Surviving the Life Sciences Data Deluge using Cray Supercomputers

Bhanu Rekepalli and Paul Giblock
National Institute for Computational Sciences
The University of Tennessee
Knoxville, TN, USA
bhanu@utk.edu, pgiblock@utk.edu

***Abstract*—The growing deluge of data in the Life Science domains threatens to overwhelm computing architectures. This persistent trend necessitates the development of effective and user-friendly computational components for rapid data analysis and knowledge discovery. Bioinformatics, in particular, employs data-intensive applications driven by novel DNA sequencing technologies, as do the high-throughput approaches that complement proteomics, genomics, metabolomics, and metagenomics. We are developing massively parallel applications to analyze this rising flood of life sciences data for large scale knowledge discovery. We have chosen to work with the desktop or cluster based applications most widely used by the scientific community, such as NCBI BLAST, HMMER, and MUSCLE. Our endeavors encompasses extending highly scalable parallel applications that scale to tens of thousands of cores on Cray's XT architecture to Cray's next generation XE and XC architectures along with focusing on making them robust and optimized, which will be discussed in this paper.**

***Keywords*-Benchmarking; Bioinformatics; HPC;**

I. Introduction

The exponential growth of data generated in biology research generated by advances in next generation sequencing technology [1] and mass-spectrometry analysis [2], from small atoms to big ecosystems, necessitates an increasingly large computational component to perform analyses. Novel next-generation sequencing (NGS) technologies and complementary high-throughput approaches involving genomics (DNA), transcriptomics (RNA), proteomics (proteins), and metagenomics (genetic material from environmental samples) drive data-intensive bioinformatics. Current NGS technologies enable researchers to sequence five human genomes in a week with a cost less than five thousand dollars per genome. In comparison, the first human genome cost three billion dollars [3] and took more than a decade to sequence. NGS data is doubling every year, outpacing Moore's law, which is adding a plethora of data to genomic databases worldwide. This trend increases the demand for computing power utilizing High Throughput Computing (HTC) and High Performance Computing (HPC) architectures for data analysis from genome assembly to annotation, which will advance scientific fields from human disease research to agriculture and evolutionary science.

The growing use of sequencing data, medical imaging data, and their byproducts used for drug discovery and biofuel generation introduces critical problems such as data movement, storage, and analysis. This paper addresses the analysis part of the critical problems in life sciences. We are developing massively parallel applications to analyze this rising flood of life sciences data, and to rapidly contribute fresh knowledge to the fields of computational biology and biomedicine. The availability and accessibility of these highly scalable parallel bioinformatics applications on Cray supercomputers will help to bridge the gap between the rate of data generation in life sciences and the speed and ease at which scientists can study this data.

These highly scalable applications are used to address the data analysis needs of our collaborators. For example, researchers at the Joint Institute for Biological Sciences at UT-Knoxville are studying permafrost, or perennially frozen ground, which underlies about 24% of the Earth's land surface and covers the Arctic landscape. It is also a source of extremely potent greenhouse gases, such as methane and carbon dioxide. Temperatures in the Arctic may increase 6 °C over the next 100 years and will increase the depth of the active layer, the seasonally unfrozen soil above the permafrost. Defining the diversity, activities, and biogeochemical parameters by carrying out large-scale genomic and metagenomics sequencing and analysis studies on various samples from active and permafrost layers will allow for a better understanding of the Arctic permafrost system as it relates to climate change factors on Earth.

Our collaborators at Seattle Children's research Institute are engaged in an unprecedented effort to identify the genetic changes that give rise to some of the world's deadliest cancers by decoding the genomes of hundreds of cancer patients. Scientists involved in the project are sequencing the entire genomes of both normal and cancer cells from each patient, comparing differences in the DNA to identify genetic mistakes that lead to cancer. As the novel genomics space is imploding, the unknown space – with no functional or structural characterization – is expanding in multiple orders [4]. Thus, the collaborative project between NICS and Seattle Children's Research Institute is focusing on improving and expanding functional annotation of newly sequenced genomes, proteomes, and meta-genomes [5] to better understand the functional behavior of various genes involved in cancer and other diseases.

II. Implementation

The following sections discuss our Highly Scalable Parallel (HSP) tools, and their optimizations for various CRAY architectures, along with discussing scaling studies performed.

A. Benchmark Programs

NCBI BLAST is an implementation of the Basic Local Alignment Search Tool (BLAST) by the National Center for Biotechnology Information [6]; one of the most widely-used tools for sequence similarity searches. BLAST can perform comparisons between protein or DNA sequences and a sequence database. There are variations of the algorithm for different research needs. The program is used by researchers for annotating newly sequenced genomes and similarity searches among other purposes.

HMMER by Howard Hughes Medical Institute's Janelia Farm is also used for similarity searches of sequence databases [7] along with protein domain identification. Moreover, HMMER can perform homolog searches. The tool uses hidden Markov models instead of the local alignment search employed by BLAST.

MUSCLE [8] is another very commonly used tool for bioinformatics research. MUSCLE executes multiple sequence alignment among a set of biological sequences. It also outperforms most other multiple sequence alignment algorithms such as CLUSTALW [9].

B. Optimizations

The tools are useful, but they are not designed to run on supercomputers, such as those provided by Cray. Instead, the software assumes a common environment such as a desktop or workstation. Unlike the target environment, supercomputers contain a massive number of cores, have little to no local storage, and rely on distributed filesystems for persistent storage. Our goal is to efficiently run these software tools on high performance computing systems. The codebase for the programs is quite large; NCBI BLAST contains over one million lines of code, HMMER with 35,000 lines, and MUSCLE with nearly 28,000. Due to the maintenance cost, the need to maintain consistency, and our desire to target even more applications, we realize that as few changes should be made to the software as possible. Therefore, we designed a reusable solution for minimizing these differences [10].

HSP-Wrap, or Highly Scalable Parallel Wrapper, is the name of our software wrapper, which acts as a harness for the underlying tools and provides scaling optimizations. The wrapper is implemented in C, uses MPI for communication, and serves as a load balancer where input is distributed across processes running on the compute nodes. The design of the wrapper stems from our experiences with the aforementioned tools, both from code review and profiling. We utilize CrayPat [11], PAPI [12], gprof [13], and our own timing code to analyze the runtime of both the tools and the wrapper itself. The CrayPat profiler was used to investigate MPI communication. PAPI and gprof were used to measure the performance of the tools. The wrapper measures the time taken for a number of tasks and this is useful for tuning the wrapper's parameters for a particular architecture or tool.

The results of our prior scaling studies suggests that input and output is one of the greatest problems to solve [10]. For example, BLAST and HMMER load large sequences databases tens of Gigabytes in size. Each BLAST or HMMER process should not need to read these files from storage. Instead we load the database once, and broadcast it across the computational nodes with the `MPI_BCast` command. The data is then stored into POSIX shared memory segments (SHMs) so that the wrapped tool processes can access the shared data. We provide two software libraries, libstdiowrap and libstdiowrap++, to facilitate the use of these shared memory segments from the wrapped applications. Libstdiowrap provides analogs to the C standard I/O routines. Libstdiowrap++ is the C++ compatibility library. It is based around an implementation of the `std::filebuf` class, as well as analogs of the `std::ifstream`, `std::ofstream`, and `std::fstream` classes for convenience. The majority of the applications' input and output routines can be ported to use libstdiowrap through a number of C preprocessor macros.

The shared memory segments are also used to supply the processes with the load-balanced input. The inputs for BLAST and HMMER are protein or DNA sequences, but MUSLE requires a set of sequences for alignment. HSP-Wrap maintains a queue of input on each compute node. The node maintains a pool of worker processes which are issued queries from the queue. Input latency is reduced since the queries already reside locally on the node. The wrapper on the compute nodes can then request more input from the designated master node as its local queue drains. We hope to compare this technique with a work-stealing algorithm in the future.

The performance of output operations was also improved. We use a two-stage buffering technique to reduce the performance penalty of hundreds or thousands of processes writing to storage simultaneously. The first buffering stage sits between HSP-Wrap and the tool process in the form of one or more shared memory segments. Writes are made asynchronously from the tool by simply copying the data into the shared memory segment reserved for the process's output. The first stage is flushed to the second stage upon task completion or to avoid overflow of the first stage buffer. The data can optionally be compressed between the two stages to reduce memory and disk consumption and to decrease the time spent writing. Additionally, HSP-Wrap can distribute the output files across multiple directories if the storage is a distributed file system. This helps improve the chance that output is written across multiple object

Table I
INPUT SELECTION

Node Count	1	2	4	8	16
`blastp`	192	384	768	1,536	3,072
`hmmscan`	1,536	3,072	6,144	12,288	24,576
`muscle`	1,200	2,400	4,800	9,600	19,200

[1] figures are the number of sequences of length 250 – 450 Amino Acids
[2] figures are the number of multiple sequences alignments of 500 proteins

storage targets within the Lustre, or similarly distributed, filesystems.

III. EXPERIMENT SETUP

We evaluated the performance of HSP-Wrap combined with the wrapped versions of NCBI BLAST, HMMER 3.0, and MUSCLE 3.8.31; and commonly used databases such as the nr (non-redundant) protein sequence database, and the Pfam (Protein Families) domain database to benchmark on the three previously mentioned Cray architectures. These HSP tools were initially developed on Kraken, a Cray XT5 supercomputer. Kraken has 9,408 compute nodes with two sockets per node with two 6-core 2.6 GHz AMD Opteron 2435 processors, 16 GB memory with 3D torus Cray SeaStar2+ interconnect, and with peak performance of 1.17 PetaFLOPs [14]. The HSP tools are further optimized to new Cray systems at the National Institute for Computational Sciences known as Mars and Darter. Mars is a Cray Nano machine with 16 XK6 compute nodes, each with a 16-core AMD 2.2GHz Opteron processor, 16 GB memory, and an NVIDIA X2090 GPGPU with 6 GB of memory. Mars also has 20 XE6 compute nodes with two 16-core AMD 2.2GHz Opteron processors and 32 GB of memory per node. All nodes are connected with a 2D torus Gemini interconnect. Darter is a Cray XC30 supercomputer with a total of 748 compute nodes. Each node has two 8-core 2.6 GHz Intel Sandy Bridge processors, 32 GB of memory per node, and the newly designed dragonfly Aries interconnect with peak performance of 248.9 TeraFLOPs.

We evaluated the Protein BLAST function through the `blastall -t blastp` tool. The database used was the nr database as of March 2012. Input query sequences were selected from the same version of the nr database. We sampled a number of sequences with a length of 250 – 450 Amino Acids according to Table I. We assessed HMMER's `hmmscan` tool in performing searches against the 24.0 version of the Pfam database. Again, sequences were chosen with 250 – 450 Amino Acids, but the number of sequences differs from the `blastp` experiments due to `hmmscan`'s faster speed. MUSCLE's `muscle` tool was also tested. We generated the inputs for MUSCLE by finding homologs with the Protein BLAST function. We ran a portion of the nr database against itself, limited the number of homologs to 500, then chose all of the results resulting in 500 proteins.

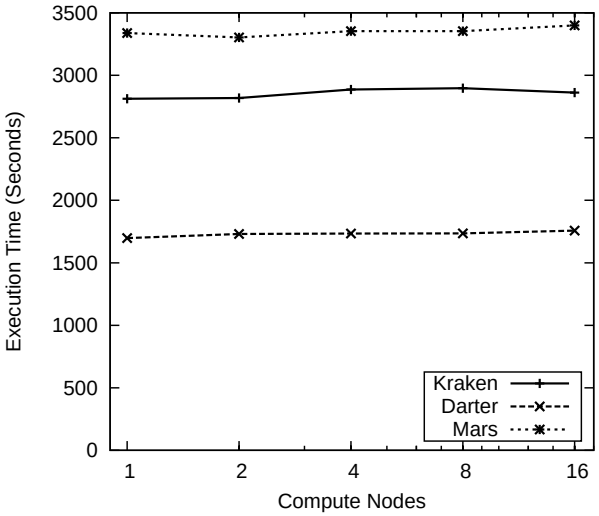

Figure 1. BLAST weak scaling results

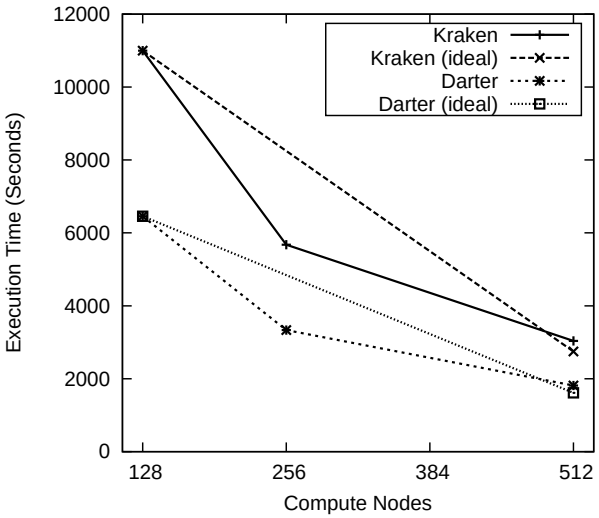

Figure 2. BLAST strong scaling results

Hence, MUSCLE tests perform multiple sequence alignments on 500 proteins, the number of multiple sequences alignments used for the experiment is shown in Table I. We derived these counts by choosing a number which allows each core to perform several operations. The job should also run sufficiently long to provide more accurate results, and the number of tasks is scaled with the number of nodes. Finally, we also looked at strong scaling performance of BLAST on Kraken and Darter; Mars is excluded due to its relatively low core count. For this, we used 98,304 protein sequences, sampled as above. The experiment is run on 128, 256, and 512 nodes.

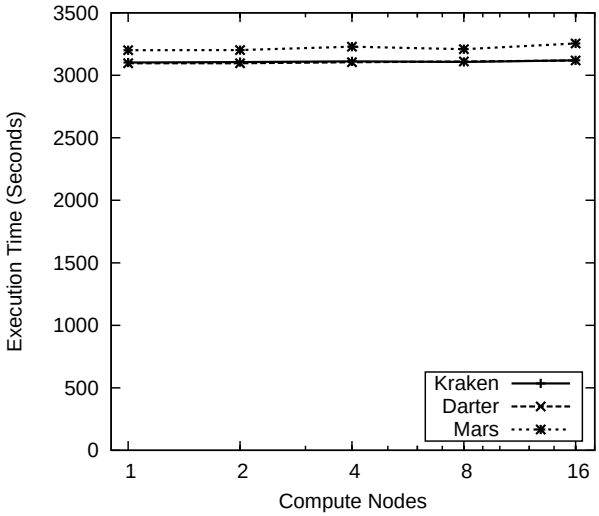

Figure 3. HMMER weak scaling results

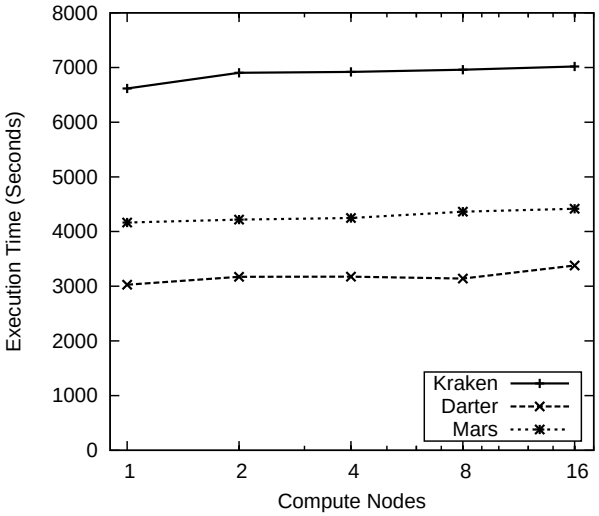

Figure 4. MUSCLE weak scaling results

IV. Results

The BLAST weak scaling results in Figure 1 show a near linear speedup. There is also a significant difference in performance between the 3 machines. Darter performs significantly better than Kraken, and Mars has the worst performance. Darter's superior performance is no surprise given the specifications of the machine. However, we are surprised to see Kraken outperform Mars. More analysis will be necessary for us to determine the cause of this. The strong scaling results for BLAST, shown in Figure 2, are near ideal. Again, Darter outperforms Kraken at these high node counts. We only consider strong scaling of BLAST due to the resource constraints and positive results seen in the weak scaling for the other applications in the past [10] [15].

HMMER performance, shown in Figure 3, is nearly constant across the node counts used for this experiment. Again, Darter outperforms the other machines, but all three are very close in execution time. HMMER is less computationally intensive than BLAST. Therefore, the faster processors of Darter do not provide much improvement, and the runtime is instead limited by network and filesystem speeds. Given our previous results with HSP-HMMER [15], we suspect that the software maintains scalability across larger amounts of nodes as well.

The MUSCLE results in Figure 4 show linear speedup. The speedup is not as good as that seen in BLAST or HMMER. The performance profile is still positive, and perhaps further some tuning can improve the speedup. Interestingly, MUSCLE is the only experiment in this study where Mars outperforms Kraken, and substantially so this is because MUSCLE is compute intensive rather than memory intensive.

V. Conclusion

We've shown that our method for scaling BLAST on the Kraken supercomputer can be extended to multiple programs as well as architectures. The HSP versions of the BLAST, HMMER, and MUSCLE tools run with linear speedup, and allow one to take advantage of new systems with very little additional effort. The power of these tools and the machines they run on will only help increase the speed at which life sciences researchers are able to conduct their research.

Acknowledgements

This research used resources at the Joint Institute for Computational Sciences, funded by the National Science Foundation (NSF) and also supported in part by the NSF grant EPS-0919436.

References

[1] E. R. Mardis, "A decade's perspective on dna sequencing technology," *Nature*, vol. 470, no. 7333, pp. 198–203, 2011.

[2] M. Beck, A. Schmidt, J. Malmstroem, M. Claassen, A. Ori, A. Szymborska, F. Herzog, O. Rinner, J. Ellenberg, and R. Aebersold, "The quantitative proteome of a human cell line," *Molecular systems biology*, vol. 7, no. 1, 2011.

[3] The Human Genome Project, "About the Human Genome Project," http://www.ornl.gov/sci/techresources/Human_Genome/project/about.shtml.

[4] B. Rekapalli, K. Wuichet, G. D. Peterson, and I. B. Zhulin, "Dynamics of domain coverage of the protein sequence universe," *BMC genomics*, vol. 13, no. 1, pp. 1–6, 2012.

[5] R. Higdon, W. Haynes, L. Stanberry, E. Stewart, G. Yandl, C. Howard, W. Broomall, N. Kolker, and E. Kolker, "Unraveling the complexities of life sciences data," *Big Data*, 2013.

[6] 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.

[7] S. R. Eddy, "Profile hidden markov models." *Bioinformatics*, vol. 14, no. 9, pp. 755–763, 1998. [Online]. Available: http://bioinformatics.oxfordjournals.org/content/14/9/755.abstract

[8] R. C. Edgar, "MUSCLE: multiple sequence alignment with high accuracy and high throughput," *Nucleic Acids Research*, vol. 32, no. 5, pp. 1792–1797, 2004.

[9] "MUSCLE," http://www.drive5.com/muscle/.

[10] B. Rekepalli, A. Vose, and P. Giblock, "HSPp-BLAST: Highly Scalable Parallel PSI-BLAST for very large-scale sequence searches," in *Proceedings of the 3rd International Conference on Bioinformatics and Computational Biology (BICoB)*, F. Saeed, A. Khokhar, and H. Al-Mubaid, Eds. BICoB, 2012, pp. 37–42.

[11] Cray, Inc., "Performance tools. in optimizing applications on the cray x1 system," 2003.

[12] S. Browne, J. Dongarra, N. Garner, G. Ho, and P. Mucci, "A portable programming interface for performance evaluation on modern processors," *International Journal of High Performance Computing Applications*, vol. 14, no. 3, pp. 189–204, 2000.

[13] S. L. Graham, P. B. Kessler, and M. K. Mckusick, "Gprof: a call graph execution profiler," *ACM Sigplan Notices*, vol. 17, no. 6, pp. 120–126, 1982.

[14] National Institute for Computational Sciences, "Kraken," http://www.nics.tennessee.edu/computing-resources/kraken.

[15] B. Rekapalli, C. Halloy, and I. B. Zhulin, "HSP-HMMER: a tool for protein domain identification on a large scale," in *Proceedings of the 2009 ACM symposium on Applied Computing*. ACM, 2009, pp. 766–770.