International Journal of
INTELLIGENT SYSTEMS AND APPLICATIONS IN ENGINEERING
ISSN:2147-6799 www.ijisae.org Original Research Paper

Optimized HPC Workload Division Strategy on Heterogeneity Computing Platform

Chandrashekhar B N*[1], Kantharaju V[2], Harish Kumar N[3], Suresh H [4], Geetha V[5]

Submitted:13/03/2024 **Revised**: 28/04/2024 **Accepted**: 05/05/2024

Abstract: Over the past ten years, an incredible expansion in the computation intensity and applications of graphics processing units. The key difficulty for CPU+GPU heterogeneous clusters is effectively allocating the HPC load among the CPU and GPUs while minimizing intra- and inter-node communication costs. To address these difficulties, in this work a systematic workload division technique was developed. The analytic load allocation model divides the HPC workload amongst CPU and GPUs efficiently by considering a number of factors. We have taken into consideration the pinned memory mechanism sole MPI process on corresponding nodes to reduce the overhead of communication between and within nodes. The proposed work used the MPI+OpenMP+CUDA to efficiently utilize CPU and GPU resources. We have evaluated our method on a random dataset using well-known compute-intensive programs like LINPACK. The findings of the experiment show that, in contrast to Adaptive partitioning technique intended analytic HPC workload partition approach worked better

Keywords: *CPU, GPU, Heterogeneous, HPC, LINPAC, workload*

1. Introduction

GPU clusters are popular since they convey extraordinary execution and high-power proficiency for different sorts of scientific applications [4]. Recent GPUs are not general-purpose because they are not standalone. GPUs are thus dependent on general-purpose CPUs. In the case of compute-intensive applications CPU-free cluster is not advisable since only GPUs are not appropriate. So, CPU+GPU heterogeneous clusters [1,2] are important platforms for compute-intensive applications. In the case of compute -intensive applications, few programming frameworks, such as OpenMP, MPI, and CUDA are consolidated to abuse the powers of CPU and GPU to acquire high performance. In any case, two complications arise when the move from homogeneous to heterogeneous CPU+GPU computations.

Firstly, an additional effort of isolating load among multi-core CPUs and Many-Core GPUs. Which requires a division proportion that is corresponding to the in respect to computing speed of the two processors? Second, inter and intra-node data communications containing both the multi-core CPUs and Many-core GPUs, must be precisely programmed with a specific end goal to accomplish high communication effectiveness. To direct the first problem, we have embraced a realistic optimal dynamic HPC load division on CPU-GPUs of each registered node. Similarly, to focus the second complication used an individual MPI activity to control in cooperation of Multi-Core and Many-Core. To overlay computations with the diverse intra and inter-node communications, we adopt different parallel program design models that involve MPI, OpenMP, and CUDA. To illustrate the workload allocation strategy using MPI, OpenMP, and CUDA models, we have chosen compute-intensive applications such as LINPACK [5, 13, 14] to calculate the execution of heterogeneous computing systems. GPUs aid the research community to enhance the high-performance LINPACK (HPL).

In this piece of work, we make the following contributions for the following scenarios,

- create a novel dynamic load allocation approach utilizing the OpenMP+MPI+CUDA on CPU+GPUs cluster to boost the execution of compute-intensive applications on heterogeneous clusters:
- Case 1: Intra-nodes heterogeneity in this case CPU-GPUs cluster which has QUADRO K2000 and QUADRO 2000 graphics cards every node.
- Case 2: Inter-nodes heterogeneity in this case CPU-GPUs cluster which has NVIDIA QUADRO K2000 on one node and QUADRO 2000 graphics cards on other node Comparison compute intensive applications LINPACK performance using our HPC workload division strategy against adaptive partitioning technique [15].

2. Related Work

Workload distribution inspires attention of researcher in

[1] *Amity University, Bengaluru – 562110, INDIA.*
ORCID ID : 0000-0001-6847-8957
[2] *BMS Institute of Technology and Management.– 10,INDIA*
ORCID ID 0009-0009-4415-0530
[3]*BMS Institute of Technology and Management Bengaluru – 560064, INDIA*
ORCID ID : 0000-0002-1003-3274
[4]*KNSIT, Bengaluru 560064-INDIA*
[5] *Reva University, Bengaluru – 64,India*
ORCID ID : 0000-0001-6069-5471
** Corresponding Author Email: cnaikodi@gmail.com*

recent years. Several research has been showed on workload distribution to improve overall performance of compute intensive applications

Rong Shi et. al. [7], advances an inventive two-level jobs segmentation approach for HPL workload on hybrid cluster hubs on a different bunch. In their manner creators disseminate the responsibility considering the figure force of computer processor/GPU hubs across the group. Additionally handles multi-GPU setups they utilized procedures, for example, process network reorganization to diminish MPI correspondence while guaranteeing load balance across hubs. Creators Present point by point examination of execution, proficiency, and adaptability of their half and half HPL plan across various bunches with various designs. Because of memory-figure unevenness loss of effectiveness on computer chip hubs, isn't considered in this paper.

SimpliceDonfack et.al [6] the creators present viable cross breed CPU+GPU approach that is convenient, progressively and effectively balances the responsibility between the computer chips and the GPU. Indeed, even the creator considered information to move bottleneck among computer processor and GPU. In the proposed approach how much, starting work allocated to the central processor before not entirely set in stone by the hypothetical model. Then, at that point, vivaciously settle the heap during execution for example moving a fragment of work progressively moved from GPUs to computer processors to keep load balance creator proposed model self-embrace on any design. Show their strategy on LU factorization.

Takuro Udagawa et.al. [9] Creator proposed another technique to adjust the responsibility between computer chips &GPUs. The proposed technique is based on figuring out and noticing responsibilities and statically disperses the work. Creator prevailed with regards to using processors more productively and speeding up reproduction utilizing NAMD. It gives 20.7% improvement contrasted with GPU ideal code. Proposed technique is shown utilizing on MD (sub-atomic elements) recreation. Writer not tended to when the quantity of cycles builds, computer processor centers must be relegated to correspondence strings. That lessens computer processor speed.

Chakkour et.al. [10] and Liu, et.al. [11] The proposed demonstrates the Indigo3 labeled benchmark suite [46], which consists of 41,790 graph analytics scripts written in parallel C, OpenMP, and CUDA. There is no limit to how many inputs can be used to run a program. They are predicated on 15 different categories of typical issues and 13 sets of alternate parallelization/implementation techniques.

Joselli et.al [12] to improve performance, authors proposed an automatic spreading of HPC load between Multi and Many core by using physics engine. In their work physics engines compute an approximate load and dynamically distribute among CPUs and GPUs. Their method used a distributor virtual base class to derive customized heuristics. To determine the user program scheduling adoption in their framework they called C scripts. This work won't talk about partitions of workload based upon processors capabilities and even they did not formulate any analytical model.

Other works of workload balancing included Becker et.al [18] describes the system that uses the ParFUM framework with that handles numerous simulations on clusters that effectively utilizes both CPU-GPU processors. Author used the similar code for GPU and CPU functions; it allows the users to have programs that stabilize the HPC workload among CPU and GPU. The problem with their approach is not developed any analytical model for Units

Use either SI (MKS) or CGS as primary units. (SI units are strongly encouraged.) English units may be used as secondary units (in parentheses). This applies to papers in data storage**.** For example, write "15 Gb/cm^2 (100 Gb/in^2)." An exception is when English units are used as identifiers in trade, such as "3½-in disk drive." Avoid combining SI and CGS units, such as current in amperes and magnetic field in oersteds. This often leads to confusion because equations do not balance dimensionally. If you must use mixed units, clearly state the units for each quantity in an equation.

The SI unit for magnetic field strength *H* is A/m. However, if you wish to use units of T, either refer to magnetic flux density *B* or magnetic field strength symbolized as $\mu_0 H$. Use the center dot to separate compound units, e.g., "A·m^2."

3. Proposed workload distribution framework

Proposed HPC load distribution framework for HPC application on CPU-GPUs hybrid platform [25]**.**

3.1. CPU-GPUs Based Intra-nodes and Inter-nodes Heterogeneity Computing Platform:

Case: 1 Intra-nodes heterogeneity**:**

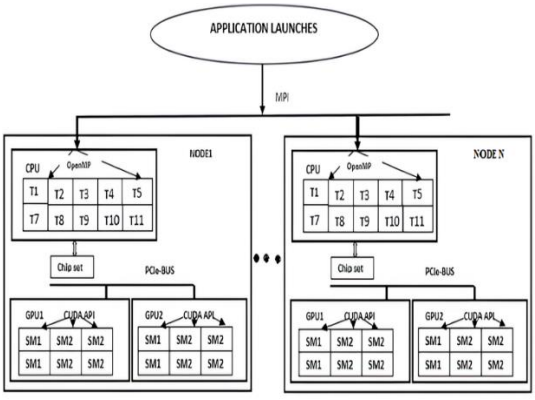

Fig. 1. Architecture of hybrid [CPU+GPU] cluster environment on evry node having two different computing capabilities GPUs.

Case: 2 Inter-nodes heterogeneity:

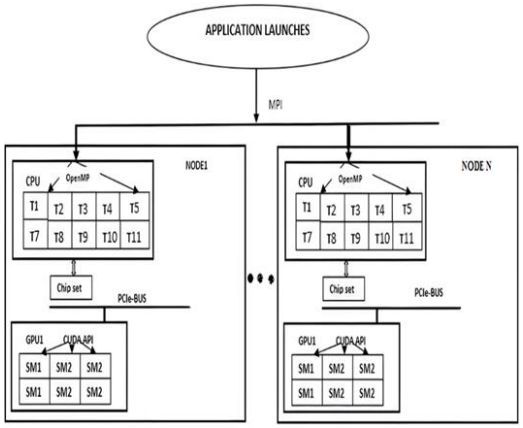

Fig. 2. Architecture of hybrid [CPU+GPU] cluster environment with each node having one CPU and varying capability of one GPU.

In the intra-nodes heterogeneity case, Figure 1 shows an outline of mix architecture, where each node consists of 1 CPU and 2 GPUs having unique processing potential endorsed on the PCI-E. We must map the suitable workload to them. The execution of GPUs changes under different workloads, while CPU performance is moderate [22], separate storage is maintained with many-core and multi-core. Many-core CPU has few numbers of cores, and each core has independent cache. The huge number Streaming Multiprocessors and streaming processors [SPs] are available on multi-core GPU.

A heterogeneous cluster architecture with several heterogeneous compute nodes is depicted in Figure 2. where a single CPU processor and a solitary GPU chip are present in each heterogeneous computing node. We have taken into consideration the pinned memory approach in each scenario, where a distinct MPI activity is used to direct and transfer the appropriate amount of work to each node in the varied cluster. This allows us to resolve the problem of resources being underutilized after the compute intensive application launches to the cluster. When compared to numerous MPI processes per node, we can improve internodes transmission overhead and memory-bandwidth by employing the pinned memory technique and a solitary MPI process, which will result in restricted data transmission [22].

The MPI activity first distributes the computed HPC load to the appropriate nodes within a diverse cluster. Subsequently, it distributes the workload dynamically on each respective node among multiple many- cores and multi-core GPUs, taking into account their individual processing power, speed, and hardware specifications.

3.2 Hybrid [MPI+OpenMP+CUDA] model

Figure 3 illustrates the hybrid [MPI+OpenMP+CUDA] parallel computing paradigm [19,20] that has been chosen in order to enhance the effectiveness of compute-intensive applications. NVIDIA developed the parallel programming language known as CUDA (Compute Unified Device Architecture), which offers a wealth of useful guidelines to help designers create high-performance computing systems that leverage the potent processing capabilities of GPUs. Within the domain of academia engineers have been utilizing CUDA a lot. to improve performance and speed up numerous applications.

OpenMP (Open Multiprocessing) in figure 3 is implemented to exploit Multicore CPU computation intensity and concurrency platform for multi-threaded, shared-memory parallel processing multi-core architecture. By using OpenMP, no need to generate the threads nor assign jobs to every thread, compiling program commands help the compiling program to generate threads data management, thread synchronization, etc. for the parallel processor platform. OpenMP even has runtime library functions to switch the parallel execution environment, control and monitor threads, control and monitor processors. Even OpenMP has environment variables to adjust the execution of OpenMP applications. The most significant benefit of the OpenMP framework is that is not required to restructure the sequential source code. The method of creating a parallel version is only insertion of appropriate compiler directives to reorganize the serial program to a parallel one.

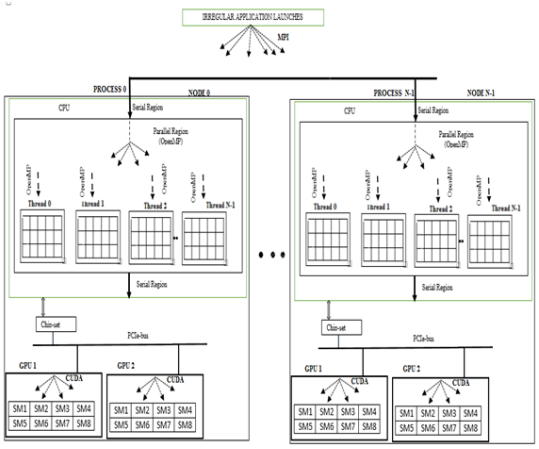

Figure 3: Hybrid prallel programming model[OpenMP+MPI+CUDA]

3.3 Optimized Workload division strategy

Allocation of the HPC load between many-core CPUs and multi-core GPUs is completed in accordance with their computational capacity [21]. While assigning the workload we should be careful with latency issues, because if we allot insignificant volume of HPC load to the many-core CPU, Maintaining CPU engagement during data allocation and kernel launch on multi-core GPUs is challenging. In contrast, if we allocate more volume of HPC load to the many-core then, the multi-core kernel has to delay for the many-core to achieve the assigned workload before

producing the result. we have taken care while assigning the workload optimally to many-core and multi-core [22].

In the projected strategy following parameters are adopted.

Parameters
HPC Workload size
Number of available nodes
Speeds of process
Transmission rate
Memory bandwidth
No. of CPU cores
No. of GPU cores
CPU FLOPS
GPU FLOPS

Utilizing these specifications in proposed frameworks optimal HPC load partition approach decides best load for the compute nodes of many-core and multi-core according to the strength of their computing ability

$$\boldsymbol{CC_{nodei}} = \sum_{i=1}^{i=n} \frac{\boldsymbol{CP_{flopsi} + Gp_{flopsi}}}{\boldsymbol{(CPbwi + GPbwi)}} \quad (\mathbf{1})$$

Where,

CCnodei - nodei computing capacity

CP flopsi - CPU processor FLOPS

GP flopsi - GPU processor FLOPS

Cpbw - CPU memory bandwidth

Gpbw - GPU's memory bandwidth

n- number of nodes

*After computing the co*mputation power of the nodes using equation 1,

Each node's HPC load is processed using equation 2

$$L_{nodei} = \frac{W}{\sum_{j=1}^{j=n} CC_{nodej}} \times CC_{nodei} \qquad (2)$$

Where,

W- Total work load

Lnodei - Nodei Work load

n- number of nodes

load on the CPU_j of the node Equation 3 can be used to calculate

$$L_{CPi} = L_{nodei} \times \frac{CP_{flopsi}}{(CP_{flopsi+GP_{flopsi}})} \qquad (3)$$

Where,

L_{CPi} - *CPU i* load on CPU

CPflopsi - - CPU processor FLOPS

GPflopsi - GPU processor FLOPS

L*nodei* - Nodei workload

After computing a chunk of the task to the many-core CPU,each core's load many-core CPU is computed using equation 4

$$L_{CPi/Cj} = \frac{L_{CPi}}{Nci} \qquad (4)$$

Where,

LCPi/cj load on every core of every CPU

Nci - number available core

Load on multi-core GPUs is processed for different inter-node and intra-node cases

Case 1: In this case CPU+ GPUs cluster has different capabilities graphics cards on all node. So, workload distribution for each GPUs is calculated in a cluster using eq (5).

$$L_{GPi} = L_{nodei} \times \frac{GP_{flopsi}}{(CP_{flopsi+GP_{flopsi}})} \quad (5)$$

Where,

LGPi - Load for GPU i present in node i

CPflopsi - FLOPS of CPU processor

GPflopsi - FLOPS of GPU processor

Lnodei - Work load for Nodei

Case 2: In this case, heterogeneous cluster has different capabilities graphics cards. Each node having only one graphics card. So, workload distribution for individual multi-core GPU in a cluster is computed using equation 6.

$$L_{GPi} = (L_{nodei} - L_{CPi}) \qquad (6)$$

Where,

LGPi - Load for GPU i present in node i

Lnodei - Work load for Nodei

LCPi - CPU i load on CPU

With equation 3 we compute portion of workload on available many-core CPU and by using equation 5 and 6 we calculated the multi core-GPU workload. By using proposed Workload division strategy we can distribute workload optimally.

3.4. Performance evaluation

As per above section, once the division of workload among CPU-GPUs in fusion cluster is calculated with a random input of data volumes. The speedup of HPC load on CPUs+GPUs against only CPU platform is determined using equation 7 and by equation 8 the speedup of HPC load on CPUs+GPUs against only GPU .

$$Speedup_{(hybrid \times cpu)} = \frac{[Tcpu]}{T_{(cpu+gpu)}} \quad (7)$$

$$Speedup_{(hybrid \times Gpu)} = \frac{[Tgpu]}{T_{(cpu+gpu)}} \quad (8)$$

Where,

Tcpu is Elapsed _Time of parallel computation

TGpu is Elapsed _Time of parallel computation

Speedup (hybrid X CPU) is speedup of compute applications of heterogeneous v/s Cpu

Performance (hybrid X Gpu) is speeedup of compute applications of heterogeneous v/s Gpu

T (cpu+gpu) is Total Elapsed _Time of hybrid computation

$$\text{Elapsed Time}=[\text{Execution TIME}]x10^{-6} \quad (9)$$

We attempted to assemble the formula for both muti-core CPU and many-core GPU, which have different configurations, by utilizing the same formula to assess the execution of LINPACK Benchmark apps on CPU-GPU. We then took the readings and displayed the comparison through a graph.

4. Experimental setup and performance analysis

Testing were carried out to confirm the suggested workload allocation strategy for Compute Intensive Applications on Hybrid Platform [MPI+OpenMP+CUDA] the trials were showed on 2 different cases of clusters:

Case 1: In this case CPU+ GPUs cluster has NVIDIA QUADRO K2000 and QUADRO 2000 graphics cards on each nodes. So workload distribution for each GPUs in a cluster is computed using equation 5. By using the above equation 3, we compute CPU workload and by using equation 5 we compute the GPUs benchmark load. With the right workload split between CPU and GPU, a heterogeneous system's processing resources may be able to handle the burden and provide improved performance.

Case 2 Features two nodes with a 2.40 GHz six-core/socket Intel Xeon CPU processor, 31 GB of RAM, and one CUDA-enabled GPU (NVIDIA Quadro K2000). The other node has a CUDA-enabled GPU (NVIDIA Quadro 2000) that serves as a device accelerator. The configuration includes a Dell Precision R5500 system with 227 and 192 cores, respectively, connected by a 100 Mbps switched Ethernet [19].

In various cluster scenarios, every node is set up with MPICH2-1.2 for the MPI library. With Fedora 24 operating system, the compilers utilized are NVIDIA nvcc version 5.0 and GCC version 4.4.7. We ran tests on systems that we have administrative authority over [19].

4.1 Experimental results on LINPACK benchmark applications

The industry-standard benchmark program LINPACK is used to measure how processing frameworks are exhibited. Addressing the NXN arrangement of direct equation is utilized.

$$Ax=b \quad (10)$$

For case 1: In this case CPU+ GPUs cluster has NVIDIA QUADRO K2000 and QUADRO 2000 graphics cards on each nodes. So workload distribution for each GPUs is calculated using equation 5 in a cluster. By consuming the beyond equation 3, we compute CPU load and by using equation 5 compute the GPU's load. In theory, HPC workloads can be appropriately distributed throughout the computational capabilities of a mixed system to improve performance by appropriate workload distribution between many and multi core. Performance comparison results of novel division of load approach against an adoptive partitioning technique [22]

4.1 Performance comparison results of novel workload division strategy against an adoptive partitioning technique

LINPACK for case 1:

Table 1 displays the comparison of performance. of LINPACK in Gflops using adaptive partitioning technique against our novel workload division strategy [23]. The application was executed on our workload distribution framework, having different capabilities based CPU and NVIDIA Quadro K2000 and Quadro 2000 GPUs on an intranode heterogeneity cluster with hybrid programming models and to minimize communication overhead we used pinned memory techniques to accelerate the performance

Table: 1 Performance comparison results of HPC load division strategy against an adoptive partitioning technique

No of data points	Performance in Gflops Adaptive partitioning technique	Performance in Gflops novel workload division strategy	% of performance improvement
2000	1.1364768	4.673553275	75.68281063
4000	1.469949107	6.138539305	76.05376402
6000	1.901270997	8.062744251	76.4190586
8000	2.45915412	10.59011625	76.77878069
10000	3.180734885	13.90972586	77.1330153
15000	4.114046504	18.26991026	77.48184614

20000	5.3212164	23.99685114	77.82535563
30000	6.882601825	31.51897609	78.16362497
40000	8.90213897	41.3990089	78.49673408
50000	11.51426165	54.37606642	78.82476169
60000	14.89285011	71.42095133	79.14778531
70000	19.26280565	93.808777	79.46588127
80000	24.91502156	123.2143577	79.77912474
90000	32.22574689	161.8374999	80.08758976
100000	41.68163211	212.5675681	80.39134921
150000	53.91212378	279.1996357	80.69047488
200000	69.73136472	366.7183911	80.98503745
250000	90.19238873	481.6710381	81.27510653
300000	116.6572176	632.6570867	81.56075068

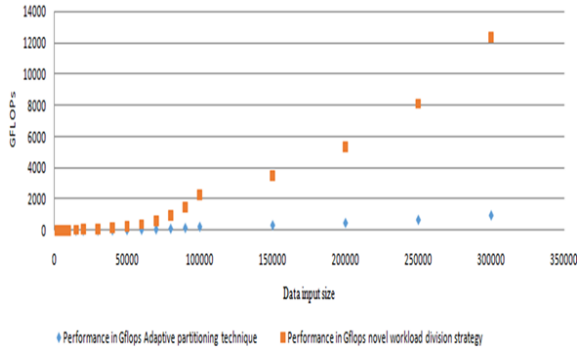

Fig 4: Performance comparison results of HPC LINPACK load partition scheme against an adoptive partioning technique

Figure 4 compares the execution of a unique workload division approach with an adoptive partitioning technique. The y-axis displays GFLOPs, while the x-axis displays the different data input sizes. When compared against adoptive partitioning strategies, the performance (GFLOPs) of our unique methodology exhibits modest average performance improvement, i.e. 76.7%, throughout the initial stage of LINPACK testing with varying input size up to assign resources according to user priorities and workload factors. Future workload distribution plans ought to include fault tolerance techniques. workload division techniques should be portable in terms of performance across various computing platforms and scalable across a broad variety of system sizes and architectures100,000. The performance of our unique approach shown improved average performance improvements, i.e. 80%, during later rounds of LINPACK trials as started to change data input size. This was achieved by lowering calculation and transfer overhead via using pinned memory strategy on intranode heterogeneity In contrast to o the adoptive partitioning techniques.

our unique workload division technique produced an overall average performance (GFLOPs) of 78.74% in LINPACK trials. Strategy an overall average performance (GFLOPs) i.e.78.74% when compared against the adoptive partitioning techniques.

LINPACK for case 2:

Table 2 shows the speed up comparison of LINPACK in GFLOPs using adaptive partitioning technique against our novel workload division strategy. The application was executed on our workload distribution framework, having different capabilities based CPU and NVIDIA Quadro K2000 and Quadro 2000 GPUs on an internodes heterogeneity cluster pinned memory and hybrid programming models techniques to accelerate the performance

Table: 2 Performance comparison results of HPC load partition strategy against an adoptive partitioning technique

No of data points	Performance in Gflops Adaptive partitioning technique	Performance in Gflops novel workload division strategy	% of performance improvement
2000	2.1364768	6.345355328	66.33006838
4000	3.000177671	9.666914063	68.96447355
6000	4.213041797	14.7271859	71.39275741
8000	5.916223347	22.43632281	73.63104731
10000	8.307940053	34.18090766	75.69420878
15000	11.66654196	52.07334813	77.59594422
20000	16.38290605	79.33181916	79.34888394
30000	23.00592682	120.8590912	80.96467002
40000	32.30639712	184.1243537	82.45403366
50000	45.36671367	280.5066402	83.8268664
60000	63.70684734	427.3414876	85.09228587
70000	89.46123865	651.0389448	86.25869629
80000	125.6272058	991.8337439	87.33384435
90000	176.4137751	1511.022011	88.3248706
100000	247.7315311	2301.986126	89.2383569
150000	347.8804955	3506.990689	90.08037014
200000	488.5160909	5342.770557	90.85650253
250000	686.0056088	8139.51326	91.5719087
300000	963.3330488	12400.24729	92.23134002

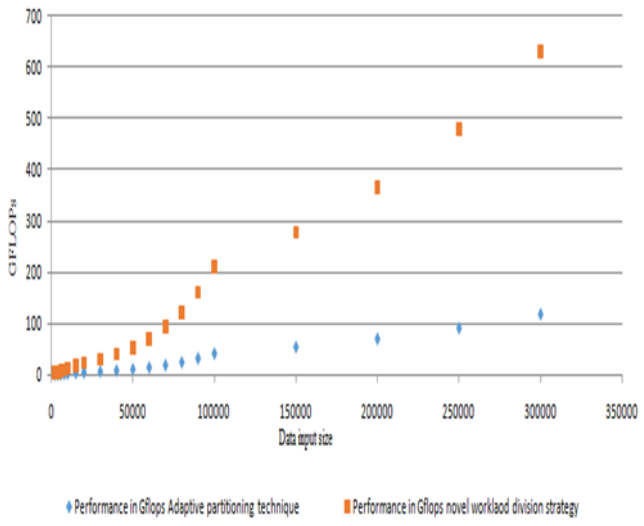

Fig 5: Performance comparison results of novel workload division strategy against an adoptive partitioning technique

Figure 5 results of novel HPC load allocation policy against an adoptive partitioning technique, where x-axis shows the varying data input size and y-axis shows GFLOPs. During initial stages of LINPACK experiment with data size up to 20000 on internodes heterogeneity, we observed that there is nominal average performance (GFLOPs) of improvement is 73.29% when compared against the adoptive partitioning techniques. During later stages of LINPACK experiments as started to vary input size up to 100000 the performance (GFLOPs) of our novel approach showed better average performance improvements i.e. 85.43% by minimizing computation and communication overhead by using pinned memory technique on internodes heterogeneity. observed the GFLOPs improvement of our novel approach for the data size 100000 to 300000 is 91.98%. As we observed from LINPACK experiments size 2000 to 300000 with our novel workload division strategy an overall average performance (GFLOPs) i.e.82.16% when compared against the adoptive partitioning techniques.

5. Conclusion and Future scope

The two primary challenges to improving the execution (GFLOPs) on the intra and internodes heterogeneous clusters are effectively distributing the HPC load concerning the multi-core and many-core and overcoming the inter- and intra-node communication overhead. To effectively divide the HPC load on intra- and internodes heterogeneity clusters, proposed a novel analytical technique in this paper, which expresses our expertise resolving these two challenges. We have also thought of using a OpenMP+MPI+CUDA to efficiently use the powers of multi and many cores. To decrease the volume of communication overhead between and within nodes, pinned memory with a solitary MPI process is managed to handle a portion of the load on each node. We have also contrasted our novel's performance (GFLOPs). a unique approach to analytical workload management using the adoptive partitioning technique with benchmark applications from LINPACK. Based on experimental results, we can conclude that our novel workload division outperforms adoptive partitioning technique when compared to a random data input size with different CPU and GPU capabilities based intra-nodes heterogeneity (CASE-1) and internodes heterogeneity (CASE-2) cluster by achieving overall average performance (GFLOPs), i.e. 78.74% and 82.16%. Future systems might have to dynamically.

6. References and Footnotes

Author contributions

Chandrashekhar B N: Conceptualization, Methodology, Software, Field study **Kantharaju V:** Data curation, Writing-Original draft preparation, Software, Validation **Harish Kumar N:** Field study, Visualization, **Suresh H**: Software, Validation Investigation, **Geetha v** Writing-Reviewing and Editing.

Conflicts of interest

The authors declare no conflicts of interest.

References

[1] Chandrashekhar, B.N., Kantharaju, V., Harish Kumar, N. et.al. (2024) “Balancing of Web Applications Workload Using Hybrid Computing (CPU-GPU) Architecture”. SNCOMPUT.SCI. Journal 5,127 Springer https://doi.org/10.1007/s42979-023-02444-2.

[2] J Zeng, L., Alawneh, S. G., & Arefifar, S. A. (2024). Parallel multi-GPU implementation of fast decoupled power flow solver with hybrid architecture. Cluster Computing, 27(1), 1125-1136.

[3] B. N. Chandrashekar, Mohan M, and Geetha V,(2023) "Forecast Model for Scheduling an HPC Application on CPU and GPU Architecture," 3rd International Conference on Intelligent Technologies (CONIT-2023), pp. 1-5, DOI: 10.1109/CONIT59222.2023.10205724.) IEEE

[4] T. Shimokawabe, T. Aoki, T. Takaki, T. Endo, A. Yamanaka,N. Maruyama, A. Nukada, and S. Matsuoka, “Peta-scale phase-field simulation for dendritic solidification on the TSUBAME 2.0 supercomputer,”in Proceedings of 2011 International Conference for High Performance Computing, Networking, Storage and Analysis, Nov. 2011,pp. 3:1–3:11..

[5] Du, Dayou, Gu Gong, and Xiaowen Chu. "Model Quantization and Hardware Acceleration for Vision Transformers: A Comprehensive Survey." arXiv preprint arXiv:2405.00314 (2024).

[6] S Chalumeau, Felix, Bryan Lim, Raphael Boige, Maxime Allard, Luca Grillotti, Manon Flageat, Valentin Macé et al. "Qdax: A library for quality-diversity and population-based algorithms with hardware acceleration." Journal of Machine Learning Research 25, no. 108 (2024): 1-16.

[7] Rong Shi, SreeramPotluri, Khaled Hamidouche, Xiaoyi Lu, Karen Tomko, and Dhabaleswar K. (DK) Panda Ohio State University "A Scalable and Portable Approach to Accelerate Hybrid HPL on Heterogeneous CPU-GPU Clusters" 978-1-4799-0898-1/132013 IEEE

[8] SimpliceDonfack , StanimireTomovand Jack Dongarra Innovative Computing Laboratory, University of Tennessee, Knoxville, USA"Dynamically balanced synchronization-avoiding LU factorization with multicore and GPUs" 2014 IEEE 28th International Parallel & Distributed Processing Symposium Workshops 978-1-4799-4116-2/14 IEEE Computer society

[9] TakuroUdagawa, Masakazu Sekijima "GPU Accelerated Molecular Dynamics with Method of Heterogeneous Load Balancing" 2015 IEEE International Parallel and Distributed Processing Symposium Workshop 978-1-4673-7684-6/15 IEEE Computer society.

[10] Chakkour, Tarik. "Parallel computation to bidimensional heat equation using MPI/CUDA and FFTW package." Frontiers in Computer Science 5 (2024): 1305800.

[11] Liu, Yiqian, et al. "Indigo3: A Parallel Graph Analytics Benchmark Suite for Exploring Implementation Styles and Common Bugs." ACM Transactions on Parallel Computing (2024).

[12] Mark Joselli, Esteban Clua, Anselmo Montenegro, Aura Conci, and Paulo Pagliosa. A new physics engine with automatic process distribution between cpu-gpu. In Sandbox '08: Proceedings of the 2008 ACM SIGGRAPH symposium on Video games, pages 149–156, New York, NY, USA, 2008. ACM.

[13] Hong, Yuxi, et al. "High performance computing seismic redatuming by inversion with algebraic compression and multiple precisions." The International Journal of High-Performance Computing Applications (2024): 10943420231226190.

[14] Shane Cook. CUDA programming: a developer's guide to parallel computing with GPUs. Newnes, 2013

[15] Dongarra J J, Luszczek P, Petitet A. "The linpack benchmark: Past, present and future". Concurrency and Computation: Practice and Experience, 2003, 15(9): 803-820.: http://dl.z-thz.com/eBook/zomega_ebook_pdf_1206_sr.pdf. Accessed on: May 19, 2014.

[16] .Massimiliano Fatica. Accelerating linpack with CUDA on heterogeneous clusters. In David R. Kaeli and Miriam Leeser, editors, GPGPU, volume 383 of ACM International Conference Proceeding Series, pages 46–51. ACM, 2009.

[17] Canqun Yang, Feng Wang, Yunfei Du, Juan Chen, Jie Liu, Huizhan Yi and Kai Lu , "Adaptive Optimization for Petascale Heterogeneous CPU/GPU Computing" National High Technology Research and Development Program of China 978-0-7695-4220-1/10 $26.00 © 2010 IEEE DOI 10.1109/CLUSTER.2010.12.

[18] Aaron Becker, Isaac Dooley, and Laxmikant Kale. Flexible hardware mapping for finite element simulations on hybrid cpu / gpu clusters. In SAAHPC : Symposium on Application Accelerators in HPC, July 2009.

[19] B. N. Chandrashekar, Mohan M, and Geetha V, "Forecast Model for Scheduling an HPC Application on CPU and GPU Architecture," 2023 3rd International Conference on Intelligent Technologies (CONIT-2023), pp. 1-5, DOI: 10.1109/CONIT59222.2023.10205724.).

[20] . Chandrasekhar B N, Sanjay .H.A "Performance Study of OpenMP and Hybrid Programming Models on CPU-GPU Cluster" Fifth Scopus International Conference on 'Emerging Research in Computing, Information, Communication and Applications', (ERCICA-2018) springer publisher

[21] Chandrashekhar B.N, Sanjay H. A "Dynamic Workload Balancing for Compute Intensive Application Using Parallel and Hybrid Programming Models on CPU-GPU Cluster" Journal of computational and theoretical Nanoscience American scientific Publishers Volume 15, Numbers 6-7, June 2018,pp. 2336-2340(5),
DOI: https://doi.org/10.1166/jctn.2018.7464.

[22] B. N. Chandrashekhar and H. A. Sanjay "Performance Framework for HPC Applications on Homogeneous Computing Platform" International Journal of Image, Graphics and Signal Processing (IJIGSP) MECS Press Publishers Vol. 11, No. 8, pp,28-39,2019, DOI: 10.5815/ijigsp.2019.08.03.

[23] Chandrashekhar B. N, Sanjay H. A, Mohan Murthy. Performance Driven Analytical Workload Division Model for the HPC Applications on CPU-GPU Heterogeneous Cluster, Springer Cluster computing Journal 28 September 2022, https://doi.org/10.21203/rs.3.rs-2096666.

B. N. Chandrashekar, K. Aditya Shastry, B. A. Manjunath

and V. Geetha, "Performance Model of HPC Application On CPU-GPU Platform," 2022 IEEE 2nd Mysore Sub Section International Conference (MysuruCon), 2022, pp. 1-6, DOI: 10.1109/MysuruCon55714.2022.9972737.