# On the Scalability of Computing Genomic Diversity Using SparkLeBLAST: A Feasibility Study

Ritvik Prabhu*, Bernard Moussad*, Karim Youssef†, Emil Vatai‡, and Wu-chun Feng*
*Department of Computer Science, Virginia Tech, USA
{ritvikp,bernardm,wfeng}@vt.edu
†Center for Applied Scientific Computing, Lawrence Livermore National Laboratory, USA
{youssef2}@llnl.gov
‡RIKEN Center for Computational Science, Japan
{emil.vatai}@riken.jp

***Abstract*—Studying the genomic diversity of viruses can help us understand how viruses evolve and how that evolution can impact human health. Rather than use a laborious and tedious wet-lab approach to conduct a genomic diversity study, we take a computational approach, using the ubiquitous NCBI BLAST and our parallel and distributed SparkLeBLAST, across 53 patients (∼40,000,000 query sequences) on Fugaku, the world's fastest homogeneous supercomputer with 158,976 nodes, where each code contains a 48-core A64FX processor and 32 GB RAM.**

**To project how long BLAST and SparkLeBLAST would take to complete a genomic diversity study of COVID-19, we first perform a feasibility study on a subset of 50 query sequences from a single COVID-19 patient to identify bottlenecks in sequence alignment processing. We then create a model using Amdahl's law to project the run times of NCBI BLAST and SparkLeBLAST on supercomputing systems like Fugaku. Based on the data from this 50-sequence feasibility study, our model predicts that NCBI BLAST, when running on all the cores of the Fugaku supercomputer, would take approximately 26.7 years to complete the full-scale study. In contrast, SparkLeBLAST, using both our query and database segmentation, would reduce the execution time to 0.026 years (i.e., 22.9 hours) – resulting in more than a 10,000× speedup over using the ubiquitous NCBI BLAST.**

***Index Terms*—computational science, NCBI BLAST, SparkLeBLAST, COVID-19, genomic diversity, pairwise sequence search, feasibility, scalability, A64FX CPU, supercomputer.**

## I. Introduction

The genomic diversity amongst viral strains creates a challenge in effectively understanding of a virus. Thus, genomic diversity studies can be significantly beneficial in order to accelerate research related to their study. One such virus, the SARS-CoV-2 coronavirus (i.e., COVID-19), infected more than 775,000,000 people and resulted in more than 7,000,000 deaths worldwide from January 2020 to May 2024 [1]. Understanding how the virus evolves within different patients in response to their immune systems or as a result of interactions with other microorganisms is crucial for predicting future outcomes and reducing complications of the infection [2].

While COVID-19 is the most recent pandemic event, it actually evolved over decades starting in 2003. The emergence of SARS-CoV-1 in 2003 highlighted the need for ongoing genomic monitoring and vaccine development. Despite the efforts for the creation of a vaccine against SARS-CoV-1 during the period of 2003-2004, the subsequent emergence of MERS in 2012 and ongoing coronavirus research underscored the importance of viral evolution [3]. To that end, we choose COVID-19 as the basis for a feasibility study on large-scale genome diversity, due to its recent outbreak as well as the abundance of data available related to the the virus [2], [4].

Genome diversity analysis involves a computationally expensive taxonomic assignment step, requiring exhaustive alignment of nucleotide sequences from collected samples against a reference sequence database, as shown in the red squares of the COVID-19 genome diversity pipeline [2] in Fig. 1. This step is typically performed using BLAST (Basic Local Alignment Search Tool) [5], a widely used algorithm for comparing biological sequence information. However, the relative slowness of BLAST [6] significantly impacts the scalability of the analysis pipeline, particularly as new projects aim to collect and analyze large-scale patient data. For example, Shen et al. [2] studied the interactions of the COVID-19 virus with various microorganisms, such as bacteria, fungi, and other viruses, necessitating a comprehensive taxonomic assignment against the `nt` reference database [7], the largest publicly available nucleotide database at 1.78 TB in size.

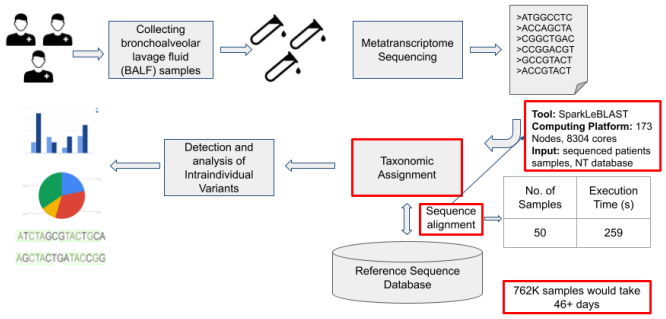



Fig. 1: The genomic diversity pipeline for detecting and analyzing intra-host variants for COVID-19.

This project seeks to conduct a feasibility study to characterize the performance and scalability of the genome diversity analysis pipeline with SparkLeBLAST [8], our massively

parallel realization of NCBI BLAST, in order to project the computational cost of performing a full-scale genome diversity study on all the 158,976 A64FX CPU nodes of the Fugaku supercomputer. To that end, we perform a smaller but similar evaluation on Ookami [9], a 173-node A64FX CPU-based cluster modeled after Fugaku, which reached #1 on the Top500 fastest supercomputers in the world in November 2021 [10], boasting a peak performance of 488 quadrillion floating-point operations per second.

By evaluating SparkLeBLAST on Ookami, we aim to identify performance bottlenecks and predict the scalability of SparkLeBLAST's hybrid approach for full-scale runs on Fugaku, which is *three orders of magnitude* larger than Ookami. This preliminary study lays the groundwork for future full-scale genome diversity analysis, ultimately contributing to more effective management of the COVID-19 pandemic as well as any future evolutions of the virus.

In short, we make the following contributions in this paper:

- A feasibility study on the Ookami supercomputer to identify bottlenecks and gather data for scaling up to the Fugaku supercomputer.
- A model that projects the run times for NCBI BLAST and SparkLeBLAST at scale.
- A projection of the runtime to conduct the entire genome diversity study on Fugaku and identification of challenges unique to Fugaku that are not observed on Ookami.

## II. Related Work

Here we summarize past work on parallelizing BLAST, along with the need for genome diversity studies for prominent viruses such as COVID-19 and HIV.

### A. Parallelization of BLAST

Past attempts to parallelize BLAST simply entailed running multiple instances of BLAST, each on a different query sequence against the same database. This technique is referred to as *query segmentation*, and while moderately effective, it still suffers from excessive I/O, due to having to frequently page different portions of the sequence database (e.g., 1.78 TB for the nucleotide (`nt`) database) between memory and storage as 1.78 TB is too large to fit within system memory. Such an approach has been leveraged in the case of [11].

Rather than segment the query file (as in [11]), mpiBLAST proposed to segment the database in such a way that each portion of the database could fit entirely in memory, effectively eliminating unnecessary and slow paging between system memory and disk, thus significantly accelerating pairwise sequence search [12], [13]. As a consequence, mpiBLAST experienced broad adoption by the scientific computing community from 2003–2016, including pre-packaging within many Linux distributions. However, because mpiBLAST was intertwined and customized to a specific version of NCBI BLAST, updating mpiBLAST with each new release of NCBI BLAST became tedious and cumbersome, thus leading to its eventual discontinued support.

SparkLeBLAST [8] eliminates the need to tightly couple a distributed processing framework, i.e., MPI, with a specific version of NCBI BLAST by leveraging Apache Spark [14] instead of MPI to pipe database segments to arbitrary instantiations of BLAST executables. That is, like mpiBLAST, SparkLeBLAST parallelizes BLAST by segmenting the reference database among compute nodes and replicating the query to each node and does so by only having to change a *single* line of code whenever a new version of NCBI BLAST is released.

### B. COVID-19 Diversity

The rapid mutation of the COVID-19 virus led to significant challenges in containment and treatment [15]. On top of the mutation rate, these variants' initial sightings occurred worldwide [16]. The benefit of having the ability to profile human genomics quickly and at scale allows for a faster understanding of how the virus mutates as well as how it spreads and between what groups of people. Beyond understanding the spread, such a level of profiling allows researchers to determine the virus' resistance to current antiviral drugs [17].

### C. COVID-19 Genomic Studies

Another prominent virus, HIV-1, experienced global spread in the 1980s and 1990s [18]. The organismic origins and spread of the HIV-1 virus are rather well known yet its "genetic variability" [19] was also a detrimental factor to vaccine development, similar to SARS-CoV-2. Due to the level of study that HIV-1 has seen, much is understood regarding its mutation rate, drug resistances, as well as the development of antiretroviral therapies [20], [21]. Progress such as this can be greatly accelerated with the ability to perform large-scale genome diversity studies.

## III. Methodology

SparkLeBLAST parallelizes BLAST by segmenting the reference database among compute nodes and replicating the query to each node. While this database segmentation optimizes I/O and load balancing between parallel workers, it suffers from communication overhead that is needed to merge and sort the final output, thus limiting its scalability. To extend the scalability of SparkLeBLAST, a hybrid query and database segmentation approach is implemented. This approach is projected to scale SparkLeBLAST across tens of thousands of cores, thereby providing a scalable genome diversity analysis pipeline, capable of handling tens of thousands of samples and delivering timely and crucial insights to enhance our understanding of viral diseases, such as COVID-19.

To identify the "ideal" segmentation size, our goal is to utilize as much of the memory of each node in order to simultaneously minimize paging and minimize the overhead cost of parallelization, thus maximizing efficiency. Given that the memory of each A64FX-based node is 32 GB, we project that the "ideal" segmentation of the `nt` database occurs at 55 nodes, where each database segment of 32 GB (i.e., 1.78 TB / 55) fits in the available 32 GB of memory per node.

At a high level, our study evaluates the performance and scalability of NCBI BLAST and SparkLeBLAST by conducting computational experiments on the 173-node Ookami A64FX-based supercomputer. We analyze the run times of NCBI BLAST and SparkLeBLAST as the node count increases, assess threading efficiency, and derive a predictive model for execution time using Amdahl's law. By plotting the run times, we identify scaling behavior and the points of diminishing returns. Additionally, we perform a bottleneck analysis to identify phases that dominate run time and determine the optimal number of nodes for efficient parallelization.

### A. Experimental Approach

We conduct a feasibility study on Ookami, a supercomputer with 173 A64FX-based CPU nodes, where each node is equipped with 32 GB of memory, as in Fugaku. We use the results from this feasibility study to project the performance and scalability of SparkLeBLAST for a full-scale genomic study on Fugaku, which consists of 158,976 nodes. The sample query file used for the initial experiments contains a subset of 50 random query sequences, while the full query file contains 761,919 query sequences collected from a single patient (out of the 53 total patients found in the study conducted by Shen et al. [2]). We reference the sample query file against the `nt` database, which is 1.78 TB in size. Since this is a genomic study on nucleotides, we use `blastn` from NCBI BLAST v2.13.0 for our experiments.

Our approach involves projecting the results from Ookami (173 nodes) to Fugaku (158,976 nodes), a system nearly 1,000 times larger. This projection is necessary because running experiments at Fugaku's full scale incurs substantial costs due to its massive power and cooling requirements (29,899 kW $\approx$ 29.9 MW). By extrapolating from our feasibility study, we can estimate the performance and resource requirements for conducting a full genome diversity study on the Fugaku supercomputer.

That is, our feasibility study processes 50 query sequences on 173 nodes of Ookami. Scaling this to a single patient's entire dataset involves 761,919 query sequences, processed on 158,976 nodes. Further expanding to our available dataset of 53 patients increases the workload to 40,381,707 sequences (with an average length of 100 nucleotides per sequence) on the same number of nodes.

A note about genome data. We often measure sequence length in base pairs (bp), which represent the paired nucleotides on complementary DNA or RNA strands. In our study, the average sequence length is 100 bp. Thus, for 53 patients, we process approximately 4.04 billion base pairs (i.e., 40,381,707 sequences × 100 bp/sequence). Additional diversification of the data would further enrich the study. For instance, incorporating the NCBI Virus database [22] would add 8,932,711 query sequences with an average length of 29,500 base pairs, equating to about 263.5 billion base pairs. Additionally, the GISAID dataset for COVID-19 [23] contributes 16,933,062 genome sequence submissions, also averaging 29,500 base pairs in length and adding another 499.5 billion base pairs. These datasets exemplify the vast amount of genome data available.

When combined, these datasets represent over **767 billion base pairs** of genetic information. These examples illustrate how the problem size can continue to expand, potentially outpacing even Fugaku's considerable resources. The exponential growth in the number of base pairs that need to be processed underscores the computational challenges in large-scale genome studies and the need for advanced supercomputing capabilities.

### B. Performance Measurement

We measure the run time of SparkLeBLAST searches on the sample query file as the number of nodes increases. All the NCBI BLAST executables wrapped by SparkLeBLAST are configured to utilize 48 threads, with each core mapped to a single thread, thereby maximizing the utilization of each node. By plotting the run time of SparkLeBLAST against the increasing number of nodes, we can visualize the relationship between node count and run time, identifying points where diminishing returns begin to appear.

### C. Predictive Model for BLAST Execution Time

To better understand the performance scaling of BLAST, we start with Amdahl's law, which is used to find the maximum improvement to an overall application when only part of the system is improved. Amdahl's law is given by:

$$S(N) = \frac{1}{(1-P) + \frac{P}{N}}$$

where $S(N)$ is the speedup with $N$ threads and $P$ is the parallel fraction of the workload [24].

For BLAST, we extend this model to handle multiple queries $Q$ by assuming that the time $T$ scales linearly with $Q$:

$$T_{\text{Db}}(N, Q) = Q \times T_1 \times \left((1-P) + \frac{P}{N}\right) \tag{1}$$

where $T_1$ is the time to process one query on one thread. Eqn. (1) can be used to forecast the performance of BLAST.

To avoid any idling of resources, a greedy approach would be appropriate where any additional patients would have their query sequences appended to the original 761,919 query sequences. Hence, when forecasting NCBI BLAST run time for all the available patients, we simply replace $Q$ in Eqn. (1) with the total number of query sequences across all the patients.

### D. Predictive Model for SparkLeBLAST Execution Time

SparkLeBLAST's segmentation by a database-only method is similar to BLAST's approach, as both involve segmenting the database. The key difference is that BLAST segments across threads, while SparkLeBLAST segments across nodes. Therefore, we simply redefine $N$ in Eqn. (1) to be the number of nodes instead of the number of threads.

Our observations, as seen on Fig. 2a, indicate diminishing returns in performance improvements with the addition of more nodes when relying solely on database segmentation.

| Implementation | Method | 1 Patient (Single Thread) | 53 Patient (Single Thread) | 1 Patient (All Nodes) | 53 Patients (All Nodes) |
|---|---|---|---|---|---|
| NCBI BLAST | Database Segmentation | **348.3 days** | **18459.9 days** | 184 days | 9,752 days |
| SparkLeBLAST | Database Segmentation | N/A | N/A | 46.4 days | 2,459.2 days |
| SparkLeBLAST | Query & Database Segmentation | N/A | N/A | **0.018 days** | **0.95 days** |

TABLE I: Summary of approaches and estimated run times for COVID-19 genomic studies. The study processes 761,919 query sequences per patient against a 1.78 TB nucleotide database. All Nodes refers to utilizing all 158,976 nodes on Fugaku. N/A indicates that the method is not applicable for single thread execution.

Integrating a hybrid approach that combines both database and query segmentation appears to offer the optimal utilization of all available computing resources. Let

$$N_d = \left\lceil \frac{D}{M} \right\rceil$$

where $N_d$ is the ideal number of nodes for database segmentation, $D$ is the size of the database, and $M$ is the memory per node.

When both database and query segmentation are considered, we distribute the queries evenly across the nodes. We will assume that all the queries are roughly the same size. If the number of nodes $N$ is greater than $N_d$:

$$\text{Groups} = \left\lfloor \frac{N}{N_d} \right\rfloor$$

Thus, the number of queries per group is:

$$\frac{Q}{\left\lfloor \frac{N}{N_d} \right\rfloor}$$

We can extend Eqn. (1) to obtain the time per group:

$$T_{\text{group}} = \frac{Q}{\left\lfloor \frac{N}{N_d} \right\rfloor} \times T_1 \times \left( (1-P) + \frac{P}{N_d} \right)$$

Hence,

$$T_{\text{DB + Query}}(N, Q) = \max(T_{\text{group}})$$

We can safely assume that each group will take roughly the same amount of time since all the queries are divided among all the threads equally and all the queries are assumed to be of equal length, therefore:

$$T_{DB+Query}(N, Q) = \frac{Q}{\left\lfloor \frac{N}{N_d} \right\rfloor} \times T_1 \times \left( (1-P) + \frac{P}{N_d} \right) \quad (2)$$

By combining Eqns. (1) and (2), we can formulate the conditional formula for the estimated search time using SparkLeBLAST:

$$T(N, Q) = \begin{cases} \frac{Q}{\left\lfloor \frac{N}{N_d} \right\rfloor} \times T_1 \times \left( (1-P) + \frac{P}{N_d} \right) & \text{if } N_d \leq N \\ Q \times T_1 \times \left( (1-P) + \frac{P}{N} \right) & \text{if } N_d > N \end{cases} \quad (3)$$

This conditional formula provides the estimated time for the most efficient method. When $N_d$ is greater than the number of nodes available, it is more efficient to perform database segmentation only, as this maximizes resource utilization and the cost of parallelism does not outweigh the performance improvement benefits. Conversely, when $N$ is greater than or equal to $N_d$, a hybrid approach of database and query segmentation is more efficient, as it ensures optimal utilization of all available computing resources.

As mentioned in §III-C we want to use a greedy algorithm to avoid idling of compute resources. Hence, when forecasting SparkLeBLAST run time for all the available patients, we simply replace $Q$ in Eqn. (3) with the total number of query sequences across all the patients.

### *E. Bottleneck Analysis*

To identify potential bottlenecks in SparkLeBLAST, we analyzed the time distribution across different phases of the search process on the sample query file.

*1) Time Distribution Across Phases:* We measured the percentage of time spent on various phases of the SparkLeBLAST search, including the BLAST search operation and the intermediate output phase. The results allowed us to identify the phases that dominated the run time.

*2) Node Scalability:* We observed the scalability of SparkLeBLAST by examining how the percentage of time spent on the distributed search phase changed with the number of nodes. This analysis helped us identify the optimal number of nodes for efficient parallelization and highlighted the diminishing returns beyond a certain point.

## IV. Results

In this section, we present the results of our experiments and analyses on Ookami. We evaluate the performance of SparkLeBLAST, our optimized version of BLAST, on the Fugaku supercomputer. Our findings highlight significant improvements in execution time, showcasing the potential of SparkLeBLAST for large-scale genomic studies. We systematically examine different configurations, node utilizations, and segmentation approaches to provide a thorough understanding of SparkLeBLAST's performance characteristics and scalability.

Table I summarizes the key findings of our feasibility study for large-scale genomic diversity analysis at exascale. These results demonstrate the significant performance improvements achieved by SparkLeBLAST, particularly when using both query and database segmentation. The approach reduces the processing time for a 53-patient study from over 26 years (using NCBI BLAST) to less than a day, making large-scale genomic diversity studies feasible on exascale systems. In

the following subsections, we systematically examine different configurations, node utilizations, and segmentation approaches to provide a thorough understanding of SparkLeBLAST's performance characteristics and scalability.

### A. Genomic Diversity: Modeling BLAST Performance

We now utilize the BLAST predictive model to estimate the time required for BLAST to perform a comprehensive COVID-19 genomic study using the entirety of Fugaku.

As previously mentioned, the database size is approximately 1749 GB, our query file contains 761,919 query sequences, and Fugaku comprises 158,976 nodes. We ran experiments on configurations with 1, 2, 4, 8, 16, and 32 threads. The experimental data was then used to fit a non-linear least squares regression line to calculate the $P$ value in Amdahl's law. The regression analysis yielded an $R^2$ value of 0.94, indicating a strong correlation between the model predictions and the actual data. An $R^2$ value of 0.94 means that 94% of the variance in the observed data can be explained by the model, showcasing a high degree of accuracy and reliability in the predictive model. The $P$ value obtained from the regression analysis was 0.4718. This $P$ value represents the parallel fraction of the workload, indicating that approximately 47.18% of the workload can be parallelized, while the remaining 52.82% is inherently serial.

When we apply Eqn. (1), we verify that a single thread is expected to take approximately 348.3 days to process all the query sequences for a single patient. This is given that a single query for one thread is 39.5 seconds. When utilizing all 158,976 nodes (which would be 7,630,848 threads), the predicted time for NCBI BLAST search is reduced to approximately 184 days, which is only $\sim$1.9 times faster than a single thread. If we extend the analysis to all of the 53 patient data available to us, the processing time scales linearly. For the 53-patient study, we multiply the single-patient processing time by 53, as each patient's genome is processed independently. When we apply Eqn. (1), we find that a single thread is expected to take approximately 18,460 days (50.6 years) to process the query sequences for all 53 patients. When utilizing all 158,976 nodes (which would be 7,630,848 threads), the predicted time for NCBI BLAST search is reduced to approximately 9,752 days (26.7 years). The poor speedup occurs because the database segments per thread become so small that the parallelization overhead dominates, leading to diminished performance despite the extensive parallelism.

### B. SparkLeBLAST Search

The blue line in Fig. 2a shows the run time of SparkLeBLAST search as the number of nodes increases. We observe a exaggerated speed up as the number of nodes increases up to 16 nodes. Beyond this point, the decrease in run time becomes less noticeable, indicating a diminishing return in performance improvement with the addition of more nodes. The parallel behaviour of SparkLeBLAST is similar to that observed in NCBI BLAST, where the database is divided across all the nodes, enabling simultaneous sequence searches across all segments. Each node searches for hits in its own database segment for one query sequence at a time. Once the sequence hits are complete, it moves on to the next query sequence. This parallel processing significantly reduces the overall run time, especially with the initial increase in nodes.

### C. Time Spent on Each Phase of SparkLeBLAST Search

Fig. 2b shows the percentage of time spent on different phases of the SparkLeBLAST search. It is evident that the search time is dominated by the BLAST search operation (depicted in blue) and the intermediate output phase, which involves grouping each query sequence result, sorting by score, and selecting the top k sequences (depicted in red). Ideally, we would like to see the majority of the time spent on the BLAST search operation.

From Fig. 2b we observe that the percentage of time spent on the distributed search phase increases with the number of nodes up to 32 nodes. Beyond this point, the percentage starts to decrease, indicating that the cost of parallelism outweighs the benefits. As analyzed in §III, the distributed search phase peaks at 55 nodes. The inefficiency beyond 55 nodes is due to each node holding a database segment that is too small to effectively parallelize. Consequently, the search phase becomes extremely fast due to the small database size, causing the entire compute time to be dominated by I/O operations. This highlights the importance of maximizing the database size on each node to ensure optimal performance and minimize I/O overhead.

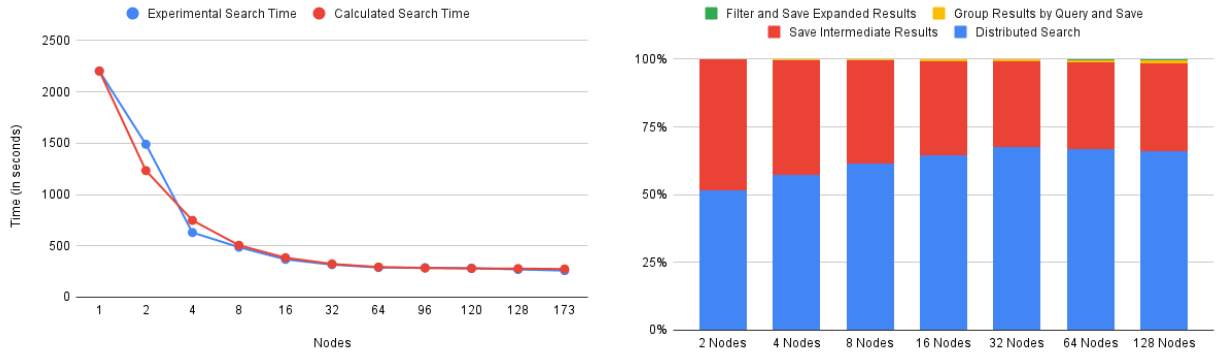
### D. Verification of SparkLeBLAST Predictive Model

To verify the predictive model for SparkLeBLAST, we conducted experiments using 50 queries against the 1.78 TB nt database. The experiments were run on configurations with 2, 4, 8, 16, 32, and 64 nodes. The experimental data was then used to fit a non-linear least squares regression line to calculate the $P$ value in Amdahl's law. The regression analysis yielded an $R^2$ value of 0.92, indicating a strong correlation between the model predictions and the actual data.

The $P$ value obtained from the regression analysis was 0.8805. This high parallel fraction suggests that SparkLeBLAST is well-suited for parallel execution, benefiting significantly from the addition of more nodes.

We validate the model only for database segmentation, which is given by Eqn. (1). The equation for query and database segmentation (and subsequently the conditional equation) is an extension of the database segmentation equation. Therefore, validating the model for database segmentation also ensures the validation of the model for the combined query and database segmentation.

To validate the predictive model, we compared the model's predictions with the results of further experiments. We extended our experiments to include configurations with 2, 4, 8, 16, 32, 64, and the previously unseen 96, 120, 128, and 173 nodes. As illustrated by the red line in Fig. 2a, the model's predictions closely matched the experimental results across the different node configurations.




(a) (b)

Fig. 2: Comparison of (a) SparkLeBLAST predictive model's estimated execution times with experimental results for various node configurations, and (b) time spent on each phase of the SparkLeBLAST search process for multi-node configurations.

Fig. 2a shows that as the number of nodes increases, the model's predictions remain closely aligned with the experimental results. This consistency reinforces the model's accuracy and reliability in predicting the execution time of SparkLeBLAST. The close alignment between the model and experimental data across a wide range of node configurations indicates that the model effectively captures the performance characteristics of SparkLeBLAST, even for configurations not previously tested during the initial regression analysis.

The verification process demonstrates the robustness of our predictive model. By accurately estimating the execution time across various node configurations, the model provides valuable insights for optimizing SparkLeBLAST deployments, ensuring efficient use of computational resources.

### *E. Genomic Diversity: Modeling SparkLeBLAST Performance*

We now utilize the SparkLeBLAST predictive model to estimate the time required for two configurations of SparkLeBLAST — (1) database segmentation and (2) database and query segmentation — to perform comprehensive COVID-19 genomic studies using the entirety of Fugaku. As previously noted, the database size is approximately 1784 GB; our query file contains 761,919 query sequences per patient; and Fugaku comprises 158,976 nodes. For database segmentation only, we apply Eqn. (1). For a single patient study, a single node is expected to take approximately 388.6 days, while utilizing all 158,976 nodes reduces the predicted time to approximately 46.4 days. Extending this to the 53-patient study, a single node would require approximately 20,595.8 days (56.4 years), while all nodes would complete the task in approximately 2,459.2 days (6.7 years). For the combined database and query segmentation approach, each group consists of 55 nodes, with each node handling roughly 32 GB of the database. Applying the extended model equation (Eqn. (2)), SparkLeBLAST on all 158,976 nodes of Fugaku is predicted to take approximately 26.2 minutes for a single patient study and 22.9 hours (0.954 days) for the 53-patient study. The significant reduction in execution time when using the combined "database and query segmentation" configuration highlights its efficiency, especially for large-scale genome diversity studies involving multiple patients. By optimally distributing both the database and query workload across Fugaku's extensive node resources, SparkLeBLAST achieves rapid and efficient processing of multiple patient genomes, enabling timely insights and analyses for genomic diversity studies.

## V. Discussion

A more expansive genome diversity study that encompasses more patients and more diverse datasets would capture newer viral variants and reveal different virus-microorganism interactions across populations, enhancing our understanding of viral evolution and improving strategies for vaccine development. Using NCBI BLAST for such extensive studies could lead to the generation of obsolete results before the genome diversity study is even completed, given the rapid evolution of viruses. The accelerated processing and scalability capabilities of SparkLeBLAST delivers timely analysis that is crucial in addressing the challenges posed by fast-evolving pathogens.

In addition to the time savings on the computationally expensive taxonomic assignment step in the genome diversity pipeline, the economic impact of the performance differences between NCBI BLAST and SparkLeBLAST is substantial. With Japan's average commercial power cost of \$233.55/MWh[1] and Fugaku's power consumption of 29.9 MW, NCBI BLAST, running for 26.7 years, would cost approximately **\$1.63 billion** in energy consumption; while SparkLeBLAST with database segmentation, taking 6.7 years, would incur about **\$0.41 billion** in energy costs. In stark contrast, SparkLeBLAST with hybrid database and query segmentation, completing in just 22.9 hours, would only cost about **\$0.00016 billion** (i.e., \$159,914). These calculations demonstrate the enormous cost savings of SparkLeBLAST, particularly the hybrid segmentation approach, when used in a genome diversity study; it could save over \$1.5 billion

[1]Average power costs in Japan: https://www.global-climatescope.org/markets/jp/

in energy costs compared to the traditional NCBI BLAST approach for this 53-patient study.

The projections made for the Fugaku supercomputer are based on runs made on a smaller Fugaku-like supercomputer called Ookami. When migrating to a larger system, like Fugaku, the latency of shared file systems starts to suffer due to increased I/O demands and network complexities. Furthermore, due to misconfigured memory-mapped files (`mmap`) on the Fugaku file system, which NCBI BLAST uses for file I/O operations, we might experience further I/O issues that increase latency.

We confirmed the above assertion when NCBI BLAST incurred a $7\times$ slowdown when running on a single node with 48 threads on Fugaku (vs. Ookami). The database and output files residing in the global file system exacerbate this issue, as constant writes and reads significantly increase latency. This is due to the contention and overhead associated with accessing the global file system across many nodes, leading to bottlenecks. To verify this, we ran BLAST on a query file consisting of 50 query sequences against a much smaller database of 284 MB. We chose a smaller database intentionally to ensure that the database file, query file, and output file could all reside within the local storage of the node, without exceeding the local storage limitation of 80 GB. By doing so, we minimized the dependency on the global file system and eliminated the associated I/O contention. This approach allowed us to match the runtime observed on Ookami, where no provisions for local storage were necessary. Thus, optimizing file I/O on Fugaku will be critical in achieving the predicted performance gains with SparkLeBLAST.

## VI. Future Work

While our study provides a comprehensive analysis of SparkLeBLAST's performance and scalability, we plan further exploration in the following areas: (1) conducting experiments on the actual Fugaku supercomputer to validate our predictions and refine the hybrid approach based on real-world performance data, (2) investigating optimization techniques for the BLAST search operation and intermediate output phase to improve runtime and enhance efficiency, and (3) tuning file system parameters and memory-management techniques to minimize I/O-induced latency introduced by the large file system of Fugaku.

## VII. Conclusion

Our feasibility study, conducted on the 173-node Ookami supercomputer and projected to the 158,976-node Fugaku supercomputer, demonstrates the transformative potential of SparkLeBLAST for large-scale genome diversity studies. For a modest 53-patient study, SparkLeBLAST with hybrid query and database segmentation is projected to complete the genome diversity study in just 0.026 years (i.e., 22.9 hours) on Fugaku, a dramatic improvement from the approximately 26.7 years estimated for NCBI BLAST.

The projected performance of SparkLeBLAST has important implications for population-scale genomics and personalized medicine. The ability to rapidly analyze large cohorts of patient genomes could accelerate the identification of disease-linked genetic variations and enhance our understanding of genetic diversity. On the other hand, our study also reveals challenges in I/O optimization and scalability that require further investigation. As genome datasets continue to expand, the development of increasingly sophisticated and optimized computing strategies will be essential to keep pace with the growing computational demands of genetic research.

## Acknowledgment

This work was supported in part under the auspices of the U.S. Department of Energy by Lawrence Livermore National Laboratory under Contract DE-AC52-07NA27344 (LLNL-CONF-866612).

With respect to computing resources, we thank Stony Brook Research Computing and Cyberinfrastructure as well as the Institute for Advanced Computational Science at Stony Brook University for access to the innovative high-performance Ookami computing system via NSF grant OAC-1927880.

## References

[1] World Health Organization, "COVID-19 Dashboard," https://data.who.int/dashboards/covid19/.

[2] Z. Shen, Y. Xiao, L. Kang, W. Ma, L. Shi, L. Zhang, Z. Zhou, J. Yang, J. Zhong, D. Yang *et al.*, "Genomic Diversity of Severe Acute Respiratory Syndrome–Coronavirus 2 in Patients with Coronavirus Disease 2019," *Clinical Infectious Diseases*, vol. 71, no. 15, pp. 713–720, 2020.

[3] G. Abdolmaleki, M. A. Taheri, S. Paridehpour, N. M. Mohammadi, Y. A. Tabatabaei, T. Mousavi, and M. Amin, "A Comparison between SARS-CoV-1 and SARS-CoV2: An Update on Current COVID-19 Vaccines," *DARU Journal of Pharmaceutical Sciences*, vol. 30, no. 2, pp. 379–406, 2022.

[4] S. Di Giorgio, F. Martignano, M. G. Torcia, G. Mattiuz, and S. G. Conticello, "Evidence for Host-Dependent RNA Editing in the Transcriptome of SARS-CoV-2," *Science Advances*, vol. 6, no. 25, p. eabb5813, 2020.

[5] S. F. Altschul, W. Gish, W. Miller, E. W. Myers, and D. J. Lipman, "Basic Local Alignment Search Tool," *Journal of Molecular Biology*, vol. 215, no. 3, pp. 403–410, 1990.

[6] I. Korf, "Serial BLAST Searching," *Bioinformatics*, vol. 19, no. 12, pp. 1492–1496, 2003.

[7] nt [Internet]. Bethesda (MD): National Library of Medicine (US), National Center for Biotechnology Information, 2004–2024 Jul, Available from: https://www.ncbi.nlm.nih.gov/nucleotide/.

[8] K. Youssef and W. Feng, "SparkLeBLAST: Scalable Parallelization of BLAST Sequence Alignment Using Spark," in *Proceedings of the 20th IEEE/ACM International Symposium on Cluster, Cloud and Internet Computing (CCGRID)*, 2020, pp. 539–548.

[9] Stony Brook University, "Ookami," https://www.stonybrook.edu/ookami/.

[10] "Supercomputer Fugaku - Supercomputer Fugaku, A64FX 48C 2.2GHz, Tofu interconnect D," https://top500.org/system/179807/.

[11] M. R. de Castro, C. d. S. Tostes, A. M. Dávila, H. Senger, and F. A. da Silva, "SparkBLAST: Scalable BLAST Processing Using In-Memory Operations," *BMC Bioinformatics*, vol. 18, pp. 1–13, 2017.

[12] A. Darling, L. Carey, and W. Feng, "The Design, Implementation, and Evaluation of mpiBLAST," *Proceedings of 4th International Conference on Linux Clusters (in conjunction with ClusterWorld)*, vol. 2003, pp. 13–15, 2003.

[13] H. Lin, A. E. Darling, P. Balaji, J. S. Archuleta, C. P. Sosa, J. D. Gans, and L. Carey, "mpiBLAST," https://sourceforge.net/p/mpiblast, Nov 2009.

[14] R. Guo, Y. Zhao, Q. Zou, X. Fang, and S. Peng, "Bioinformatics Applications on Apache Spark," *GigaScience*, vol. 7, no. 8, p. giy098, 2018.

[15] B. Korber, W. M. Fischer, S. Gnanakaran, H. Yoon, J. Theiler, W. Abfalterer, N. Hengartner, E. E. Giorgi, T. Bhattacharya, B. Foley *et al.*, "Tracking Changes in SARS-CoV-2 Spike: Evidence that D614G Increases Infectivity of the COVID-19 Virus," *Cell*, vol. 182, no. 4, pp. 812–827, 2020.

[16] P. Gupta, V. Gupta, C. M. Singh, and L. Singhal, "Emergence of COVID-19 Variants: An Update," *Cureus*, vol. 15, no. 7, 2023.

[17] D. Zella, M. Giovanetti, E. Cella, A. Borsetti, M. Ciotti, G. Ceccarelli, G. D'Ettorre, A. Pezzuto, V. Tambone, L. Campanozzi *et al.*, "The Importance of Genomic Analysis in Cracking the Coronavirus Pandemic," *Expert Review of Molecular Diagnostics*, vol. 21, no. 6, pp. 547–562, 2021.

[18] J. Hemelaar, "Implications of HIV Diversity for the HIV-1 Pandemic," *Journal of Infection*, vol. 66, no. 5, pp. 391–400, 2013. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S016344531200312X

[19] M. M. Santoro and C. F. Perno, "HIV-1 Genetic Variability and Clinical Implications," *International Scholarly Research Notices*, vol. 2013, no. 1, p. 481314, 2013.

[20] World Health Organization, "HIV Drug Resistance," https://www.who.int/teams/global-hiv-hepatitis-and-stis-programmes/hiv/treatment/hiv-drug-resistance.

[21] National Institute of Allergy and Infectious Diseases, "Antiretroviral Drug Discovery and Development," https://www.niaid.nih.gov/diseases-conditions/antiretroviral-drug-development.

[22] NCBI Virus [Internet]. Bethesda (MD): National Library of Medicine (US), National Center for Biotechnology Information, 2004–, Available from: https://www.ncbi.nlm.nih.gov/labs/virus/vssi/.

[23] S. Elbe and G. Buckland-Merrett, "Data, Disease and Diplomacy: GISAID's Innovative Contribution to Global Health," *Global Challenges*, vol. 1, no. 1, pp. 33–46, 2017.

[24] Hill, Mark D. and Marty, Michael R., "Amdahl's Law in the Multicore Era," *Computer*, vol. 41, no. 7, pp. 33–38, 2008.