# Supercomputing Pipelines Search for Therapeutics Against COVID-19

Josh Vincent Vermaas, *Oak Ridge National Laboratory, Oak Ridge, TN, 37830, USA*

Ada Sedova, *Oak Ridge National Laboratory, Oak Ridge, TN, 37830, USA*

Matthew B. Baker and Swen Boehm, *Oak Ridge National Laboratory, Oak Ridge, TN, 37830, USA*

Jeff Larkin, *NVIDIA Corporation, Santa Clara, CA, 95051, USA*

Jens Glaser, *Oak Ridge National Laboratory, Oak Ridge, TN, 37830, USA*

Micholas D. Smith, *Oak Ridge National Laboratory, Oak Ridge, TN, 37830, USA, and the University of Tennessee Knoxville, TN, 37996, USA*

Oscar Hernandez, *Oak Ridge National Laboratory, Oak Ridge, TN, 37830, USA*

Jeremy C. Smith, *Oak Ridge National Laboratory, Oak Ridge, TN, 37830, USA, and the University of Tennessee Knoxville, TN, 37996, USA*

David M. Rogers, *Oak Ridge National Laboratory, Oak Ridge, TN, 37830, USA*

***The urgent search for drugs to combat SARS-CoV-2 has included the use of supercomputers. The use of general-purpose graphical processing units (GPUs), massive parallelism, and new software for high-performance computing (HPC) has allowed researchers to search the vast chemical space of potential drugs faster than ever before. We developed a new drug discovery pipeline using the Summit supercomputer at Oak Ridge National Laboratory to help pioneer this effort, with new platforms that incorporate GPU-accelerated simulation and allow for the virtual screening of billions of potential drug compounds in days compared to weeks or months for their ability to inhibit SARS-COV-2 proteins. This effort will accelerate the process of developing drugs to combat the current COVID-19 pandemic and other diseases.***

DRUG discovery is a lengthy process that merges efforts from computational and bench-top scientists in search of small-molecule therapeutics.[1] At the molecular level, these small compounds interact with proteins or nucleic acids to alter cellular pathways associated with disease progression. For instance, angiotensin receptor blockers act to inhibit the action of certain proteins in the body to reduce blood pressure, and antiretroviral drugs combat proteins created and used by the HIV virus, interfering with viral replication and other pathogenic mechanisms.


Digital Object Identifier 10.1109/MCSE.2020.3036540
Date of publication 6 November 2020; date of current version 26 February 2021.

Drug discovery must work to assure that drugs are not only efficacious but also are safe, and do not inadvertently affect the function of proteins that are not targeted, in order to minimize potential drug side effects.[2] Early drugs were often nonspecific, with substantial side effects and their mechanisms of action were not fully understood. Before computing power was readily available, laboratory experiments were the only way to determine which small-molecule compounds would have the desired effect. Chemicals from natural products played a substantial role, as did serendipity, exemplified by Alexander Fleming's accidental discovery of penicillin from a natural fungal antibiotic.[4] The speed of the discovery process is still hampered by the trial-and-error aspects of the experimental methods: Compounds must be synthesized or isolated from a natural source, then extensively tested.

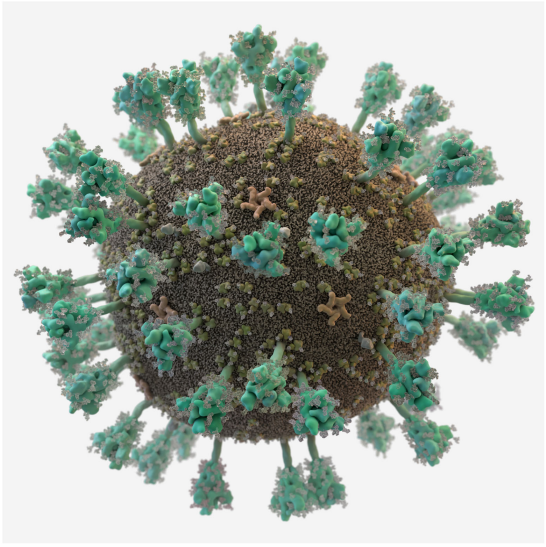

**FIGURE 1.** SARS-CoV-2 virion visualized by Thomas Splettstoesser scistyle.com under commission. Used with permission.

Several technological advances would be required before the field could start to move away from the above paradigm. High-throughput experimental tests (assays) of binding to specific protein targets using libraries of synthetic molecules allowed for a more systematic searching of the chemical space. Techniques such as X-ray crystallography became able to resolve the locations of the atoms within target proteins, thus providing a detailed map of their three-dimensional structures. When matched with advances in computing, it then became possible to try to predict *in silico* how strongly a small molecule interacts with a given protein target. Critically, computational approaches are much faster and cheaper than *in vitro* work for filtering combinations of compounds and proteins, winnowing chemical space for specific protein targets.

That computational speed is very useful in exploring chemical space. Today, chemical synthesis companies, such as Enamine, promise to be able to synthesize any of over a billion different compounds that might demonstrate utility as drugs, which is five orders of magnitude beyond all the molecules currently approved for therapeutic use. Only by applying computational tools we evaluate all these molecules. By acting as a selective sieve, *in silico* molecular docking, an approach that simulates the small-molecule interactions 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 assays are still 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. Nevertheless a pool of compounds selected by computation is likely to be significantly enriched with "hits," often by a factor of 10–100.[3] 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.

The ongoing COVID-19 pandemic and the uncertainty surrounding this novel coronavirus have presented unique challenges for this synergistic, interdisciplinary approach, including a new urgency, sparked by the lack of existing pharmaceutical treatments and relatively high mortality rates. Typical drug discovery pipelines, even with computational efforts, can take years to go from the identification of a promising drug target to the use of a drug at the bedside; the drug discovery process cannot be fully automated and involves many complex, multidisciplinary steps. However, the pandemic has encouraged us to rethink the computational portion of this effort, and we focused on how best to apply supercomputing to accelerate drug discovery by reducing the time and cost associated with molecular docking.

*THERE ARE MANY UNKNOWNS REGARDING THE LIFE CYCLE OF THE SARS-COV-2 VIRUS.*

There are many unknowns regarding the life cycle of the SARS-CoV-2 virus (see Figure 1), particularly as it relates to choosing the best proteins to use as drug targets to mitigate the symptoms of COVID-19. We have been using the computational horsepower afforded by Summit to try targeting all SARS-CoV-2 proteins with known structure through a consistent and scalable drug discovery pipeline. A set of nine 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 our recent efforts using high-performance computing (HPC) on the Summit supercomputer, housed 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

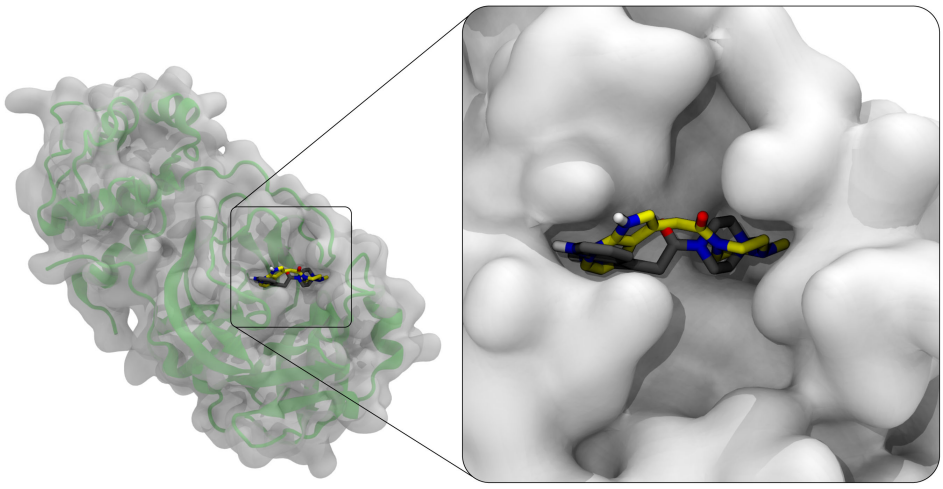

**FIGURE 2.** Docking example, emphasizing how binding pockets in a protein (left, with secondary structure shown as a green cartoon below a semitransparent surface), in this case, the SARS-CoV-2 Main Protease, can be occupied by small molecules. Two alternative bound molecule configurations are shown, black representing a crystallographic structure, and yellow representing the pose predicted by AutoDock-GPU.

reduce computational time to solution as part of a larger drug discovery pipeline.[5,6]

## ACCELERATING MOLECULAR DOCKING

At the outset 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. Experimental laboratories used these for screening with against live-virus infected cells. The goal was to quickly dock compounds with known safety profiles in an attempt 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 of the Summit supercomputer, which rests mainly in its more than 27,000 GPUs. We addressed these problems with several HPC-focused modifications to accelerate docking.[6]

### Simulating Molecular Recognition

Molecular docking consists of modeling the interactions between two chemical entities—typically a protein and a small chemical compound (ligand)—and predicting both the most energetically favorable positions for interaction and also the binding strength, or affinity, of the interaction (see Figure 2). High affinity drugs tend to be more efficacious, as they are needed in smaller concentrations to produce a therapeutic effect. The search is often focused on the region of the protein known as the active site that affects the chemical function of the protein. Many applications use an interaction field, represented as a grid, to describe one or both of the binding partners for efficient computation. Representing the protein's binding pocket with a grid, and the ligand as a set of individual atoms, is an approach often used in drug discovery, and this type of application is the primary focus for the protein–ligand interaction calculations in our recent work.

*HIGH AFFINITY DRUGS TEND TO BE MORE EFFICACIOUS, AS THEY ARE NEEDED IN SMALLER CONCENTRATIONS TO PRODUCE A THERAPEUTIC EFFECT.*

The Scripps AutoDock4 program and its predecessors,[7] together with the accompanying suite of tools, have been used by researchers worldwide for decades. AutoDock4 uses a genetic optimization algorithm where an evolving "population" of solutions mixes features (a "crossover") at each generation, moving toward an optimal solution for a ligand geometry within the field described on the grid. AutoDock thus combines the genetic algorithm with a local optimization step to arrive

at a final solution. Recently developed machine learning methods can further supplement these techniques to extrapolate from a limited training set of compounds to produce a larger set of similar compounds. Several types of machine learning have already been applied to COVID-19 research,[8] but may miss the best compounds overall. Recent large docking campaigns have demonstrated that small changes in molecular structure and interaction that are difficult to encode into machine learning frameworks significantly influence the score.[9]

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

Simulating 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 were largely designed in a CPU-computing era and may not be optimal for today's accelerator-driven HPC environments. On a pre-exascale machine like Summit with a theoretical performance of 200 petaFLOPs, the FLOPs are provided primarily by the six NVIDIA V100 GPUs on each node (97%) rather than the two 21-core POWER9 CPUs. CPU-exclusive algorithms, when deployed on Summit, effectively provide performance at the level equivalent to a commodity cluster. As a result, using CPU-based codes our docking calculations into the 10,000 or so FDA approved compounds took on the order of weeks to carry out and analyze.

*AUTODOCK-GPU WAS PORTED TO CUDA WITHIN A MONTH, ALLOWING SUMMIT'S FULL CAPABILITIES TO BE BROUGHT TO BEAR FOR DOCKING.*

To scale up and reap the full benefits of the 200 petaFLOP Summit hardware it was imperative that we use a GPU-accelerated docking solution. AutoDock-GPU, a recent GPU port of AutoDock for OpenCL released by Scripps Research,[10] was particularly appealing, as the AutoDock family is a well-known set of docking tools. However, AutoDock-GPU's dependence on OpenCL was a barrier on Summit, as no OpenCL drivers exist for Summit's combination of POWER9 CPUs and NVIDIA GPUs. Working with industrial partners both big (NVIDIA) and small (Jubilee Development), driven to help with the ongoing pandemic, AutoDock-GPU was ported to CUDA within a month, allowing Summit's full capabilities to be brought to bear for docking.

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. In the older CPU docking codes, where docking of one compound could take minutes to calculate with significant variability, the time spent reading small input files is insignificant. However, on the GPU where FLOPs are abundant, a high-throughput version of the application became limited by I/O and set-up time, as 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. With the help of Jubilee Development, OpenMP threading was added that enabled multiple threads to parse protein and ligand input files and transfer data to the GPU while a previous calculation was being performed. This allowed for prefetching data and overlapping data movement with the GPU-based docking calculation, and provided over 3× speedup compared to the original GPU-accelerated version of AutoDock-GPU.[6]

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 protein input files were used for subsequent docking calculations. Transfer of data from CPU host to GPU is often one of the most time consuming portions of a GPU-based calculation 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 were loaded only once into GPU global memory, and is resident on the large global memory found on newer GPUs such as NVIDIA's V100 with 16 or 32 GB HBM2 for hundreds of dockings.[6] The creation of a CUDA context is also a time-consuming step, and we observed that reusing CUDA contexts also provided speedup. Overall, the total speedup from all of these I/O optimizations combined was over 9× compared with the CUDA version for typical druglike molecules such as those found in fragment libraries.[6] When compared to the serial CPU version of AutoDock4, the speedup was up to 300×.

Together, these advances prepared us for increasing the scope of our docking calculations to billions of compounds. The Enamine REAL database contains billions of compounds that have never previously been synthesized. Explicit docking allowed us to filter that vast chemical space down to a tractable size for follow-on experimental study.

## ADDRESSING PROTEIN FLEXIBILITY

A single 3-D structure of a protein target obtained from a crystallography experiment is not necessarily representative of the dynamic population of possible configurations the protein adopts in a living cell. Within a docking context, these fluctuations of internal protein structure are important to consider, since the interaction between the protein and small molecule will change as contacts are formed or broken. Docking to a set of protein structures rather than a single snapshot helps to take these fluctuations into account, and can be used to test if top-scoring compounds can bind to many protein states. Among approaches to addressing protein conformational change are the use of a flexible protein in the docking calculation itself, the use of several crystal structures to provide different protein active site poses, if available, and the use of molecular dynamics (MD) simulations to simulate a set of poses, or conformations, of the protein's native state. We complement billion-ligand docking calculations by docking with both multiple crystal structure and poses generated by MD.

### Structural Ensemble Generation for SARS-CoV-2 Proteins Using MDs

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 are allowed to propagate forward in time subject to Newton's equation of motion. The resulting "trajectory," or molecular movie, is then analyzed to provide a representative set of poses (an ensemble), which can be sampled by docking. In early applications to HPC, molecular simulations rode on the coattails of Moore's Law to longer simulation timescales and larger system sizes. More recently, MD applications, such as AMBER, GROMACS, NAMD, HOOMD-Blue, and LAMMPS, have been at the forefront in the transition to GPU use,[11] and are highly optimized for Summit hardware. In practice, the GPU transition means that a single GPU simulating a small protein in water will generate a few hundred nanoseconds of simulated trajectory in a day of walltime.

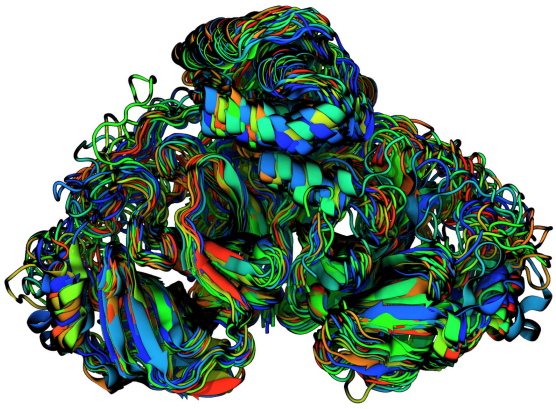

**FIGURE 3.** Graphical representation of the structural ensemble generated from parallel-tempering MD simulations. Each of the 26 protein snapshots taken from the ensemble has a different color, and is drawn in a cartoon representation. The spaghettilike strands represent loop elements where the greatest variation in structure takes place in the simulation.

However, obtaining a single continuous trajectory for timescales comparable to protein structural changes (milliseconds or more) still remains impractical without special purpose hardware, such as Anton or Anton2 developed by DE Shaw Research. Exploiting the parallelism inherent to modern supercomputer design is paramount. The key methodological advances come from algorithms that can provide sampling statistically equivalent to a single long trajectory from many shorter trajectories.

We used parallel tempering, frequently called temperature replica exchange in biomolecular simulations, to accelerate sampling of the ensemble of structures for each protein system. In parallel tempering, multiple copies of a simulation system are run simultaneously, each at a different simulated temperature enforced by a numerical thermostat. The higher temperatures allow what would otherwise be large energetic barriers to structural change to be easily overcome via thermal motion. The enhanced sampling in the higher temperature space can then be incorporated into low temperature ensembles that reflect room temperature conditions by periodically exchanging configurations between different temperatures using rules based on statistical physics. By crossing barriers to conformational change at high temperature and sampling these states at low temperature, the generated structural ensemble (see Figure 3) is statistically equivalent to a longer continuous trajectory at the low temperature. Recasting the problem in this way can provide

us with an ensemble of structures in a greatly reduced amount of time. For modestly sized proteins such as found in the SARS-CoV-2 proteome, Summit can deliver the equivalent of several microseconds per day of simulation time per protein using these parallel replica methods.

### Using a Set of Different Crystal Structures

The pandemic has spurred a massive effort in crystallography. As a result, hundreds of structures of viral proteins have been deposited in the crystallographic databases, with some well-studied proteins having dozens of different structures available. This provides a unique set of data with which to explore different conformations of proteins found among the various crystal structures. While the range of poses and the magnitude of changes in atomic positions is much less than that produced by MD, a set of crystal structures provides experimental snapshots of protein poses that are free of potential model-induced biases introduced by simulation.

## 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 our new methods, a computational screen of this magnitude can now be performed in under 24 h 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-CoV-2. One structure was crystallized with a bound ligand, and the other had an empty active site, which created small 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. This large, but intuitively parallel, computation can be greatly aided by HPC. However, these types of calculations were not designed to be treated by HPC approaches, and require novel software solutions in order to make this type of scaling not only possible but rapidly attainable, given the urgency in delivering hits to experimental groups.

Our drug-discovery pipeline consisted of these components: data preprocessing, workflow management for billion-ligand docking calculations, and postprocessing analysis (see Figure 4). Each benefits from HPC tools and libraries to exploit parallelism within and across nodes.

### Python Improves Productivity

Python plays a critical role as a "glue language" within our 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 of these files have to 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 preprocessing, for instance, the `tarfile` library can read, manipulate, and create new tarfiles entirely in memory, the `joblib` 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 AutoDock that carries out the file conversion was transformed into a loadable module with minimal effort and greatly sped up the input conversion process.

*SLATE PROVIDES CONTAINER ORCHESTRATION FOR USERS AND CONSISTS OF TWO USER-FACING CLUSTERS*

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 postprocessed; 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 1 Billion Compounds

With our optimized AutoDock-GPU code, on average a single GPU could complete a docking calculation on the SARS-CoV-2 Main Protease every 1.6 s, with the fastest compounds docking in half a second or less.[6] Thus, when distributing work over Summit, roughly 17,600 compounds had to be loaded every second to

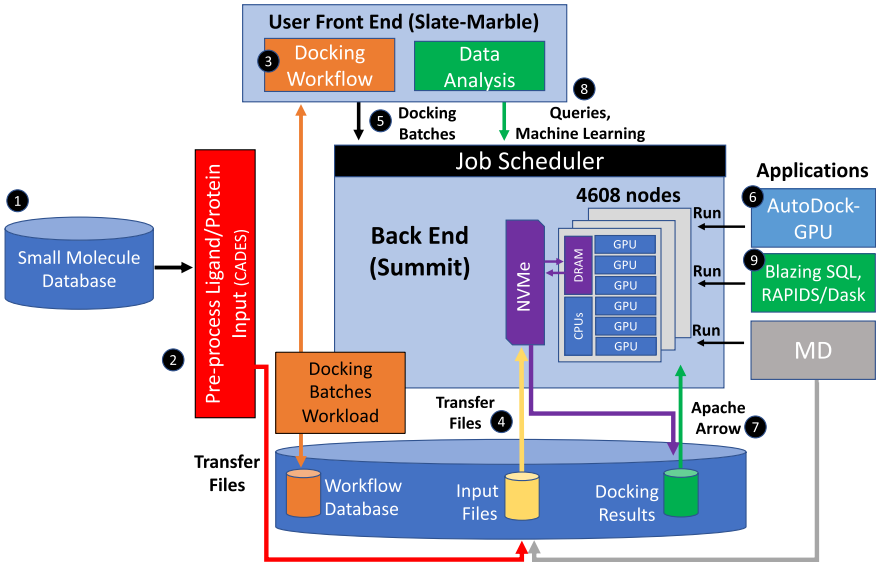


**FIGURE 4.** Overview for the workflow, starting from the small molecule database of potential drugs (1) and ending at a query-able database of results (9). Preprocessing the small molecule database (2) creates files needed for docking, and in our case was done on a small commodity cluster (CADES). 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 NVMe 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).

occupy all of Summit's ~27,000 GPUs. Our workflows (see Figure 4) moved tar archives with thousands of input ligands to the compute node's nonvolatile memory, decompressed the archive, created an input file for the docking code, and finally compressed the results and moved them back to the center-wide file system once the batch of docking runs was complete. 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.

### *FireWorks and the Slate Resource at OLCF*

One framework used the FireWorks 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. Marble is a heterogeneous cluster of 30 nodes with 10 Gb Ethernet connectivity. With marble, we drive workflows that run on Summit but are controlled externally, providing a persistent state for the workflow in case of job 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 driving Oak Ridge resources, this setup allows workflows to be deployed simultaneously on Summit and other computing facilities. This provides the capability for even more computing power using collective resources across institutions.

FireWorks provides a workflow framework written in Python, and utilizes a MongoDB database management program as its backend, 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 backend and the dashboard were deployed on marble. Task managers called "FireWorkers" are deployed on Summit's compute nodes, and contact the MongoDB backend 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. For docking

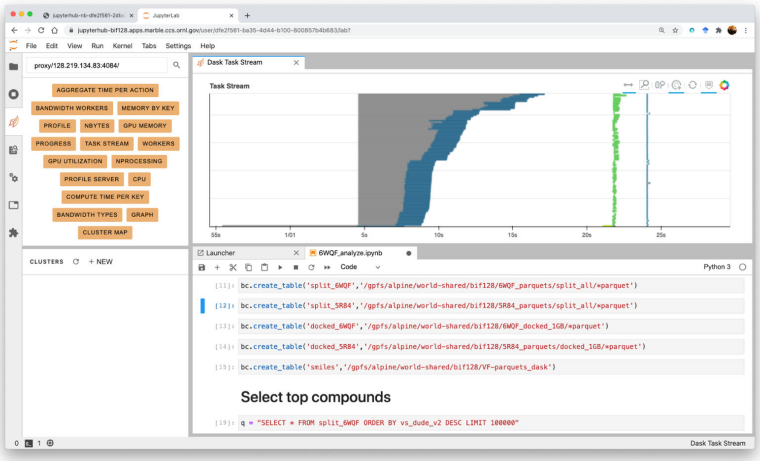


**FIGURE 5.** Example of the Jupyter notebook interface for exploring the billion compound docking results.

the 1.4 billion compound dataset to one of the Main Protease structures, 4,602 compute nodes were used simultaneously on Summit, each running six FireWorkers per node, one for each GPU, bringing the total number of FireWorkers to 27,612.

### *Simple Task-List on a Summit Node*

At this scale, we encountered some temporary problems related to the database backend and connection limits when using FireWorks, which were solved after reconfiguring Marble/FireWorks system settings. Since each FireWorker keeps a persistent database connection, the FireWorks server can become a point of failure. As an alternative to mitigate this risk, we also developed an in-house solution using Python and an in-memory Redis database to provide the compute nodes lists of batches to run. Although a list of 1.4 billion ligands was 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 ligand–protein pairs. Although this setup generated a huge number of database requests on startup, there were only tens of requests per second during continuous operation for a day, a rate Redis handled easily. The database latency for using Redis was about 0.4 s out of the entire day of running at full scale on Summit.

### *Recovering From Failure*

Both the FireWorks and Redis workflow management schemes for the docking calculation, as well as the initial preprocessing steps, needed to account for failures, either at the hardware or software level. 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 Enamine REAL database satisfied the assumptions AutoDock makes about compounds, such as by including an organic metal within the structure. 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. These were all traced to nonstandard molecules that were unsuitable inputs to AutoDock due to geometry or constituent atoms.

## 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. AutoDock-GPU, like other AutoDock 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 BlazingSQL to sort, search, and query the database interactively, facilitated through a Jupyter notebook.

What separates BlazingSQL from other SQL implementations is that BlazingSQL 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 Dask, 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 (see Figure 5), increasing researcher's productivity. The interactive analysis supported can return results quickly for compound SQL queries, such as the top-scoring compounds that are within a selected distance from a protein residue, or the score compared with atomic feature vectors used for machine-learning-based rescoring functions. Combining the speed and flexibility of the underlying SQL queries is a further boon to understanding these large datasets. By being able to analyze and reanalyze the results quickly, we can use machine-learned rescoring functions to attack our output from multiple directions, sending compounds on for experimental validation only if they look promising from multiple angles. These multiple selection criteria make the docking sieve more stringent and hopefully further enriching the output set passed to experimental collaborators. The rapid analysis and visualization integration enabled by Python are 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. Streamlining the drug discovery pipeline is part of a larger collective effort to not only address the current crisis, but also prepare for future challenges the community will face in treating other diseases. While the unique circumstances of the pandemic brought together many key resources for this work, the durable results of an accelerated docking application (https://github.com/ccsb-scripps/AutoDock-GPU) integrated into a scalable drug discovery pipeline are now available to the wider research community, including integration into 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.

## ACKNOWLEDGMENTS

The authors would like to thank Scott Le Grand (NVIDIA), Aaron Scheinberg (Jubilee Development), and Andreas Tillack (Scripps Research) along with the rest of the NVIDIA and Scripps teams for their work on AutoDock-GPU on Summit and advice and support. We also acknowledge the contributions Rupesh Agarwal and Mathialakan Thavappiragasam for their input and expertise in the early stages of pipeline development. This work was supported by the Laboratory Directed Research and Development Program at Oak Ridge National Laboratory, which is managed by UT-Battelle, LLC, for the U.S. Department of Energy (DOE) under Contract DE-AC05-00OR22725. This work also used resources, services, and support provided via the COVID-19 HPC Consortium (https://covid19-hpc-consortium.org/), which is a unique private–public effort to bring together government, industry, and academic leaders who are volunteering free compute time and resources in support of COVID-19 research, and used resources of the Oak Ridge Leadership Computing Facility at the ORNL.

## REFERENCES

1. J. Drews, "Drug discovery: A historical perspective," *Science*, vol. 287, no. 5460, pp. 1960–1964, 2000.
2. N. S. Pagadala, S. Khajamohiddin, and J. Tuszynski, ," Software for molecular docking: A review," *Biophys. Rev.*, vol. 9, no. 2, pp. 91–102, 2017.
3. S. T. Kurkinen, S. Lätti, O. T. Pentikäinen, and P. A. Postila, "Getting docking into shape using negative image-based rescoring," *J. Chem. Inf. Model.*, vol. 59, no. 8. pp. 3584–3599, 2019.
4. S. Y. Tan and Y. Tatsumura, "Alexander Fleming (1881-1955): Discoverer of Penicillin," *Singapore Med. J.*, vol. 56, no. 7, pp. 366–367, 2015.
5. A. Acharya *et al.*, "Supercomputer-based ensemble docking drug discovery pipeline with application to Covid-19," *J. Chem. Inf. Model.*, vol. 60, no. 12, pp. 5832–5852, 2020.
6. S. LeGrand *et al.*, "GPU-accelerated drug discovery with docking on the summit supercomputer: Porting, optimization, and application to COVID-19 research," *Proc. ACM Conf. Bioinf., Comput. Biol., Biomedicine*, 2020, pp. 43:1–43:10. [Online]. Available: https://doi.org/10.1145/3388440.3412472

7. G. M. Morris *et al.*, "AutoDock4 and AutoDockTools4: Automated docking with selective receptor flexibility," *J. Comput. Chemistry*, vol. 30, no. 16, pp. 2785–2791, 2009.
8. A. Ton, F. Gentile, M. Hsing, F. Ban, and A. Cherkasov, "Rapid identification of potential inhibitors of SARS-CoV-2 main protease by deep docking of 1.3 billion compounds," *Mol. Informat.*, vol. 39, 2020, Art. no. 2000028.
9. J. Lyu *et al.*, "Ultra-large library docking for discovering new chemotypes," *Nature*, vol. 566, pp. 224–229, 2019.
10. D. Santos-Martins *et al.*, "D3R Grand Challenge 4: prospective pose prediction of BACE1 ligands with AutoDock-GPU," *J. Comput.-Aided Mol. Des.*, vol. 33, no. 12, pp. 1071–1081, 2019.
11. Á. Jász, Á. Rák, I. Ladjánszki, and G. Cserey, "Classical molecular dynamics on graphics processing unit architectures," *WIREs Comput. Mol. Sci.*, vol. 10, no. 2, 2020, Art. no. e1444, doi: 10.1002/wcms.1444.
12. A. Jain *et al.*, "FireWorks: A dynamic workflow system designed for high-throughput applications," *Concurrency Comput.: Pract. Experience*, vol. 27, no. 17, pp. 5037–5059, 2015.



**JOSH V. VERMAAS** is currently a Computational Scientist with the National Center for Computational Sciences Division, Oak Ridge National Laboratory, Oak Ridge, TN, USA. His research interest includes large-scale molecular dynamics simulations on high-performance computing systems. He is the corresponding author of this article. Contact him at vermaasjv@ornl.gov.

**ADA SEDOVA** is currently a Research and Development Staff Member with the Biological Sciences Division, Oak Ridge National Laboratory, Oak Ridge, TN, USA. Her research interest includes achieving more accurate models of physical systems with high-performance computing and connecting experiment to simulation. Contact her at sedovaaa@ornl.gov.

**MATTHEW B. BAKER** is currently a Technical Staff Member with Oak Ridge National Laboratory, Oak Ridge, TN, USA. He was instrumental in the development of the UCX platform and has been developing ways to improve the use of Python interfaces for UCX such as those used in Dask. Contact him at bakermb@ornl.gov.

**SWEN BOEHM** is currently a Research and Development Staff Member in the Computer Science and Mathematics Division, Oak Ridge National Laboratory, Oak Ridge, TN, USA. His research is focused on runtime environments, tools, and programming models for high-performance computing environments. Contact him at boehms@ornl.gov.

**DAVID M. ROGERS** is currently a Computational Scientist with the National Center for Computational Sciences Division, Oak Ridge National Laboratory, Oak Ridge, TN, USA. His research interests include mathematical and computational theory and methods for multiscale modeling using high-performance computing. Contact him at rogersdm@ornl.gov.

**JEFF LARKIN** is currently a Senior Developer Technology Software Engineer with NVIDIA, Santa Clara, CA, USA. His research interests include high-performance computing application profiling and optimization in GPU computing, MPP, cluster, and grid-computing platforms. Contact him at jlarkin@nvidia.com.

**JENS GLASER** is currently a Computational Scientist with the National Center for Computational Sciences Division, Oak Ridge National Laboratory, Oak Ridge, TN, USA. He is interested in developing efficient parallel simulation algorithms to investigate assembly pathways of novel synthetic nano- and biomaterials. Contact him at glaserj@ornl.gov.

**MICHOLAS D. SMITH** is currently a Research Assistant Professor with the Department of Biochemistry and Cellular and Molecular Biology, University of Tennessee, Knoxville, TN, USA. His research interests include the application of computational modeling and molecular simulations to accelerate the development of molecular therapeutics and novel high-value plant-derived materials. Contact him at msmit316@utk.edu.

**OSCAR HERNANDEZ** is currently a Senior Research and Development Staff Member with the Computer Science and Mathematics Division, Oak Ridge National Laboratory, Oak Ridge, TN, USA. He works on all aspects of high-performance computing including programming models, compilers, and high-performance tools. Contact him at oscar@ornl.gov.

**JEREMY C. SMITH** is currently a Biophysicist and a Governor's Chair with the University of Tennessee, Knoxville, TN, USA, and the Director of the UT/ORNL Center for Molecular Biophysics. He has performed and directed research in high-performance computer simulation of biological macromolecules, neutron scattering in biology, the physics of proteins, enzyme catalysis, bioenergy, environmental biogeochemistry, and the analysis of structural change in proteins for the past four decades. Contact him at smithjc@ornl.gov.