Comparative Study: MD Simulation with different Load Balancing Technique on Heterogeneous Environment

Jitesh M. Sapariya
Computer Engg. (M.E.)
Pimpri Chichwad College of Engg.
Pune, India

Sudershan Deshmukh
Computer Engg. (Faculty)
Pimpri Chichwad College of Engg.
Pune, India

ABSTRACT
We present the new approach to utilize all heterogeneous resources like CPU cluster, GPU cluster in multi core and multi GPU environment. MD simulation are used for deeper understating of fluid flows, chemical reaction, and other phenomena due to molecular interaction. The main drawback in the MD simulation is that it require computationally demanding more resource because of its amount of O(n2). The use of heterogeneous resources is an attractive solution and has been applied to MD problems. However, such heterogeneous resources cause load imbalances between CPUs and GPUs and they were not utilize all available computation resources.

Keywords
Molecular Dynamics Simulation, GPU, CPU, Heterogeneous Resources.

1. INTRODUCTION
MD simulation is a method used to track the motions of molecules based on classical mechanics and gives detailed information on molecules and phenomena. This method is used for various objects such as gaining better understanding of the thermodynamics of biomolecules [1], refining protein structures [2] and calculating the binding free energy between proteins and ligands [3]. A critical problem with Molecular dynamics simulation is that it is require more computing time. For example MD simulation requires 2() ON calculation to evaluate the interactive forces in N atoms systems. Solvated protein systems contain more than 4 10 atoms, and evaluations of these involve heavy calculations. One more reasons is that it requires number of simulation steps to obtain meaningful results. For example, protein folding generally occurs more than in the order of micro seconds. This physical time is equivalent to more than 9 10 steps in MD simulations [4]. The use of heterogeneous resources is one solution to this problem. Various implementations of Molecular Dynamics using GPUs have been reported thus far [5] [6] [7]. These can be divide into two types according to what calculations are executed on GPUs and CPUs. The first type is all calculations including force evaluations and solving Newton's equations of motion are executed on GPUs. The second type is that only some calculations such as evaluations of non-bonded interaction forces are calculated on GPUs. AMBER [8] uses the first type of method while GROMACS [9], NAMD [10] and LAMMPS [11] use the latter. Although both methods have succeeded in speeding up simulations, more acceleration can be expected by using processors more efficiently. For example, NAMD only uses GPUs for evaluations of non-bonded forces and uses CPUs for other calculations including evaluations of bonded forces. Even though GPU calculations can be overlapped with CPU calculations, processors cannot be efficiently utilized if the workloads of CPUs and GPUs are imbalanced because all force calculations must be synchronized before the coordinates of atoms are updated.

2. LITRATURE SURVEY
MD is a simulation method of computing dynamic particle interactions on the molecular or atomic level. The method is based on knowing, at the beginning of the simulation, the mass, position, and velocity of each particle in the system (in general in a 3D space). Each particle interacts with other particles in the system and receives a net total force. This interaction is performed using a distance calculation, followed by a force calculation. Force calculations are usually composed of long range, short range and bonded forces. While bonded forces are usually among few atoms composing molecular bonds, the long range and short range forces are gated by a predetermined cutoff radius. When the net force for each particle has been calculated, new positions and velocities are computed through a series of motion estimation equations. MD is an application well known for its load-imbalance behavior especially when the simulated systems have no uniform densities. STARPU, Is a runtime system that efficiently exploiting heterogeneous multicore architecture. It provides an execution model, a unique framework to design scheduling policies and the library automated data transfers. Author have written several scheduling strategies and observed how they works on some classical problems. In which author applying simple scheduling strategies can reduce load balancing issues and improve data quality.

STARPU is a representative platform for Heterogeneous multicore architectures. It has a heterogeneous task scheduling system supporting multi-core processors, NVIDIA GPUs, OpenCL devices, and Cell Processors [12]. Author present heterogeneous algorithms with hybrid tiles to solve a class of dense matrix problem that affine a loop structure. They treat a multicore and multi-GPU system as distributed machine, and deploy a heterogeneous multi-level block cyclic data distribution to minimize communication and also introduced an auto tuning method to determine the best tile sizes. Author also design a new run time system for the heterogeneous multicore and multi-GPU architectures.

Author provides static approach to CPU-GPU load balancing in linear algebra [14].They developed heterogeneous tile algorithms and applied them to Cholesky and QR factorizations. Load balancing between CPUs and GPUs was achieved by hybrid tile sizes determined by the speed of each device. Their method could keep imbalance rate within 5 %. Authors discuss several important issues in porting a large molecular dynamics code for use on parallel hybrid machines. Choosing a hybrid parallel decomposition that works on CPUs with distributed memory and accelerator cores with the shared memory, and also minimizing the amount of code that must be ported to efficient acceleration and utilizing the

available processing power from both multicore-CPU and accelerator and choosing a programming model for acceleration.

Author also provide the solution to each of above issue for short range force calculation in the MD package LAMMPS, however the method can be applied in many molecular dynamics codes. They provides algorithm for efficient short range force calculation on hybrid high performance machines and provide approach for dynamic load balancing of work between CPU and accelerator cores and provide Geryon library that allows a single code to compile with both CUDA and OpenCL for use on a verity of accelerators. LAMMPS applied a dynamic load balancing strategy for van-del-Waals and electrostatic interactions [15] [16]. They implemented a system to dynamically move atoms between CPUs and GPUs to balance the calculation times for CPUs and GPUs. Brown et al. [15] reported they could achieve 22.8 % of acceleration with this system in double precision simulations but this system did not have effect in single precision simulations.

3. PROPOSE SYSTEM

Molecular Dynamics simulations are mostly used for simulating the motions of molecules to gain a deeper understanding of chemical reactions, fluid flows and other physical phenomena due to molecular interactions. An MD simulation is a numerical solution of the Newton's equation of motion.

$$m^T \frac{d^2 q}{dt^2} = -\nabla U(q) \qquad (1)$$

Where m, q and U correspond to the mass vector, position vector and potential energy. This continuous differential equation is broken down into discrete small time steps, each of which is an iteration of two parts: the calculation of force (calculating forces from evaluated conformational energies) and atom updates (calculating new coordinates for molecules).

$$\begin{aligned} U(q) = & \sum_{bonds} k^b (b - b_0)^2 + \\ & \sum_{angles} k^\theta (\theta - \theta_0)^2 + \\ & \sum_{torsion} V_n [1 + \mathrm{Cos}^2 (n\phi - \phi_0)] + \\ & \sum_{i,j \in atoms} \in_{ij} [(\frac{r_{ij}}{r_{ij}})^{12} - 2(\frac{r_{ij}}{r_{ij}})^6] + \sum_{i,j \in atoms} \frac{q_i q_j}{r_{ij}} \end{aligned} \qquad (2)$$

The former three terms of bonds, angles and torsions, are called bonded forces and they are derived from chemical bonds. The other two terms, van-del-Waals and Coulombic electrostatic forces, are called non-bonded forces and they affect atom pairs without chemical bonds. Most computational time in Molecular Dynamics is spent in calculating non-bonded forces, which scale with the second power of the number of atoms.

We proposed a method of distributing the workload heterogeneous resources like CPU cluster or GPU cluster. The most important concept is distribute the workload or calculation is depending upon the availability of resources. Therefore, it enables the workload to be balanced across heterogeneous resources and efficiently utilize all resources. In our method the query first came to our scheduler, it checks the availability of the resources after that our algorithm takes the decision about that query execute on the GPU cluster, CPU cluster, distribute the workload between CPU and GPU.

Case 1. In which that all calculation including force evaluation and solving Newton's equation of motion are executed on CPUs. This is time consuming process because it needs to distribute the calculation among the no. of CPU available and result needs to integrated then same process are going on till the number of evaluation needs to calculation. Its disadvantages is that it needs to communicate among the all cores available in the CPU cluster.

Case 2. In which that all calculation including force evaluation and solving Newton's equation of motion are executed on GPUs. Because Graphics Processing Units(GPUs) have become popular as accelerators for scientific computing due to their low cost, impressive floating-point capabilities, and high memory bandwidth. GPU overcome the drawback of CPU communication overhead between CPU over network.

Case 3. In which that only some specific calculation such as evaluation of non-bounded interaction forces are calculate on GPUs. And the bounded forces are evaluation on CPUs.

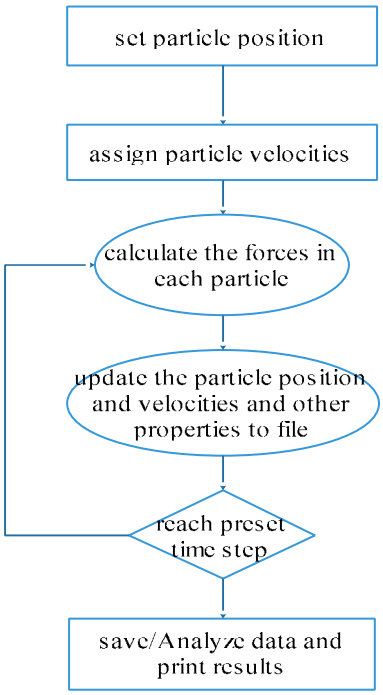

Fig 1: Flow chart of MD Simulation

4. GLOBAL MD ALGORITHM

Step 1. Input Initial Condition: Potential interaction V as a function of atom positions. Position r of all atoms in the system velocities v of all atoms in the system Repeat 2, 3, and 4 for the number of steps.

Step 2. Compute forces: the forces of any atom.

$$F_i = -\frac{dv}{dr_i} \qquad (3)$$

Is computed by calculating the forces between non-bounded atoms

$$F_i = \sum_j F_{ij} \qquad (4)$$

Plus the forces due to bounded interaction (which may depends on 1, 2, 3 or 4 atoms), plus restraining and/or external forces. The potential and kinetic energies and pressure tensor may be computed.

Step 3. Update configuration: The movement of atoms is simulated by numerically solving Newton's equation of motion.

Step 4. Output step: write position, velocities, energies, temperature, pressure etc.

5. CONCLUSION

Long CPU idle time and low CPU utilization were observed in current GPU-accelerated Molecular Dynamics simulation due to load not balanced between CPUs and GPUs. There are number of method are available of load balancing between CPUs and GPUs for Molecular Dynamics simulation. The method was based on the formulation and observation of individual workload and used to achieve dynamical load balance on single and multi-GPU systems to be employed in a molecular dynamic application that exhibit significat load unbalance.

6. REFERENCES

[1] M. Sekijima, C. Motono, S. Yamasaki, K. Kaneko, and Y. Akiyama, "Molecular dynamics simulation of dimeric and monomeric forms of human prion protein: insight into dynamics and properties," Biophysical journal, vol. 85, no. 2, pp. 1176–1185, 2003.

[2] K. Sugawara, S. Saito, M. Sekijima, K. Ohno, Y. Tajima, M. Kroos, A. Reuser, and H. Sakuraba, "Structural modeling of mutant alphaglucosidases resulting in a processing/transport defect in Pompe disease," J Hum Genet, vol. 54, pp. 324–330, 2009.

[3] H. Fujitani, Y. Tanida, M. Ito, G. Jayachandran, C. D. Snow, M. R. Shirts, E. J. Sorin, and V. S. Pande, "Direct calculation of the binding free energies of FKBP ligands," The Journal of chemical physics, vol. 123, p. 084108, 2005. [4] P. L. Freddolino, F. Liu, M. Gruebele, and K. Schulten, "Tenmicrosecond molecular dynamics simulation of a fast-folding ww domain," Biophysical journal, vol. 94, no. 10, p. L75, 2008 Tavel, P. 2007 Modeling and Simulation Design. AK Peters Ltd.

[4] T. Udagawa and M. Sekijima, "Energy Consumption of GPU with Molecular Dynamics," in Proceedings of the 22th IASTED International Conference on Parallel and Distributed Computing and Systems (PDCS 2010). ACTA Press, 2010, pp. 40–44.

[5] S. Du, T. Udagawa, T. Endo, and M. Sekijima, "Molecular Dynamics Simulation of a Biomolecule with High Speed, Low Power and Accuracy Using GPU-Accelerated TSUBAME2.0 Supercomputer," in Proceedings of the Asia-Pacific Signal and Information Processing Association Annual Summit and Conference 2011 (APSIPA ASC2011), 2011, pp. 1–5.

[6] T. Udagawa and M. Sekijima, "The power efficiency of GPUs in multi nodes environment with molecular dynamics," in Proceedings of The 2011 International Conference on Parallel Processing Workshops. IEEE, 2011, pp. 399–405.

[7] D. Case, T. Darden, C. S. T.E. Cheatham, III, J. Wang, R. Duke, R. Luo, R. Walker, W. Zhang, K. Merz, B. Roberts, S. Hayik, A. Roitberg, G. Seabra, J. Swails, A. Goetz, I. Kolossv´ary, K. Wong, F. Paesani, J. Vanicek, R. Wolf, J. Liu, X. Wu, S. Brozell, T. Steinbrecher, H. Gohlke, Q. Cai, X. Ye, J. Wang, M.-J. Hsieh, G. Cui, D. Roe, D. Mathews, M. Seetin, R. Salomon-Ferrer, C. Sagui, V. Babin, T. Luchko, S. Gusarov, A. Kovalenko, and P. Kollman, "Amber12," 2012, university of California, San Francisco.

[8] B. Hess, C. Kutzner, D. van der Spoel, and E. Lindahl, "GROMACS 4: Algorithms for highly efficient, load-balanced, and scalable molecular simulation," Journal of chemical theory and computation, vol. 4, no. 3, pp. 435–447, 2008.

[9] J. C. Phillips, R. Braun, W. Wang, J. Gumbart, E. Tajkhorshid, E. Villa, C. Chipot, R. D. Skeel, L. Kal´e, and K. Schulten, "Scalable molecular dynamics with NAMD," Journal of Computational Chemistry, vol. 26, no. 16, pp. 1781–1802, Dec. 2005.

[10] S. Plimpton, "Fast parallel algorithms for short-range molecular dynamics," Journal of Computational Physics, vol. 117, no. 1, pp. 1 – 19, 1995.

[11] C. Augonnet, S. Thibault, R. Namyst, and P. Wacrenier, "StarPU: A unified platform for task scheduling on heterogeneous multicore architectures," Concurrency and Computation: Practice and Experience, vol. 23, no. 2, pp. 187–198, 2011.

[12] E. Agullo, B. Bramas, O. Coulaud, E. Darve, M. Messner, and T. Takahashi, "Pipelining the Fast Multipole Method over a Runtime System," CoRR, vol. abs/1206.0115, 2012.

[13] F. Song and J. Dongarra, "Enabling and Scaling Matrix Computations on Heterogeneous Multi-Core and Multi-GPU Systems," in Proceedings of the 26th ACM international conference on Supercomputing. ACM, 2012, pp. 365–376

[14] W. M. Brown, P. Wang, S. J. Plimpton, and A. N. Tharrington, "Implementing molecular dynamics on hybrid high performance computers - short range forces," Computer Physics Communications, vol. 182, no. 4, pp. 898 – 911, 2011.

[15] W. M. Brown, A. Kohlmeyer, S. J. Plimpton, and A. N. Tharrington, "Implementing molecular dynamics on hybrid high performance computers - particle-particle particle-mesh," Computer Physics Communications, vol. 183, no. 3, pp. 449 – 459, 2012.

[16] Takuro Udagawa and Masakazu Sekijima, "GPU Accelerated Molecular Dynamics with Method of Heterogeneous Load Balancing," 2015 IEEE International Parallel and Distributed Processing Symposium Workshops