Journal of Advanced Research in Computer Graphics and Multimedia Technology
Volume 3, Issue 1 - 2021, Pg. No. 12-16
Peer Reviewed Journal

Article

Supercomputers Help in COVID-19

Himani Rajendra Salvi[1], Rishabh Subhash Mehta[2]

Research Scholar, MCA, Thakur Institute of Management Studies, Career Development and Research (TIMSCDR) Mumbai University, Maharashtra, India.

I N F O

Corresponding Author:
Himani Rajendra Salvi, MCA, Thakur Institute of Management Studies, Career Development and Research (TIMSCDR) Mumbai University, Maharashtra, India.
E-mail Id:
himanisalvi18@gmail.com
How to cite this article:
Salvi HR, Mehta RS. Supercomputers Help in COVID-19. *J Adv Res Comp Graph Multim Tech* 2021; 3(1): 12-16.

Date of Submission: 2021-04-09
Date of Acceptance: 2021-05-16

A B S T R A C T

The earnest quest for medications to battle COVID-19 has incorporated the utilization of supercomputers. The utilization of advancements like universally useful graphical preparing units (GPUs), huge parallelism, new programming for elite registering (HPC) has permitted investigated to look for the huge synthetic space of potential medications quicker. Another medication conveyance pipeline was created utilizing Summit supercomputer at Oak Ridge National Laboratory with the assistance of arising stages which use GPUs and permit virtual screening of potential medication compounds in no time. This exertion will speed up the way toward creating medications to battle the current COVID-19 pandemic.

Keywords: Summit, COVID-19, Supercomputer, GPU, Molecular Docking, Drugs

Introduction

The corona virus outbreak (COVID-19), which occurred in China in December2019, spread rapidly around the world and was declared a pandemic by the world health organization (WHO).[1] It is extremely important to be able to diagnose COVID-19 in a patient who is infected during the pandemic process.

From the early days of the pandemic research teams have been trying to refined treatment to improve outcomes and hence reduce the number of deaths from novel coronavirus disease (COVID-19). While some medications have proven effective at keeping the disease from progressing and preventing complications, the recently invented vaccines are also on trial basis and the side effects of it might develop later.

Drug Discovery is a lengthy process that merges efforts from computational and scientists in search of therapeutics. At the molecular level, these small compounds interact with proteins or nucleic acids to alter cellular pathways associated with disease progression.

Drug discovery must work to assure that drugs are not only effective but also are safe. In earlier times, drugs were often non-specific, with significant side effects and their mechanisms of action were not fully understood. Earlier, laboratory experiments were the only way to determine the effects of compounds. as computational resources were not available. The speed of the discovery process is still hindered by the trial-and-error aspects of the experimental methods. Techniques such as x-ray crystallography came able to resolve the locations of the atoms within target proteins, thus providing a detailed map of their three-dimensional structures. Due to advances in computing, it then became possible to try to predict in silico how strongly a small molecule interacts with a given protein. Computational approaches are much faster and cheaper.

Computational speed is especially useful in exploring the vast chemical space. By acting as a selective sieve, in silico molecular docking, an approach that simulates the interaction of the small molecule with the three-dimensional structure of a target protein [2], reduces the chemical search space to a reasonable subset that can be further refined by resource-intensive experimental techniques.

The experimental tests are a critical step within the drug discovery pipeline. Even the best computational methods are nowhere near 100% accuracy, and still predict a high proportion of false positives. Advanced rescoring techniques that incorporate more features of the protein or ligand can further improve these results, with successful machine learning methods representing significant advances. Thus, drug discovery depends on a tight interplay between benchtop and laptop scientists.

Due to the ongoing COVID-19 pandemic, the whole world has come to a standstill, thus, urgency sparked by the lack of the pharmaceutical's treatments.

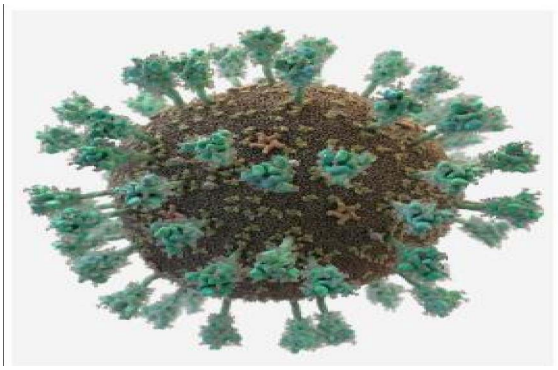

Figure 1.The SARS-CoV-2 virion, visualized by Thomas Splettstoesser scistyle.com under commission. Used with permission

There are many unknowns regarding the SARS- CoV-2 virus (Figure 1) life cycle, particularly as it relates to choosing the best proteins to use as drug targets to mitigate the symptoms of COVID-19. Computational horsepower afforded by Summit is being used to target all SARS-CoV-2 proteins with known structure through a consistent and scalable drug discovery pipeline. A set of 9 distinct viral proteins formed the basis for our structural studies combining molecular simulation, small molecule docking, and analysis to provide a set of compounds predicted to bind to SARS-CoV-2 targets for experimental validation.[5] Emerging results from the recent efforts using high-performance computing (HPC) on the Summit supercomputer, located at the Oak Ridge Leadership Computing Facility (OLCF), to discover potential COVID-19 therapeutics highlight the impact of GPU acceleration and massive parallelism to substantially reduce computational time to solution as part of a larger drug discovery pipeline.[5,6]

Accelerating Molecular Docking

In the early days of the COVID-19 pandemic, collaborating groups from across the world, each with their own docking techniques and philosophies, began applying molecular simulation and protein- small molecule docking methods to provide lists of potential compounds to experimental laboratories for screening with biochemical methods and assays that used live-virus infected cells. The goal was to quickly dock compounds with known safety profiles to meet an immediate need for therapeutics.[5] This experience revealed some computational limitations of current small-molecule docking techniques: with I/O bottlenecks, high variability in time-to solution for docking different molecules, a lack of optimization for the high- throughput docking problem, and overall, performance that did not harness the true power ofthe Summit supercomputer which rests mainly in its 27,000+ GPUs. Therefore, these problems were addressed with several HPC-focused modifications to accelerate molecular docking.[6]

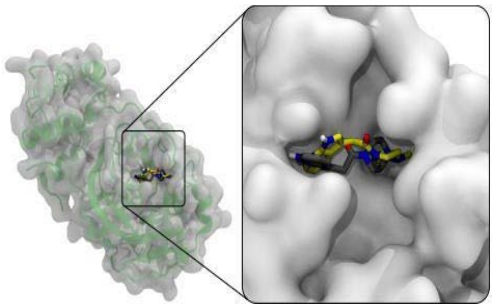

Figure 2.A docking example, emphasizing how binding pockets in a protein (left, with secondary structure shown as a green cartoon below a semitransparent surface)

For this situation, the SARS-CoV-2 Main Protease, can be involved by little atoms.

Two elective bound particle setups are appeared, dark addressing a crystallographic construction, and yellow addressing the posture anticipated via Auto Dock-GPU.

The Scripps AutoDock4 program and its predecessors[7], with the accompanying suite of tools, has been used by researchers worldwide for decades. These techniques have already been applied to COVID-19 research[8], but may miss the best compounds overall, as even minute changes in molecular structure and interaction that are difficult to encode into machine learning frameworks have been shown to affect the score significantly in previous large docking campaigns[9] Figure 2.

GPU Acceleration and Addressing I/O for the High-Throughput use Case

The molecular recognition (binding) events using these structure-based tools is important in virtual drug discovery. However, a majority of small- molecule docking programs are designed for use as single-instance executables to calculate an interaction for one ligand-protein pair. These codes may not be optimal for today's accelerator driven HPC environments. CPU-exclusive algorithms, when deployed on Summit, effectively provide performance at the level equivalent to a commodity cluster. As a result, it took weeks to carry out and analyze docking calculations into the 10,000 or so FDA approved compounds using CPU-based codes.

The speed of the new implementations shifted the bottleneck from floating-point operations to other portions of the program that were bandwidth and I/O bound. Now, the docking calculations themselves only took a few seconds at most.[6] If the docking is performed on the GPU, using idle CPUs to prefetch data for the next calculation can yield a substantial improvement in performance.

Further reductions in I/O and data transfer time were achieved by reusing shared data. Previously, the program required the grid representation of the protein to be both read from file and transferred to the GPU for each ligand processed, even if the same input files were used for consequent docking calculations. Transfer of data from CPU host to GPU is often time consuming and this time increases with data size. Thus, repeated transfers of identical large data objects are to be avoided if reuse is possible. Since our interest was in docking many different small molecules to the same protein, where the small molecule is represented by a much smaller data structure, it made little sense to reload megabytes of data needed to describe the binding region of the protein. Therefore, the program was modified so that the protein data was loaded only once into GPU global memory and is resident on the large global memory found on newer GPUs for hundreds of dockings.[6] These advances prepared us for increasing the scope of our docking calculations to billions of compounds.

Addressing Protein Flexibility

A single 3D structure of a protein target obtained from a crystallography experiment is not necessarily representative of the dynamic population of the protein adopts in a living cell. Within a docking context, these fluctuations of internal protein structure are important to consider, as the interaction between the protein and small molecule will affect further calculations as well. Docking to a set of protein structures rather than a single snapshot helps to keep these fluctuations into account and can be used to test if top scoring compounds can bind to many protein states.

Structural Ensemble Generation for SARS- CoV-2 Proteins Using Molecular Dynamics (MD)

Collecting a structural ensemble for each of the targets using MD involves a calculation using a classical mechanics representation of protein motion. In MD, biological systems can propagate forward in time subject to Newton's equation of motion.

Deploying at Scale on The Summit Supercomputer

Screening one billion compounds against a protein experimentally is effectively impossible with current technologies, as the fastest experimental assays only manage hundreds of thousands of compounds a day. Using the new methods, a computational screen of this magnitude can now be performed in under 24 hours on Summit, or for under half a million dollars on cloud resources. The first such screens used all of Summit to dock the full 1.4 billion compound Enamine REAL dataset against two different crystal structures of the Main Protease of SARS-CoV2. One structure was crystallized with a bound ligand, and the other had an empty active site, which resulted in differences in the active site geometries. Screening datasets of millions to billions of independent small molecules against the viral proteins is a big-data problem.

Our medication revelation pipeline comprised of these segments: information pre-preparing, work process the board for billion-ligand docking computations, and post-handling examination (Figure 3). Each profit by HPC instruments and libraries to misuse parallelism inside and across hubs.

Python Improves Productivity

Python plays a critical role as a "glue language" within the drug delivery pipeline, and the flexibility of Python increased programmer productivity. As distributed, the input data for a billion ligands are small text files representing individual molecules collected into compressed archives and must be converted into a format that the docking program requires. Therefore, before docking a billion ligands, all these files must be extracted, converted into the appropriate format for docking, and repackaged into compressed formats.

The Python ecosystem has utilities to perform tasks important to this pre-processing, for instance, the trifle library can read, manipulate, and create new trifles entirely in memory, the job lib library allows for parallel tasks to be executed within a node, and the mpi4py library was used to distribute preprocessing steps across multiple nodes in an HPC cluster (CADES, Figure 4). The Python processing script provided by Auto Dock that carries out the file conversion was transformed into a loadable module with minimal effort and greatly sped up the input conversion process.

Python was also used in launching workflows on Summit. Python-based workflow managers were used to run the new CUDA-based docking program on the 6 V100 GPUs and 42 Power9 cores of each of the 4608 nodes of Summit. After the simulations are run, output data must be post processed; sorting and analyzing the terabytes of data produced is challenging. Python was used to facilitate the rapid development of solutions using new tools that provide GPU back-ends for Python interfaces, which are expanded upon in subsequent sections.

Workflows for Docking I Billion Compounds

With the optimized Auto Dock-GPU code, on average a single GPU could complete a docking calculation on the SARS-CoV-2 Main Protease every 1.6 seconds, with the fastest compounds docking in half a second or less.[6] For efficiency it is important to deploy the many independent

tasks using a dataflow execution model, where any resources that are available can be assigned a new task on the fly, leaving few resources unused at any time. Two separate frameworks were tested to orchestrate launching these workflows without hitting resource limits or leaving idle GPUs.

Fire Works and the Slate Resource at OLCF

One framework used the Fire Works workflow management software [12], deployed via an external computing cluster at the OLCF called Slate. Slate provides container orchestration for users and consists of two user-facing clusters, with Marble being the cluster that interfaces with Summit. With Marble, we drive workflows that run on Summit but are controlled externally, providing a persistent state for the workflow in case of Summit failure and allowing system configurations and services that are not possible on Summit. Specifically, Fireworks depended on libraries and configurations that were simplest to deploy on containerized resources such as Slate. Rather than exclusively serving Oak Ridge resources, this setup allows workflows to be deployed simultaneously on Summit and other computing facilities, providing the capability for even more computing power using collective resources across institutions.

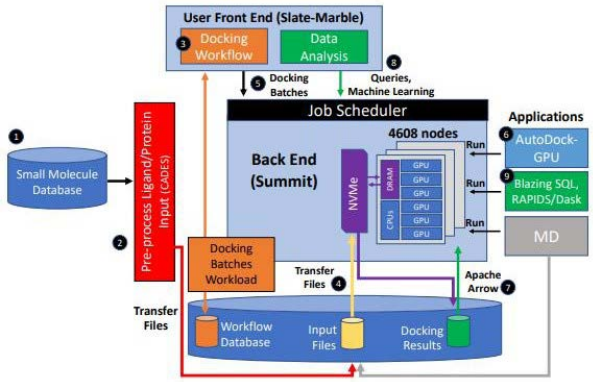

Figure 3.Overview for the workflow

Starting from the small molecule database of potential drugs (1) and ending at a query-able database of results (9). Preprocessing the small molecule database (2) creates files needed for docking, and in our case was done on a small commodity cluster (CADES). Once on the file system, the docking workflow is prepared (3), and is stored within the Slate-Marble database system. Subsequently, the input files are loaded onto the fast NV Me drives on Summit (4), and the batches of compounds stored within the workflow database (5) are docked on Summit (6). Output is then converted to Apache parquet format (7) for storage into a GPU-accelerated Blazing SQL database (8). Queries on this database are fast and enable potential machine learning applications (9) Figure 3.

Fireworks provides a workflow framework written in Python and utilizes a MongoDB database management program as its back end, for queuing tasks. It provides a set of utilities to interact with the workflow system from the command-line and a programming interface to control workflows, including facilities for monitoring task status in real time. The Fireworks back-end and the dashboard were deployed on Marble. Task managers called “Fireworks” are deployed on Summit’s compute nodes and contact the MongoDB back-end on Marble to retrieve sets of tasks to execute on that compute node. If a set fails to complete, due to a node failure or a temporary input issue, it is automatically requeued within the database.

Simple Task-List on a Summit Node

At this scale, some temporary problems related to the database back-end and connection limits were encountered when using Fireworks, which were solved after re-configuring Marble/Fireworks system settings.

Since each Firework keeps a persistent database connection, the Fireworks server can become a point of failure. Although a list of 1.4 billion ligands were too large to fit into the memory of a single machine, a list of 1 million batches of ligands worked well and was hosted on a Summit launch node. The combination of memory caching and running on a launch node minimized the overhead time for querying the database and lowered the risk of connection issues between the database and compute node delaying docking calculations. This step used 27,600 concurrent docking processes, with each GPU calculating on a distinct set of lagans-protein pairs.

Recovering from Failure

If a node failed or otherwise a task could not be completed, the easiest symptom to detect was missing output files. In some instances, the issue was with the input, as not all compounds in the database satisfied the assumptions Auto Dock makes about compounds. In other cases, a node failed, and these tasks simply were rerun by restarting that stage of the pipeline, which would attempt to fill in missing outputs. Eventually, less than 1% of outputs could not be generated due to input issues.

Processing and Analysis of Terabytes of Generated Data: New Tools Using Gpu Acceleration

The next challenge we faced was analyzing the billions of docking results due to the sheer size of the dataset. Auto Dock-GPU, like other Auto Dock programs before it, produces human readable output that includes the geometries and scores of the small molecule poses. While the individual output files are small and on the order of a few hundred kilobytes, collectively the output for a billion dockings spans terabytes of data to process, store, and query. We were able to transform the output data into Apache parquet files so that we can use GPU-based data

tools such as Blazing SQL to sort, search, and query the database interactively, facilitated through a Jupiter notebook.

Figure 4.Example of the Jupiter notebook interface for exploring the billion compound docking results

What separates Blazing SQL from other SQL

implementations are that Blazing SQL is built on top of the NVIDIA RAPIDS analysis stack, and leverages GPUs to make data manipulation faster. When combined with distributed task management via Desk, searching through the collection of thousands of parquet files for a compound of interest takes seconds rather than hours. This level of interactivity allows researchers to explore the data in real time with familiar tools (Figure 4), increasing researcher productivity. The rapid analysis and visualization integration enabled by Python and powerful tools to enable close communication between computational and wet-lab scientists.

Conclusion

Since the COVID-19 pandemic began, there has been a substantial mobilization of scientific resources to address the pressing need for COVID-19 treatments. Together with vaccine development, finding drugs to combat the virus is at the forefront of activity. While the unique circumstances of the pandemic brought together many key resources for this work, the durable results of an accelerated docking application integrated into a scalable drug discovery pipeline are now available to the wider research community, including the use of leadership computing facilities to assist in these efforts. We also see that as the scale of the data and calculations grow, workflow tools and analysis will become a larger part of the scientific toolkit, and we hope that our experience is informative to future researchers tackling problems at the boundary of what is possible.

References

1. Zhu N, Zhang D, Wang W etal. A novel coronavirus from Patients with pneumonia in China, 2019.New England Journal of Medicine 2020; (382): 727-733. doi: 10.1056/NEJMoa2001017
2. Pagadala NS, Khajamohiddin S, Tuszynski J. Software for molecular docking: a review. *Biophysical reviews* 2017; 9(2): 91-102.
3. Kurkinen ST, Latti S, Pentik OT et al. Getting Docking into Shape Using Negative Image Based Rescoring. *Journal of Chemical Informatics and Modeling* 2019; 59(9): 83584–3599.
4. Tan SY, Tatsumura Y. Alexander Fleming (1881- 1955): discoverer of penicillin. *Singapore medical journal* 2015; 56(7): 366.
5. Acharya A, Agarwal R, Baker M et al. Supercomputer-Based Ensemble Docking Drug Discovery Pipeline with Application to Covid-19. *ChemRxiv*, July 2020 doi: 10.26434/chemrxiv. 12725465.v1.
6. LeGrand S, Scheinberg A, Tillack AF et al. GPU Accelerated Drug Discovery with Docking on the Summit Supercomputer: Porting, Optimization, and Application to COVID-19 Research. ACM BCB 2020 (pending publication, https://doi.org/10.1145/3388440.3412472)
7. Morris GM, Huey R, Lindstrom W et al. AutoDock4 and AutoDockTools4: Automated docking with selective receptor flexibility. *Journal of Computational Chemistry* 2009; 30(16): 2785-2791.
8. Ton A, Gentile F, Hsing M et al. Rapid Identification of Potential Inhibitors of SARS-CoV2 Main Protease by Deep Docking of 1.3 Billion Compounds. Molecular Informatics, 39, 2000028, 2020.
9. Lyu J, Wang S, Balius TE et al. Ultra-large library docking for discovering new chemotypes. *Nature* 2019; 566: 224-229.
10. Santos-Martins D, Eberhardt J, Bianco G et al. D3R Grand Challenge 4: prospective pose prediction of BACE1 ligands with Auto Dock-GPU. *Journal of Computer-Aided Molecular Design* 2019; 33(12): 1071-1081.
11. AJ, AR ak, Ladj I anszki, G. Cserey, Classical molecular dynamics on graphics processing unit architectures. WIREs *Computational Molecular Science* 2020; 10(2) doi:10.1002/wcms.1444.
12. Jain A, Ong SP, Chen W, B. Medasani, X. Qu, M. Kocher, M. Brafman, G. Petretto, G. M. Rignanese, G. Hautier, and D. Gunter.