This is much useful in many commercial and legacy applications where multiple requests (e.g. web requests) earlier used to be computed on single machines, now capable to be processed on multiple nodes and the result(response) is provided to the requesting systems.

Subsequently the Distributed Computing, Parallel Computing and Cloud Computing come as major contributing factors for high performance factors. In distributed computing, same task is processed in different geographically located systems (single node or cluster nodes). Usually cluster nodes are configured in each geographical location and connected over the network with appropriate network topology. Parallel computing is the part of the computation where different tasks are being processed in different processors and the result is gathered from different processors. The integrated combined results are provided as the output. This computing technique is very much helpful for the heterogenous tasks which executed in parallel to ensure the fastness of the processing.

The organization of this article is as follows. In next section, we'll overview each field of HPC and subsequently focus on related advancement of research work. Following sections, we'll go over the recent technological advancements of the HPC. We'll conclude this article with our conclusion.

II. THE IMPORTANCE & INTERNATIONAL FOCUS

In this section, as reference [12], the different nation's HPC initiatives and the investment is shown.

TABLE I. HPC INITIATIVES AND NATIONS INVESTMENTS

Country	HPC Initiatives	M$/Yr
US	NSCI	320
China	13th 5-Year Dev Plan [DMES]	200
Japan	Flagship 2020 Plan	200
Eur Union	ExaNeSt; PRACE; ETP4HPC	1100
India	National Super Computing Mission	140
South Korea	National Super Computing Act	20
Russia	HPC Focus Modernization Program	N/A

This provides the significance of the high-performance computing research in the field of computer science and how the leading nations are much thoughtful and are ensuring the high investments to enhance the computing accessibility and faster communications in world's geography.

III. Overview & History of High-Performance Computing

A. Overview

The concept of high-performance computing refers to the achievements of the improved performance over the existing system by upgrading the design, flow or integrating additional computational units. The central idea of high-performance computing is to accomplish the objective to process high load of data and information with a greater speed to provide the processed result in less amount of time. This idea is always an ever-changing concept to acquire the high-speed performance over current systems or current platforms. In a nutshell, the concept of high-performance computing is the concept of continuous evolution and improvements of itself.

B. History & Background

There is always a competition that the high-speed could always be archived through the modification of the underlying hardware units and the network topology Vs the improvements of the system software (operating systems and related other usable software) and the application software. However, in reality, all the components on high level contributes towards the high-performance computation. In the computer science world union and integration of multiple hardware units along with the installation of supporting system and application software establish the way for solving the growing computational need. Research data shows that there are mainly two models of task execution e.g., Single Instruction Multiple Data (SIMD) and Multiple Instruction Multiple Data (MIMD).

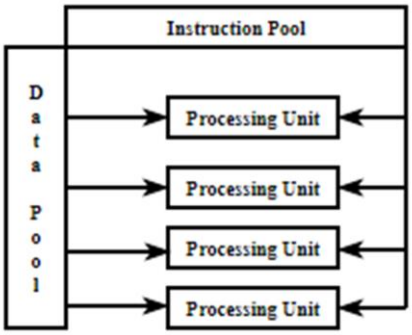

Fig. 1. SIMD Model

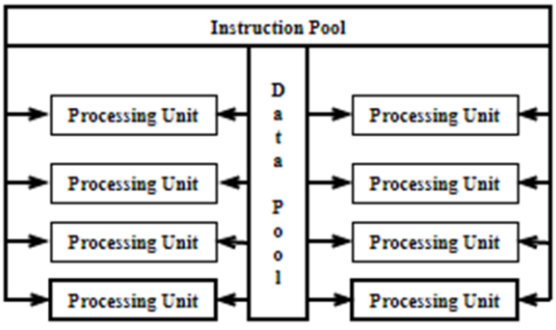

Fig. 2. MIMD Model

The High-Performance computing concept has started its initial evolution in 1950. At the time, only mainframe is the only commercial computing mechanism and serve the handful leading organizations. Billing was considered to be the main task for almost all business organizations and at most a batch process might require the billing consolidation or day end process.

TABLE II. Difference SIMD Vs MIMD

SIMD	MIMD
Defined as Single Instruction Multiple Data	Defined as Multiple Instruction Multiple Data
Less Memory Required	Large Memory Required
Less Expensive	More Expensive than SIMD
Based on Single Decoder	Based on Multi Decoder
Synchronous Programming	Asynchronous Programming
Efficient than Single Data Processing Model	Efficient than SIMD
Defined as Single Instruction Multiple Data	Defined as Multiple Instruction Multiple Data

The jobs are sequentially processed one after another as there was no concept of parallel execution mechanism. There was no interaction required and multiple jobs were sequentially programmed which does not require manual human interaction. JCL (Job Control Language) was the language mainly used for the sequence batch job programming.

In the era of Personal Computing, there was a requirement of incremented performance which came in the year of 1970s. In personal computing, the computer speed had to match the moderately faster computing expectations. Inventions of vector computing in the field of super computers CRAY-1 in 1976, was a major achievement in the history of HPC. This computer systems use Reduced Instruction Set (RISC) processors and vector-based registers fulfills the computing need till the period of late 80s. IBM with RISC microprocessors used the butterfly interconnection network allows the programmers and engineers to integrate shared

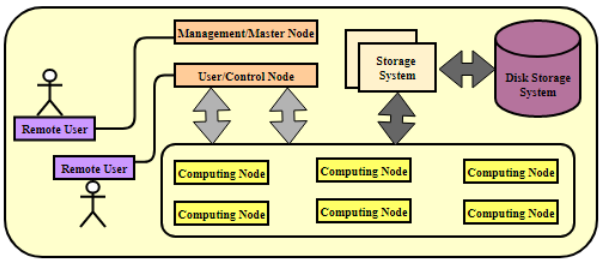

Fig. 3. Computing Node Clustering in HPC

memory caching systems with data storage units and processing units. In beginning of 90s, Dual Access Storage Handling (DASH) was designed by Stanford University. Distributed Memory Cache consistency was archived with multiple data directory structures. This widens ideation and research for innovative architectures in shared caches and landed onto major processors' architecture developments. More and more commercial microprocessors and interconnections with data storages using several layers of caches are one of the major achievements at that period. This concept of interconnectivity between the different computational units leads to the idea of interconnection between the computers, resulting the concept of clustering.

IV. Literature Survey – HPC Evolutions

We listed the historical progress of the High Performance Computing below. The traditional approach for the evolution is based on the specific server systems and scoped to the clustered platforms. While few authors attempted and

successful in the performance improvements of single machines and its underlying architecture, the other authors mainly focused onto the multi-systems and cluster platform's performance.

TABLE III. HISTORICAL PROGRESS SURVEY AND ANALYSIS OF HIGH PERFORMANCE COMPUTING

Author, Year	Study Abstract
Jones et. Al, 2011	Single Machines Performance limitation of 10^{15} FLOPS
Gerber et. Al, 2012	Multiple independent units construct the HPC System. A control node/unit then capable to establish the distributed working load for the HPC.
E-sciencecity. Org, 2015	Processing & Data Storage with memory caching & sharing system. Parallel
Buyya et. al, 1999	Discuss about cluster management, deployment, interconnection and its monitoring
Kalcher et. al, 2007	Control node failure and how it fails the entire cluster systems. Discuss about the blade servers, simple installation and low power consumptions
Kohlmeyer et. al, 2010	Introduction of Load-Reduced Dual In-line Memory Module (LRDIMM) and reduce the load on server bus and low power consumption
Jacob et. al, 2015	Multi Stage Pipeline run to n-times fasten the process.
Barragy, 2007	HPC cluster includes Computing Node, head node, Gigabit Ethernet, Global Storage Systems
TOP500.org, 2015	Computing Performance up to 308.9 PFLOPS in 2014 as compared to 1.1 TFLOPS 1993
Barney, 2015	Time Parallelism and Space Parallelism with concurrent competing by multiple processors

The figure shown is a classic organization of different components which contributes towards the cluster computing. In this cluster-based model, multiple computing nodes are closely working together with the shared storage systems and with a common disk storage system. There is a user control node which manages all users; their user request and submit the user requests as a job to be processed by multiple computer node. The master node basically a master control node which controls all the computing nodes and other different computing units incase of any issues, errors or fault occurs during the system process. This concept is a very widely used model in the era of ~2000 and till date the base multi node concept is still in use in today's advanced clustering concept.

V. HIGH PERFORMANCE COMPUTING – MAIN STREAM COMPONENTS

We gathered few of the main study of the high-performance computing. In 2020 Stephen J Ezell and et. al provide a detailed study on the importance of high-performance computing and how every nation is more focused on the high-performance computing research. The importance of the programming languages, the parallel computing concept advancement was deeply discussed by H.J Sips et. al. The library design, performance tuning and compatibility with different processor and computing components and it's adequate designing for tomorrow's high-performance computing is well analyzed and massive development activities are in progress by DK Panda et. al. We also refer the various base conceptual views and it's historical relevance in today's world high performance computing is discussed in the book written by A.D. Kshemkalyani et. al. In 2012, M. Pharr discussed about a language feature which provides a 35X high performance on Intel R® SPMD Program Compiler (ispc).

TABLE IV. RECENT PROGRESS SURVEY ON HPC

Author, Year	Study Abstract
Stephen J. Ezell and Robert D. Atkinson, 2020	Importance of High-Performance Computing
H.J. Sips, 2009	Programming Languages and the Parallel Programming
DK Panda et. al, 2021	High-Performance MPI Library and Designing of MPI Library for AMD GPUs
A.D. Kshemkalyani, M. Singhal, 2015	Discussed on the Conceptual view of Distributed Computing Principles, Algorithms, and Systems
Li Luxingzi, 2015	High Performance Computing Applied to Cloud Computing
Andrews, Gregory R, 2000	Discuss on foundation of Multithreaded, Parallel, and Distributed Programming
M. Pharr and W. R. Mark, 2012	Intel R® SPMD Program Compiler (ispc) Language Feature with 35x speedups on a 4-core system for high-speed performance

We are now proceeding to briefly discuss on different computing model and units which contributed a lot of performance scales towards today's world high-performance computing model.

A. *Instruction-level parallelism*

Instruction level parallelism is a very thoughtful concept provided by Seymour Cray in late 1970, and later this concept was headed by Cray Research. Initially this Instruction level Parallelism was developed for the Super Computers using the pipelining for multiply and add/subtract module. Later this was extended by Roger Chen, James Bradley and By Mid of 1980s, almost multiple companies implemented this instruction level parallelism from different parts of the world.

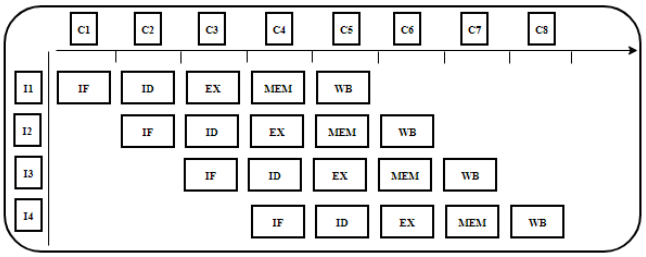

Fig. 4. Instruction Level Parallelism

As shown in the figure, there are different instruction commands e.g. Instruction Fetch (IF), Instruction Decode (ID), Instruction Execution (EX), Memory Operation (MEM) and Write Back (WB). Before the introduction of instruction level parallelism, the instructions were executed sequentially i.e. the second instruction I2's Instruction Fetch Cycle (IF) operation was starting at Cycle 6 ticks i.e. at C6 and continues to end at C10 clock tick. However, with Instruction Parallelism, it starts at clock tick C2 and ends at C6 cycle tick.

This saves 4 clock ticks than the sequential mechanism of instruction pipeline execution.

B. The storage hierarchy & IO Processing

The Storage hierarchy plays a major role in the field of high-performance computing. The major discovery of the structural alignments of storage hierarchy is to optimize the storage capacity vs the access latency.

The more storage capacity unit, the more access latency time. Therefore, the early researchers and designers has

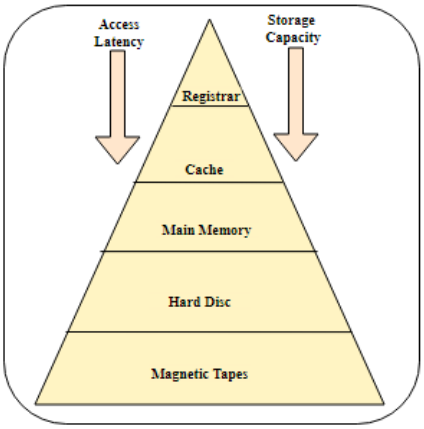

Fig. 6. Access Latency and Capacity for Storage Hierarchy

implemented a system which is based on the faster access unit closer to the CPU.

That in turn means, the high-capacity devices are accessed at the end if the data and/or instructions not found in the faster access units e.g., cache or main memory. Cache is also stacked into different levels e.g., level-1 cache, level-2 cache, so to avoid the cache miss situation during the data/instruction search by the CPU in the cache. If the data could not detect in the cache, the main memory is fetched with the mechanism direct mapping, associative mapping and direct associative mapping techniques.

The figure shows for the Access latency and storage capacity. Registrar which is the closest variable for the Processor has the least capacity with faster access response time(low latency time). Similarly Caches (often used as level 1 cache and level 2 cache) has the faster access time and less capacity than main memory.

Main memory component is the unit where all the instruction code and data to be made available before any program execution. Main memory has the low latency and with moderate storage capacity but could not hold the data as like the hard disc or magnetic tapes. Hard disk or Magnetic tapes has the high latency time and has the high storage capacity.

C. High Performance Compilers & Program Parallelism

There is a significant contribution of the compilers and program parallelism [13] towards the high-performance computation. Program parallelism was not achieved till the year 1990s and used the following steps for the evaluation and examination.

- Evaluation of compiler transformation ability
- Examine the output of compiler execution
- Performance measurement against another compiler
- Comparison of fully optimized compiler performance
- Comparison of compiler parallel execution performance

The parallelism in compilers execution ability is extremely powerful for achieving the parallel execution of program

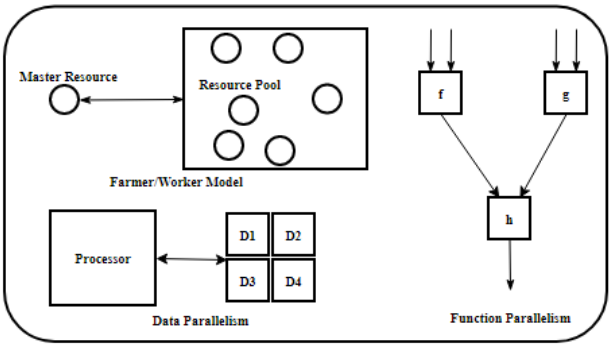

Fig. 7. Task/Program Level Parallelism Model – Farmer/Worker Model, Data Parallelism, and Function Parallelism

instruction and hence a true contribution towards the high-performance computation.

In real-time high-performance execution there are a lot of exercise is performed in planning a designing of the application. The concurrent program unit execution, data results integration is one of the major challenges for the program installation in scope. The underlying programming model must be suitable to adopt the different models' parallelism view. In general, there are various models are considered e.g., farmers workers model, data parallelism and functional parallelism mode.

Farmers Worker Model: In framer worker model, the main program creates multiple similar tasks. Those small tasks will be executed by different working threads and then the small task execution results will be produced. When the smaller task execution completed, it'll request the parent process to get assigned more task to be processed. The farmer worker model is best suitable to produce consistent and constant statistics, just to ensure each of the worker component evenly loaded and executed in a constant manner.

Data comparisons: In a computational data model for calculation, the division of data frames is a great force for finding similarities. The developer, using this model, begins by classifying all the appropriate data structures and then assigns each structure to a processor (or process). The following calculation follows this division of the allocation, meaning that the allocation of the data structure is done in the processor (or process) in which the item is allocated (e.g., this allocation includes the right-hand count for this allocation statement.

Functional Parallelism: In certain cases, or parallelism model, there is a scope to make a task to be split into two or more number of dissimilar subtasks and each subtask executed by multiple number of processors to achieve the parallelism. This model most confirms to an acyclic process, and the respective continuous flow of the task is divided into the respective number of subtasks and is assigned to the sub module responsible unit to get executed. All the dissimilar tasks executed results then again collected through a common module as shown in the figure. The integrated results then provided as the output of the program structure. This model

most often referred as task parallelism. However, an eviction clash formalism is essential for some of the specific demand driven search operation so as to proceed for the next operation execution.

D. Shared Memory Parallelism

In initial days of early 1970, the designers mostly focus to inject a greater number of processors to achieve the hpc.

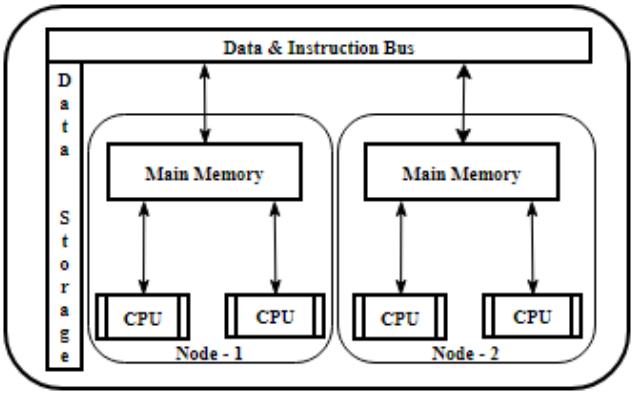

Fig. 8. A General Construct of Shared Memory Parallelism

In the shared memory parallelism, the data and instructions are fetched through the Data & Instruction Bus which is interconnected to the main memory and the data storage. Multiple processors are connected to the main memory to fetch the instructions and data for the parallel processing.

E. Distributed Parallelism

Andrews, Gregory R in his study found that the OS Architectures explored the message-passing technique in 1960s. The predecessor of internet (APRANET) invented and installed the e-mail which was considered the first distributed system in the history of computer science.

Later in 1980s, with multiple conferences and symposiums (e.g., Principles of Distributed Computing (PODC), 1982 and International Symposium on Distributed Computing (DISC), 1985), the distributed computing was mainly focused and the advancements on the programmatical, platform level was carried through.

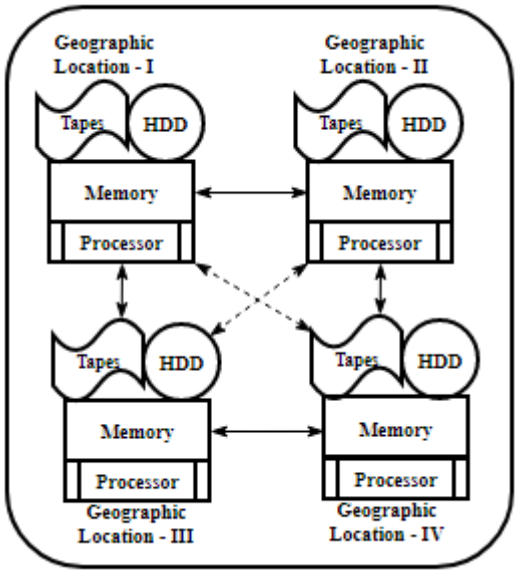

Fig. 9: Distributed Computing in different Geographical location

Distributed computing is based on the message passing techniques. Most of the computing task depends on the data fetch from the remote sites. While it could be accomplished by retrieving the data from the remote location and process at an centralized location, many architectures inclined to process the data at the site where the data is available and then get the processed results. For example, as shown in figures, there are 4 different geographical locations, and hence the complete computing nodes available in different geographical locations and the data is available for patterning to the respective geographical location.

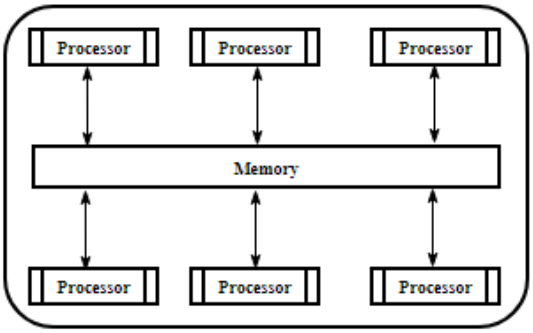

Fig. 10: Parallel Computing with Single Memory Unit with Multi-Processor Access

If any data is required then the cross-region data communication could be established and the relevant data could be retrieved. This model serves to it's best capacity to ensure to serve maximum customers at their local site including the rest of the needed clients/roaming clients at a different geographical location through cross region communication.

TABLE V. DIFFERENCE IN PARALLEL COMPUTING VS DISTRIBUTED COMPUTING

SIMD	MIMD
Multiple Operations Execution	Independent Computing Systems in different geographical location
Single Computing Unit	Multiple Computing Unit
Multi Processors with Shared/Distributed Memory	Multiple Computers with Distributed Memories
Internal Service & Data Bus for intra communication withing Single Computing multi-processor unit	Message passing based communication
Improved performance than Single Processor System	High Scalability and fault tolerance. Effective resource sharing.
Multiple Operations Execution	Independent Computing Systems in different geographical location
Single Computing Unit	Multiple Computing Unit

In Todays' most advance cloud computing model, this concept of distributed computed is heavily implemented and the study shows that most of the in- region traffic served with very faster access mode and with cross region access it's able access the cross region with the acceptable latency and the bottleneck in centralized system model is greatly reduced.

F. Scientific libraries

Scientific and Engineering libraries has a great impact onto the high-performance computing model. These libraries are mostly used to make application programming compatible

with parallel computing environment, distributed computing environment etc. Recent study and developments show there is some more development on implementational level and designing level research in in progress in the field of MPI libraries.

In the figure 11 as shown, the program initialization starts at the initialize module through the program library, it then creates the works process. The library then responsible to send the data to different parallel functional module to get executed on different data as shown.

The master node again collects the data from the worker module and processes the integrated results to produce the final results. The entire process execution is mainly supported

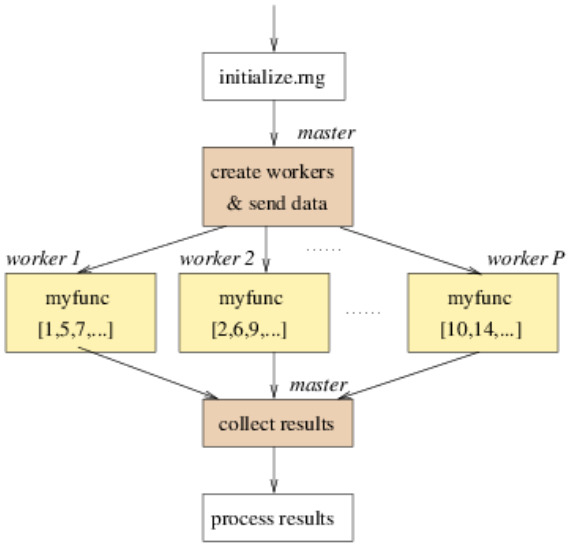

Fig.11: Library support for parallelism for high performance computing

by the underlying programmed library which helps to achieve the high-performance computing.

VI. CONCLUSION

In this article we study the historical details and evolution of high-performance computing. We saw the differences in concepts, the design concepts usability and the implementational theory behind the proposed structure to achieve the high-performance computation. Many models have been adopted in different level of computational layers e.g., instruction level parallelism, program level parallelism, computer clustering concept to overcome the single unit super computers, compilers support, program libraries contribution towards high-performance computation. The evolution of high-performance computation is a continuous improvisation process and today's world's significant development achievements are the outcome of the historical progress discussed and reviewed in this article.

As a future scope, these studies refresh the high-performance historical design models and it helps to refresh few design improvements in today's new concept on machine and deep learnings. As a immediate scope, we will discuss the advancements of the cloud models for the high-performance computation as part of our future scope of the work.

ACKNOWLEDGMENT

This paper is a dedicated support and extensive motivation from the members of The Science and Engineering Research Council (TSERC).

REFERENCES

[1] J. Lindström, A. Hermanson, F. Blomstedt and P. Kyösti Multi-Usable Cloud Service Platform: A Case Study on Improved Development Pace and Efficiency in SCi Applied Sciences vol 7(316) pp 1-14, 2018

[2] S. Sridharan et al.(Intel Corporation) On Scale-out Deep Learning Training for Cloud and HPC, SysML Conference, pp 1-3 Jan 2018

[3] L.F. Añover The High Performance Computing strategy and the European Cloud Initiative, HPC User Forum, Horizon 2020, September, 2017

[4] A. Tataru, Metrics for Evaluating Machine Learning Cloud Services, JonKoping University School of Engg. Thesis Report, October 2017

[5] Steffen Rendle et. al (Google Inc.) Robust Large-Scale Machine Learning in the Cloud ACM SIGKDD pp. 1125-1134 August, 2016

[6] M. Parekh, B. Saleena, Designing a Cloud based Framework for HealthCare System and applying Clustering techniques for Region Wise Diagnosis. in Elsevier ScienceDirect Procedia Computer Science Vol 50 pp. 537 – 542, 2015

[7] J. L. Hennessy and D. A. Patterson, Computer Architecture, A Quantitative Approach, Morgan Kaufmann - Elsevier, San Mateo, CA, 5th edition, 2012.

[8] G. Hager and G. Wellein, Introduction to High Performance Computing for Scientists and Engineers, CRC Press - Taylor & Francis Group, 1st edition, 2011.

[9] R. Gerber. The Software Optimization Cookbook: High-performance Recipesfor the Intel Architecture. Intel Press, United States, 2002.

[10] S. Goedecker and A. Hoisie. Performance Optimization of Numerically Intensive Codes. Society for Industrial and Applied Mathematics, Philadelphia PA, 2001.

[11] W. Triebel, J. Bissell, and R. Booth. Programming Itanium-based Systems. Intel Press, United States, 2001.

[12] Stephen J. Ezell and Robert D. Atkinson; "The Vital Importance of High-Performance Computing to U.S. Competitiveness; in INFORMATION TECHNOLOGY & INNOVATION FOUNDATION April 2016 Page 1-57

[13] H.J. Sips, "Programming Languages For High Performance Computers" Delft University of Technology, Delft, the Netherlands https://cds.cern.ch/record/400331/files/p229.pdf, 2009 pp 1-10

[14] DK Panda, H. Subramoni, C. Chu, and M. Bayatpour, The MVAPICH project: Transforming Research into High-Performance MPI Library for HPC Community , Journal of Computational Science (JOCS), Special Issue on Translational Computer Science , Oct 2020.

[15] S. Sur, S. Potluri, K. Kandalla, H. Subramoni, K. Tomko, and DK Panda, Co-Designing MPI Library and Applications for InfiniBand Clusters , IEEE Computer , Nov 2011.

[16] K. Khorassani, J. Hashmi, C. Chu, C. Chen, H. Subramoni, D. Panda; "Designing a ROCm-aware MPI Library for AMD GPUs: Early Experiences" ISC HIGH PERFORMANCE 2021, Jun 2021.

[17] Andrews, Gregory R. (2000), Foundations of Multithreaded, Parallel, and Distributed Programming, Addison–Wesley, ISBN 978-0-201-35752-3.

[18] M. Pharr and W. R. Mark, "ispc: A SPMD compiler for high-performance CPU programming," 2012 Innovative Parallel Computing (InPar), 2012, pp. 1-13, doi: 10.1109/InPar.2012.6339601.