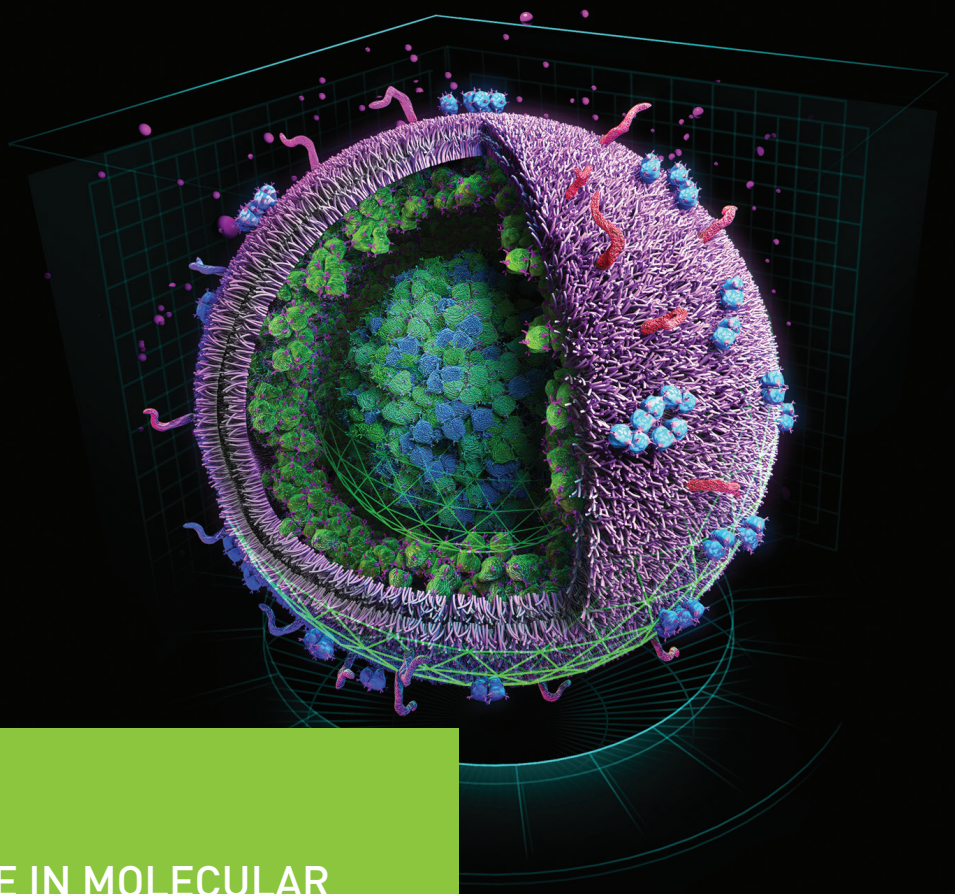

GPU USAGE IN MOLECULAR DYNAMICS TRAJECTORY ANALYSIS

CASE STUDY
IIT DELHI

EXECUTIVE SUMMARY

INTRODUCTION

This case study shows the advantages of using GPU's in the compute intensive field of molecular simulations during the analysis of resultant trajectories. The problem analyzes the probability of finding three particles at a certain distance by measuring the multi-particle distribution function.

CHALLENGE

The core issue that the research team faced was the need to compute complex calculations from a large volume of data in a timely fashion. Common algorithms used for analyzing molecular dynamics trajectories traditionally run as serial codes which the existing CPU based compute infrastructure was taking a day or more for each simulation. With better availability of compute power and availability of off-the-shelf package softwares accelerated using GPU's, the data size has increased to a point where analysis part needs to be parallelized as well in order to gain overall performance. In such situations, the serial analysis could take much longer than the simulation itself.

SOLUTION

The research team ported their algorithm to CUDA® and addressed the compute issue by using NVIDIA® GPUs, eventually scaling to an NVIDIA® Tesla® K20, with the objective of migrating to Tesla® K40. Also, since the problem is embarrassingly parallel, multiple GPU's in the same node – or across nodes can be used as well.

OUTCOME

The result was successful completion of calculation computation with an output of results, as well as reduction of simulation time to hours and hence faster and successful completion of the project. The conclusion was that GPUs, with their parallel computing architecture, accelerate applications and enhance computation speed even when handling a large volume of data, and when used in conjunction with a GPU programming language such as CUDA. The basic CUDA code gives a 30X scale up as compared to the serial code whereas with using multi-gpu with 2 cards the scalability was as high as 60x over a single serial process.

Image for representation purpose only

DETAILED DESCRIPTION

CALCULATION OF TRIPLET CORRELATION FUNCTIONS USING CUDA®

Gourav Shrivastav, Manish Agarwal and Charusita Chakravarty
Department of Chemistry, Indian Institute of Technology Delhi, India
July 1, 2014

INTRODUCTION

Using Graphical Processing Units (GPUs) for conventional computation is very common today, with many applications implemented on CUDA (Common Unified Device Architecture), including algorithms used in molecular dynamics, quantum chemistry, physics, bioinformatics, etc. A large number of traditional software suites have added GPUs as accelerators, and packages such as LAMMPS, GROMACS, NAMD and AMBER showing 3-5x improvement in speeds "off the shelf". With simulation sizes growing, the data generated and the required calculations which follow have grown tremendously. As a result, common algorithms to analyse molecular dynamics trajectories, traditionally run as serial codes need to be ported onto parallel architectures. Since these codes are embarrassingly parallel, CUDA on GPUs is a logical choice. This case study deals with analysis of data from molecular dynamics simulations of large systems.

Among the many static and dynamical properties which are calculated from an MD trajectory is the radial distribution function, or the pair correlation function. The property measures the probability of finding two particles at a certain distance. This code has been successfully parallelized and ported onto CUDA with a very high efficiency [1]. Among the properties which can be obtained from the RDF is the translational entropy or the "pair" entropy. As the name suggests, the pair entropy of a system measures that part of the entropy of the system which originates due to pairs. Similarly, the triplet entropy, i.e. entropy contribution due to triads[2] can be computed.

CHALLENGE

Calculation of triplet correlation functions involves selection of all triads of particles which obey certain distance constraints. This yields the problem into a computation which grows as N3. Calculation of triplet entropy additionally needs the radial distribution function (RDF), which is of the order of N2. In this case study, we use the RDF computed on the GPU to further accelerate the triplet calculation. The input consists a trajectory of "ncon" frames, each of "N" particles. All frames and all particles are independent - and hence calculation of both RDF and the triplet histogram can be parallelized. The output would consist of (a) the RDF histogram (1 D integer array) (b) the triplet histogram (3 D integer array). The serial FORTRAN code takes approximately 24 hours for N = 4096 atoms and ncon = 5000 configurations

SOLUTION

CUDA Kernels
There are two CUDA Kernels involved in the calculation - (a) calculates the pair correlations, and pre-processes the distance information for the triplet correlations, and (b) calculates the triplet correlations using the information from the (a).

Pair Correlation calculation
In this kernel, "pair gpu", all pairs within a distance rcut are counted and the histogram is updated. Each instance of the kernel processes a single atom i. Additionally, the neighbor list of the atom i is stored in the form of vector data for use by the next kernel. This kernel takes a fraction of the overall triplet calculation. e.g. for N = 4096 this kernel takes < 0.5s.

Triplet correlation calculation
In this case, kernel grids are spawned for each atom i. This is done either sequentially, or via streams. Each instance of the kernel works on an i-j pair and updates the 3D histogram by looping over the third atom k. Both j and k are picked from the neighbor list generated in the pair gpu kernel.

The data obtained from this CUDA-C kernel is passed on to the original FORTRAN code for further processing. This processing is relatively fast, and is independent of the number of frames.

OUTCOME

For a system of 4096 particles, for 5000 configurations, time taken by the serial program is 17-18 hours on the "best" machine available (3.0 GHz 8GB RAM). In comparison, the CUDA code takes 30 minutes on a single K20 card. The code has also been modified to use 2 GPU cards simultaneously. This cuts down the time to ~18 minutes. The basic CUDA code gives a ~30x scale up, as compared to the serial code. With 2 cards, the scale up is ~60x. Currently, the code is being actively used for calculations of three particle entropies of alkanes and other systems of interest in the group. The system size for alkanes is 6720 carbon atoms. The CUDA code takes 2-3 hours for the calculation; the serial code is predicted to take ~60 hours.

CONCLUSION AND FUTURE SCOPE

Since the problem itself is embarrassingly parallel, bigger systems with large number of frames can be distributed across multiple GPUs with the help of MPI. We are in the process of optimizing the code and expanding it to be run over multiple nodes. The study can also be extended to analysis of data obtained from confocal microscopy experiments. Higher order multi-particle correlation functions can also be potentially investigated using this approach.

References

[1] Levine, B. G., Stone, J. E., and Kohlmeyer, A. J. Comput. Phys. 230(9), 3556–3569 May (2011).

[2] Singh, M., Dhabal, D., Nguyen, A. H., Molinero, V., and Chakravarty, C. Phys. Rev. Lett. 112(14), 147801 April (2014).

NVIDIA