Molecular Dynamics Simulations Accelerate on Elastic Multi-GPU Architecture Build with FP64/TF32 Latest Streaming Multiprocessor Ampere Architecture

Nisha Agrawal, Abhishek Das, Rishi Pathak, Pankaj Dorlikar, and Manish Modani

Abstract Newtonian equations of motion for systems with millions of particles and bonded complex interactions describe the behavior of a physical system in terms of its motion as a function of time. Simulating such ordinary differential equations of second order with molecular dynamics application GROMACS for biochemical molecules is being used extensively. GROMACS's performance is optimized over the years. This paper focuses on the study of the behavior of molecular dynamics (MD) simulations using GROMACS on the PARAM Siddhi-AI system. The application performance is analyzed with the number of latest NVIDIA A100 GPUs, MPI tasks, and OpenMP threads. It is observed that 12 and 16 OpenMP threads give better performance for PME-GPU and PME-CPU runs, respectively. GPU-enabled GROMACS demonstrated ~18× and ~3× of relative speedup on single-node with 8× NVIDIA A100 GPUs for PME-GPU and PME-CPU, respectively, when compared to single-node CPU only performance. The GPU-enabled GROMACS multiple nodes each node with 8× A100 NVIDIA GPUs performance with CPU multi-node performance were also analyzed. Relative speedup shows the performance gain with GPU-enabled GROMACS is 3.4×, 9×, 8.9×, 7.6×, and 6× for 1, 2, 3, 4, and 5 nodes, respectively. NVIDIA Ampere architecture using recent CUDA version 11×, OpenMPI version 4.0×, multiples of millions of transistors fabricated on a TSMCs 7 nm N7 manufacturing technology, new third generation NVLink with lower latencies, and HPC software complexity. The introduction of a new sparsity feature accelerates mathematical operations while the faster and larger L1 cache and shared memory in Ampere architecture (192 KB per streaming multiprocessor (SM)) delivers additional speedups for high-performance computing (HPC) workloads.

N. Agrawal · A. Das · R. Pathak · P. Dorlikar
Centre for Development of Advanced Computing, Pune, India
e-mail: anisha@cdac.in

M. Modani (✉)
NVIDIA, Pune, India
e-mail: mmodani@nvidia.com

M. S. Kaiser et al. (eds.), *Information and Communication Technology for Competitive Strategies (ICTCS 2021)*, Lecture Notes in Networks and Systems 401,
https://doi.org/10.1007/978-981-19-0098-3_12

Keywords NVIDIA Ampere architecture · A100 · Accelerated computing · GPU · Molecular dynamics · Simulation · GROMACS · High-performance computing

1 Introduction

Recently, PARAM Siddhi-AI, the supercomputer established under the National Supercomputing Mission (NSM) at Center for Development of Advanced Computing (C-DAC) Pune, has achieved the global ranking of 62 in TOP 500 most powerful non-distributed computer systems in the world, released on 16th November 2020.

The system has 42 racks of NVIDIA DGX A100 servers, connected with HDR 200 Gbps InfiniBand Network and 10.5 PB of storage sub system. Each rack consists of 8 powerful NVIDIA A100 GPUs, Dual AMD EPYC Rome 7742 CPU (128 CPU cores) results in 19.5 Teraflops, double-precision peak performance per node. The PARAM Siddhi-AI system can deliver 210 AI Petaflops and 6.3 Petaflops of double-precision peak performances. The majority of the scientific applications are now capable to exploit the heterogeneity of the modern high-performance computing architecture, however, to extract, optimal performance on these state-of-the-art systems (e.g., PARAM Siddhi-AI), applications require further tuning and optimizations.

Molecular dynamics (MD) simulations used extensively to investigate and understand biomolecular function in atomic detail [1]. Almost all the MD applications (e.g., GROMACS, NAMD, LAMMPS, CHARMM, GENESIS, ACEMD, AMBER, DESMOND, OPENMM) are GPU-enabled and exploiting the performance of heterogeneous architecture. Turner et al. [2] discussed the performance of variety of scientific workloads, when executed only on CPUs and when GPUs are used. Groningen machine for chemical simulations (GROMACS) is one of the fastest MD engines, freely available, and widely used applications [1]. GROMACS CPU performance provided a challenge [3] as the optimized CPU SIMD kernels can achieve iteration rates of around 1 ms per step. To improve performance further, GROMACS was run on heterogeneous (GPU-enabled) architecture. The relative speedup on accelerators (GPUs), without extensive tuning, becomes less impressive.

Kutzner et al. [1] benchmark GROMACS 2018 performance on a diverse set of compute nodes with various generations of GPUs (GTX, RTX, K40, Quadro, and V100). Pall et al. [3] shared the GROMACS parallelization approach, GPU enablement, and performance. Konke et al. [1] implemented a performance-optimized, highly efficient GPU-enabled FMM and integrated it into the GROMACS code base as an alternative to particle mesh Ewald (PME). The performance advantage of FMM over PME is demonstrated over V100 GPUs. However, GROMACS performance is not discussed, on the latest NVIDIA GPU Ampere (A100) architecture.

This paper focuses on the performance analysis of GROMACS on the PARAM Siddhi-AI system. In the next section, the system architecture is discussed. GROMACS and its setup are discussed in the third section. GROMACS performance analysis on one node as well as on multi-node is discussed in the results and discussion section. GROMACS performance is analyzed with the number of OpenMP threads, number of GPUs, PME execution on CPU as well as when PME offloads on GPUs. GROMACS performance improvement over A100 GPUs in comparison with the previous generation GPUs (V100) was also demonstrated. The findings are summarized in conclusions section.

2 PARAM Siddhi-AI

The PARAM Siddhi-AI system is used to study GROMACS behavior on elastic multi-GPU architecture. Each node of the PARAM Siddhi-AI system consists of 8 NVIDIA A100 GPUs and Dual AMD EPYC Rome 7442 CPU (128 total CPU cores).

2.1 AMD EPYC Rome 7742 CPU

The NVIDIA DGX A100 system consists of the latest Dual AMD EPYC Rome 7742 with 128 total CPU cores. Apart from the higher core count, system consists of an L3 configuration which may also impact the performance of the CPU-based applications. The 64 cores in EPYC are organized into sixteen core complex (CCX), groups of four cores with each of the cores having private L1 and L2 cache, with L3 cache shared by all four cores in CCX. The L3 cache size of AMD EPYC 7742 is relatively on the larger side in comparison with the other CPU architectures.

2.2 NVIDIA A100 GPU

The newly released NVIDIA A100 GPU based on NVIDIA Ampere architecture is used for this application behavior analysis. A100 GPU has a third generation tensor core which deliver 300 teraFlops deep learning performance; next generation NVLink when connected with NVSwitch deliver up to 600 Gbps GPU-to-GPU bandwidth to unleash the highest application performance; 40 GB of high-bandwidth memory (HBM2) provides bandwidth of up to 1.6 TB/sec which is $1.7\times$ higher than the previous generation of NVIDIA Volta GPU; the structural sparsity feature provides $2\times$ higher performance for the sparse model.

The fives nodes of the PARAM Siddhi-AI system are used to perform the study of the behavior of GROMACS on elastic multi-GPU architecture. Each node consists of 8 NVIDIA A100 GPUs and Dual AMD EPYC Rome 7442 CPU (128 total CPU cores).

3 GROMACS

PARAM Siddhi-AI is designed to provide National Supercomputing Facility for HPC, AI, and HPC-AI applications. The goal of this work is to understand the behavior (scaling efficiency) of the widely used molecular dynamics application GROAMCS on a multi-GPU multi-node system. GROMACS is one of the most highly used HPC applications globally for biomolecular systems.

It evolves systems of particles using the Newtonian equations of motion, forces between particles dictate their movement (e.g., two positively charged ions will repel).

The force (F) on any atom

$$F_i = -\frac{\partial V}{\partial r_i} \tag{1}$$

is computed by calculating the force between non-bonded atom pairs:

$$F_i = \sum_j F_{ij} \tag{2}$$

plus the forces due to bonded interactions (which may depend on 1, 2, 3, 4, or 5 atoms), restricting and/or external forces. Energies (potential and kinetic) and pressure tensor are computed. Here, v represents bond potential.

Calculating forces is the most expensive part of the simulation as—all pairs of particles in the simulation can potentially interact. Force calculations typically fall into three major classes in GROMACS:

- Bonded forces: captures the specific behavior of bonds between particles, e.g., the harmonic potential when two covalently bonded atoms are stretched
- Non-bonded (short range) forces: These help to capture the behavior of particles within a certain cutoff range interact directly
- PME (long range) forces: Accounted through a “particle mesh Ewald” scheme, where fast Fourier transforms (FFTs) are used to perform calculations in Fourier-space. It is much cheaper than calculating all interactions directly in real-space.

PME implementations in molecular dynamics (MD) packages (e.g., GROMACS and NAMD) are highly optimized. For GROMACS, typical MD systems reach iteration rates of O (1000) steps per second, and the computation of all the forces takes less than a millisecond. However, with the increasing demand to study the larger system size (system of 108–109 particles), on high-performance systems [4, 5], the performance of PME runs into a communication bottleneck.

FFT needs to communicate with all (all-to-all communication) processes, implies quadratic scaling with the number of processes. Accordingly, PME scaling breaks down at an intermediate number of processes. At the same time, the FFT grid becomes memory intensive, particularly if high accuracy is required or for highly inhomogeneous charge distributions [6].

Bonded, non-bonded forces, and PME are offloaded on GPUs. However, the performance of FFTs in PME becomes critical in the case of single-node run (all-to-all communication remains in the node) and multi-node run (communication over a network). Therefore, in the present study, PME performance on CPU as well as on GPUs is analyzed in detail. The user community has put a lot of effort to port GROMACS on accelerators. All three types of force calculation can now be executed on the accelerator, but PME performance is a bottleneck as PME run on one MPI rank only.

It is not decomposed to run on many MPI ranks. For the performance study, GROMACS version 2021-dev is compiled with CUDA Version: 11.0, OpenMPI-4.0.5, and ucx-1.9.0. Satellite tobacco mosaic virus (STMV) proteins which consist of 1066 K atoms have been selected due to their relatively large problem sizes. The performance is measured in nanoseconds per day (ns/days).

4 Results and Discussion

GROMACS performance depends on several factors, such as the usage of GPUs, the number of MPI tasks and OpenMP threads, the loading balance algorithm, the ratio between the number of particle–particle (PP) ranks and particle mesh Ewald (PME) ranks, and the type of simulation being performed, force field used, etc.

Performance analysis is divided into two major sections such as (A) single-node multi-GPU and (B) multi-node multi-GPU. GROMACS performance is analyzed with the number of OpenMP threads, MPI tasks, and compared with CPU only performance as well as with the previous generation (V100) GPUs.

4.1 Single-Node Multi-GPU

A number of experiments were performed to study GROMACS single-node behaviors on PARAM Siddhi—AI system. These include the OpenMP thread and GPUs scalability analysis.

4.1.1 OpenMP Thread Scalability

This analysis was carried out to study the impact of the number of OpenMP threads on GROMACS performance. The experiment was performed by varying the number of OpenMP threads for the fixed number of GPU and MPI processes. The GPU-enabled GROMACS spans one MPI task per GPU. Therefore, we used 8 MPI ranks, which spans job on all the 8 GPUs in one node. Two kinds of experiments were performed:

- Case 1: PME, non-bonded, and bonded all the three types of force calculation are offloaded to GPUs. This case will be referred to as PME-GPU.
- Case 2: PME calculations performance on CPU whereas non-bonded and bonded force calculation is offloaded to GPU. It will be known as PME-CPU.

Figure 1 shows the variations in performance for OpenMP threads 4, 8, 10, 12, and 16. Here, a maximum of 16 threads per MPI ranks (total 16 × 8 = 128) is used as the CPU node has 128 cores.

Figure 1 also shows that for PME-GPU, GROMACS performance is increasing with the increase in OpenMP threads up to 12 threads. For 16 OpenMP threads, GROMACS performance decreased slightly. The performance variation is mainly because of the run time spent in waiting due to PP/PME imbalance, PME wait time for PP increased in the case of 16 OpenMP threads per MPI process when compared to 12 OpenMP threads per MPI process performance (Table 1).

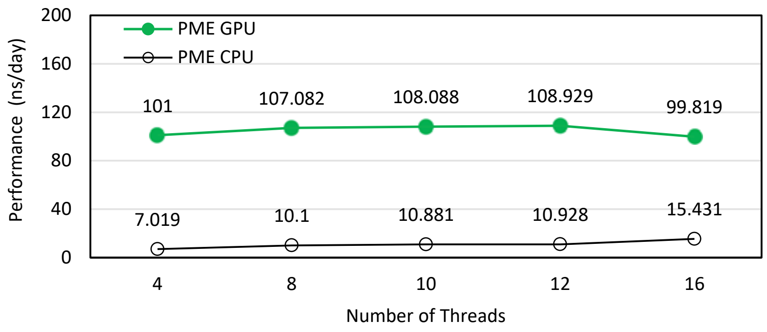

Fig. 1 Single-node PME-GPU and PME-CPU performance for varying number of OpenMP threads

Table 1 Summarize the hardware and software environment used for experiments

Node configuration	
CPU	2× AMD EPYC 7742 64C 2.25 GHz; 128 cores total
System memory (RAM)	1 TB
Accelerator	8× NVIDIA A100 –SXM4
Accelerator memory	40 GB/accelerator card
Storage	1.8 TB; raid:14 TB
OS	Ubuntu 18.04.5 (Kernel: 5.4.0–48-generic)
NVIDIA software	Cuda: 11.0, NVIDIA Driver Version:450.51.06
Message passing interface	OpenMPI-4.0.x

Table 2 Key functions/subroutines time with the change in OpenMP threads for the run on one node and 8 GPUs

Functions or subroutines	OpenMP threads				
	4	8	10	12	16
Domain decomp.	0.74	0.58	0.55	0.52	0.57
Send X to PME	1.1	0.00	1.09	1.12	1.15
Neighbor search	0.47	0.28	0.23	0.21	0.18
Force	0.14	0.08	0.08	0.07	0.07
PME mesh	0.21	0.21	0.21	0.21	0.21
PME wait for particle-particle	8.32	7.85	7.78	7.72	8.44
Wait + Recv. PME F	0.4	0.39	0.34	0.34	0.29
Wait PME-GPU gather	0.0	0.0	0.0	0.0	0.0
Wait bonded GPU	0	0	0	0	0
Wait GPU NB nonlocal	0.02	0.02	0.02	0.02	0.02
Wait GPU NB local	0.0	0.0	0.0	0.0	0.0
Wait GPU state copy	0.32	0.31	0.32	0.33	0.34
NV X/F/buffer ops	0.01	0.01	0.01	0.01	0.0
Comm. energies	0.12	0.12	0.12	0.13	0.18
Rest	0.53	0.42	0.40	0.39	0.4
Total	8.53	8.07	7.99	7.93	8.66

For PME-CPU, when bonded and non-bonded forces are offloaded to GPU and PME computed on CPU, the GROMACS performance increasing linearly with the increase in the OpenMP threads, and maximum performance is observed with 16 OpenMP threads (Fig. 1). It shows PME-CPU performance improves significantly with the increase in OpenMP threads (Table 2).

Figure 1 reveals the single-node performance of GPU enablement of PME (PME-GPU) is 7× better than that obtained by PME-CPU. It also shows GPU enablement of PME improved GROMACS performance significantly.

4.1.2 GPU Scaling

Here, the GROMACS single-node performance is analyzed with the increasing number of GPUs (up to 8). The OpenMP threads are fixed to 12 and 16 for PME-GPU and PME-CPU, respectively, to have optimum performance as discussed in the previous sections.

Figure 2 shows single-node GROMACS scalability with the variation in the number of GPUs. In the case of PME-GPU, 4× GROMACS performance is increased when the number of GPUs increased from 2 to 4. The performance further increased by 2× when GPUs increased from 4 to 8. In the case of PME-CPU, there is a linear

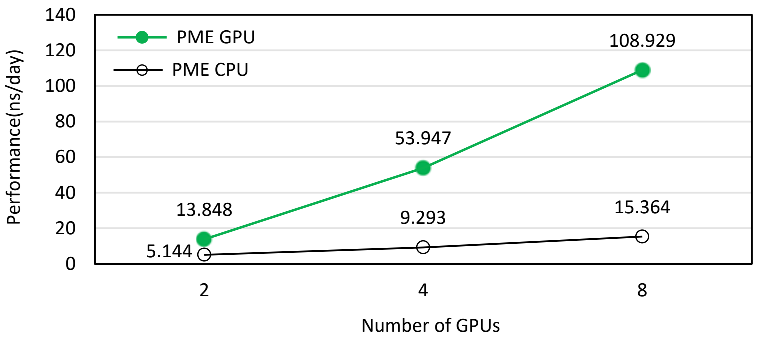

Fig. 2 Single-node PME-GPU and PME-CPU performance with the varying number of GPUs

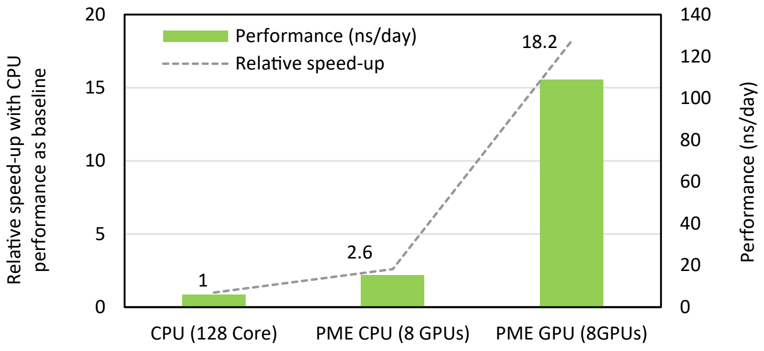

Fig. 3 Single-node CPU, PME-CPU, and PME-GPU performance and speedup

performance increase in GROMACS with 2, 4, and 8 GPUs. PME-GPU enablement (PME-GPU) shows approximately 8× improvement over PME-CPU (Fig. 2).

Figure 3 shows the relative speedup of PME-CPU and PME-GPU with CPU performance as a baseline for the performance. PME-CPU runs 2.6× faster, whereas PME-GPU runs 18.2× faster when compared with CPU runs.

For CPU only run, all the forces of GROMACS remain on CPU whereas in the case of GPUs, bonded and non-bonded force calculations are offloaded to GPUs. CPU runs performed using 16 MPI processes and 8 threads on each node as each node having 128 cores and for the configuration, 16 processes and 8 threads per process achieved maximum single-node CPU performance with GROMACS default parameters.

4.2 Multi-node Multi-GPU Scaling

In this section, GROMACS scalability with the number of GPUs and nodes is analyzed. In addition, GROMACS multi-node performance is also compared with CPU only run is performance is analyzed.

4.2.1 Scaling

As demonstrated above, PME is critical to have GROMACS optimum performance. PME is offloaded to GPUs; however, PME is not decomposed further. In other words, PME runs on the single MPI rank on GPUs. The single-node performance detailed in the previous section shows the PME-GPU performed better than PME-CPUs. For multi-node, multi-GPUs also, GROMACS run is performed for PME-GPU as well as PME-CPU.

In the case of PME-GPU two nodes run, the performance decreased is observed for 16 GPUs run (approximately ~ −10×). Run time logs show the execution time taken by domain decomposition statistics (e.g., Domain decomp., DD comm. Load, Send X to PME, Wait + Recv. PME_F, etc.) is increased significantly for 16 GPUs (2 nodes) run in comparison with 8 GPUs (1 node) run. It again illustrates for PME-GPU scalability over 2 nodes is limited by PME decomposition.

In the case of PME-CPU, GROMACS scalability is observed with the increase of the number of nodes. Figure 4 shows the performance increase with the increase in the number of GPUs and nodes. The performance increases up to 4 nodes (32 GPUs) and remains the same for 5 nodes (40 GPUs) run. It should also be noticed, the performance gain with PME-CPU multi-node run is there; however, observed performance is still less than PME-GPU single-node run (~3×). In other words, PME offload to GPU increases GROMACS performance significantly.

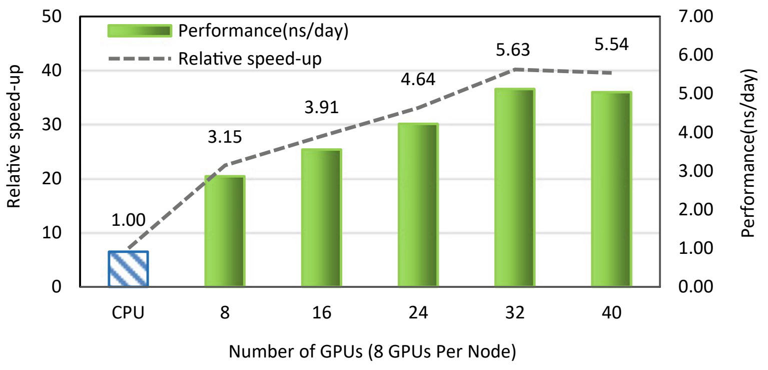

Fig. 4 GROMACS PME-CPU, multi-node multi-GPU scaling

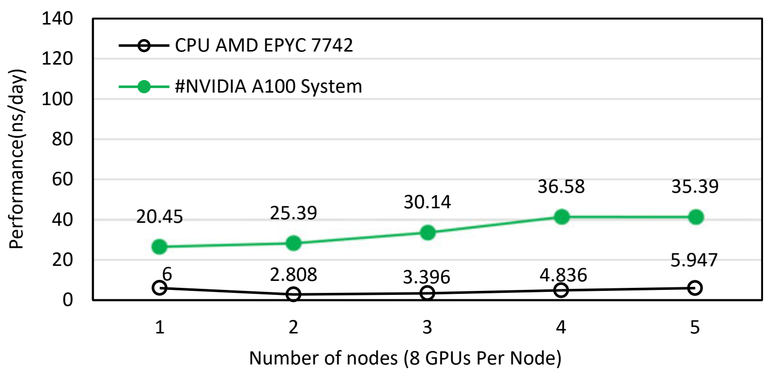

Fig. 5 GROMACS single-node multi-GPU and multi-node CPU scaling

4.2.2 Multi-node CPU and GPU Performance

In this section, GROMACS CPU only multi-node performance is compared with those observed with GPU-enabled, PME-CPU, multi-node performance.

For CPU only run, all the forces of GROMACS remain on CPU, whereas in the case of GPUs, bonded and non-bonded force calculations are offloaded to GPUs. CPU runs performed using 16 MPI processes and 8 threads on each node as each node having 128 cores and for the configuration, 16 processes and 8 threads per process achieved maximum single-node CPU performance with GROMACS default parameters.

In Fig. 5, GROMACS multi-node relative speedup is shown for CPU only and GPU-enabled (PME-CPU). In the case of CPU only runs, the performance decreases with the increase of CPU node. However, in the case of the PME-CPU run, GROMACS performance increases with the increase in the number of nodes. A significant performance gain is observed with the GROMACS GPU enablement. On 5 nodes, GROMACS ~7× performance gain is observed in comparison with CPU only run.

4.3 GPU-Enabled GROMACS Performance on Current Generation (NVIDIA A100) and Previous Generation (NVIDIA V100) GPUs

In this section, the performance of GROMACS is analyzed, on the current generation of GPUs (NVIDIA A100) versus the previous generation GPUs (NVIDIA V100) for 8 GPUs (single node) run. GROMACS V100 performance is considered from the results provided on the NVIDIA developer site [https://developer.nvidia.com/hpc-application-performance] for the STMV input dataset.

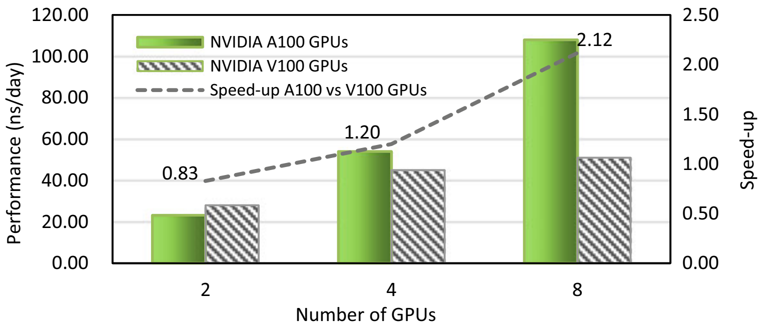

Fig. 6 GPU-enabled GROMACS performance on A100 and V100 GPUs

As shown in Fig. 6, in both cases, GROMACS is scaling well up to 8 GPUs. GROMACS on NVIDIA A100 GPU performed ~2× times faster in comparison with the performance observed on NVIDIA V100 GPUs. It is in sync with the capability difference of A100 and V100.

In the present work, GROMACS performance is analyzed on the PARAM Siddhi-AI cluster. The performance of GROMACS is analyzed for the single node as well as for multi-node runs (up to 5 nodes) on the latest GPU architecture. It is demonstrated that GROMACS scale well on multi-node. The significant performance gain is obtained with the GPU enablement of GROMACS forces. There is a need to further analyze the GPU performance and OpenMP threads behavior for further performance improvement. PME-GPU enablement is critical to have multi-node, multi-GPU performance gain. Other methods to PME, e.g., FMM [6] need to be analyzed on A100 GPUs. In the present, GROMACS GPU enablement, only 1 MPI rank spawns the job on 1 GPU. Therefore, a maximum of up to 8 MPI ranks is used for one node performance analysis. However, GPUs memory is not fully utilized. To further optimize, the multi-instance GPU (MIG) and multi-process services (MPS) need to be used to spawn jobs on GPUs from more than one MPI rank. GROMACS performance needs to be further analyzed on more than 5 nodes of the PARAM Siddhi-AI system.

5 Conclusions

GROMACS is one of the most popular and freely available codes for molecular dynamics (MD) simulations. Over the years, GROMACS performance improved with the advancement of GPU architectures and generations. In this article, we analyzed the characteristic of molecular dynamics application GROMACS on the new and most powerful supercomputer PARAM Siddhi-AI. The system has the latest generation AMPHERE (A100) GPUs. Single node, as well as multi-node GROMACS performance, is demonstrated. Detailed scalability analysis for OpenMP thread, number of GPUs, number of nodes, etc., is carried out. It is observed that 12 and 16 OpenMP threads give better performance for PME-GPU and PME-CPU runs, respectively. GROMACS GPU-enabled performance is also compared with the GROMACS CPU only performance for the single node as well as multiple nodes. GPU-enabled GROMACS demonstrated ~18× and ~3× performance gain on PARAM Siddhi-AI system for one and five nodes. The GROMACS multiple nodes performances with CPU multiple nodes performance were also compared. Relative speedup shows the performance gain with GPU-enabled GROMACS is 3.4×, 9.04× , 8.88×, 7.56×, and 5.95× when 1, 2, 3, 4, and 5 nodes are used to run GROMACS, respectively. In end, limitations of the present work and future work are discussed.

Acknowledgements The support and the resources provided by the Center for Development of Advanced Computing (C-DAC) and the National Supercomputing Mission (NSM), Government of India are gratefully acknowledged. We acknowledge the use of computing resources at PARAM Siddhi-AI (C-DAC, Pune).

References

1. Kutzner C, Pall S, Fechner M, Esztermann A, Groot BL, Grubmuller H (2020) Improved use of GPU nodes for GROMACS 2018. arXiv: 1903.05918v2
2. Turner D, Andresen D, Hutson K, Tygart A (2018) Application performance on the newest processors and GPUs. In: Proceedings of the practice and experience on advanced research computing July 2018 Article No. 37, pp 1–7
3. Pall S, Zhmurov A, Bauer P, Abraham M, Lundborg M, Gray A, Hess B, Lindahl E (2020) Heterogeneous parallelization and acceleration of molecular dynamics simulations in GROMACS. Chem Phys 153:134110. https://doi.org/10.1063/5.0018516
4. Berendsend HJC, van der Spoel D, van Drunen R (1995) GROMACS: a message-passing parallel molecular dynamics implementation. Comput Phys Commun 91(1-3):43–56
5. Abraham MJ, Murtola T, Schutz R, Pall S, Smith JC, Hess B, Linkahl E (2015) GROMACS: high performance molecular simulations through multi-level parallelism from laptops to supercomputers. SoftwareX 1–2:19–25
6. Bartosz Kohnke B, Kutzner C, Grubmüller H (2020) A GPU-accelerated fast multipole method for GROMACS: performance and accuracy. J Chem Theory Comput 16:6938−6949