DOI: http://dx.doi.org/10.26483/ijarcs.v10i2.6397

ISSN No. 0976-5697

Volume 10, No. 2, March-April 2019

International Journal of Advanced Research in Computer Science

REVIEW ARTICLE

Available Online at www.ijarcs.info

IMPROVING PERFORMANCE IN HPC SYSTEM UNDER POWER CONSUMPTIONS LIMITATIONS

Muhammad Usman Ashraf
Department of Computer Science
Government College Women University
Sialkot, Pakistan

Amna Arshad
Department of Computer Science,
Government College Women University
Sialkot, Pakistan

Rabia Aslam
Department of Computer Science,
Government College Women University
Sialkot, Pakistan

Abstract: Today's High-Performance Computing (HPC) systems require significant usage of "supercomputers" and extensiveparallel processing approaches for solving complicated computational tasks at the Petascale level of performance (10^{15} calculations per second). The next breakthrough in the computing revolution is the Exascale level of performance that is 10^{18} calculations per second-a remarkable achievement in computing that will have a fathomless influence towards everyday life. Current supercomputers can't achieve such a high level of performance under power dissipation constraints. Even though the Exascale performance can be achieved by multiplying the number of cores according to Exascale computing system constraints, the challenge of power consumption still persists. However, the primary focus of this study is to analyse how to enhance performance under power consumption limitations for emerging technologies. Leading to objectives, the current study presents a comprehensive analysis of existing strategies that can be considered to enhance performance and reducing power for emerging Exascale computing system. Consequently, we have suggested a massive parallel programming mechanism which is promising to achieve HPC Exascale system goals.

Keywords: HPC, Exascale computing, Massive parallelism, Intra-node communication, Inter-node communication.

I. INTRODUCTION

High-Performance Computing (HPC) practice the computing power in such a way that can provide greater performance than a traditional desktop computer to resolve extensive enigmas in medicine, engineering and commercial enterprise [1], [2]. A traditional computer generally consists of a single processing unit (a Central Processing Unit), whereas an HPC system embodies a community of CPUs where each processor contains multi-cores along with its local memory to execute a variety of complicated tasks and software applications. HPC systems utilize supercomputers and parallel processing techniques to perform extensive jobs. In supercomputing, thousands of processors work in parallel to solve extensive problems. Parallel Computing is a sort of computation in which many calculations or execution of tasks are accomplished concurrently. Massive problems can frequently be divided into smaller ones that could then be solved at the same time to enhance the overall performance of HPC systems.

The detailed example can explain the fundamental concept behind the HPC i.e. a traditional desktop computer takes 200 hours to complete a specific task while it could be completed in 1 hour by utilizing 200 computers at once. In short, a single desktop computer might not be as useful as it could be while utilizing all the resources collectively as a community. Every advancement in HPC system's performance from GigaScale, to Terascale, to Petascale, to Exascale constitute an extraordinary improvement in computing performance. The usage of high-performance computing has to turn out to be globally significant throughout the sectors of education & government and all the areas of enterprise and business. High-Performance Computing provides high-end designing and simulation environment, helps applications under development to deal with marketing delivery challenges by providing the facility to accelerate or even get rid of prototyping and testing phases. And also, for the decision-making, enhancing the quality and for predicting the overall performance and failure rate of the product [5], [14]. Moreover, HPC acts as promoting support for the research and development process in science and technology by providing support to improve existing technologies and to deliver the products to the marketplace more efficaciously and quickly. Similarly, organizations and industries use supercomputers to build and examine their strategies before the actual implementation [6].

Nowadays, the fastest supercomputer at the global level solves complex problems using Petascale systems capable of performing10^{15}(quadrillion) calculations each second. Though these Petascale systems are going well in this era, the next milestone in computing advancement is to pace relatively towards high-performance Exascale systems offering outstanding computing power. These advance and powerful high-performance will reveal many scientific mysteries and will have a fathomless impact on everyday life [1], [2], [4]. Current supercomputers cannot deliver such a high level of computation under the power consumption limitation. Although developers can extend the cores in the current Petascale systems to devise a way towards Exascale

computing system, the challenge of power consumption still persists.

The rest of the paper has been presented in a way that Section II introduces HPC challenges for emerging Exascale computing systems. Section III defines the approaches to improve performance and reduce power consumption in HPC systems. Section IV has carried out a comparative analysis of these approaches. Section V has highlighted discussions and recommendations for the best approach and we have concluded the work in Section VI.

II. CHALLENGES IN EXASCALE COMPUTING

The roadmap towards Exascale computing systems, the United States Department of Energy (DOE) has pointed out some primary constraints taking into account the financial as well as power consumption limitations. These constraints include the power consumption not more than 25 to 30 Mega Watts, system development cost near about to 200 million USD, system time to delivery almost 2020 and integrated multi-cores no more than 100 million [7]. The major challenge in the way towards Exascale is that it does not exist yet and to meet the above-defined barriers current technologies are facing many challenges, from which power consumption is the most influential challenge [8]. These challenges are further elaborated and classified as follows:

A. Power Consumption Management

Exascale systems, which includes heaps of nodes drawing excessive Megawatt power, mandate a want for brand new, system-wide methodologies and strategies for power monitoring, controlling, and scheduling. The power consumption of both individual nodes and the overall system is, therefore, an essential issue to cope with [9]. However, new energy-efficient algorithms and devices are needed to manage power consumption.

B. Programming Models

The emerging Exascale systems can face many challenges as complex jobs make use of billions of threads, so there is a need for novel models to deal with thread management and synchronization overhead. Moreover, to utilize the power well there will be a need to handle all the significant resources for memory intensive operations. Failures can be expected in the novel architecture and therefore error handling can limit the computing performance. There is a need for a functional programming model that takes into account all the issues and make use of those resources that could be managed either at software end or compiler end and have a tremendous impact on the overall performance of the system. Furthermore, we cannot assume whether the emerging Exascale system is of a homogenous or heterogeneous environment. There is a need for such programming models that can support both the homogeneous and heterogeneous frameworks [10].

C. Novel Architectures

While Exascale computing remains a great challenge, it is most probably for incremental advances in current technology to attain performance 50x better than contemporary HPC systems [11]. While conventional computer systems continue to make substantial advances, it is argued that radical new architectures and frameworks might be needed for high-performance computing to attain Exascale-level of performance [12].

D. Massive Parallelism

The one conventional way to enhance the Petascale performance up to ExaFlops is to increase the clock speed of CPU. But shortly the clock speed could be restricted to 1G Hz. An alternative way towards Exascale is to extend the cores in current Petascale systems. But according to Exascale limitations defined by the United State Department of Energy, we can only exceed the number of cores up to 100 million. Moreover, the increase in computing resources (the number of cores) will ultimately consume much power. Therefore, another option in the way towards the Exascale level of performance is to achieve massive parallelism by takingadvantage of modern programming models, accelerated GPGPU devices and many-core processors [23].

E. Resiliency

There is a need for considerably new computing strategies for having the roadmap towards Exascale computing environment. Massive parallelism, delivered by many-core processors will open the way for massive computing with more than 1018 floating point operations per second. A considerable number of practical components (computing cores, memory chips, network interfaces) will extensively increase the possibility of partial disasters, load balancing and reliability issues [4]. Developers can't be intended to continually cope with load balancing and reliability issues. The operating system has to discover an efficient way that offers an effective way for load management and checkpointing while allowing software developers to complete control over the performance of the system.

F. Memory Management Mechanism

The cost of data movement has continually been a concerned subject matter in high-performance computing (HPC) systems. It has now a substantial effect on both power consumption and performance. Locality management has acquired a new urgency in emerging HPC systems providing massive parallelism and complicated memory hierarchy. Data locality abstractions with the goal to increase productivity without sacrificing overall performance are available in the varieties of libraries, data structures, languages and runtime systems. Because of the complex memory hierarchy of HPC systems, developers cannot persist with low-level solutions of data management. Novel memory management mechanisms are required to perform large tasks without compromising performance [13].

III. PERFORMANCE ENHANCEMENT MECHANISMS IN HPC SYSTEMS

The one conventional way to enhance the Petascale performance up to ExaFlops is to increase the clock speed of CPU. But shortly the clock speed could be restricted to 1G Hz. Another way towards Exascale is to extend the cores in current Petascale systems, but the challenge is to achieve all under predefined Exascale computing constraints. This paper illustrates well the various approaches to improve performance using energy efficient models, performance efficient models and approaches to improve intra-node communication that will ultimately result in enhancing performance as discussed below:

A. Performance Efficient Models

Parallelism has played an outstanding role in system performance enhancement. Many single hierarchical models were introduced to parallelize large independent processes for Terascale computing systems. A message passing library specification used for clusters, heterogeneous networks and parallel computers. It is used in all connected nodes to communicate among host CPU processors. It has two processes

master and slave process. The job of the master process is to distribute data to all the connected nodes via the slave processes [15]. It is used for distributed computing applications and provides an efficient and portable way to address parallel programs. Moreover, to distribute and parallelize data at the inter-node level MPI offers coarse-grained parallelism and maintains synchronization via blocking methods [16]. But the major limitation of this model is that MPI designer did not take into account the futureExascale systems that will require novel MPI configurations and runtimes.

OpenMP is a single hierarchical programming model used to parallelize and synchronize the threads by utilizing the shared memory-based architecture. It offers two primary principles: sequential equivalence and incremental parallelism [18], [19]. It provides fine-grained parallelism by parallelizing and distributing the data at the intra-node level via accelerated GPU devices. Synchronization among host CPUs and GPU cores along with GPU computation were improved to run the tasks concurrently [20]. It provides high-performance support for both heterogeneous and homogeneous systems for parallel applications. The primary limitation for this model is that it only supports shared memory architecture on a single node, and doesn't give the support for cluster system [21].

CUDA abbreviated as "Compute Unified Device Architecture" is an efficient hybrid model utilizing accelerated GPUs and threads for massive parallelism [23]. It makes use of the application comprising of a program referred to as "CUDA kernel" that helps out to execute the tasks concurrently on GPU devices. CUDA refers to as the most competent model for thread-level optimization that allows application flexibility. But GPUs supporting CUDA are rendered only by Nvidia and has interoperability with rendering languages such as OpenGL. It provides lesser performance as compared to OpenACC and supports heterogeneous computation where the application uses both the CPU and GPU devices [22].

OpenACC appeared as a high-level programming model that makes use of high-end and supportive directives to achieve parallel computing. It affords better performance than CUDA and enables portability to a broad field of computing architecture. However, it does not provide flexibility, thread management, thread synchronization and optimization for the programs, along with other high-level features available in the CUDA framework [24].

OpenCL refers to as "Open Computing Language" is an efficient parallel programming model for heterogeneous frameworks. OpenCL supports run-time compilation that excludes dependencies on instruction sets, allowing hardware providers to make remarkable changes to instruction sets, drivers, and supporting libraries. It grants portability and compatibility of kernels across multiple hardware and platforms [25], [26]. But the restriction is that OpenCL demands a complicated setup which includes preparation of settings, command queues, in addition to a compilation of kernel codes [26].

Later on, dual-hierarchical models were introduced to improve the performance for Petascale computing systems. In the hybrid of MPI + OpenMP, MPI provides coarse-grain parallelism by parallelizing and distributing the data at the inter-node level, whereas OpenMP provides fine-grain parallelism by parallelizing and distributing the data at the intra-node level. This hybrid of MPI and OpenMP for coarse-grained and fine-grained parallelism gives the best performance as opposed to single hierarchical models. But restriction for this model is that it uses a couple of threads in a hybrid model that will increase the thread management overhead and synchronization extensively. Also, this hybrid only supports homogeneous frameworks and provides no support for heterogeneous frameworks [28], [29].

The hybrid of MPI + OpenACC programming model was proposed to resolve the portability and scalability issues for heterogeneous frameworks. The hybrid of these two models introduces unusual inefficiencies including excessive data transfer and configuration overhead among the models [24].

The hybrid model of MPI + CUDA was proposed for heterogeneous frameworks utilizing multi-cores embedded within accelerated GPU devices. MPI distributes work among multiple computers, each of which uses CUDA to execute its share of work. CUDA and MPI can be considered separate entities: CUDA handles process per GPU and accelerate the computational kernels with CUDA. This model achieves coarse grain parallelism through MPI and fine-grain parallelism through GPU computations. But the problem with this model is that it causes portability and scalability issues [22].

The hybrid of OpenCL-MPI makes use of Finite-difference Time-Domain (FDTD) technique primarily based on Open Computing Language (OpenCL) and the Message Passing Interface (MPI). OpenCL provides better portability and gives support for both the distributed shared memory clusters (typically based on multicore CPUs) and GPU-accelerated clusters. Because of the remarkable computational power of GPUs for massive enigmas, execution time could be equal to the communication time which leads to the decline of the scalability. Furthermore, this model does not support dynamic memory architecture [27].

An alternative to the MPI/OpenMP hybrid model is to use a Partitioned Global Address Space (PGAS) model, attempts to use the Single Program Multiple Data (SPMD) model generally support distributed memory systems. In this model, a portion of the memory could be exposed by one process to other processes, though each one has its memory address. PGAS languages propose a one-sided approach where a process locates instantly the remote memory of another process without disrupting its execution. Potential reduction in memory footprint is equal to the reduction in energy consumption. It provides explicit support for parallelism. But this model also has some drawbacks that it does not support distributed memory architecture as used in GPU clusters. Copies severely limit the performance. Further, Compiler can help the programmer with performance, scalability, and programmability is another challenge.

Toward massive parallel computing, a Tri-Hierarchy hybrid MOC (MPI + OpenMP + CUDA) model is proposed. MPI distributes data to overall connected nodes at inter-node and thus provides coarse-grain parallelism. OpenMP is used to achieve fine-grain parallelism and to parallelize CPU threads over intra-node. CUDA is used to achieve finer grain parallelism by executing data over accelerated GPU cores [23]. Though, the United State Department of energy has pointed out some primary constraints taking into account the financial as well as power consumption limitations. These constraints include the power consumption not more than 25 to 30 Mega Watts, system development cost near about to 200 million USD, system time to delivery almost 2020 and integrated multi-cores no more than 100 million [7]. MOC model does not provide an Exa-scale level of performance even by using all the resources mentioned above.

B. Energy Efficient Models

Integer linear programming-based technique (ILP) is used for selecting the optimal configuration of the chip that reduces its power intake. Before the actual execution of the scheduled job on the chip, the ILP optimizer starts its execution on a specific chip to ascertain the best configuration for the job being scheduled. This approach does not require any extra

resources for the selection of optimal configuration for the job. By eliminating the overhead of finding the optimal configuration for the job it minimizes the power consumption and saves resources [30].

C. Models for communication reduction in intra- node

Userspace memory copy-based design utilizes the advanced features of modern systems NUMA and CMP to promote MPI intra-node communication. It requires a shared memory area so that the processes can utilize it as a communication channel. The sending process copies its data and message to the shared memory region and consequently, the receiving process copies the data to its buffer from the shared memory. This technique is more portable than the kernel-assisted memory mapping scheme as it does not demand any services from the kernel thus providing high bandwidth and low latency among processes, offering better performance than the NIC-based loopback scheme. Some limitations of this approach include that it could not improve CPM latency for large message size and it wastes CPU cycles and bandwidth for large messages. It depends on a cache-block replacement scheme to complete its job [31].

Another approach is vShark which reduces the burden in clusters of SMPs for intra-node communication by utilizing thread-based design. Rather than utilizing the communication stack of the message-passing library, the vShark makes use of threads. Instead of starting many processes on SMP nodes, it starts the same number of threads that exist in the same memory location as the process does and thus provides a better communication environment among the processes. One benefit of this library is that it avoids deadlocks and additionalmemory necessities via the usage of communication protocol. The limitations of vShark are, it uses an additional communication protocol and reduces the bandwidth for a more significant number of processors [32].

One more approach to reduce intra-node communication is Kernel-based memory mapping approach that takes advantage from the kernel of the operating system to copy messages directly from one user process space to another without utilizing any shared memory resource. The kernel copies the message from the sender buffer to the receiver buffer, only when the other process arrives at the exchange point taking into account the kernel-based memory mapping address space. Hence the advantage of this technique is, it entails only one copy and use fewer memory transactions and makes use of cache efficiently. But limitations of this approach are, it disturbs OS kernel and has memory mapping overhead [31].

The Network Interface Card offers NIC-level loopback. When the message is initiated from the source, NIC locates the position of the destination address. If the source and destination are the identical nodes, it merely loopback as opposed to injecting it into the network, therefore offer high latency and put off overheads on the network link. The limitations of NIC loopback are, it does not distinguish inter-node or intra-node traffic and no longer utilize the cache impact, and relatively it depends on NIC to locate source and destination [31].

We have studied various models and approaches for achieving performance indirectly by massive parallelism or by reducing communication overhead. In Table 1, we have done the critical analysis on all the above-discussed approaches to deciding the promising approach for future Exascale systems.

Table I. Models To Improve Performance In HPC System

Sr. no	**Performance Efficient Programming Models (Single Hierarchy)**			
	Approaches / Models	***Description***	***Features***	***Limitations***
1	MPI (Message Passing Interface)	MPI is a popular distributed-memory single hierarchical programming model used to communicate between host CPU processors in all associated nodes.	It provides an efficient and portable way to address parallel programs. Moreover, to distribute and parallelize data at the inter-node level MPI offers coarse-grained parallelism and maintains synchronization via blocking methods.	But the major limitation of this model is that MPI designer did not take into account the future Exascale systems that will require novel MPI configurations and runtimes.
2	OpenMP (Open Specification of Multi-Processing)	OpenMP is a single hierarchical programming model used to parallelize and synchronize the threads by utilizing the shared memory-based architecture. It offers two primary principles: sequential equivalence and incremental parallelism.	It provides fine-grained parallelism by parallelizing and distributing the data at the intra-node level via accelerated GPU devices. It provides high-performance support for both heterogeneous and homogeneous systems for parallel applications.	The primary limitation for this model is that it only supports shared memory architecture on a single node, and doesn't give the support for the cluster system.
3	CUDA (Compute Unified Device Architecture)	CUDA abbreviated as "Compute Unified Device Architecture" is an efficient hybrid model utilizing accelerated GPUs and threads for massive parallelism	A useful model to perform thread level optimization that facilitates program flexibility.	GPUs supporting CUDA are only rendered by Nvidia and provides lesser performance as compared to OpenACC and supports heterogeneous

				computation where application use both the CPU and GPU devices.
4	OpenACC (Open Accelerators)	OpenACC appeared as a high-level programming version that makes use of high-level compiler directives to detect parallelism within the code and parallelizing compiler.	It provides high performance than CUDA and facilitates portability to a variety of computing architecture.	However, it does not provide flexibility, thread management, thread synchronization and optimization for the programs, along with other high-level features available in the CUDA framework.
5	OpenCL (Open Computing Language)	OpenCL refers to as "Open Computing Language" is an efficient parallel programming model for heterogeneous frameworks that supports run-time compilation that excludes dependencies on instruction sets, allowing hardware providers to make remarkable changes to instruction sets, drivers, and supporting libraries.	This model provides the guarantee of correctness and portability of kernels over a variety of hardware.	OpenCL demands a confusing setup, for example, the formation of contexts, command queues, and compilation of kernel codes. It doesn't guarantee that a selective kernel will gain peak performance on various architectures.

Performance Efficient Programming Models (Dual Hierarchy)				
6	MPI + OpenMP	The hybrid model of MPI and OpenMP introduced to improve the performance for Petascale computing systems. MPI parallelizes data at the inter-node level and provides coarse-grain parallelism, whereas OpenMP parallelizes data at the intra-node level and provide fine-grain parallelism.	This hybrid model shows good scalability as compared to single-hierarchy-level parallelism.	It gives support only for homogeneous systems, not for the heterogeneous cluster systems. This model makes use of multiple threads in the scheme that ultimately results in thread synchronization and management overhead.
7	MPI + OpenACC	The hybrid of MPI + OpenACC programming model was introduced to write the portable and scalable application for heterogeneous accelerator clusters.	It provides high performance, scalability, and portability from MPI and programmability & portability from OpenACC.	This hybrid model introduces some inefficiencies such as unnecessary data transfer and extreme synchronization between the models.
8	MPI + CUDA	The hybrid of MPI + CUDA supports heterogeneous cluster system in which multiple CPU processors are configured with high-speed NVIDIA GPU devices.	This hybrid model achieves coarse grain and fine-grain parallelism using MPI and GPU computations respectively.	The problem with this model is that it causes portability and scalability issues.
9	MPI + OpenCL	The hybrid of OpenCL-MPI is a hybrid parallelization of the Finite-difference Time-Domain (FDTD) [27] technique primarily based on Open Computing Language (OpenCL) and the Message Passing Interface (MPI).	Due to the portability feature of OpenCL, the advanced code can not only be used for distributed shared memory clusters typically based on multicore CPUs but can also be used for GPU-accelerated clusters.	Due to the remarkable computational power of GPUs for massive enigmas, execution time could be equal to the communication time which leads to the decline of the scalability. Moreover, this model doesn’t support dynamic memory handling.

10	PGAS	(PGAS) the model attempts to use the Single Program Multiple Data (SPMD) model generally support distributed memory systems.	PGAS languages introduced a one-sided approach where a process can access directly the remote memory of another process without interrupting its execution. This model provides explicit support for parallelism.	It does not support distributed memory architecture as used in GPU clusters. Copies severely limit the performance.
Performance Efficient Programming Models (Tri Hierarchy)				
11	MOC (MPI + OpenMP +CUDA)	MPI is used to achieve coarse-grain parallelism and to distribute data overall connected nodes at inter-node. OpenMP is used to achieve fine-grain parallelism and to parallelize CPU threads over intra-node. CUDA is used to achieve finer grain parallelism by executing data over accelerated GPU cores.	It provides Coarse, Fine and Finer Granularity. This model minimizes energy consumption and enables computing on inter-node, intra-node and accelerated GPU devices.	United States Department Of Energy (DOE) has pointed out some primary constraints that include power consumption not more than 25 to 30 Mega Watts, system development cost near about to 200 million USD, system time to delivery almost 2020 and integrated multi-cores no more than 100 million [7]. MOC model does not provide the Exa-scale level of performance even by using all the resources mentioned above.
Models To Reduce Intra-Node Communication				
12	Userspace memory copy	Userspace memory copy-based design utilizes the advanced features of modern systems NUMA and CMP to promote MPI intra-node communication.	This technique is more portable than the kernel-assisted memory mapping scheme as it does not demand any services from the kernel thus providing high bandwidth and low latency among processes, offering better performance than the NIC-based loopback scheme.	Some limitations of this approach include that it could not improve CPM latency for large message size and it wastes CPU cycles and bandwidth for large messages. Further, it depends on a cache-block replacement scheme to complete its job.
13	vShark, A C++ Library	vShark introduces a thread-based architecture to reduce the overhead of intra-node communication in clusters of SMPs. It entails a shared memory area so that the processes can utilize it as a communication channel.	One advantage of this library is that it avoids deadlocks and extra memory requirements through the use of communication protocol.	The limitations of the vShark library are, it uses additional communication protocol and reduces the bandwidth for large number of processors.
14	Kernel-based memory mapping	A method to enhance intra-node communication is Kernel-based memory mapping which takes help from the operating system kernel to duplicate information without delay from one user process to any other without any shared memory region.	The benefits of this approach are, it involves only one message copy with fewer memory transactions and utilizes cache efficiently.	Limitations of this approach are, it disturbs OS kernel and has memory mapping overhead.
15	NIC Loopback	The Network Interface Card offers NIC-level loopback. When the message is initiated from the source, NIC locates the position of the destination address. If the	NIC loopback provides high latency and eliminates overheads on the network link.	The limitations of NIC loopback are, it does not distinguish inter-node or intra-node traffic and not utilize

		source and destination are the identical nodes, it merely loopback as opposed to injecting it into the network.		the cache effect, and somewhat it depends on NIC to locate source and destination.

IV. DISCUSSIONS AND RECOMMENDATIONS

The most challenging step towards Exascale computing systems is that it does not exist yet. However, the performance of HPC systems has been improved on the basis of current results and predictions to achieve Exascale performance. Also, it would require applications to take advantage of billion-way parallelism provided by an estimated Exascale system. The technology challenges mentioned in this paper would require targeted investments to acquire Exascale computing. This is about a quarter million-way parallelism in contrast with modern-day Petascale systems. Performance and power consumption are the two primary HPC metrics that have been taken into consideration the most challenging factors for Exascale computing systems. Node architectures are predicted to change dramatically within the subsequent decade with the increase of power and cooling constraints restriction in microprocessor clock speeds. Therefore, computer corporations are dramatically growing on-chip parallelism to enhance performance. The conventional doubling of clock speeds each eighteen to twenty-four months make the system less efficient. Moreover, doubling the number of cores increased the number of resources and as a result, it automatically increased the power consumption for computation. These techniques are being replaced by doubling of threads or different parallelism mechanisms. Exascale systems will be designed to gain excellent performance within both power and cost constraints. Additionally, hardware breakthroughs might be required to gain beneficial Exascale computing, at least within an affordable power and price range. In step with improvement to Exascale systems, it has been foretold that it will likely be created from a massive variety of heterogeneous systems in which each system will be configured with traditional multicore CPUs and many-core high-speed GPU devices.

The primary goal of Exascale computing systems is to handle massive data HPC applications. For this purpose, many Parallel Programming Models has been introduced to enhance the performance of HPC systems such as:

- Single Hierarchy Models: To attain Terascale (1012 calculations per second)
- Dual Hierarchy Models: To achieve Petascale (1015 calculations per second)
- Tri Hierarchy Models: To accomplish Exascale (1018 calculations per second)

Furthermore, different mechanisms come up expressly to limit the power consumption along with the performance improvement of HPC systems such as:

- User Space Memory Copy Mechanism
- Vshark, A C++ Library
- Kernel-Based Memory Mapping Mechanism
- NIC Loopback Mechanism

In this report, we have observed that a tri-level MOC (MPI+OpenMP+CUDA) model has achieved a tremendous performance by providing coarse-grained, fine-grained and finer granularity parallelism. This model has not only focused on inter-node and intra-node level but also on accelerated GPU devices anticipated for Exascale performance. Many pioneers have critically analyzed the performance of the MOC model. The experimental results have shown that achieved performance is up to 1 Teraflops within 130W power consumption by using the MOC model. Though attaining 1TeraFlops is not a big deal but in contrast, consuming just 130W is an efficient way of utilizing resources. If we try to find more approaches that work in parallel with the MOC model, performance could be enhanced. It could be a promising approach for the Exascale level of performance if the communication at inter-node or intra-node level is reduced to some extent that will reduce the power consumption and ultimately enhance the performance.

V. CONCLUSION

Towards the race of achieving Exascale performance, the power has been the most significant constrained resource among all the other constraints. Therefore, achieving practical Exascale computing with optimum performance will be under the control of power constraint. This advanced computing system will deliver a thousand-fold performance improvement contrasted to the current Petascale computing practice and mandate a need for new, system-wide methodologies and methods for power monitoring and administration. Although the novel programming models and programming methodologies are being proposed day-by-day in HPC culture; but the quest for enhanced programming models always exists. There are significant questions and research regarding the models that will be used at Exascale level to achieve better performance than the current Petascale systems. Contributing to the quest for the optimum programming model for Exascale systems, a comprehensive analysis has been conducted on the existing programming models and approaches. Based on a critical analysis, current study suggested that the MOC model (a tri-level hybrid of MPI +OpenMP + CUDA) a promising model which can be taken into consideration for emerging Exascale computing system to gain massive performance under the power consumption constraints.

VI. ACKNOWLEDGEMENT

This work was performed under the auspices of the Department of Computer Science and Information Technology, Govt. College Women University, Sialkot, Pakistan by Heir Lab-78. The Authors would like to thank Dr M. Usman Ashraf for his insightful, and constructive suggestions throughout the research.

VII. REFERENCES

[1] Perarnau, Swann, Rinku Gupta, and Pete Beckman. "Argo: An Exascale Operating System and Runtime." (2015).

[2] Shalf, John, Sudip Dosanjh, and John Morrison. "Exascale computing technology challenges." International Conference on High Performance Computing for Computational Science. Springer Berlin Heidelberg, 2010.

[3] B. S. J. E. A. R. D. ATKINSON, “The Vital Importance of HighPerformance Computing to U.S. Competitiveness.” (2016).

[4] Reed, Daniel A., and Jack Dongarra. "Exascale computing and big data."Communications of the ACM 58.7 (2015): 56-68. Cappello, Franck, et al. "Toward exascale resilience." International Journal of High Performance Computing Applications (2009).

[5] Zhou, Min. Petascale adaptive computational fluid dynamics. Diss. RENSSELAER POLYTECHNIC INSTITUTE, 2009.

[6] Dongarra, Jack J., and David W. Walker. "The quest for petascale computing." Computing in Science & Engineering 3.3 (2001): 32-39.

[7] Reed, Daniel, et al. DOE Advanced Scientific Computing Advisory Committee (ASCAC) Report: Exascale Computing Initiative Review. USDOE Office of Science (SC)(United States), 2015.

[8] M. Snir, R. W. Wisniewski, J. A. Abraham, S. V. Adve, S. Bagchi, P. Balaji, J. Belak, P. Bose, F. Cappello, B. Carlson, A. A. Chien, P. Coteus, N. A. Debardeleben, P. Diniz, C. Engelmann, M. Erez, S. Fazzari, A. Geist, R. Gupta, F. Johnson, S. Krishnamoorthy, S. Leyffer, D. Liberty, S. Mitra, T. Munson, R. Schreiber, J. Stearley, and E. V. Hensbergen, "Addressing failures in exascale computing," Tech. Rep. ANL/MCS-TM-332, Argonne National Laboratory, Mathematics and Computer Science Division, Apr. 2013

[9] DOE. Report from the Architectures and Technology for Extreme Scale Computing Workshop, 2009.

[10] K. Yoshii, K. Iskra, R. Gupta, P. Beckman, V. Vishwanath, C. Yu, and S. Coghlan. Evaluating power-monitoring capabilities on IBM Blue Gene/P and Blue Gene/Q. In Proceedings of the IEEE International Conference on Cluster Computing (CLUSTER '12), Beijing, China, 2012. (to appear).

[11] Rajovic, Nikola, et al. "The low power architecture approach towards exascale computing." Journal of Computational Science4.6 (2013): 439-443.

[12] P. M. Kogge and J. Shalf. "Exascale computing trends: Adjusting to the new normal' for computer architecture." Computing in Science and Engineering, 15(6):16–26, 2013.

[13] P. Participants. "Workshop on programming abstractions for data locality,PADAL'15".https://sites.google.com/a/lbl.gov/padalworkshop/,2015.

[14] Shafto, Mike, et al. "Modeling, simulation, information technology & processing roadmap." NASA, Washington, DC, USA, Tech. Rep 11 (2012).

[15] Gabriel, Edgar, et al. "Open MPI: Goals, concept, and design of a next generation MPI implementation." European Parallel Virtual Machine/Message Passing Interface Users" Group Meeting. Springer Berlin Heidelberg, 2004.

[16] Message passing Interface, https://computing.llnl.gov/tutorials/mpi/ , 20 June, 2017 [03 Aug, 2017]

[17] Dinan, James, et al. "An implementation and evaluation of the MPI 3.0 onesided communication interface." Concurrency and Computation: Practice and Experience (2016).

[18] Jin, Shuangshuang, and David P. Chassin. "Thread Group Multithreading: Accelerating the Computation of an Agent-Based Power System Modeling and Simulation Tool--C GridLAB-D." 2014 47th Hawaii International Conference on System Sciences. IEEE, 2014.

[19] Martineau, Matt, Simon McIntosh-Smith, and Wayne Gaudin. "Evaluating OpenMP 4.0's Effectiveness as a Heterogeneous PP Model." Parallel and Distributed Processing Symposium Workshops, 2016 IEEE International. IEEE, 2016.

[20] Terboven, C., Hahnfeld, J., Teruel, X., Mateo, S., Duran, A., Klemm, M., Olivier, S.L. and de Supinski, B.R., 2016, October. Approaches for Task Affinity in OpenMP. In International Workshop on OpenMP (pp. 102-115). Springer International Publishing.

[21] Podobas, Artur, and Sven Karlsson. "Towards Unifying OpenMP Under the Task-Parallel Paradigm." International Workshop on OpenMP. Springer International Publishing, 2016.

[22] NVIDIAAcceleratedComputing "developer.nvidia.com/cuda-downloads", 02 Nov 2016.

[23] Ashraf, Muhammad Usman, FadiFouz, and Fathy Alboraei Eassa. "Toward Exascale Computing Systems: An Energy Efficient Massive Parallel Computational Model", International Journal of Advanced Computer Science and Applications, 2018

[24] The OpenACC Application Programming Interface Version 1.0, 2011.[Online]. Available: http://openacc.org

[25] Khronos OpenCL Working Group, The OpenCL Specification Version 1.2, November 2011. [Online]. Available: http://www.khronos.org/

[26] NVIDIA Corporation, OpenCL Best Practices Guide, 2011.

[27] C. Ong, M. Weldon, D. Cyca, and M.Okoniewski, "Acceleration of large-scale FDTD simulations on high performance GPU clusters," in Proc. IEEE APS/URSI '09, 2009.

[28] Jin, Haoqiang, et al. "High performance computing using MPI and OpenMP on multi-core parallel systems." Parallel Computing 37.9 (2011): 562-575.

[29] Mininni, Pablo D., et al. "A hybrid MPI–OpenMP scheme for scalable parallel pseudospectral computations for fluid turbulence." Parallel Computing 37.6 (2011): 316-326.

[30] E. T. U. S. P. L. V. K. e. Akhil Langer, "Energy-efficient Computing for HPC Workloads on Heterogeneous Manycore Chips", pp. 11-19, 2015.

[31] L. C. A. H. D. K. Panda, "Designing High Performance and Scalable MPI Intra-node Communication Support for Clusters", 2006.

[32] S. H. a. T. Rauber, "Reducing the Overhead of Intra-Node Communication in Clusters of SMPs", 2005.

