Making Campus Bridging Work for Researchers: A Case Study with mlRho

Abhinav Thota
Indiana University,
Bloomington, IN, 47408
athota@iu.edu

Scott Michael
Indiana University,
Bloomington, IN, 47408
scamicha@iu.edu

Sen Xu
Indiana University,
Bloomington, IN, 47408
senxu@indiana.edu

Bernhard Haubold
Max-Planck Institute for
Evolutionary Biology, Plön,
Germany
haubold@evolbio.mpg.de

Thomas Doak
Indiana University,
Bloomington, IN, 47408
tdoak@indiana.edu

Robert Henschel
Indiana University,
Bloomington, IN, 47408
henschel@iu.edu

ABSTRACT

An increasing number of biologists' computational demands have outgrown the capacity of desktop workstations and they are turning to supercomputers to run their simulations and calculations. Many of today's computational problems, however, require larger resource commitments than even individual universities can provide. XSEDE is one of the first places researchers turn to when they outgrow their campus resources. XSEDE machines are far larger (by at least an order of magnitude) than what most universities offer. Transitioning from a campus resource to an XSEDE resource is seldom a trivial task. XSEDE has taken many steps to make this easier, including the Campus Bridging initiative, the Campus Champions program, the Extended Collaborative Support Service (ECSS) [1] program, and through education and outreach.

In this paper, our team of biologists and application support analysts (including a Campus Champion) dissect a computationally intensive biology project and share the insights we gain to help strengthen the programs mentioned above. We worked on a project to calculate population mutation and recombination rates of tens of genome profiles using mlRho [2], a serial, open-source, genome analysis code. For the initial investigation, we estimated that we would need 6.3 million service units (SUs) on the Ranger system. Three of the most important places where the biologists needed help in transitioning to XSEDE were (i) preparing the proposal for 6.3 million SUs on XSEDE, (ii) scaling up the existing workflow to hundreds of cores and (iii) performance optimization. The Campus Bridging initiative makes all of these tasks easier by providing tools and a consistent software stack across centers.

Ideally, Campus Champions are able to provide support on (i), (ii) and (iii), while ECSS staff can assist with (ii) and (iii). But (i), (ii) and (iii) are often not part of a Campus Champion's regular job description. To someone writing an XSEDE proposal for the first time, a link to the guidelines and a few pointers may not always be enough for a successful application. In this paper we describe a new role for a campus bridging expert to play in closing the gaps between existing programs and present mlRho as a case study.

Categories and Subject Descriptors

B.8.2 [**Hardware**]: Performance and Reliability—*Miscellaneous*; D.2.8 [**Software Engineering**]: Metrics—*complexity measures, performance measures*; J.3 [**Computer Applications**]: Life and Medical Sciences—*Biology and genetics*

General Terms

Performance, Experimentation, Design, Reliability, Human Factors

Keywords

high-throughput, XSEDE, BigJob, pilot-job, genetics, mlRho, performance tuning, optimization

1. INTRODUCTION

As first defined by the National Science Foundation Advisory Committee for Cyberinfrastructure's Task Force on Campus Bridging [3], and later expanded upon by Stewart et al., campus bridging is:

> "...the seamlessly integrated use of cyberinfrastructure operated by a scientist or engineer with other cyberinfrastructure on the scientist's campus, at other campuses, and at the regional, national, and international levels as if they were proximate to the scientist, and when working within the context of a Virtual Organization (VO) make the 'virtual' aspect of the organization irrelevant (or helpful) to the work of the VO. [4]"

In applying this definition of campus bridging to XSEDE, one of the biggest challenges for researchers moving from campus resources to XSEDE resources is being able to scale up their workflows so that they are efficient and can achieve high throughput on the larger XSEDE machines. While Stewart et al. identify key use cases where campus bridging tools can improve a researcher's experience using XSEDE resources, one aspect that is not included in their analysis is the dramatic increase in complexity that is inherent in the computational and data storage systems when a researcher moves from his workstation to an XSEDE resource. This scale up in complexity is in many cases a daunting prospect for a researcher new to XSEDE, who may have relatively little computational experience. Even savvy users who have experience with campus clusters can encounter issues when scaling up to XSEDE resources.

This challenge is distinct from the challenge of providing a canonical software stack, but can be compounded by widely divergent software environments between campus and XSEDE resources. Even when the operating environment is not very different on XSEDE resources, compared to campus resources, researchers can face challenges with the sheer scale and complexity of XSEDE resources. In the situation described in this paper, campus bridging included helping the researchers redesign their experiments and scale up their workflows to move effectively from their local workstations to IU's modest sized cluster environment, and then on to using the much larger XSEDE machines at Texas Advanced Computing Center (TACC). It should be noted that, in general, the differences in software environment between the IU campus resources and the TACC supercomputers were superficial; the real challenges lie in scaling up the applications and navigating the complexity of a much larger system.

In this paper we propose a new role for a campus bridging expert to help address the challenges faced by researchers when moving from their workstation to XSEDE resources. A campus bridging expert helps to bridge the gap between domain science and computer science and helps researchers move their applications from small-scale campus resources to much larger XSEDE resources. A campus bridging expert is mostly a technologist, who is able to apply many different approaches to scaling up applications, but should also be familiar with the common challenges of the scientific domain. The campus bridging expert must be able to communicate effectively about XSEDE resources with researchers who may have little or no supercomputing experience. The campus bridging expert role is distinct from the current Campus Champion and the role filled by the Extended Collaborative Support Service (ECSS) team. As the name implies, the campus bridging expert closes the gap between the Campus Champion role – which is designed to provide information, guidance, and facilitate access to XSEDE resources, and the ECSS role, which is designed to provide in-depth analysis, insight and development for a particular scientific or engineering challenge. The role of campus bridging expert may, in fact, be filled by the same people who are currently Campus Champions or ECSS team members; in many ways the functions of the campus bridging expert are an extension of ECSS and Campus Champion functions. In this paper we discuss how, by filling the role of campus bridging expert, we were able to bridge the gap between many different sized resources and help researchers in the field of population genomics use some of the largest XSEDE resources to conduct a research program at an unprecedented magnitude.

Researchers in the Indiana University (IU) Department of Biology have been conducting studies into population genomics and evolution for some time. A newly developed computer program called mlRho [2] is being used to study ecological and genetic parameters in different populations. While mlRho is a serial code, the investigation we have conducted with our XSEDE allocation was embarrassingly parallel in nature. The work for each species was divided among many computational processes. To manage hundreds of processes for each of the 40+ species of interest, we used the SAGA BigJob pilot-job tool [5], which is currently available on many XSEDE resources. BigJob allowed us to distribute the analyses for each species across thousands of processor cores, depending on the genome sizes. Thus far we have investigated a total of 46 individual genomes. In addition to the use of the BigJob tool, we have done an extensive performance analysis of the mlRho code and have been able to improve the runtime of the code by a factor of more than 50.

The remainder of the paper is organized as follows: section 2 gives some insight into the computational methodologies and principles of biology being explored by the mlRho code. Section 3 outlines our initial estimates for the computational resources necessary to accomplish the research agenda and the steps that were taken to secure an XSEDE allocation. In section 4, we describe how, through tracing and profiling of the code, we were able to successfully optimize it and dramatically increase its performance. Section 5 details the current status of the research program and initial scientific results and in section 6 we present conclusions.

2. SCIENTIFIC BACKGROUND

The amazing biodiversity on our planet has fascinated humans for thousands of years. To understand how this diversity arises and is maintained, it is critical to determine fundamental ecological and genetic parameters (e.g., population sizes, recombination rates) for a range of species. These parameters play important roles in creating opportunities for increasing genetic diversity, population divergence, and speciation. mlRho is a software package which uses next-generation genomic sequencing to generate a novel measure of linkage disequilibrium. Deploying mlRho on XSEDE resources has allowed us to study these important population-genetic parameters in a broad assembly of eukaryotic genomes.

Although there are several methods for determining recombination rates, they can be both time and resource intensive. The mlRho software employs a novel analytic approach that uses a new metric called the zygosity correlation coefficient, which is estimated using maximum likelihood (ML) methods. It only requires single individual genome sequences, but is extremely data intensive. Using mlRho and XSEDE computational resources, we have been able to examine the recombination rates of a plethora of species with accuracy that was previously unachievable.

2.1 Program Description

The mlRho software is a serial program that estimates mutation, recombination, and sequencing error rates from genome sequences [2]. The underlying data consists of assembled sequencing reads obtained from a single diploid individual. Such data are collected, for example, for the 1000 human genome project. mlRho reads a profile consisting of the number of each nucleotide (`A`, `C`, `G`, and `T`) from a file at each sequenced position. Given a mutation and error rate, mlRho computes two probabilities for each profile: The probabilities of observing the profile given that the position is either mutated (heterozygous), or not (homozygous). These probabilities depend on the mutation and error rates. By varying them, mlRho finds the values that maximize the overall likelihood of the data.

While mutation and sequencing error affect individual genome positions, recombination uncouples the evolutionary history of pairs of positions. This is observable as a decorrelation of the zygosity states between pairs of positions. To estimate recombination, mlRho computes the probability of observing profile pairs separated by, say, 1000 nucleotides. This is a function of the recombination rate and the single position likelihoods.

2.2 Linkage Disequilibrium and Recombination Rate

Linkage disequilibrium (LD), i.e., the non-random association of alleles at two or more loci, is an important parameter for many areas of population genetics. In recent years, there has been growing interest in measuring LD across a broad range of species, because a proper understanding of LD would greatly facilitate identifying the genetic loci which underlie important phenotypic variation in natural populations, as well as human diseases. More importantly, LD is a population-genetic property that can help ascertain recombination rate, because recombination is the primary evolutionary force that breaks down LD among genetic loci. Although a substantial body of research has been devoted to elucidating the evolutionary consequences of recombination [6], the forces that determine recombination rates remain poorly understood. For example, we have little idea what determines the occurrence of recombination on a chromosome, what impact local DNA polymorphism has on recombination processes, and how recombination rates change over evolutionary time [7].

Conventional approaches to measuring LD and recombination rates use population-genetic surveys. These surveys require hundreds of individuals and dozens of genetic loci. Using this method, the sampling variance associated with conventional measures of LD, such as D (a measure of LD) and r^2 (the square of the correlation coefficient), is huge. Recombination rates can also be investigated by pedigree analyses and crossing experiments, but these methods cannot provide information on fine-scale recombination rates and are often difficult to perform in model organisms, let alone non-model species. Thus, while we know these values for a few species, there is very little comparative data to understand how these processes vary across many species.

2.3 A Novel Approach to Estimating LD and Recombination Rate

Whole genome sequences of diploid organisms include both alleles at every site of the genome (two copies of each chromosome), which can be used to determine a number of very useful population-genetic parameters in the evolution a species [8]. With the rapid accumulation of whole genome sequences from a large number of species, a maximum likelihood (ML) approach that capitalizes on these data has recently been developed to estimate LD and examine genomic recombination patterns [9, 2].

The general idea behind this approach is that two allelic chromosomes had a common ancestor some time in the past. Since that time, they have become increasingly different, due to mutations changing their sequence. In addition, recombination shuffled the mutations between chromosomes. If we catalog the differences between the two alleles, we can learn about this history of mutation and recombination.

The ML approach uses this information to determine the zygosity correlation (Δ) between all the pairs of sites that are separated by various distances (i.e., the probability of two sites being both homozygous or both heterozygous, or mixed) in a single diploid genome, using the entire set of assembled individual reads. Population genetic theory then links the expected value of Δ to conventional measures of LD, such as the population recombination rate ρ, suggesting that Δ can be used as a valid measure of LD on the population level.

Our simulation results have shown that this ML procedure generates unbiased estimates for theta and zygosity-correlation at a range of sequencing coverage (10-20x) [9]. mlRho is able to handle genomes of all sizes, and linkage analyses over hundreds of thousands of base pairs. It computes the maximum-likelihood estimators of the population mutation rate, θ, the sequencing error rate, ϵ, and the population recombination rate, ρ using the Nelder and Mead algorithm, as implemented by the GNU scientific library. Nonetheless, this analysis is computationally intensive: millions to tens of billions of pairs of sites at a given distance have to be extracted from the input files of up to giga-byte sizes and Δ calculated for every distance ranging from 10^2 to 10^4.

3. TRANSITIONING TO XSEDE

We began with an initial research plan that called for 15 million core hours to completely analyze all target genomes. At this point, we were using Quarry, a local IU resource, but it was clear that we needed a bigger machine to complete the calculations in a reasonable period of time. Quarry is a 2960 core machine, whereas XSEDE machines such as Ranger, Stampede, and Kraken have 10^5 cores (Figure 1). The software and scheduling environment on Quarry is similar to that of XSEDE machines referred to here. The central difference is in the size of the machines and we explain how we made the transition.

3.1 Research Plan

As a first step, we applied for a startup allocation on Ranger and Kraken to benchmark the mlRho application. We also planned to use the startup allocation to conduct a scaling study. A service unit in XSEDE is defined as one core hour on a machine. It became apparent that conducting the study for all 70+ genomes would require at least 15 million SUs. However, given that the researchers are new to the XSEDE ecology and this is the first time they are applying for a large scale allocation, we wanted to stay under the 5 million SU mark. To do this, we pared down the list to 46 diploid eukaryotic genomes. We initially determined to measure 100 kilo basepair (kbp) distances to limit the number of SUs required. Given that in eukaryotes, recombination rates range between $0.001-1$ event per Mbp [10], a 100 kbp window across a genome provides a good basis for capturing the signature of crossover events that occur 1 or 2 times per chromosome arm per meiotic event.

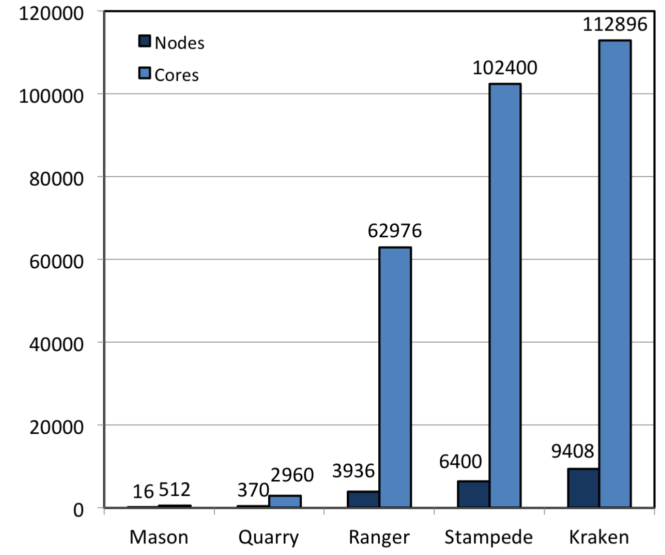

Figure 1: Mason and Quarry are IU resources, while Ranger, Stampede and Kraken are XSEDE resources. The graph shows the large change in the size of the machines that the users experience when they move to XSEDE.

3.2 Scaling Up to XSEDE

For some time the researchers had been running the mlRho code on local IU resources serially, by requesting one compute-node at a time. Given that the IU resource, Quarry, had a serial queue that placed multiple serial jobs from a single user on the same node, this was a feasible course. Unfortunately, most of the major XSEDE resources do not provide such a high throughput queue. The users are charged for the whole node, irrespective of the number of cores their job actually utilizes. Moreover, the larger machines are configured for highly parallel jobs and the schedulers are configured to prioritize larger jobs. Many XSEDE centers have established workarounds for this problem by providing users with wrappers and other software which bundle many serial jobs into larger jobs. We decided to use Ranger for our project, as BigJob was still in an experimental state on Kraken at this point.

3.2.1 SAGA BigJob framework

We used the SAGA BigJob to bundle our serial mlRho simulations. We have integrated mlRho with BigJob and conducted several experiments on Ranger and Stampede which have shown good performance and scalability. BigJob is a pilot-job tool available on many XSEDE resources such as Kraken, Ranger, and Lonestar. Many researchers have successfully used BigJob [11] to bundle hundreds of smaller jobs into larger, more manageable groups of jobs [12, 13].

BigJob maintains the list of processors allocated after the job request becomes active. The user can design the BigJob to assign these processors to start and manage smaller jobs. The main benefit of doing this is that instead of submitting thousands of single core job requests to the queue, we can submit hundreds of large job requests ($\approx$ 500 to 5000 cores) to the queue. This reduces the overall number of job submissions to the queue and thereby, to some extent, time spent waiting in the queue. This job size is also more appropriate for many of the XSEDE machines.

3.2.2 mlRho Scalability tests

We used the BigJob installation already in place on Ranger. The major focus of our tests was to discover how mlRho scaled when we increased the number of concurrent mlRho processes. We took three different organisms based on the size of their genome: a small genome, *F. cylindrus* (diatom), a medium genome, *P. ornithorhynchus* (platypus), and a large genome, *C. familiaris* (dog). Data sizes are provided in Table 1. We ran the mlRho program with each of these genomes starting with 16 concurrent mlRho instances reading from the same data file. We let the processes run for 24 hours and measured how many distances were computed in aggregate. It was clear that the distance travelled per second directly depended on the size of the genome.

Organism Type	Size of profile	Distance/sec	
	(GB)	V 1.10	V 2.1
F. cylindrus (diatom)	0.72	0.0034	0.323
P. ornithorhynchus (platypus)	11	$2.3\times10-4$	0.020
C. familiaris (dog)	31	8.1×10^{-5}	0.005

Table 1: The table lists three organisms which were chosen to represent profiles of different size. The second column shows the size of the profile in gigabytes. The third column shows the distance travelled per second by V1.10 and V2.1 on one core on Ranger and Stampede, respectively. The rate of distance is a function of the size of the genome.

We repeated this for each of the genomes with 32, 64 and 128 concurrent instances. The results are shown in Figure 2. Figure 2 shows that the distance traveled, irrespective of the genome, size, and the number of concurrent instances, increases nearly linearly with number of concurrent instances being run. We have verified this behavior on Stampede with an optimized version of mlRho (V2.1) and obtained similar results. We are unable to repeat this scaling study on Ranger with the improved version of mlRho as Ranger has since been decommissioned.

3.3 XSEDE Allocation

In general, obtaining an XSEDE allocation with sufficient SUs is an important step in the process of executing a computationally intensive research plan. Even to someone who has a computer science background, writing a successful XSEDE proposal is not a trivial task. The computational justification for the allocation needs to be concrete and this is especially true if the request is for more than a few million SUs. XSEDE allocation committees are routinely faced with the fact that the machines are oversubscribed by a factor of 2:1.

We believe that allocation proposal preparation is one key area where users new to XSEDE need significant help. In the case of the mlRho project, research staff from the Indiana University Pervasive Technology Institute (PTI) helped biologists to prepare an allocation proposal. We started by requesting startup allocations on Ranger and Kraken. We benchmarked mlRho on a single core on Ranger and Kraken.

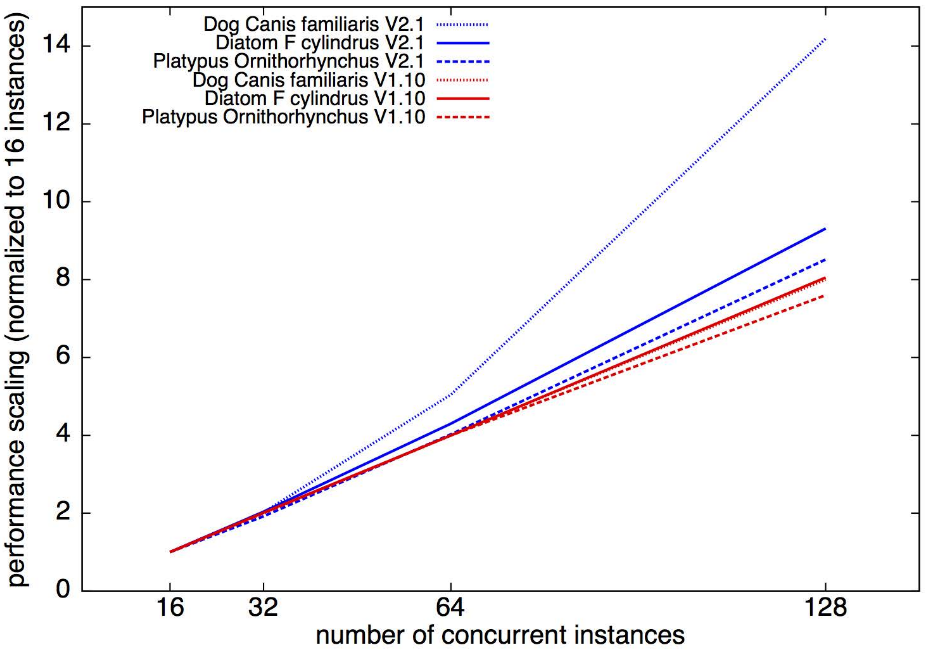

Figure 2: After running 16, 32, 64 and 128 instances of mlRho concurrently using BigJob on Ranger and Stampede with versions 1.10 and 2.1, respectively, we see that amount of work done increases nearly linearly with the number of instances of mlRho. The y-axis shows the total distance travelled by all the instances put together and normalized to 16 instances. The x-axis shows the number of concurrent mlRho processes running.

We then did a scalability study as described in section 3.2.2. It is important to accurately estimate and justify the total number of SUs that we request from XSEDE. We usually also need to justify the science, but if the research is supported by a current grant award from a federal science agency, further justification is not needed. Given that we are requesting shared resources, it is important that we make an effort to analyze and optimize our code. This reduces our own usage time and also turnaround time. We began to look into code optimization shortly after submitting the proposal. Our efforts on the analysis and optimization of the code is discussed in section 4 in more detail.

3.3.1 Design of Experiments

We have carefully constructed a detailed plan on how to proceed with our experiments. In section 3.2, we showed that the mlRho code can scale to 128 processes, with each process reading from the same file with no performance degradation. We estimated that we can feasibly scale up to ≈500 processes reading from the same data file, without noticeable performance degradation. By making multiple copies of the data file, we could scale up to ≈5000 processes. Job requests of 5000 processes on Ranger and Kraken are appropriate and we have not experienced extremely long wait times (more than 48 hours) with past BigJob experiments. We proposed that with this design, we could complete our analysis within four to five months from the time of the allocation award.

In our initial testing of the three sample genomes, we worked out a few issues that could have affected scalability. One of those issues was data access and file striping. When we had scaled up to 128 cores we noticed some intermittent I/O issues. Since all the concurrent instances read from a single file, if that file is singly striped in a Lustre file system, it will introduce a large load on the Lustre servers. We have remedied this issue by moving the data files to the scratch file system on Ranger and striping the data files 16 ways. While this only occurred with the larger (11 and 31 GB) data files, we are aware that this might limit our scalability. In the future, if I/O becomes an issue for scalability, we plan to use multiple copies of the same data file to prevent an excessive number of concurrent processes from reading from the same file at the same time.

4. OPTIMIZATION OF MLRHO

We can not stress enough how important it is for each and every user of a shared supercomputer to analyze and optimize their code. This is especially true in the case of users who develop their own code. Most community codes are analyzed and optimized by their developers. But still, it is not unusual to see a 10% gain in performance just by moving from one compiler to another. Given that XSEDE distributes hundreds of millions of SUs every year, even a 10 or 20 percent improvement will save millions of SUs and lower queue waiting times.

4.1 Performance Analysis

Following its initial release in 2010[2], mlRho is being developed through a collaboration of research labs at IU and the Max Planck Institute for Evolutionary Biology (MPI) in Germany. Following initial benchmarking and scalability testing for an XSEDE allocation proposal, staff members from the IU Pervasive Technology Institute worked together with the mlRho application developer at MPI to improve the serial performance of the code. As a result of the investigations by PTI and MPI, a new version which is vastly more efficient and delivers much better performance when compared to the original version has been released. While the work described in the previous sections could be addressed by the campus bridging expert role, the analysis and optimization described in this section would most likely be associated with the XSEDE ECSS team. In this particular case both roles were fulfilled by a single team member at PTI. However, it is certainly possible that these two roles could be filled by different teams at different institutions.

4.2 Implementation of Performance Analysis Findings

Research staff at PTI began code optimization with mlRho version 1.10. As a first step, we compiled the code with compilers other than the standard GNU compiler. Both the Intel and PGI compilers produced a runtime improvement of ≈10% over the GNU compiler on Ranger, this is a fairly typical result that the optimization team at PTI has seen in many instances. From this point we determined that further improvements would most likely be gained by modifications to the source code. To inform the core developer at MPI as to where his efforts would be best spent, we conducted a detailed analysis of mlRho version 1.10 using the Vampir toolchain. The analysis led the core developer to focus on two aspects of the code: data handling and repeated computation. As to data handling, many profiles occur repeatedly in a data set. To address this, an additional program, formatPro, was written to compress the raw profiles. The formatPro program reads profiles either from a text file or from a BAM file, the standard format for distributing genome alignment files [14].

This new method of data storage increased performance by more than a factor of two from version 1.16 to 1.21 (see Figure 3). The formatPro program writes a binary table of unique profiles. The binary file can then be inspected using the program inspectPro. In addition to the profiles file, formatPro writes a binary file listing the profile ID at every genome position. Finally, formatPro writes a binary file of the contig lengths. The mlRho program can then read the profiles from the files produced by formatPro. The profiles are read individually rather than in a single step, because we found that this improved stability on the Lustre filesystem. The improvement in data handling introduced with these changes resulted in an overall speedup of the serial code by a factor of 2-4X. The next focus was to look at the repeated computation that occurred in the mlRho program.

In the version 1.10 of the mlRho program, the likelihood computation iterated over all positions. By introducing several improvements in how the likelihoods are calculated, the program now iterates over a much smaller number of unique profiles. This improvement can be noticed in Figure 3 between versions 1.21 and 1.23. In addition, the unique likelihoods are now written to disk for future reference.

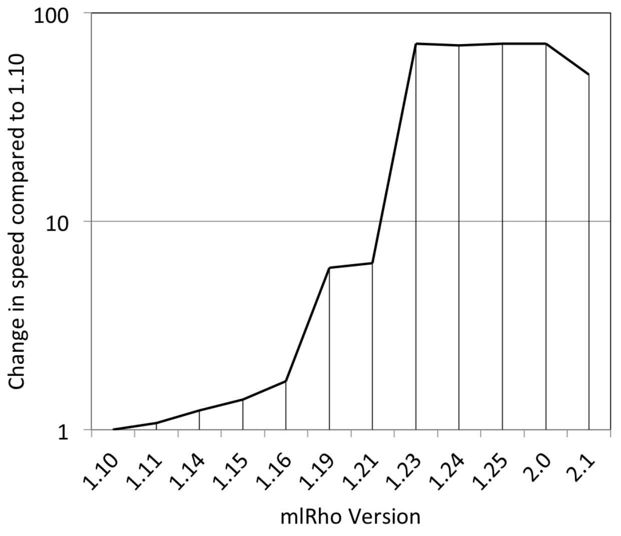

Figure 3: The graph shows the different versions of mlRho benchmarked on a single core on Stampede. The y-axis shows the change in time steps per hour with respect to version 1.10 in logarithmic scale. Compared to version 1.10, 2.1 is more than 50 times faster.

In the disequilibrium analysis of the 1.10 version, the single-site likelihoods were recomputed for every step. To improve efficiency, they are now either read from disk or stored after the first pass across the data. In addition, we noticed that by ignoring the order of two profiles, we could halve the number of distinct profile pairs stored in a search tree. The likelihood computation during traversal of this tree is now based entirely on precomputed probabilities.

The combination of better data handling and careful avoidance of repeated computation led to an overall 50-fold speedup of mlRho without increasing the minimal memory requirement observed in version 1.10.

5. USER EXPERIENCE, PRELIMINARY SCIENCE RESULTS, AND FUTURE WORK

5.1 User Experience

Overall the end user experience of migrating from the relatively modest Quarry and Mason local clusters to the much larger Ranger resource on XSEDE was a smooth transition. This was mostly due to two factors, the first of which was the similar computational environment on campus and XSEDE resources. We were able to transfer all of the input data files from campus resources to XSEDE resources via the XSEDE network using standard transfer mechanisms like `scp`. Since both the campus resources and Ranger and Stampede at TACC use a variant of the modules software environment management system, replicating the software environment on XSEDE resources was a simple matter of finding and loading the correct modules. Although the scheduling systems used on the campus resources and XSEDE resources were different (TORQUE vs. SGE), there were enough similarities and documentation on converting submission scripts and made the transition relatively easy.

The other factor in simplifying the transition was the use of the BigJob framework coupled with the consultation advice from the PTI staff. The initial transition to running mlRho in a massively parallel way on Ranger was fairly straightforward (mainly due to the embarrassingly parallel nature of the problem), and produced excellent throughput. Using BigJob to run thousands of mlRho processes was not without issues, though. Without the aid of the PTI staff, working effectively on Ranger and Stampede would have been extremely challenging.

5.1.1 Technical Issues

The initial benchmarking and scaling tests that we did for the allocation proposal progressed smoothly. But as with any research project, we ran into many technical challenges when we started running experiments at scale on Ranger. There were scaling issues, problems with BigJob design/usage, and file system issues. Some of the major issues:

- We had to remain vigilant for issues related to the Lustre file system, given that hundreds of mlRho processes read from a single input file. We tried to address this problem by striping the directories containing the input files. We gradually scaled up our job size to 4000 cores on Ranger but this put excessive stress on the file system. Due to this issue, we decided to stay below 2000 cores. Another solution could be to have a different copy of the input file for every group of 500 mlRho processes. During the course of attempting to resolve these issues, Ranger was decommissioned and Stampede came online. We did not see a similar issue on the Stampede file system, which is likely due to the improved hardware of the Stampede file system.

- The BigJob tool is and has been under active development. There were major design changes in process during the time that we started using it on Ranger. BigJob addresses a range of compute and data problems and multiple example scripts are available on its website. The initial version we deployed was not suitable for our task, which is bundling and running hundreds of serial jobs, and hampered performance.

- Another potential issue with BigJob is that the master process needs to be active from the time the job is submitted until the end of the job. This means that BigJob needs to be active on the login node of the system and cannot be disconnected. Another solution to this issue is to run BigJob in a screen session or use another tool like nohup on the login node. In the end we were told not to run more than four interactive sessions at a time, which was less than optimal for our use case. This issue has now been resolved by the BigJob developers by removing this requirement.

- We also had problems with some mlRho processes failing, which forced us to identify and re-run these jobs. This was a major problem, but with both mlRho and BigJob were rapidly evolving. This made it difficult to diagnose the issue.

5.2 Preliminary Science Results

The analyses we were able to perform on Ranger and Stampede provided an extensive amount of data that would have been unimaginable with campus resources. We began our study planning to analyze 100,000 basepair distances per genome. However, with the optimized version of mlRho performing at more than 50 times the efficiency of the version we began with, we have now completed 10 times the work we had initially planned. For many genomes we have been able to compute up to 1 million basepair distances, and we have been able to investigate many more genomes than originally proposed. Results from the new data indicate that the zygosity correlation at large distances deviates significantly from theoretical expectation. This finding has prompted us to begin new simulation and theoretical work to explain the observed discrepancy between the theoretical prediction and our ML measurements on actual data. We believe that these new data will provide insights into the evolution of a large number of organisms.

5.3 Future Work

We have begun initial work on deploying the mlRho program on the Intel Xeon Phi coprocessor boards available on Stampede. Since the Phi runs an embedded Linux Operating System [15], it is relatively easy to launch multiple copies of the mlRho binary on the Phi board, assuming that the input data set can fit in the memory footprint of the Phi board. In our case we copied input data sets to the Phi RAM disk and computed against this copy. Figure 4 compares the performance of a single Phi board on the *F. cylindrus (diatom)* genome with the scaling measurements presented in Figure 3. Here we compare to the scaling numbers for the Stampede timings using version 2.1 of the mlRho software. By using a relatively large number of processes, in this case 488, on the Phi board we are to achieve throughput that is roughly equal to the throughput of the CPUs on two Stampede nodes. We are currently focusing our efforts on integrating the Phi scripts into the BigJob framework and adding the Phi workload to the CPU workload. While writing a script for the BigJob framework is fairly straightforward, the challenge is in properly balancing the load between CPU and Phi, particularly when the input data access patterns (i.e. the input I/O) is very different for the CPUs and the Phi board.

6. CONCLUSIONS

In all, the project of transitioning and scaling up mlRho workloads from campus computational resources to XSEDE resources has been very successful, not only from the perspective of accelerating scientific discovery, but also from the perspective of providing a real and useful example of the value of a campus bridging expert. Through this project we have shown that a relatively small investment of effort by people with the right mix of skills can make a big difference when transitioning from local resources to XSEDE resources. We propose that XSEDE consider including the campus bridging expert role in more of its supported projects, particularly those projects with PIs who are relatively new to high performance computing concepts like batch scheduling, application scalability, and high performance file systems. This effort could be funded at the XSEDE level by providing experts at each of the XSEDE centers to assist with campus bridging, or at the level of the individual university where

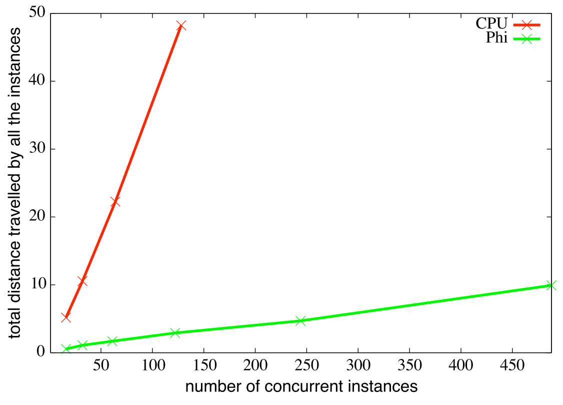

Figure 4: The graph compares the performance scaling behavior of mlRho on CPUs and Phis. Each mlRho process on a CPU was run on an individual core while multiple instances of mlRho were run on a single Phi. Four hundred and eighty-eight (488) mlRho instances on a single Phi gave us the same throughput as running 32 instances on two nodes of Stampede.

staff could be funded to assist local researchers in making the transition from campus resources to XSEDE. As the mlRho software continues to be improved and applied to more data sets, we hope to continue the excellent synergistic relationship between domain scientists, computer scientists, and cyberinfrastructure professionals.

7. ACKNOWLEDGMENTS

We would like to thank C. Stewart for his insightful comments on the initial manuscript. We would also like to thank Y. El Khamra at TACC and the SAGA-BigJob team at Rutgers University for their help with technical issues. This work funded in part by NSF grants NSF EF-0827411 and NSF DEB-1257806. This work was supported by XSEDE allocation TG-MCB120162. This research was also supported in part by the Pervasive Technology Institute Initiative. The Pervasive Technology Institute Initiative of Indiana University is funded by Lilly Endowment, Inc.

References

[1] XSEDE Extended Collaborative Support Service(ECSS). XSEDE Extended Collaborative Support Service(ECSS). `https://www.xsede.org/ecss`, 2012.

[2] Bernhard Haubold, Peter Pfaffelhuber, and Michael Lynch. mlRho - a program for estimating the population mutation and recombination rates from shotgun-sequenced diploid genomes. *Molecular Ecology*, 19:277–284, 2010.

[3] NSF Advisory Committee for Cyberinfrastructure Task Force on Campus Bridging. Technical Report *Final Report*, March 2011.

[4] Craig A. Stewart, Richard Knepper, James Ferguson, Felix Bachmann, Ian Foster, Andrew Grimshaw, Victor Hazlewood, and David Lifka. What is campus bridging and what is XSEDE doing about it? In *Proceedings of the 1st Conference of the Extreme Science and Engineering Discovery Environment: Bridging from the eXtreme to the campus and beyond*, XSEDE '12, pages 47:1–47:8, Chicago, Illinois, 2012. ACM.

[5] SAGA BigJob. SAGA BigJob. `https://github.com/saga-project/BigJob`, 2012.

[6] Sarah P. Otto and Thomas Lenormand. Resolving the paradox of sex and recombination. *Nat Rev Genet*, 3 (4):252–261, 04 2002.

[7] Michael P. H. Stumpf and Gilean A. T. McVean. Estimating recombination rates from population-genetic data. *Nat Rev Genet*, 4(12):959–968, 12 2003.

[8] Montgomery Slatkin. Linkage disequilibrium – understanding the evolutionary past and mapping the medical future. *Nature Reviews Genetics*, 9:477–485, 2008.

[9] Michael Lynch. Estimation of nucleotide diversity, disequilibrium coefficients, and mutation rates from high-coverage genome-sequencing projects. *Molecular Biology and Evolution*, 25(11):2409–2419, 2008.

[10] Michael Lynch, Louis-Marie Bobay, Francesco Catania, Jean-FranÃğois Gout, and Mina Rho. The Repatterning of Eukaryotic Genomes by Random Genetic Drift. *Annual Review of Genomics and Human Genetics*, 12 (1):347–366, 2011.

[11] A. Luckow, L. Lacinski, and S. Jha. SAGA BigJob: An Extensible and Interoperable Pilot-Job Abstraction for Distributed Applications and Systems. In *The 10th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing*, pages 135–144, 2010.

[12] A. Luckow, S. Jha, J. Kim, A. Merzky, and B. Schnor. Adaptive Replica-Exchange Simulations. *Royal Society Philosophical Transactions A*, 2009.

[13] Abhinav Thota, Andre Luckow, and Shantenu Jha. Efficient large-scale replica-exchange simulations on production infrastructure. *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 369(1949):3318–3335, 2011.

[14] H. Li, B. Handsaker, A. Wysoker, T. Fennell, J. Ruan, N. Homer, G. Marth, G. Abecasis, R. Durbin, and 1000 Genome Project Data Processing Subgroup. The sequence alignment/map format and SAMtools. *Bioinformatics*, 25:2078–2079, 2009.

[15] Intel Xeon Phi Coprocessor System Software Developers Guide. Intel Xeon Phi Coprocessor System Software Developers Guide. `http://software.intel.com/sites/default/files/article/334766/intel-xeon-phi-systemsoftwaredevelopersguide.pdf`, 2012.