

Estimating Molecular Dynamics Chemical Shift with GPUs

Eric Wright
Computer and Information
Sciences
University of Delaware
Newark DE, USA
efwright@udel.edu

Mauricio Ferrato
Computer and Information
Sciences
University of Delaware
Newark DE, USA
mferrato@udel.edu

ABSTRACT

Experimental chemical shifts (CS) from solution and solid state magic-angle-spinning nuclear magnetic resonance spectra provide atomic level data for each amino acid within a protein or complex. However, structure determination of large complexes and assemblies based on NMR data alone remains challenging due to the complexity of the calculations. Here, we present a hardware accelerated strategy for the estimation of NMR chemical-shifts of large macromolecular complexes. We demonstrate the feasibility of our approach in systems of increasing complexity ranging from 2,000 to 11,000,000 atoms.

CSS CONCEPTS

Parallel Computing Technologies

KEYWORDS

Molecular Dynamics, Programming Model, OpenACC, GPUs

BACKGROUND

Nuclear magnetic resonance (NMR) is a practice long-cherished in the fields of biochemistry, biophysics and structural biology. Chemical shift, the principle observable in NMR instrumentation, provides valuable insight into protein secondary structure by allowing inference about conformation to be drawn based on peak shift. Measured in parts per million (ppm), chemical shift is a metric that describes the resonant frequency of a nucleus by comparing its observed frequency to that of a standard in a magnetic field. In protein studies, namely structural determination or refinement, chemical shift is poised as one of the most powerful stores of information, quantifying the aberrational electrostatics that result from organized protein structure. The utility of chemical shift in biomolecular research, however, is not limited to spectroscopic strategies.

NMR-based software solutions have materialized into a rich domain of protein study and protein chemical shifts have been used in varying ways to successfully elucidate structure. Commonly, these programs employ perusal of NMR databases to establish and parse relationships between shifts, sequence and structure. In addition to standalone software designated for structural assignment, NMR data has been used to refine molecular dynamics, having been used directly to optimize backbone torsion angle sampling in the CHARMM36 protein

forcefield, improving the quality of protein dynamics in studies of ranging scales. Given the multitude of software and packages to meaningfully infer structure from protein chemical shifts and refine dynamic structural modeling, predicting chemical shift based on coordinate information is naturally recognized for its potential.

Applying NMR data as reference rather than convolving sequence homology assignment, the quantum chemical realizations of chemical shifts' dependence on local structure have been distilled to relatively low computing cost algorithmic prediction model and implemented in a software named PPM. The aim of PPM was to develop a prediction model that could operate over large systems and conformational ensembles, providing new dimensions of protein forcefield refinement, structural refinement, and ensemble validation.

In a departure from ensemble analysis, PPM's successor PPM_One introduced a static-structure based chemical shift prediction method that showed competitive accuracy with empirical and semi-empirical competitors.

Drawing from the lightweight approximations of first principle calculations and coupled with accessible protein and NMR data, PPM_One considers chemical shift as a sum of discrete descriptors.

These descriptors; which quantify chemical shifts due to ring current effects, hydrogen bond effects, dihedral, and more, take the form of relatively simple, and differentiable, functions of atomic-coordinate data.

Considering these factors, PPM_One is a prime target for parallelization and optimization, to enable its application in the study of largescale biomolecular study. However, the original PPM_One code was not written in a way to exploit massive compute power of accelerators such as GPUs. In our work, we refactor and optimize the sequential code thus preparing the code to benefit from massively parallel processors. We have parallelized and accelerated this code using OpenACC, a directives-based parallel programming model, that enables programmers to create a parallel code that is not only portable across different types of platforms but also increases the speed of the code by a large magnitude.

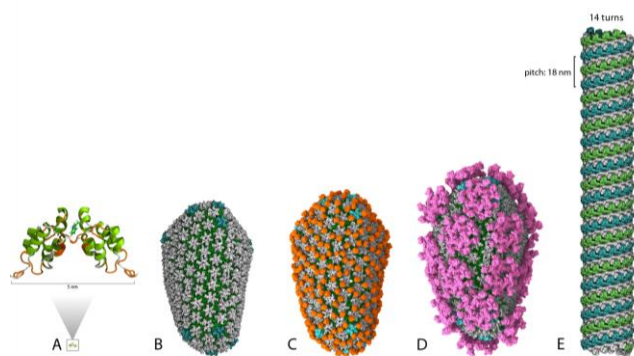


Figure 1: 3D representation of pdb inputs we used in PPM_One code. Protein (A) is a single HIV dimer, consisting of ~2,000 atoms (the image is blown up, and rendered to scale below). Protein (B) is a full HIV capsid composed of thousands of dimers and contains ~2.1 million atoms. Protein (C) and (D) are variations of B, with up to 5 million atoms. Protein (E) is a molecular motor, and we have large versions of it that contain over 10 million atoms

OpenACC, since its inception in November 2011, is being widely used to port large scale applications such as ANSYS,

GAUSSIAN, Icosahedral non-hydrostatic, smoke propagation, transport code to massively parallel heterogeneous architectures. All industry codes that run on TITAN at ORNL utilizing GPUs are using OpenACC.

METHODS

Starting with the PPM_One code, we compiled it with the PGI OpenACC enabled compiler. Then, we ran it through the PGPROF profiler to identify computational hotspots and code performance. Using this information, we began prepping the code to be ported to GPU accelerators. While doing this, we also found a few portions of the code that would benefit from some performance optimization. After applying these tweaks, we were looking at a code that was already running up to 5x faster depending on the dataset.

Next, we used OpenACC directives to parallelize the code, and tested it on several different GPUs. We used the results we were seeing from our test runs, and additional information from reprofiling the code, and fine-tuned the code until we were happy with it. To capture our final results, we were granted access to the NVIDIA PSG cluster and gathered runtimes using both Pascal P100 and Volta V100 nodes.

Results

The best result we achieved was a 67x speedup when comparing the single core performance of the code versus a single Volta V100 GPU. This is not considering the overall speedup over the original, unoptimized code. Now, using GPU acceleration, a code

that was taking upwards of 2.5 hours on large datasets can be completed in just over 2 minutes. However, when comparing the performance of our multicore CPU implementation versus the Volta V100 GPU, we are only seeing ~3.4x speedup, which is far less than ideal.

	Very Small (100K) Atoms	Medium (2.1M) Atoms	Large (6.8M) Atoms	Very Large (11M) Atoms
Serial (Unoptimized)	167.11s	3547.07 (1 hour)	7 hours approx.	14 hours approx.
Serial (Optimized)	32s	2209.64s (37 min)	2939s (48 min)	9035s (2.5 hours)
Multicore (32 cores)	2.93s	109s	172s	427s
NVIDIA PASCAL P100 GPU	1.72s	36s	69s	170s
NVIDIA VOLTA V100 GPU	1.68s	29s	56s	134s

Figure 2: Table containing various runtimes of different datasets on different architectures. All results using the Intel Xeon E5-2698 (32 cores) CPU and single GPU, using the PGI 18.4 compiler and CentOS 7.5 operating system.

We have also experienced some significant limitations with this project. At the beginning of the PPM_One code is a very substantial data pre-processing step. The raw data will be read from the PDB (protein databank) files and organized into several different array structures. As an example, the pre-processing step for our Very Large dataset takes ~110 seconds. This is many multitudes faster than it was in the unoptimized code, but it is still the main limiting factor of our performance. If we were able to disregard this pre-processing step we would see that our multicore implementation runtime is ~317 seconds and our Volta V100 GPU runtime is ~24s, resulting in ~13x speedup.

There is still room for improvement for the pre-processing step, but we feel that to optimize anymore would require a major rewrite of a large portion of the code, which we feel is not within the scope of this project.

ACKNOWLEDGMENTS

We gratefully acknowledge the support of NVIDIA PSG Cluster with the donation of the P100 and V100 GPUs used for this research. We also gratefully acknowledge Prof. Andy Novocin at the University of Delaware as the work was partially done under Vertically Integrated Project (VIP).

REFERENCES

- [1] XP, X., and D. Case, 2001. Automated prediction of ^{15}N , $^{13}\text{C}\alpha$, $^{13}\text{C}\beta$, and $^{13}\text{C}'$ chemical shifts in proteins using a density functional database. *Journal of Biomolecular NMR* 321–333.
- [2] Seavey, B. R., E. A. Farr, W. M. Westler, and J. L. Markley, 1991. A relational database for sequence-specific protein NMR data. *Journal of Biomolecular NMR* 1:217–236. <https://doi.org/10.1007/BF01875516>.
- [3] Shen, Y., and A. Bax, 2007. Protein backbone chemical shifts predicted from searching a database for torsion angle and sequence homology. *Journal of Biomolecular NMR* 38:289–302. <https://doi.org/10.1007/s10858-007-9166-6>.
- [4] Kohlhoff, K. J., P. Robustelli, A. Cavalli, X. Salvatella, and M. Vendruscolo, 2009. Fast and Accurate Predictions of Protein NMR Chemical Shifts from Interatomic Distances. *Journal of the American Chemical Society* 131:13894–13895. <https://doi.org/10.1021/ja903772t>, PMID: 19739624.

- [5] Jing, H., and M. A. D. CHARMM36 allatom additive protein force field: Validation based on comparison to NMR data. *Journal of Computational Chemistry* 34:2135–2145. <https://onlinelibrary.wiley.com/doi/abs/10.1002/jcc.23354>.
- [6] Li, D.-W., and R. Brüschweiler, 2012. PPM: a side-chain and backbone chemical shift predictor for the assessment of protein conformational ensembles. *Journal of Biomolecular NMR* 54:257–265. <https://doi.org/10.1007/s10858-012-9668-8>.
- [7] Li, D., and R. Brüschweiler, 2015. PPM_One: a static protein structure based chemical shift predictor. *Journal of Biomolecular NMR* 62:403–409. <https://doi.org/10.1007/s10858-015-9958-z>.
- [8] Sathe, S., 2016. Accelerating the ANSYS Fluent R18.0 Radiation Solver with OpenACC. <http://ondemand.gputechconf.com/supercomputing/2016/presentation/sc6126-sunil-sathe-accelerating-ansys-fluent-radiation-solver-openacc.pdf>.
- [9] Gomperts, R., 2016. Quantum Chemistry on GPUs. <http://images.nvidia.com/content/tesla/pdf/quantum-chemistry-may-2016-mb-slides.pdf>.
- [10] Sawyer, W., G. Zaengl, and L. Linardakis, 2014. Towards a multi-node OpenACC Implementation of the ICON Model. *In* EGU General Assembly Conference Abstracts. volume 16.
- [11] Küsters, A., S. Wienke, and L. Arnold, 2017. Performance Portability Analysis for Real-Time Simulations of Smoke Propagation Using OpenACC. *In* International Conference on High Performance Computing. Springer, 477–495.
- [12] Searles, R., S. Chandrasekaran, W. Joubert, and O. Hernandez, 2018. Abstractions and Directives for Adapting Wavefront Algorithms to Future Architectures. *In* 5th International