A Low Power Consuming Model of Parallel Programming for HPC System

Mohammed Nawaf Altouri[1], Abdullah M. Algarni[2]
Department of Computer Science, King Abdulaziz University (KAU)
P.O. Box 80221, Jeddah 21589, Saudi Arabia

***Abstract*—For most of the past five decades, the growing computational power of supercomputers has come primarily from a doubling of clock frequency every 18 months. Over this time period, the clock rate increased by six orders of magnitude, while the number of processors increased by three orders of magnitude. The major challenge caused by the increasing scale and complexity of HPC systems is the massive power consumption. Due to constraints on heat and the power requirements of today's microprocessors, vendors have shifted to putting multiple processors (cores) on a chip. The number of cores per chip is expected to continue increasing exponentially over the next decade. One expected strategy is the correct usage of parallel programming models that decrease power consumption and increase system performance through massive parallelism (concurrency). In the current study, we have proposed a Hybrid MVAPICH-2 + CUDA (HMC) parallel programming model that outperformed other state-of-the-art dual and tri hierarchy level approaches with respect to power consumption and execution time. Moreover, the HMC model was evaluated by implementing the matrix multiplication benchmarking application. Consequently, it can be considered a leading model for the emerging Exascale computing system.**

***Keywords*—HPC; parallel computation; power consumption; hybrid programming; MVAPIC2; CUDA**

I. Introduction

In the next decade, an extreme level computing system called Exascale computing is anticipated to revolutionize computational science and engineering by providing 1018 FLOPS operations per second, which will be comprise hundreds of thousands of heterogeneous compute nodes linked by complex networks [1]. A projection from the world's most powerful system with the capability of handling Petaflops per second developed in the recent past (2014,) creates the possibility of producing Exascale systems deployed in the 2020 timeframe [1][2]. For this Ultra-scale computing system, an extensive change in node architectures is expected, replacing the current trend of increasing clock speed by doubling the number of cores in a system [3][4]. However, a prominent level of computation for the Exascale system has some valid limitations, such as energy consumption (20MW), time of delivery (2020), number of cores (100 million) and cost ($200M) [3][5]. According to the US Department of Energy (DOE), energy consumption per flop must be less than 20 Pico-Joules (PJs) [6]. Under these hard limitations, the development community must rethink its use of existing technologies and expand the co-design space to find better solutions, new applications, algorithms, better technology, and performance.

In attaining the emerging supercomputing goal, one faces number of effective challenges (such as massive power consumption, extraordinary parallelism, memory, bandwidth, latency, hierarchy level, communication and synchronization mechanism, resilience and heterogeneity) that must be determined by introducing new hardware, general-purpose multi-cores and special-purpose accelerators, frameworks, programming models and algorithms. In the last two decades, the microprocessor-based single CPU has increased system performance and decreased cost; however, because of heat dissipation and energy consumption issues, this approach reached a limit [7], [8]. The solution was switched to a model in which the microprocessor has multiple processing units known as "cores" [9], [10]. Therefore, two approaches were introduced and are currently being used "multi-core" [11] integrates a few cores into a single microprocessor while with "many-core" [12] , a large number of cores are integrated into a single device, called a GPU (Graphical Processing Unit) or GPGPU (General Purpose Computing Graphical Processing Unit). Therefore, the aim is to minimize power consumption in the system by increasing system performance through a combination of course-grain and fine-grain parallelism using heterogeneous parallel programming models [13].

In the current study, our primary objective was to introduce a new parallel programming model that can reduce power consumption in the system to deal with several linear/non- linear HPC benchmarking applications. In terms of introducing such a model, we have conducted an empirical study in which we investigated the existing parallel programming models being considered for emerging supercomputing systems. In terms of consequences, we proposed a new parallel programming model: Hybrid MVAPICH-2 + CUDA (HMC), which section III describes in detail.

The rest of the content is organized follows. Section II demonstrates the related work of parallel computing software and hardware technologies and renowned approaches. Section III describes the proposed HMC model. Section IV describes the empirical study, consequences and comparative analysis.

II. Related Work

In [14], [15], [16] and [17], the authors investigated the primary challenges for emerging Exascale computing systems. According to the authors, supercomputer architectures have gone from 1000 processors to 100,000 processors in the last five years while next-generation systems will have more than one million processors. The rate of growth of parallelism is in fact accelerating, it will likely exceed one hundred million when

Exascale systems appear. Some estimates even predict that the need for multiple threads to cover main memory and communication latency means that scientific codes will contain billions of threads. However, they determined that the major challenge caused by the increasing scale and complexity of HPC system's the massive power consumption. One expected strategy is the correct usage of parallel programming models that decrease power consumption and increase system performance through massive parallelism (concurrency). The increase of concurrency from hundreds of thousands to hundreds of millions is also a tremendous challenge for system software to manage; in addition, it is a challenge with respect to application's ability to achieve good performance at this level of parallelism.

Further, Carretero. J, et al. [8] said that an Exascale architecture should be both energy-efficient and power proportional. The subject of reducing the energy consumption in computing brought up two main research directions. The first direction is concerned with power-aware and thermal- aware hardware design, including low-power techniques on all levels. The second research direction is based on the development of power-aware software for the entire software stack, including operating systems, compilers, applications and algorithms. The authors investigated several energy- saving mechanisms in hardware and software including DVFS (Dynamic voltage frequency scaling), clock Gating, power gating, and coprocessors or accelerators. Clock gating reduces power consumption by disabling the clock in those parts of the circuit that are idle or, as in the case of flip-flops, maintain a steady state that does not need to be refreshed. Similarly, power gating achieves a better power reduction than does clock gating. This had led emerging technologies- such as GPU (Graphics Processing Unit), MIC (Many Integrated Cores) and FPGA (Field Programmable Gate Array) becoming the most promising technologies to overcome the power consumption challenge.

Maglione et al., [18] investigated the Advanced Configuration and Power Interface (ACPI), which is an open standard for device power management co-developed by Hewlett-Packard, Intel, Microsoft, Phoenix, and Toshiba. Other advanced tools including Memscale and PGCapping were also investigated. Memscale includes a control algorithm that minimizes the overall system energy based on performance counter monitoring while PGCapping integrates power gating with DVFS for chip multiprocessors.

Feng and Xixhou [19] analyzed different modern architectures and common applications and illustrated that some system components such as caches and network links- disproportionately consume extensive power for common HPC applications. They demonstrated that a large percentage of power consumed in caches and networks can be saved using our approach automatically. Regarding energy optimization, energy spent on cooling accounts for about 40% of total energy consumption in a data center. The author's focus was to extend energy optimization work beyond machine energy savings so that cooling energy could be reduced. They worked on a runtime system which used Dynamic Voltage and Frequency Scaling (DVFS) to minimize the occurrence of hotspots by keeping core temperatures in check. The consequences showed that we can save considerable cooling energy using this temperature aware load balancing.

Aniruddha, et al. [20] discussed performance optimization under power consumption constraints. According to the authors, the power consumption of Intel's Sandy Bridge family of processors can be user-controlled through the Running Average Power Limit (RAPL) library. It has been shown that an increase in the power allowed for the processor (and/or memory) does not yield a proportional increase in the application's performance. Consequently, for a given power budget, it could be better to run an application on a larger number of nodes with each node capped at lower power rather than fewer nodes, with each running at its TDP.

Geist, Al- and Daniel A. Reed [21] conducted a survey about primary high-performance computing challenges. They explored the state of the art and reflected on some of the challenges likely to be faced during the building of trans- Petascale computing systems. The energy required for Petascale clusters is now measured in megawatts; commercial cloud data centers consume 25–60 megawatts. Putative Exascale systems would consume hundreds of megawatts using current technologies. For future technologies, two architectural paths are emerging to address the three key challenges including reliability, energy consumption and software complexity. Consequently, energy consumption is a major driver in the emergence of the above two architectural designs. For large-scale heterogeneous system, energy efficiency can be obtained through energy constrained scheduling and adaptive parallelism. Energy consumption and power costs must be managed with as much care as performance and resilience. By contrast, software complexity must be managed to decrease software development costs.

According to [22], hybrid techniques are solutions that allow for emerging HPC computing systems to deal with the primary issues of power consumption and enhancing performance. The authors proposed different hybrid techniques of MPI+X where X is any parallel programming model to compute GPU. Conventionally this X is considered CUDA or Open ACC.

Further, the authors in [23] proposed a tri-level hybrid of MPI+OMP+CUDA (MOC) to achieve massive parallelism. The primary purpose of this model was to achieve all levels of parallelism including coarse grain, fine grain and finer granularity in the system during the execution of any benchmarking application over a large cluster system. Therefore, the proposed MOC model was implemented on different HPC systems and evaluated the performance as well as power consumption. The MOC model reduced the system's power consumption by 20 MW overall by enhancing the performance by 20 PFLOPS in HPC systems.

III. Proposed Hybrid Parallel Programming Model

In this section, we present the proposed dual-level parallel programming model for the high-end computing system. The proposed approach is a hybrid of MVAPICH-2 and CUDA, named the Hybrid MVAPICH-2+CUDA (HMC) Model.

A. *Selected Model for Enhancement*

Indeed, as more and more compute cores become available on a single node, the expectation is that communication of the local node will play an increasingly important role in the overall performance of parallel applications such as MPI applications. Therefore, it is crucial to optimize intra-node communication paths utilized by MPI libraries. As much communication of data between processors as it will consume more power, so by reducing that communications overhead between processors we can increase that performance and limit that power consumption through intra-node communication. We have decided that MPI+CUDA is the best hybrid model. The main reason for that is that it has less communication overhead; other reasons are described in detail in Section IV. This is done by conducting an empirical study and analysis for all the parallel programming model that shown in Table I and Table II.

Many versions of MPI can deliver the best performance. One way is to use MVAPICH-2, an open source implementation of the Message Passing Interface that can deliver the best scalability, performance and fault tolerance for high-end computing systems and servers using InfiniBand, which is used for interconnect communication and was first popular in high performance computing environments, Internet Wide Area RDMA Protocol (IWARP) and RDMA Over Converged Ethernet (ROCE) networking technologies. It facilitates the task of porting MPI applications to run on clusters with NVIDIA GPUs by supporting standard MPI calls from GPU device memory. Furthermore, it optimizes the data movement between host and device and between GPUs in the best way possible while requiring no effort from the application developer.

Random Direct Memory Access (RDMA) is hardware architecture we used to implement our hybrid model. It is especially useful in massively parallel computer clusters. After the initialization of MPI, a global communicator will contain all processors on that library. Therefore, unprecedented scalability, resiliency, and overhead limitations will be on MPI application. However, MVAPICH-2 has more directives that can decrease power consumption and deliver the best performance. This library will have MPI Sessions- a fundamental change in how we address and organize MPI processes that remove the known scalability barriers by no longer requiring the inclusion of all possible communication peers on the global communicator.

TABLE I. EXECUTION TIME (PERFORMANCE)

Matrix Size	MPI + OMP	OMP + CUDA	MPI + Open ACC	MPI + CUDA	MPI + OMP+ CUDA
1000^2	4.5	3.22	3.79	3.01	5.01
2000^2	14.55	8.55	5.99	4.81	10.81
3000^2	45.99	37.83	38.21	34.2	32.2
4000^2	107.06	85.18	60.86	52.29	46.29
5000^2	202.39	145.62	91.39	88.74	70.74
6000^2	341.56	189.59	128.22	109.94	107

TABLE II. POWER CONSUMPTION

Matrix Size	MPI + OMP	OMP + CUDA	MPI + Open ACC	MPI + CUDA	MPI + OMP+ CUDA
1000^2	311	187.7	198.92	195.4	235.67
2000^2	334.7	200	207.07	205.13	248.91
3000^2	329.6	207	226.58	213.05	256.34
4000^2	332	245.82	243.41	233.21	263.56
5000^2	323	263.26	262	255.84	268.29
6000^2	326	271.44	279.52	271.72	273.603

Session facilitates these efforts with two key contributions:

- A scalable representation of communication groups.
- A tighter integration of MPI applications with the underlying runtime system.

B. HYBRID MVAPICH-2+CUDA (HMC) Model

As shown in Fig. 1 below, MPI initializes by creating a session with a specific run time and get the session name. Therefore, any named set of processes that are exposed by that session can be converted into a group 'MPI group' and define the ranks. These groups will communicate through a parent communicator" which is used by MPI to orchestrate (matching) the communication needed to create a new communicator". Thereafter, it will initialize the data to be computed and every processor will retrieve with a rank number. If the rank does not master the processor these data distributed over processors otherwise, its release and exit. So here will call for the CUDA kernel which means it will parallelize the data from the host (CPU) to device (GPU) cores for actual execution of DMM.

In CUDA, the GPU block dispatcher will schedule the grid by assigning each thread to one of computational core and these threads will be synchronized by self-cooperation. Each block has its own shared memory so that the thread will process the data using this shared memory within the block then return the result to the scheduler .Therefore, this processed data will be on GPU global memory that is visible to the host or CPU memory so it will copy the processed data from Device to the host memory. Finally, if the rank is a master processor, it will receive the processed data from all ranks then it will finalize and print the result.

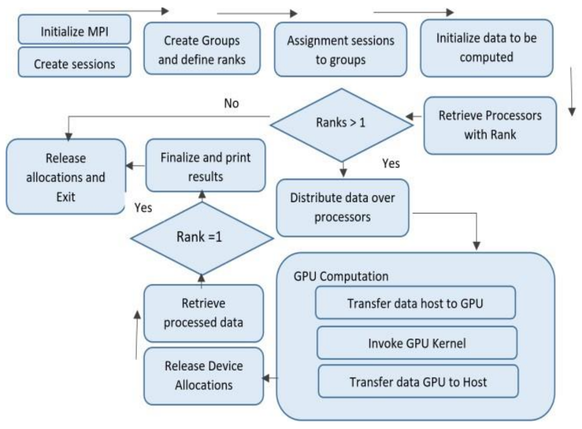

Initialize MPI
Create sessions
Create Groups and define ranks
Assignment sessions to groups
Initialize data to be computed
Retrieve Processors with Rank
Ranks > 1
No
Yes
Release allocations and Exit
Finalize and print results
Distribute data over processors
Yes
Rank =1
GPU Computation
Transfer data host to GPU
Invoke GPU Kernel
Transfer data GPU to Host
Retrieve processed data
Release Device Allocations

Fig. 1. HMC Architecture.

IV. EXPERIMENTAL RESULTS

This section present the results of experiments conducted for the proposed study. All the experiments were performed on HPC Xeon Phi with GPU 1070-ti. To quantify the primary factors (including performance and power consumption) in the proposed HMC model, we firstly carried out an empirical study in which we computed the different datasets of matrix multiplication and Implemented multiple parallel programming models, including single [24], dual [22], [25] and tri-hybrid [23]. Then we executed the same datasets on HMC model. Leading to quantification factors, we measured the system's performance by quantifying execution time against each matrix size. By contrast, the second factor of power consumption was measured using the TechpowerUp GPU-z 2.6.0 software. A running screen of TechpowerUp GPU-z is shown in Fig. 2.

According to existing state-of-the-art models and strategies, single level parallel programming models are not individually applicable for emerging HPC systems. Therefore, it must be in hybrid to achieve massive parallelism through a cluster system with multiple nodes. However, we excluded single level models from our comparative analysis, and followed dual- and Tri-hybrid parallel programming models. Leading to dual-level models, we compared the results with HMC in matrix multiplication with different size executions in the range of 1000x1000 to 6000x6000 matrix size, as shown in Fig. 3.

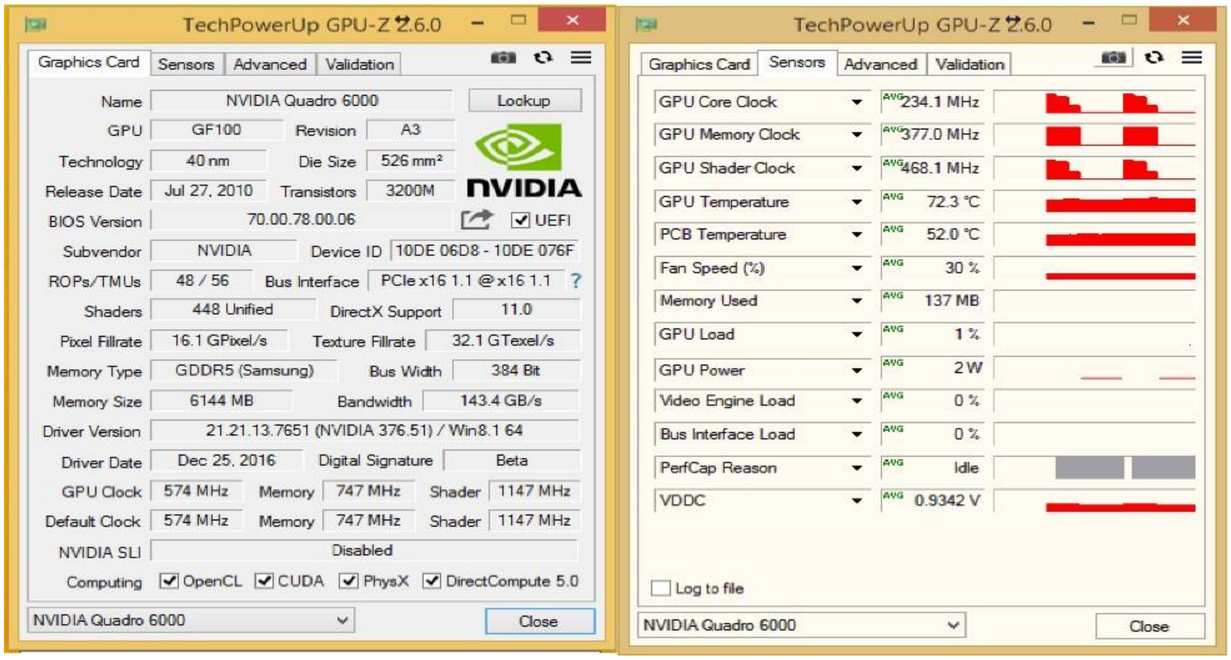
As per Fig. 3, we noticed that at an initial execution 1000 and 2000 matrix sizes, no big difference was seen in all executions, through a tremendous change was seen when matrix size was increased. We can see a major difference in execution time at matrix size 4000x4000 where heterogeneous Computation over cluster system outperformed the homogeneous/single computing system. From this position, the graph changed gradually onward, where the peak reading was noted for the hybrid of MPI+OMP with 342 number of seconds for 6000 matrix size execution. By contrast, the heterogeneous model on single node executed in less time, as OMP+CUDA took 190 seconds .We noticed that other heterogeneous models with the hybrid of MPI+ Open ACC and MPI+CUDA executed the same matrix size in 125 and 110 seconds, respectively, as MPI version 3 improved in terms of enhancing performance under power consumption limitations. Therefore, we integrated MPI-3 (MVAPICH-2) with CUDA which is the proposed model of current study and executed all these datasets in similar way. We discovered a drastic improvement in performance which was 80 second execution time against 6000x6000 matrix size. This is because on MPI-3 it has more directives, like sessions and groups, and it will be scalable. These directives facilitate an improvement in the performance of our HMC model.

According to [23], the proposed MOC model was considered promising for emerging HPC systems to attain a massive performance. Also, the MOC model introduced three level of parallelism, including coarse grain, fine grain and finer grain. Therefore, we also executed the similar matrix multiplication dataset on MOC and compared it with the proposed HMC model as shown in Fig. 4.

It was observed that HMC outperformed throughout the execution as compared to MOC in all datasets. We observed a 20-second difference in the execution for a large dataset, which is the improved performance in HMC.

Further, to quantify the second objective of study, we observed the power consumption in all the selected models and compared it to proposed HMC model. The analysis mechanisms were similar to those used for execution time. We firstly evaluated dual-level homogeneous and heterogeneous models and compared them to HMC with respect to power consumption, as shown in Fig. 5.

MVAPICH-2 introduced the new directives, which are used to optimize the communication cost among the processors. Consequently, it causes a reduction in the system's power consumption during execution. Based on these improvements, we observed that HMC decreased the power consumption throughout the executions. For small computations, up to 50 watts' power consumption was observed that was reduced in HMC by comparing the consumed power measured in the best model MPI+CUDA from existing state-of-the-art mechanisms. As shown in Fig. 6, the power consumption was evaluated in the MOC model to conduct a comparative analysis with HMC model.



Name	NVIDIA Quadro 6000		
GPU	GF100	Revision	A3
Technology	40 nm	Die Size	526 mm²
Release Date	Jul 27, 2010	Transistors	3200M
BIOS Version	70.00.78.00.06	✓ UEFI	
Subvendor	NVIDIA	Device ID	10DE 06D8 - 10DE 076F
ROPs/TMUs	48 / 56	Bus Interface	PCIe x16 1.1 @x16 1.1
Shaders	448 Unified	DirectX Support	11.0
Pixel Fillrate	16.1 GPixel/s	Texture Fillrate	32.1 GTexel/s
Memory Type	GDDR5 (Samsung)	Bus Width	384 Bit
Memory Size	6144 MB	Bandwidth	143.4 GB/s
Driver Version	21.21.13.7651 (NVIDIA 376.51) / Win8.1 64		
Driver Date	Dec 25, 2016	Digital Signature	Beta
GPU Clock	574 MHz	Memory 747 MHz	Shader 1147 MHz
Default Clock	574 MHz	Memory 747 MHz	Shader 1147 MHz
NVIDIA SLI	Disabled		
Computing	✓ OpenCL ✓ CUDA ✓ PhysX ✓ DirectCompute 5.0		

NVIDIA Quadro 6000

GPU Core Clock	AVG 234.1 MHz
GPU Memory Clock	AVG 377.0 MHz
GPU Shader Clock	AVG 468.1 MHz
GPU Temperature	AVG 72.3 °C
PCB Temperature	AVG 52.0 °C
Fan Speed (%)	AVG 30 %
Memory Used	AVG 137 MB
GPU Load	AVG 1 %
GPU Power	AVG 2 W
Video Engine Load	AVG 0 %
Bus Interface Load	AVG 0 %
PerfCap Reason	AVG Idle
VDDC	AVG 0.9342 V

NVIDIA Quadro 6000

Fig. 2. TechpowerUp GPU-z Running.

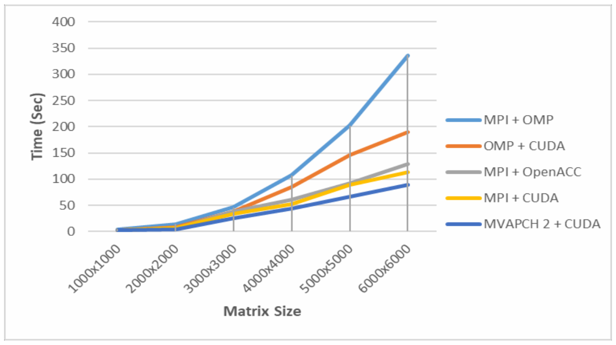


Fig. 3. (Performance) Execution Time for Matrix Multiplication in Dual Level Models vs HMC.

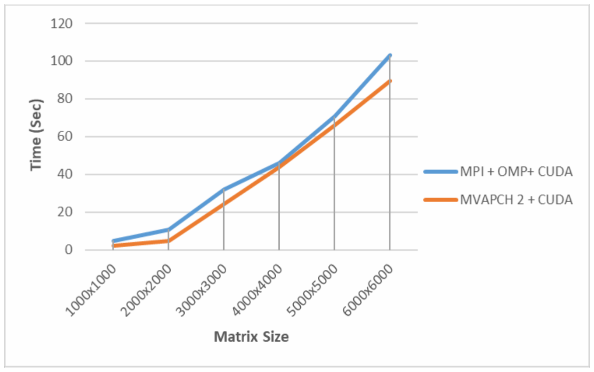


Fig. 4. (Performance) Execution Time for Matrix Multiplication in MOC Model vs HMC.

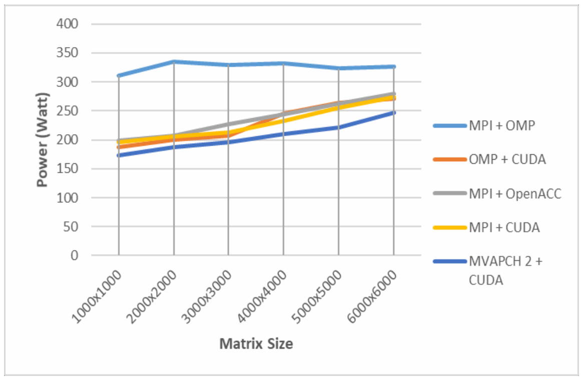


Fig. 5. (Power consumption) Dual Level Models vs HMC.

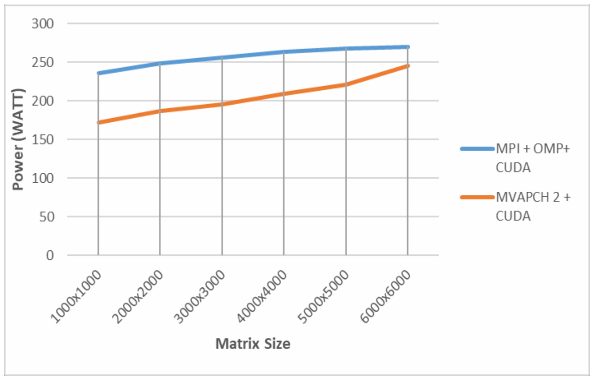


Fig. 6. (Power Consumption) MOC Tri-Hybrid vs HMC Model.

This time we observed a vital difference in power consumption throughout the executions. For small dataset, the observed power consumption was up to 80 watts, which is big difference and an achievement in terms of study's second objective. The same ratio was discovered for all other executions. Finally, HMC consumed 240 watts for 6000x6000 matrix size. We critically noted that this level of power consumption was observed in 2000x2000 for the MOC model, which was computed in a very small time. However, a drastic change in both objectives was achieved in proposed HMC model, which is a big achievement with respect to satisfying the requirements of emerging HPC systems.

V. Discussion

The proposed study was primarily concerned with emerging High-Performance Computing and its perspectives objectives, which are majorly concerned to enhancing performance under the power consumption limitations. These concerns are vital challenges now a day for current and future ICT. According to research communities, there are two solutions to these primary challenges, increasing number of cores in the system to achieve massive performance in the system. This approach is not feasible, as it will increase the power consumption in the system, there another solution is required which is achieving massive parallelism in the system to reduce the execution time that will eventually decrease the power consumption in the system. Leading to the second option, this study proposed a new parallel programming model called Hybrid MVAPICH-2 and CUDA (HMC).

HMC is fundamentally an extension of the dual level model of MPI and CUDA. The issue in a hybrid of MPI+CUDA was similar that it could not fulfill the demand of HPC systems. MPI-3 (MVAPIC2) it has more directives like sessions and groups and it will be scalable. We observed that the quantified execution in HMC was 10% less as compare the other parallel programming models. Further, we also noted that the power consumption was much consumed in dual and tri-level hybrid models but HMC consume 50 to 60 WATTs less comparatively.

VI. Conclusions

Emerging HPC technologies are experiencing more priority and demand in all scientific fields. It has been anticipated that Exascale HPC systems will be introduced at the end of 2020. According to current state-of-the-art technologies, Exascale systems face two vital challenges, including Power consumption when increasing system performance to achieve Exa-flops level of calculations. Various research communities have taken initiatives to address these challenges. With respect to these objectives, the current study proposed a new model named Hybrid MVAPICH-2+CUDA (HMC) to address such challenges. The HMC model implemented in matrix multiplication benchmarking application and compared the quantified performance and power consumption with existing dual- and tri-hybrid parallel programming models. We observed that the HMC model outperformed in all cases when we compared it to other dual- and tri-level parallel programming models. HMC reduced the power consumption up to 80 watts with the same dataset execution within 70 sec less time, comparatively. These improvements can serve as the foundation of an initiative to consider HMC as a leading model in the era of HPC systems.

From future perspectives, HMC is required to implement a large cluster system through which we can quantify the said attributes on different platforms. Moreover, we must implement HMC on different benchmarking applications to observe the behavior of the proposed model when we change the benchmark.

REFERENCES

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

[2] Carretero, Jesus, et al. "Energy-efficient Algorithms for Ultrascale Systems."Supercomputing frontiers and innovations 2.2 (2015): 77-104.

[3] Shalf, John, Sudip Dosanjh, and John Morrison. "Exascale computing technology challenges." High-Performance Computing for Computational Science–VECPAR 2010. Springer Berlin Heidelberg, 2010. 1-25.

[4] Abraham, Erika, et al. "Challenges and Recommendations for Preparing HPC Applications for Exascale." arXiv preprint arXiv:1503.06974 (2015).

[5] Hall, Mary, et al. "ASCR Programming Challenges for Exascale Computing." (2011).

[6] Tolentino, Matthew, and Kirk W. Cameron. "The optimist, the pessimist, and the global race to exascale in 20 megawatts." Computer 45.1 (2012): 0095-97.

[7] Diaz, Javier, Camelia Munoz-Caro, and Alfonso Nino. "A survey of parallel programming models and tools in the multi and many-core era." Parallel and Distributed Systems, IEEE Transactions on 23.8 (2012): 1369-1386.

[8] Rajamony, Ramakrishnan, and Alan L. Cox. "Parallel programming tools." Wiley Encyclopedia of Electrical and Electronics Engineering (1998).

[9] W. Hwu, K. Keutzer, and T.G. Mattson, "The concurrency challenge," IEEE Design and Test of Computers, vol. 25, no. 4, pp. 312-320, July 2008.

[10] Macedonia, Michael. "The GPU enters computing's mainstream." Computer 36.10 (2003): 106-108.

[11] Geer, David. "Chipmakers turn to multicore processors." Computer 38.5 (2005): 11-13.

[12] Satish, Nadathur, Mark Harris, and Michael Garland. "Designing efficient sorting algorithms for manycore GPUs." Parallel & Distributed Processing, 2009. IPDPS 2009. IEEE International Symposium on. IEEE, 2009.

[13] Nakajima, Kengo. "Hybrid vs. flat mpi on the earth simulator: Parallel iterative solvers for finite-element method." Applied Numerical Mathematics 54.2 (2005): 237-255.

[14] Geist, Al, and Robert Lucas. "Major computer science challenges at exascale." The International Journal of High- Performance Computing Applications 23.4 (2009): 427- 436.

[15] Bergman, Keren, et al. "Exascale computing study: Technology challenges in achieving exascale systems." Defense Advanced Research Projects Agency Information Processing Techniques Office (DARPA IPTO), Tech. Rep 15 (2008).

[16] Reed, Daniel A., and Jack Dongarra. "Exascale computing and big data." Communications of the ACM 58.7 (2015): 56-68.

[17] Bergman, Keren, et al. "Exascale computing study: Technology challenges in achieving exascale systems peter kogge, editor & study lead." (2008).

[18] Maglione, Stephen C., and Edward Stanley Suffern. "Power control of servers using advanced configuration and power interface (ACPI) states." U.S. Patent No. 8,250,382. 21 Aug. 2012.

[19] Feng, Xixhou, Rong Ge, and Kirk W. Cameron. "Power and energy profiling of scientific applications on distributed systems." Parallel and Distributed Processing Symposium, 2005. Proceedings. 19th IEEE International. IEEE, 2005.

[20] Marathe, Aniruddha, et al. "A run-time system for power- constrained HPC applications." International conference on high-performance computing. Springer, Cham, 2015.

[21] Geist, Al, and Daniel A. Reed. "A survey of high- performance computing scaling challenges." The International Journal of High-Performance Computing Applications 31.1 (2017): 104-113.

[22] Ashraf, M. Usman, Fathy Alboraei Eassa, and Aiiad Ahmad Albeshri. "High-performance 2-D Laplace equation solver through massive hybrid parallelism." 2017 8th International Conference on Information Technology (ICIT). IEEE, 2017.

[23] Ashraf, M. Usman, et al. "Performance and power efficient massive parallel computational model for HPC heterogeneous Exascale systems." IEEE Access 6 (2018): 23095-23107.

[24] Ashraf, Muhammad Usman, Fadi Fouz, and Fathy Alboraei Eassa. "Empirical Analysis of HPC Using Different Programming Models." International Journal of Modern Education & Computer Science 8.6 (2016).

[25] Ashraf, Muhammad Usman, and Fathy Elbouraey Eassa. "Hybrid model-based testing tool architecture for an exascale computing system." International Journal of Computer Science and Security (IJCSS) 9.5 (2015): 245.