Chore: Molecular Dynamics Case Study

Parker Diamond, Elliot Greenlee, Jason Liang
Department of Electrical Engineering and Computer Science
University of Tennessee
Knoxville, TN 37996
jdiamon3@vols.utk.edu
egreenle@vols.utk.edu
jliang7@vols.utk.edu

1 Objective

As data generation surges exponentially, the computational requirements of companies and researchers exceed growth. For companies, this means development in cloud based solutions like Amazon Web Services, which provide efficient and scalable tools for any solution. Alternatively, organizations can build dedicated clusters like Anton[8], which can simulate 17,000 nanoseconds of simulation time per day for a 24,000 atom system. However, researchers need minimally expensive or free solutions to match their funding.

In the domain of molecular dynamics for protein folding, Folding@home and CureCoin focus on using the unused computer power of individual users. In the case of Folding@home by Stanford, altruistic individuals volunteer idle time on their computers towards molecular dynamics. The project, launched publicly by Pande Lab in 2000, has explored diseases from malaria to Alzheimer's disease in a directed effort spanning two decades. In 2016 the group reached estimated 100 petaflops of computational power [1]. Curecoin, on the other hand, tries to encourage folding on the Folding@home system using a new cryptocurrency to provide value [4]. Users mine the currency by completing simulations in an idle browser window. Unfortunately, cryptocurrencies have values which fluctuate rapidly, have strong natural limits placed on use beyond peer to peer transactions, and are challenging to understand for the broader public. We propose a paid system of computational bounties.

In this paper, we evaluate molecular dynamics simulations as a job for the system. Subtropical regions in poverty and in close contact with many infectious vectors are at risk to sundry neglected tropical diseases (NTDs). Each year, more than one billion people are affected and billions of dollars are lost in developing economies. The Pathogen Box from Medicines for Malaria Venture (MMV, Switzerland) [2] provides 400 drug-like compounds as seen in Figure 1. We study their aggregation behavior in water, as drugs should distribute in the blood rather than coagulate. We use the metadata fields ID, disease category, and SMILES structure for our work.

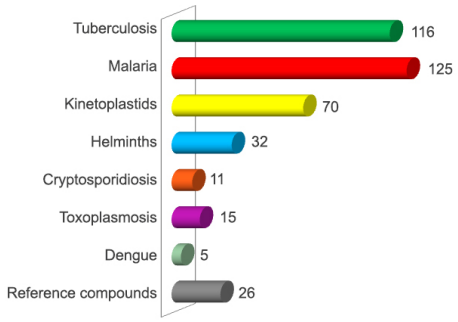

Figure 1: Distribution of molecules for each disease.

2 Approach

The Chore system, visualized in Figure 2 is made up of two components: a server run by a central authority in charge of the system, and an easy to use client run without expert knowledge. The server system is set up in Flask using an SQLAlchemy database to store information about both Users and Jobs. The code is available at `https://github.com/ParkerDiamond/OpenMM-PubSub`.

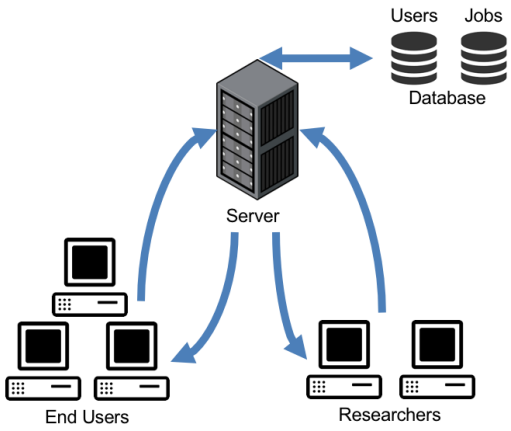

Figure 2: The full Chore system.

Users include both researchers and end users. When a user registers, they send both the details of their payment account and their computer system, both of which are verified by the central authority and the client. At this point, the system creates an ID number, fills in their account value, and adds the user to the database. This information is used differently depending on if the user is a researcher or and end user.

First, a researcher submits a job by sending the job files and code, the time it takes to run the job, and how much money to spend on the job. At this point, the system creates an ID Number and calculates the standard job time by multiplying the researcher computer rating by the time it takes to run the job, before adding the job to the database. The client keeps track of the researchers specific jobs. The system also taxes the job at a constant rate, and distributes the money equally to all other jobs. See Figure 3 for an example of two system states, one previous to and one after a researcher submits two jobs for a total of $500, which is being taxed at 20% and distributed to the rest of the already existing jobs.

Job ID	Value	Hours
Dengue 1	$10 per hour	10
Dengue 2	$10 per hour	10
Malaria 1	$0 per hour	10
Malaria 2	$0 per hour	10
Malaria 3	$0 per hour	10

Job ID	Value	Hours
Dengue 1	$12 per hour	10
Dengue 2	$12 per hour	10
Tuberculosis 1	$10 per hour	20
Tuberculosis 2	$10 per hour	20
Malaria 1	$2 per hour	10
Malaria 2	$2 per hour	10
Malaria 3	$2 per hour	10

Figure 3: Example states of the database before and after two new jobs are submitted.

Next, an end user requests a job. The highest paying job is sent to the user, along with the appropriate files. At this point, the files are installed and run by the client, before sending back the results files generated. When the results files are sent back, they are stored with the job, and the end user is payed. At this point, the researcher can request their files and the job is deleted. The number of completed jobs for each end user, and the number of submitted jobs for each researcher, is tracked as a measure of trustworthiness.

There are several security issues that must be considered when essentially outsourcing computations like these. If an adversary wants to obtain the payout of a job without computing the actual simulation tasks, he/she can simply submit wrong results to the central authority. Solutions include recomputing the simulation data by other clients and keeping track of user reputation. If in the future these computations contain sensitive data, a mechanism to keep the job hidden from the client is needed. Encrypting the simulation inputs and performing operations under a partially homomorphic encryption scheme could preserve confidentiality but is computationally expensive. If a researcher submits a computationally infeasible or malicious job, the central authority should detect and remove the job. Employing heuristics to predict job difficulty and testing example jobs, as well as a similar reputation system for researchers are all possible solutions.

3 Methods

Before any real computation, the raw metadata of the Pathogen Box Database is filtered and a molecule string in Simplified Molecular Input Line Entry Specification (SMILES) format that is labeled with the Pathogen Box ID is extracted. SMILES strings are basic line input that include connectivity of atoms but no coordinated positions. A simple example to consider would be Ethanol, C_2H_6O which is represented in SMILES format as CCO. The SMILES format molecule is converted using an online Protein Data Bank (PDB) file creation tool from the National Cancer Institute [5]. The PDB format is a fixed width specification of atom coordinates in angstroms along connectivity bonds [6]. The online tool automatically chooses an orientation and placement for the atoms, making guesses at the relative orientation of certain bonds. The general flow for this process is summarized in Figure 4.

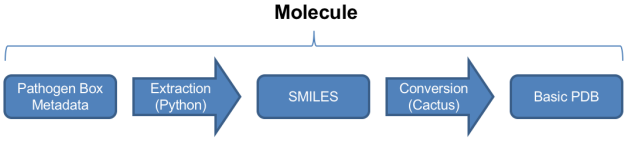

Figure 4: The data flow path for preparing a drug molecule.

In order to adjust the PDB files for this project, we then replicate each drug molecule multiple times a specified number of angstroms apart in a cubic grid and clean up the PDB files using appropriate labels for each residue. We use the Amber tools package in order to create an appropriate force field for simulation in the form of parameter/topology (prmtop) and coordinate (inpcrd) files [3]. Using solvatebox and a standard water model (water.tip3p), we fill the remaining space with water

molecules. Finally the antechamber function converts everything to the correct format for simulation. The general flow for this process is summarized in Figure 5.

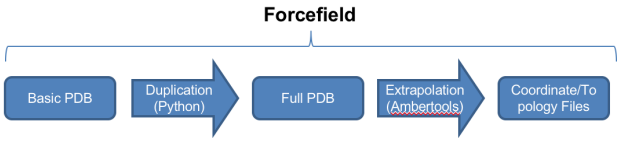

Figure 5: The data flow path for preparing the forcefield.

Each drug molecule in water solvent is then simulated using OpenMM, part of the Omnia Suite. For this project Python is used. Using the standard settings, we simulate the behavior of our systems over small timescales while recording atom positions and PDB information. This data can be used to determine if the molecules aggregate in water. The general flow for this process is summarized in Figure 6.

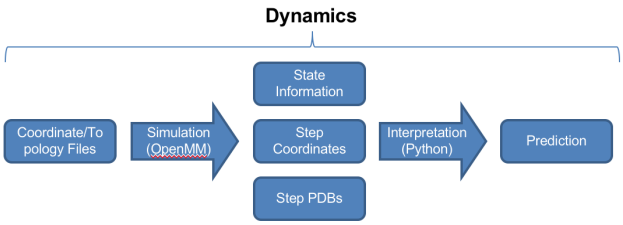

Figure 6: The data flow path for providing a prediction.

In order to determine if a drug molecule coagulated in a given simulation, the radial distribution function is used. The radial distribution function characterizes the density of a system relative to an arbitrary specified particle, and is the probability of finding a particle a given distance from that reference particle. This is calculated by computing the distances between all pairs of particles and binning them into a histogram, as in figure 7. This histogram is typically normalized relative to an ideal gas [7].

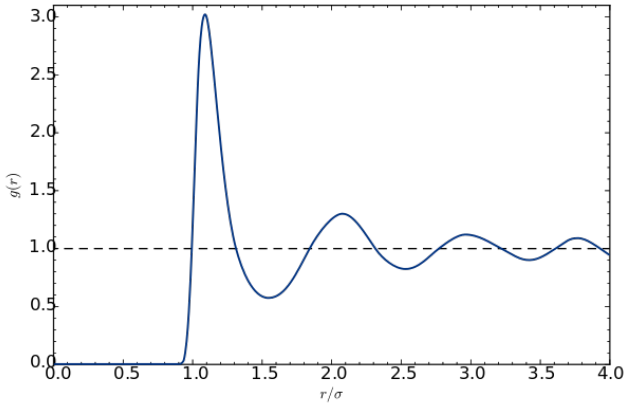

Figure 7: Density of a molecular fluid system.

For our system, we compute the histogram for the water molecules and the drug molecules separately. We expect that the water molecules will be distributed uniformly at a close distance, and will

in fact have an initial peak similar to the histogram shown. If the drug molecule is uniformly distributed, its histogram will have peaks at a larger distance; however, if the drug molecule coagulates, its histogram will have a similar large peak at the start of the graph. We go further as well, computing histograms at each results time step for the system. As time progresses, we can observe the peak moving as the system coagulates, or remaining stable as the system stays uniformly distributed.

4 Schedule

Our initial project idea was formulated in January and was significantly more ambitious – we intended to use molecule interactions as computationally hard proofs of work for a cryptocurrency. Given our time constraints, we instead decided to simply make a centralized system for performing molecule interaction simulations over a client-base. The final schedule is as follows:

1. January – Initial brainstorming and project idea formulation
2. February – Pared down project idea to make it feasible as a class project, and created initial molecular dynamics workflow
3. March – Initial work on OpenMM Client/Server started, and gradually made fixes to workflow bugs
4. April – Finished prototype system

5 Future Directions

There are a variety of directions for future work on this project. For this specific application of molecular dynamics, we would first and foremost like to deliver the predictions for the Pathogen Box drug molecules, preferably with more appropriate standards and settings. We would also suggest using a more descriptive basepoint for the molecule rather than the SMILES string. On the server side, currently the system runs on a single computer, and needs a dedicated platform set up that is accessible using the clients. Although research has gone into designing security solutions, the more complicated approaches still need to be implemented. Most importantly, new job types should be added beyond molecular dynamics simulations in order to test the generalized approach we have taken.

References

[1] About. `http://folding.stanford.edu/about/`. Accessed: 2018-02-27.

[2] About the pathogen box. `https://www.pathogenbox.org/about-pathogen-box`. Accessed: 2018-02-27.

[3] Amber file formats. `http://ambermd.org/formats.html`. Accessed: 2018-03-20.

[4] Frequently asked questions. `https://curecoin.net/knowledge-base/knowledge-base-faq/`. Accessed: 2018-02-27.

[5] Online smiles translator and structure file generator. `https://cactus.nci.nih.gov/translate/`. Accessed: 2018-02-27.

[6] Protein data bank contents guide. `ftp://ftp.wwpdb.org/pub/pdb/doc/format_descriptions/Format_v33_Letter.pdf`. Accessed: 2018-03-20.

[7] David Chandler. *Introduction to modern statistical mechanics*. New York: Oxford University Press, 1987.

[8] David E. Shaw; Martin M. Deneroff; Ron O. Dror; Jeffrey S. Kuskin; Richard H. Larson; John K. Salmon; Cliff Young; Brannon Batson; Kevin J. Bowers; Jack C. Chao; Michael P. Eastwood; Joseph Gagliardo; J.P. Grossman; C. Richard Ho; Douglas J. Ierardi; Istvn Kolossvry; John L. Klepeis; Timothy Layman; Christine McLeavey; Mark A. Moraes; Rolf Mueller; Edward C. Priest; Yibing Shan; Jochen Spengler; Michael Theobald; Brian Towles; Stanley C. Wang. Anton, a special-purpose machine for molecular dynamics simulation. *Communications of the ACM*, 51(7):9197, July 2008.