

A. Artifact Description Appendix: [Paper Title] Submission and reviewing guidelines and methodology:

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.

Description:

This project is based from an application called PPM_One developed by Ohio State University (<http://spin.ccic.ohio-state.edu/index.php/ppm/document>). The only substantial changes that we made were porting the code for GPU acceleration and implementing a few general performance optimizations over the base code. Our revision of PPM_One has all the same uses as the original code, and you should expect it to return the same results as the original as well.

Availability:

The code can be found on the University of Delaware CRPL (Computational Research and Programming Lab) github repository: https://github.com/UD-CRPL/ppm_one.

Algorithm:

While porting PPM_One for GPU accelerated execution, the original algorithm was retained. For any algorithm specific information, see the documentation found with the original application documentation.

Datasets:

PPM_One requires as input a single pdb (protein data bank) file. You may find a database of pdb files at <https://www.rcsb.org/>. If you wish to run on one of our large datasets, this is a direct link to a HIV Capsid pdb file with ~2.1 million atoms that we used in the poster. <https://www.rcsb.org/structure/3J3Q>

Hardware:

Our project is designed with OpenACC, meaning that it can be compiled and ran on several different parallel architectures. However, we wrote the code with GPU execution as a priority. The code only uses a single GPU. We have successfully run and verified the code on a few different compute machines:

NVIDIA PSG Cluster:

The NVIDIA Professional Service Group (PSG) Cluster has state of the art computer resources that we used to gather results from. Information regarding the cluster is listed below:

1. P100 Node:

- CPU: Intel Haswell, 16 cores (hyperthreading), dual socket
- GPU: NVIDIA Tesla P100, (16 GB HBM2, Pascal architecture)

2. V100 node:
 1. CPU: Intel Haswell, 16 cores (hyperthreading), dual socket
 2. GPU: NVIDIA Tesla V100, (32 GB HBM2, Volta architecture)

University of Delaware CRPL Internal Machines:

The two internal machines used in the project are part of the Computational Research Programming Lab (CRPL) at the University of Delaware. These are listed below:

- 1) Vader
 - CPU: Intel x990, 6 cores (hyperthreading), dual sockets, 11GB RAM
 - GPUs: NVIDIA Titan V (12GB HBM2, Volta architecture), 2x NVIDIA GTX1080 (8GB GDDR5X, Pascal architecture)
- 2) Yoda
 - CPU: Intel Xeon, dual socket, 24GB RAM
 - GPUs: 4x NVIDIA Tesla K20x (6GB GDDR5, Kepler architecture)

Installation:

We have included compilation instructions with the code in our git repository, along with scripts that can automate the build process using cmake. These scripts also assume you will be using the PGI compiler to build the OpenACC code. You may substitute any other OpenACC enabled compiler, but the code was only verified with PGI versions 17.10, and 18.4.

If you wish to attempt to build without the cmake scripts, you will need to build the common and Immin directories separately before linking them in the root directory.

Evaluation and Expected Results:

The program outputs the estimated chemical shift of a given pdb file. This output represents chemical shift as floating point numbers to the 1e-03 level of precision. You should expect the output to be equivalent to the output generated by the base PPM_One code.

A.4 Experiment workflow

When compiling the code, an executable called *ppm_stable* will be created inside the ppm directory. There are a few different run configurations, but our code changes uses the default configurations. If you use any other options, then the code will not be run on the GPU. The only thing you need to specify when running the executable is which pdb file to use. Below is a sample command line showing us launching the program.

```
$ ./ppm_stable -pdb sample.pdb
```