



Find the code
on github!

MOTIVATION AND GOAL

MOTIVATION

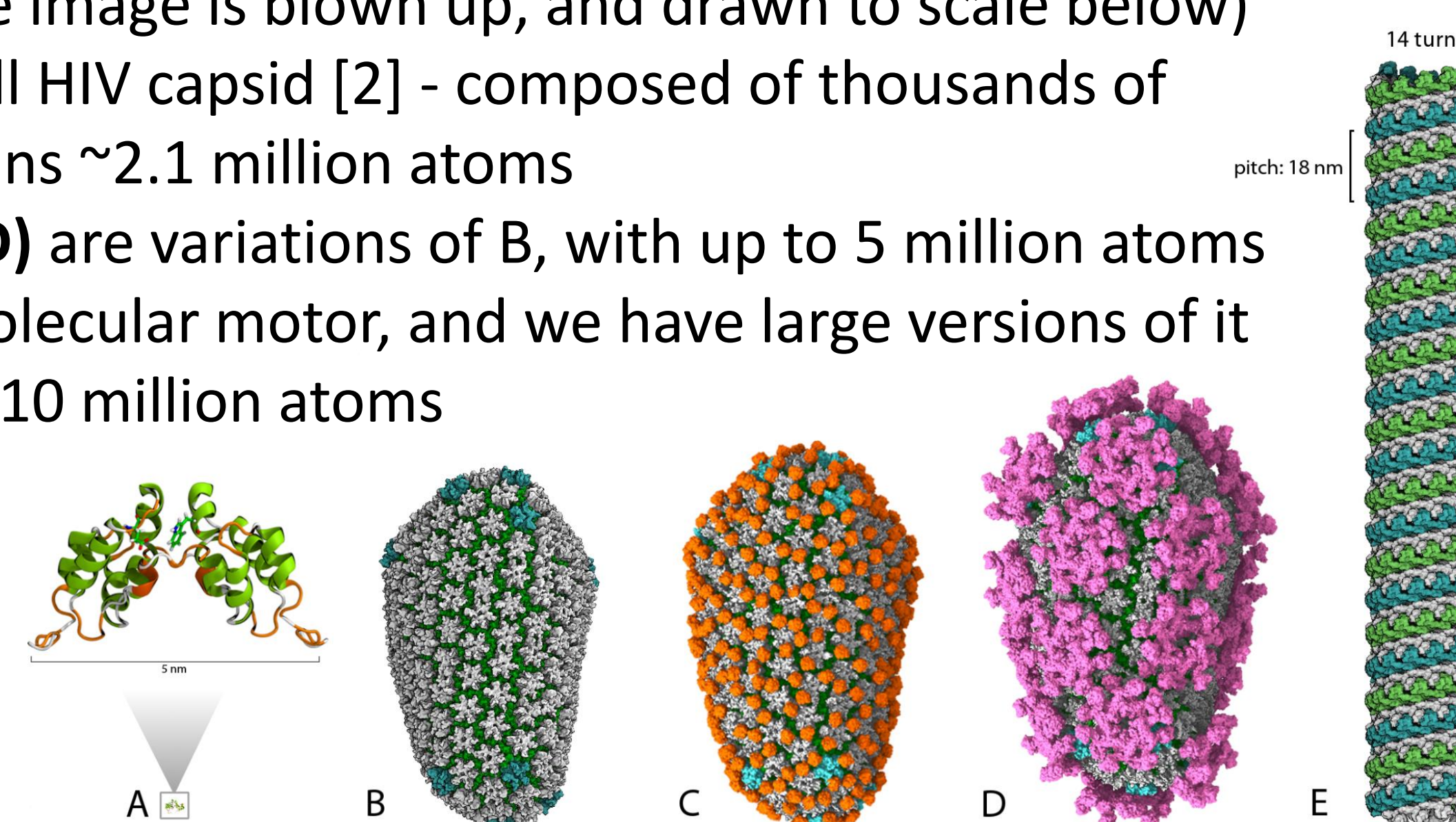
- Chemical shift, a principle observable in Nuclear Magnetic Resonance (NMR) instrumentation provides valuable insight into protein secondary structure
- In structural determination or refinement, chemical shift is poised as one of the most powerful stores of information
- Structure determination of large complexes and assemblies based on NMR data alone remains challenging due calculation complexity
- Chemical shift prediction algorithms have not been previously implemented for accelerated hardware, and computation of very large molecular structures was simply extremely time consuming

GOAL

- Study PPM_One [1], a new empirical knowledge-based chemical shift predictor of protein backbone atoms
- Create a GPU accelerated chemical shift prediction application based on the PPM_One code

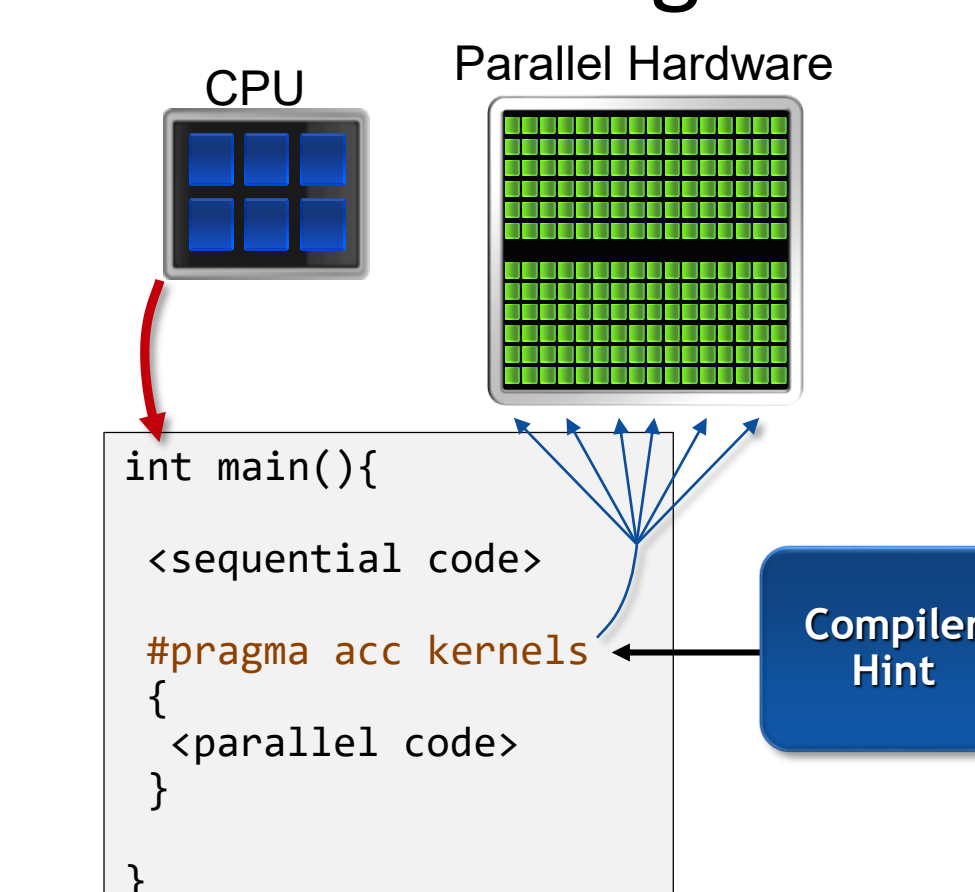
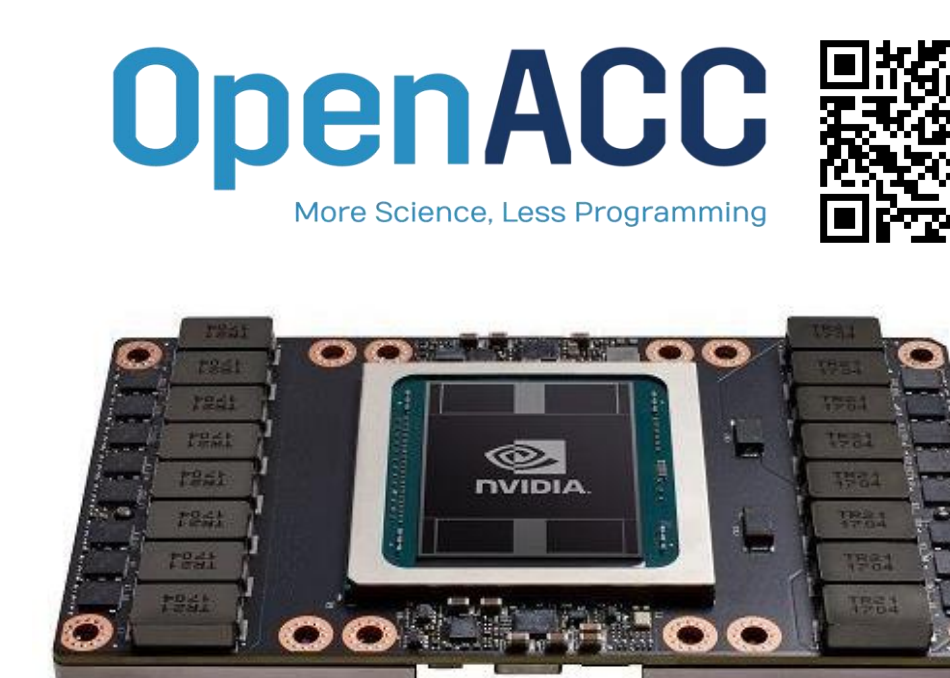
DATASETS

- A single molecular structure (typically a protein)
- Size of the molecule (number of total atoms) affects the application's runtime
- Protein (A)** is a single HIV dimer, consisting of ~2,000 atoms (the image is blown up, and drawn to scale below)
- Protein (B)** is a full HIV capsid [2] - 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



USING OpenACC DIRECTIVES

- OpenACC [3], a directive-based parallel programming model for accelerators
- By using OpenACC directives we are able to offload any major computation to accelerators while maintaining a functioning serial code



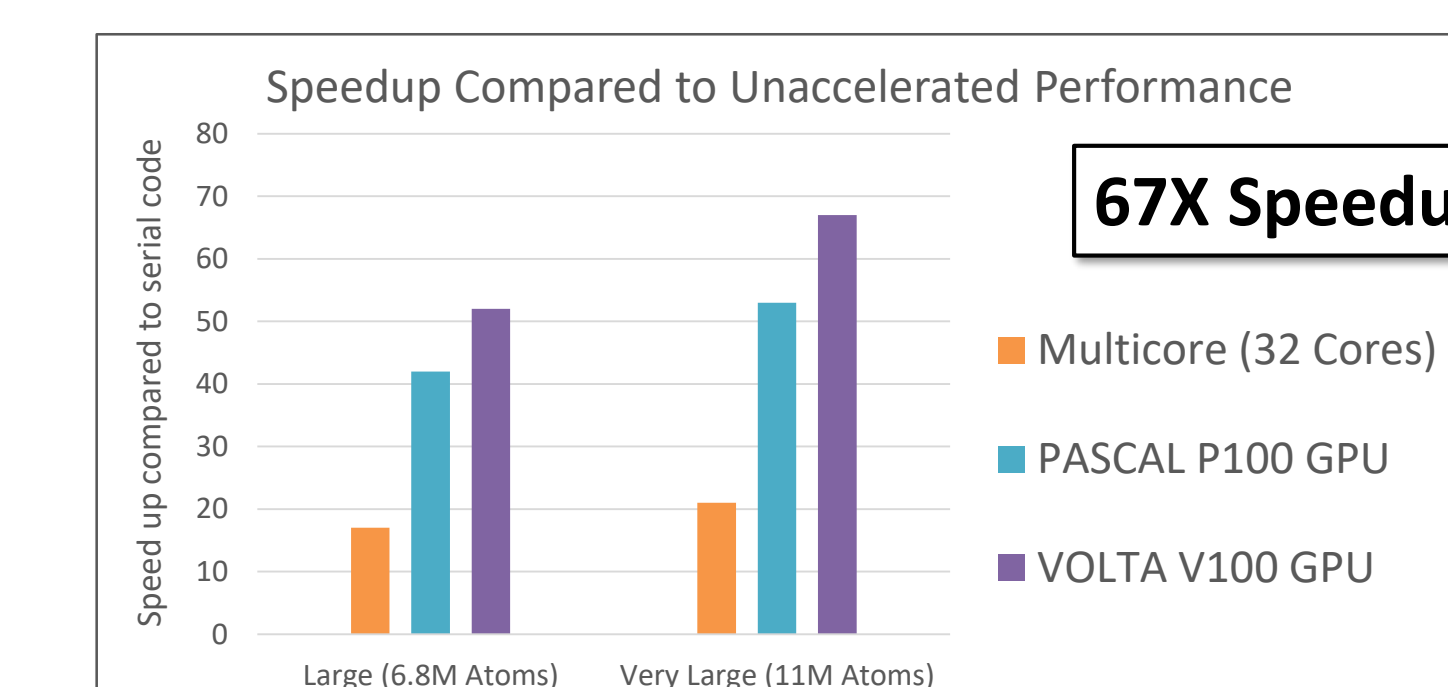
Results

All results using the Intel Xeon E5-2698 (32 cores) CPU and single GPU
 Using PGI 18.4 compiler and CentOS 7.5 OS

	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

- Here we show the absolute runtime of the code with various datasets and GPUs
- We see a huge performance increase when comparing any of our runtimes with the original, unoptimized code
- However, we use the optimized serial performance when measuring accelerated speedup

- When using our largest available dataset and an NVIDIA V100 GPU, we see up a **67x** speedup respectively when compared to single core optimized performance
- OpenACC multicore with a dual socket 32 core CPU is at 21x speedup
- The application originally existed as a serial application only, so the multicore and GPU runtimes are the same code running on different accelerators
- Compared to a fully utilized CPU node (32 cores), the V100 GPU is seeing ~3.4x speedup
- The main limitation we are facing is the extensive data pre-processing step



- Of the 134 seconds of our best runtime, pre-processing takes about 110 seconds of it
- We have made huge improvements to this so far, but we feel that we are reaching the limit of what we can achieve without entirely rewriting the pre-processing
- If we could better optimize this step, we would see a ~13x speedup of V100 GPU over 32-core CPU

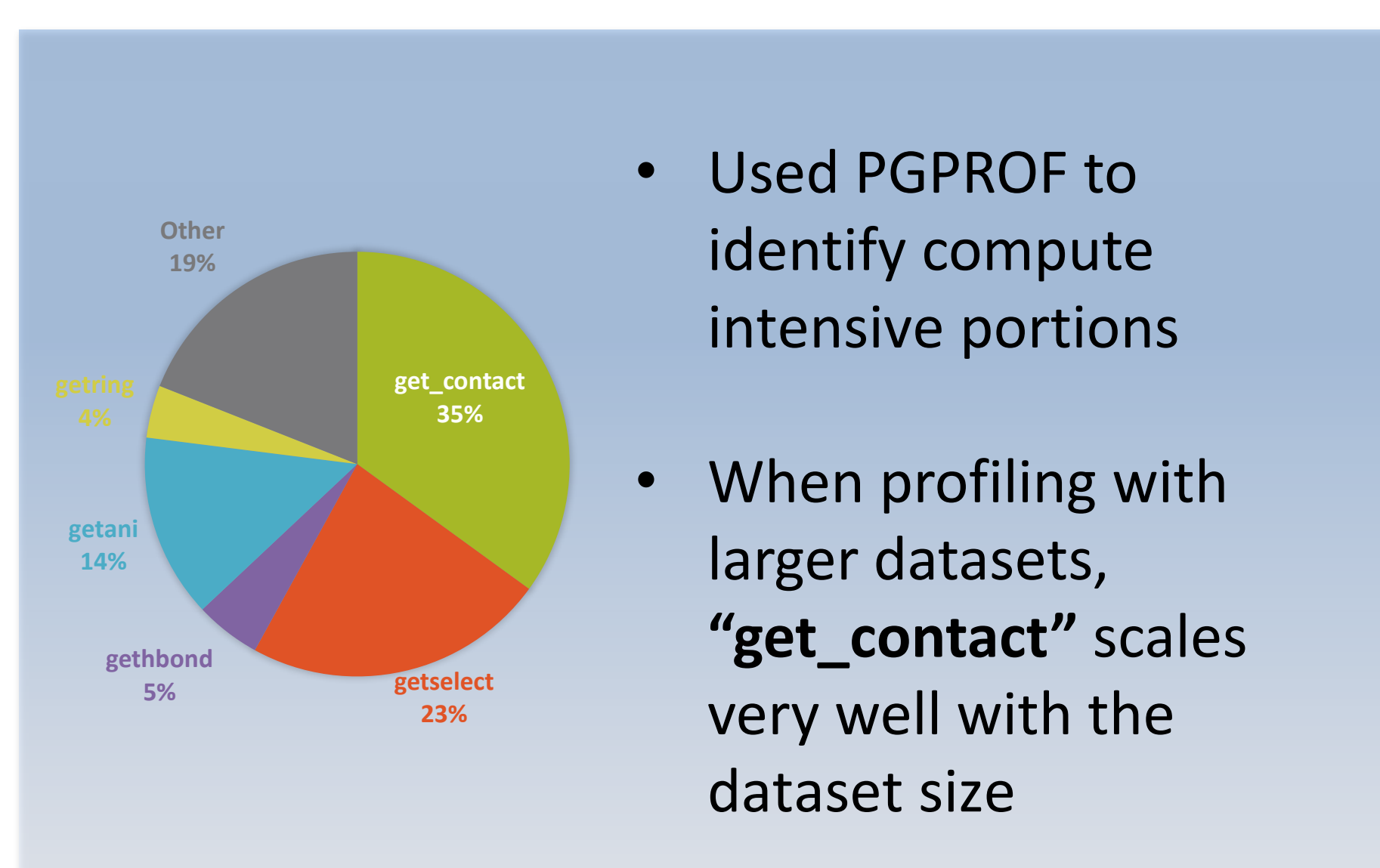
Future Work

- Scale PPM_One across nodes and multiple GPUs using MPI + OpenACC
- Incorporate the GPU accelerated PPM_One chemical shift prediction into
 - NAMD (Nanoscale Molecular Dynamics) enabling protein structure refinement combined with other experimental techniques
 - VMD (Visual Molecular Dynamics) enabling scientists to perform structure validation

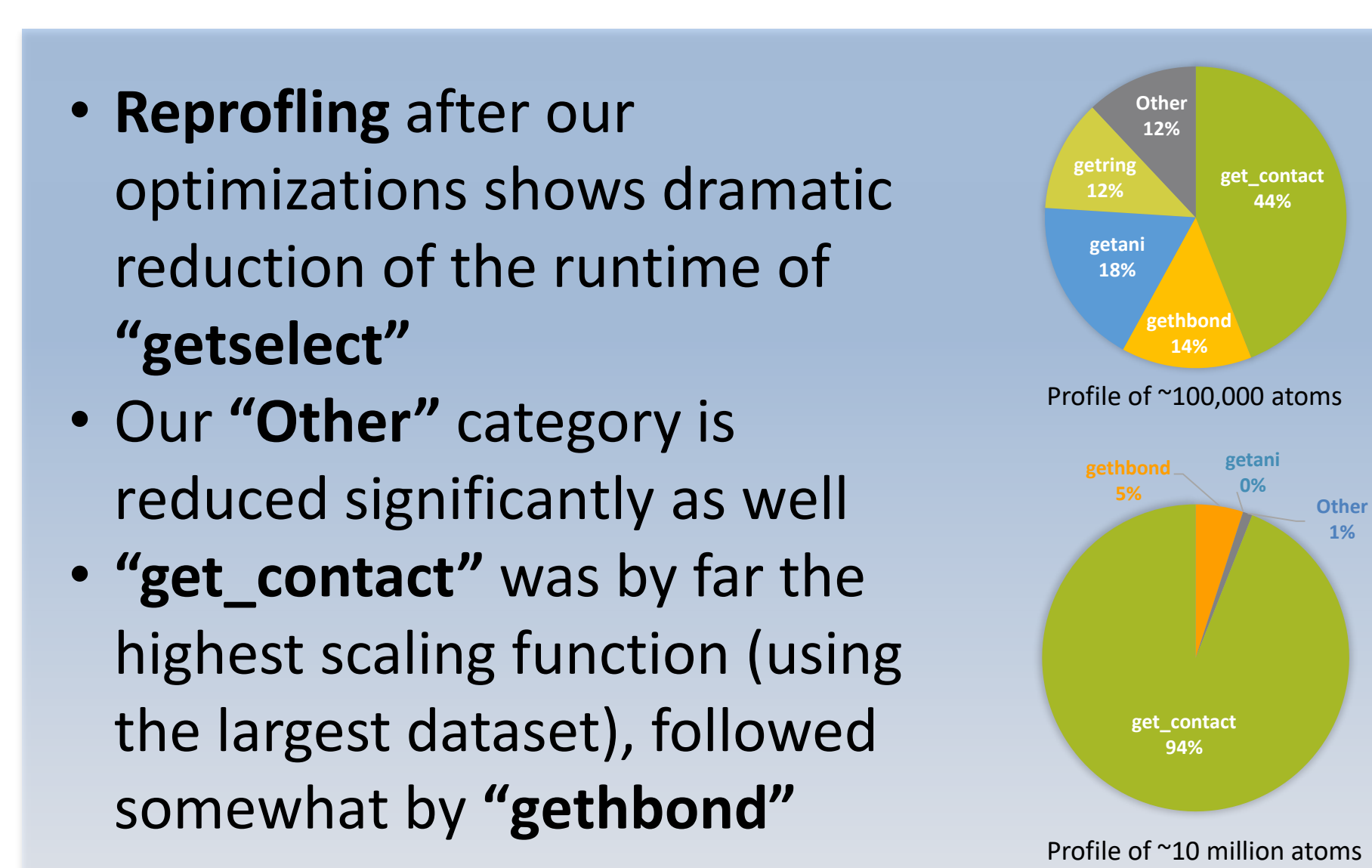
References & Acknowledgements

- Li, D., and R. Brüschweiler, 2015. PPM_One: a static protein structure based chemical shift predictor. Journal of Biomolecular NMR 62:403–409
- Perilla, J.R., Zhao, G., Lu, M., Ning, J., Hou, G., Byeon, I.J.L., Gronenborn, A.M., Polenova, T. and Zhang, P., 2017. CryoEM structure refinement by integrating NMR chemical shifts with molecular dynamics simulations. The Journal of Physical Chemistry B, 121(15), pp.3853–3863.
- <https://www.openacc.org/>

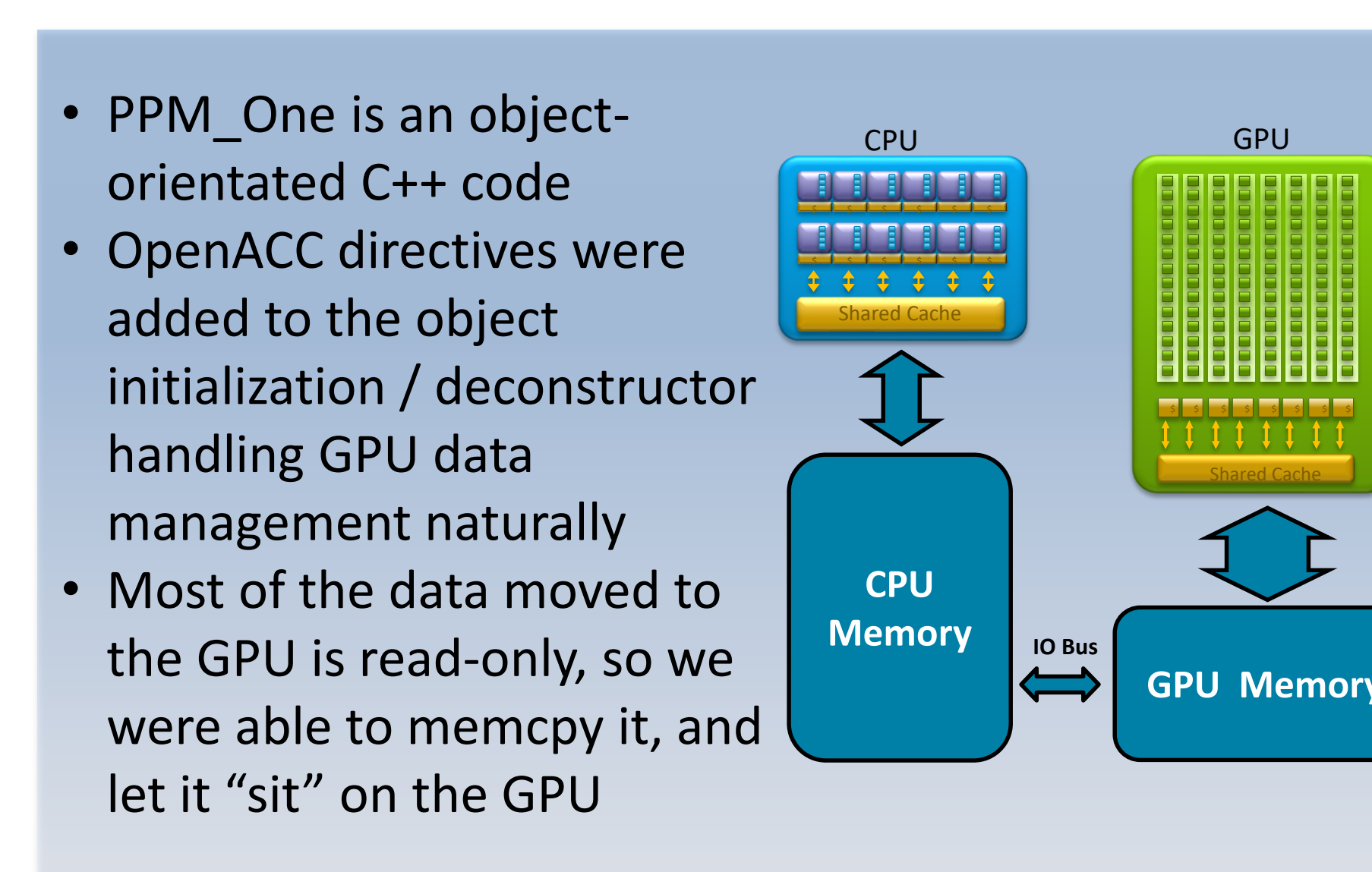
PROJECT ROADMAP



- Used PGPROF to identify compute intensive portions
- When profiling with larger datasets, “get_contact” scales very well with the dataset size



- Reprofiling** after our optimizations shows dramatic reduction of the runtime of “getselect”
- Our “Other” category is reduced significantly as well
- “get_contact” was by far the highest scaling function (using the largest dataset), followed somewhat by “gethond”



- PPM_One is an object-orientated C++ code
- OpenACC directives were added to the object initialization / deconstructor handling GPU data management naturally
- Most of the data moved to the GPU is read-only, so we were able to memcpy it, and let it “sit” on the GPU

Serial Profile

Re-Profile

Data Movement

Serial Optimization

Function Parallelization

```
vector<proton> protons;
vector<double> results;
traj->getani(protons.data(), ..., results.data());
```

STL Vectors

- A lot of data was stored in the STL vector class
- We used the data() method to expose the underlying data pointer, and used that for GPU enabled functions

getselect

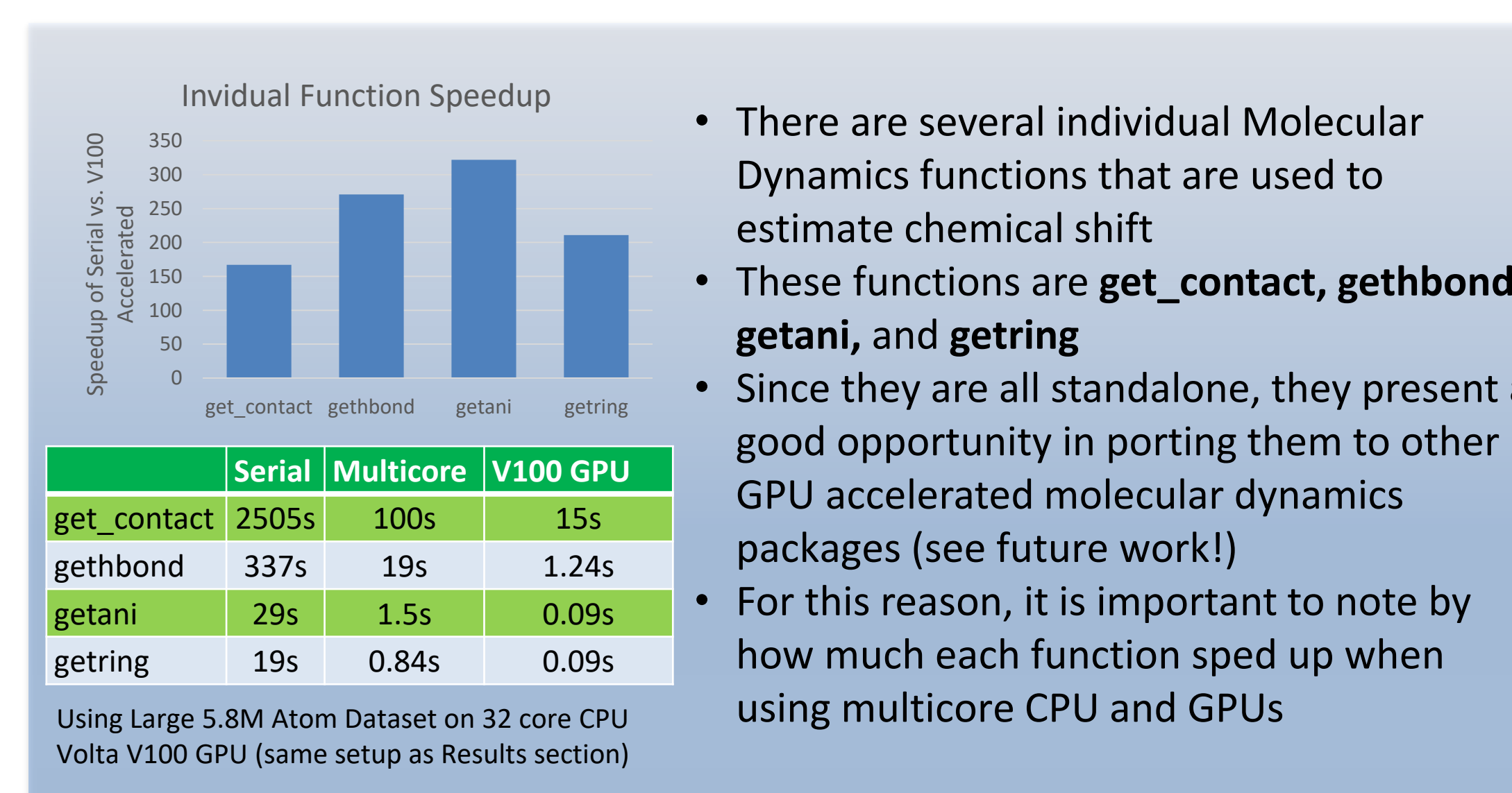
- getselect accounted for a significant portion of our runtime
- After analyzing the code, we found that this was due to getselect being called an unnecessary number of times
- We changed the code to store and reuse the getselect values, and reduced its time taken to <1%

```
c2 = getselect("1-%@allheavy");
for( ... ) { // Large main loop
  // c2 = getselect("1-%@allheavy");
  get_contact(..., c2, ...);
}
```

Filter

- There is a group of functions that filter a list of protons
- The filter function was implemented such that it was taking an unnecessarily long for large data
- In terms of Big O Notation, it was an O(n²) function

- We rewrote these functions to use a better implementation of a C++ filter
- After this change, the function was O(n), and for large datasets was taking over 90% less time



- There are several individual Molecular Dynamics functions that are used to estimate chemical shift
- These functions are **get_contact**, **gethond**, **getani**, and **getring**
- Since they are all standalone, they present a good opportunity in porting them to other GPU accelerated molecular dynamics packages (see future work!)
- For this reason, it is important to note by how much each function sped up when using multicore CPU and GPUs