EPiC Series in Computing

Volume 60, 2019, Pages 1–14

Proceedings of 11th International Conference
on Bioinformatics and Computational Biology

HPC-BLAST: Distributed BLAST for Modern HPC Clusters

Shane Sawyer[1], Mitchel Horton[1], Chad Burdyshaw[1], Glenn Brook[1] and Bhanu Rekapalli[2]

[1] Joint Institute for Computational Sciences, University of Tennessee, Knoxville, Tennessee, USA
[shane-sawyer,mhorton5,chadburdyshaw,glenn-brook]@tennessee.edu
[2] BioTeam Inc., Middleton, Massachusetts, USA
bhanu@bioteam.net

Abstract

The near exponential growth in sequence data available to bioinformaticists, and the emergence of new fields of biological research, continue to fuel an incessant need for increases in sequence alignment performance. Today, more than ever before, bioinformatics researchers have access to a wide variety of HPC architectures including high core count Intel Xeon processors and the many-core Intel Xeon Phi.

In this work, the implementation of a distributed, NCBI compliant, BLAST+ (C++ toolkit) code, targeted for multi- and many-core clusters, such as those containing the Intel Xeon Phi line of products is presented. The solution is robust: distributed BLAST runs can use the CPU only, the Xeon Phi processor or coprocessor, or both by utilizing the CPU or Xeon Phi processor plus a Xeon Phi coprocessor. The distributed BLAST implementation employs static load balancing, fault tolerance, and contention aware I/O. The distributed BLAST implementation, HPC-BLAST, maintains greater than 90% weak scaling efficiency on up to 160 Xeon Phi (Knights Landing) nodes.

The source code and instructions, are available under the Apache License, Version 2.0 at https://github.com/UTennessee-JICS/HPC-BLAST.

1 Introduction

Recent advances in sequencing technology (DNA sequencing, amino-acid and protein characterization, gene expression data, and whole-genome descriptions) are both providing bioinformatics researchers with a bonanza of biological sequence datasets, and broadening the applications for sequencing [6, 15, 17, 18, 24, 39, 40, 50, 20, 53, 26]. GenBank, a publicly available sequence database maintained by NIH, has experienced exponential-like growth [1, 27].

Sequence similarity searching remains among the most important and challenging tasks in bioinformatics [49, 26, 59]. The Basic Local Alignment Search Tool (BLAST) is the most popular sequence search and alignment tool in use today [35, 29, 63, 20, 9, 3, 44, 55, 60, 7], widely adopted for its sensitivity and speed [16]; the seminal papers for BLAST and Position-Specific Iterated (PSI)-BLAST have a combined 129,925 citations [4, 5]. The level of usage for

BLAST suggests that any improvement to its performance would have a pervasive effect on the field of bioinformatics [55].

With each passing day, sequential BLAST becomes less able to keep pace with the performance requirements placed upon it by the bioinformatics community. This phenomenon is described in many different ways. A 2003 study found that searching the GenBank database requires approximately 64% more time each year, always using current hardware [10]. It is reported that metagenomics studies can take 800,000 CPU hours, and that BLAST can make up to 99.97% of that runtime [20, 34, 22]. It is reported that the search speed of BLAST has become insufficient for the typical metagenomic analysis [53]. Researchers report that the minutes required for a single protein search against NCBI's NR database [57] can be too long to be practical [54]. People say that biological sequence data is accumulating more quickly than the growth in computing efficiency predicted by Moore's Law [8, 16], or that expansion of genetic sequence information exceeds the growth in computing power available at a constant cost [49].

In the recent past, many classes of BLAST acceleration efforts have occurred. The National Center for Biotechnology Information (NCBI)-BLAST software supports multithreading in the preliminary stages of the BLAST algorithm [9]. More recently, the traceback phase of the search was also parallelized starting with the 2.4.0 version of the toolkit. Others have implemented both multithreaded and vectorized BLAST algorithms [28, 12, 23]. Many groups have implemented Field Programmable Gate Array (FPGA) BLAST acceleration [58, 25, 21, 41, 51, 11, 58]. There are BLAST implementations distributed across a cluster [13, 31, 43, 47, 14, 19, 35, 30, 7], distributed using web services [56], and distributed across a grid of clusters [2, 60, 61]. MapReduce implementations of BLAST exist [52, 38, 36]. NCBI maintains a BLAST server farm, that parallelizes BLAST runs at the job level, scheduling more than 100,000 jobs per day [37]. People have accelerated BLAST for the GPU [55, 32, 63, 33], and the Cell Broadband Engine architecture [62]. In this work, an implementation of a distributed, NCBI compliant, BLAST+ (C++ toolkit) code intended for use on high core count clusters and emerging technologies such as the Intel Xeon Phi is presented.

2 Methodology

In this work, a new parallel implementation of the NCBI BLAST search software called HPC-BLAST is presented. HPC-BLAST is designed to overcome many of the limitations in previous parallel implementations. Previous distributed implementations, including mpiBlast [13, 47, 30] and ScalaBlast [43, 44], focused exclusively on scaling through process distribution using the Message Passing Interface (MPI) library [48]. In the advent of clusters featuring multicore processors, this approach will not provide optimal performance as compared with a hybrid approach of using MPI plus a threading scheme such as OpenMP [46]. In this paradigm, MPI can be used to distribute processes across compute nodes/processors of the cluster and the threading scheme to distribute computational work at the node/socket level. A hybrid approach is more amenable to the Xeon Phi which has more logical cores than Xeon processors. With the hybrid approach, the distribution of ranks and threads can be modified to suit the particular architecture.

As mentioned, other parallel implementations are designed to batch NCBI BLAST searches and distribute these tasks across nodes in a cluster or supercomputer [19, 35, 61]. Often, these implementations utilize a command line tool to break the input queries or database into separate tasks which are passed to the system's scheduler to be computed [19, 35, 61]. A particular example of this approach is DC BLAST [61]. An advantage of this approach is that since the controlling script is separate from the NCBI toolkit, newer versions of the

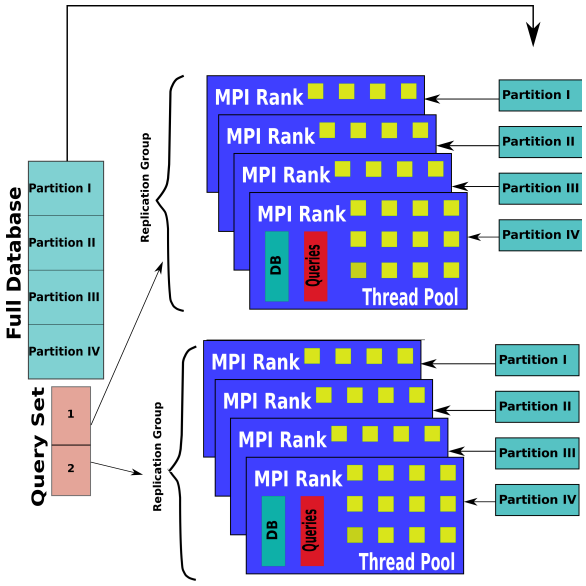

Figure 1: Top level task distribution.

NCBI executables can seamlessly be added. Another advantage is that there are no restrictions made on the users with regard to command line options, such as output format. However, in [61], there is no mention of splitting the database and only the Swiss-Prot protein database is used, so that parallelization is made possible only by dividing the queries. This is particularly limiting when the reference database is large, such as with the full non-redundant databases, as these databases may not fit into the memory of a single node. This approach also does not take advantage of the finer-grained parallelism that can be achieved by parallelizing within the NCBI toolkit code.

A further issue with many parallel implementations is that they use the older C version of the BLAST toolkit. HPC-BLAST is built using the newer C++ BLAST toolkit. As such, HPC-BLAST is able to benefit from future improvements to the underlying BLAST source code released by NCBI.

3 Design

HPC-BLAST uses a two-level hierarchical design to efficiently distribute search tasks across MPI processes and threads; a top level where tasks are distributed across processes and a low level where tasks are distributed across threads to each process. At the process level, an HPC-BLAST search can be distributed amongst MPI processes by partitioning the queries and the database provided as input to the search. For instance, partitioning the queries into 2 partitions, and partitioning the database into 4 partitions, would result in 8 MPI processes, each executing an HPC-BLAST search using a unique query-partition database-partition pair, as depicted in Figure 1. Many parallel BLAST implementations, including mpiBLAST [13, 47] and ScalaBLAST [43] use a conceptually similar approach.

At a conceptual level above the process level, there is the notion of a replication group. If, for example, the queries are partitioned into two partitions and the database is partitioned into

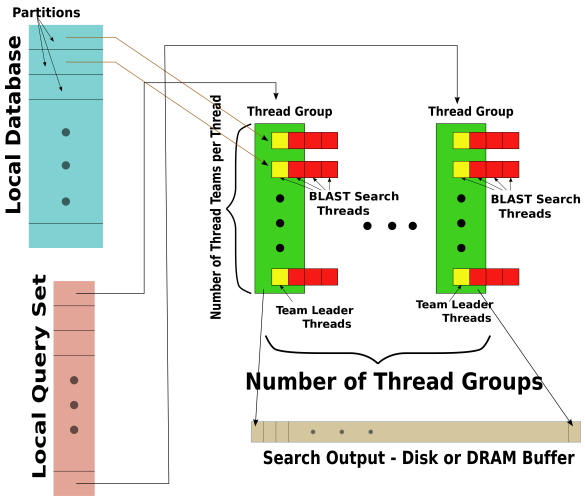

Figure 2: Process level task distribution.

four partitions, then the group of database partitions that will be searched against by query partition one, is a replication group. It is called a group because it is a group of database partitions. A characteristic of a replication group is that it contains every sequence in the full database. The group of database partitions that will be searched against by query partition two, is also a replication group. It consists of the same database sequences that are in replication group one, but it is thought of as a second replication group. The replication group view can also be seen in Figure 1, where there are two replication groups, each with the same four database partitions.

Just like at the MPI level, an HPC-BLAST search can be parallelized at the thread level, amongst OpenMP threads, by partitioning the queries and the database that was assigned from the process level. This second level of the hierarchy is unique to HPC-BLAST, increasing the amount of parallelization. As an example, partitioning the queries into two partitions, and partitioning the database into two partitions, would result in four OpenMP threads, each executing a distinct NCBI-BLAST search instantiation, using a unique query-partition, database-partition pair. The thread level is depicted in Figure 2 which shows, as an example, a number of database partitions and a number of query partitions.

Analogous to the process level, there is one thread per database partition. This collection of threads is known as a thread group. If there are multiple query partitions, there will be multiple thread groups, one per query partition. Each thread group searches the same partitions of the database, but with a different query partition. Each thread inside a query group will execute an NCBI-BLAST search with a unique query-partition, database-partition pair. A given thread inside a query group is called a team leader. This is because the NCBI-BLAST search itself can spawn additional threads during its execution. In version 2.2.31, blastp searches could be parallelized by threads during the preliminary search phases.

4 Implementation

HPC-BLAST was designed with the goal of leveraging the ideas of previous successful parallel implementations but adapted towards multi- and many-core CPU systems. Most modern CPUs,

including consumer variants, have several to tens of cores, and server CPUs are expected to reach hundreds of cores in the near future. One mechanism of parallelization, generally referred to as coarse-grained parallelization, is to map the MPI ranks to CPU cores. Search tasks can then be distributed to the MPI ranks. This is not always an ideal way to distribute work since each MPI process has a separate address space for memory. This has several implications including increased memory requirements since each process needs to have a copy of the executable in memory and requiring that each rank would need a copy of the database in its memory space.

HPC-BLAST addresses the limitations of a purely MPI parallel approach to task distribution by the use of OpenMP threads as discussed in its design. The threading model allows for better utilization of resources. Since threads share a common memory address space, a single copy of the database can be used by multiple threads within a single MPI rank. Another performance benefit seen on multi- and many-core CPUs where cores share cache is an improvement in throughput, particularly when the threads are used to distribute the query sequences. In this scenario, different thread groups scan through the same database. Once one thread group has loaded a subject sequence, subsequent requests for the same sequence from other thread groups may be serviced from cache rather than main memory, as long as the data is resident in the cache.

It is worth noting that NCBI BLAST does include multithreading support. But as of release version 2.2.31, the threads are limited to only the initial part of the search algorithm. Development of HPC-BLAST used the most recent version of the NCBI toolkit (2.2.31) when initially designed, tested, and benchmarked. However, HPC-BLASTp has subsequently migrated to NCBI toolkit release 2.7.1. The OpenMP threads used by HPC-BLAST allow for greater concurrency since these threads are parallelizing the entire search; and, as indicated above, HPC-BLAST allows for the use of both the OpenMP threads and the native NCBI BLAST search threads simultaneously.

At the conclusion of a search iteration through the BLAST algorithm, the results are distributed among the different HPC-BLAST threads. Initially, HPC-BLAST, denoted as version 0.5, allowed each thread to create a unique output file. In a separate post-processing step, the collective output was merged and sorted so that the results were presented in a single output file with the default pairwise alignment format. This solution could result in placing undue pressure on a parallel filesystem when the number of HPC-BLAST threads exceed several tens of thousands. To address this, in release 1.0 of HPC-BLAST, a single thread per MPI process is designated as the writer for that process. At the end of a search iteration, the threads cooperatively aggregate their search results into a single data structure mimicking how the NCBI search threads merge their results. The post-processing step then only has to merge results from the different MPI processes. If each process has a complete copy of the database, the outputs from each process will be disjoint with respect to the queries. In this case, the behavior is like that of running many BLAST jobs in a batched environment and is similar to other results [14, 35].

5 Results

Several benchmarking tests were conducted to study the scaling of HPC-BLAST relative to NCBI-BLAST on several different architectures including Intel Xeon processors (Sandy Bridge and Skylake varieties) and the many-core Intel Xeon Phi (Knights Corner (KNC) and Knights Landing(KNL)). For the tests, the reference database used was a partition of the non-redundant protein database (nr) containing 2,993,096 sequences and 1,001,005,291 total residues. The query input consisted of 1058 sequences totaling 350,276 residues randomly sampled from the subject database. The remaining technical specifications are given in Table 1.

NCBI BLAST+ version 2.2.31.
Intel Compiler versions 2016.3.210 and 2017.2.174.
Intel MPI version 5.1.3.210 and 2017.2.174.
Intel MPSS Stack version 3.7 (for KNC).
Sandy Bridge model E5-2670.
Skylake Gold 6148 model.
Xeon Phi KNC model 5110P.
Xeon Phi KNL model 7210.

Table 1: Technical Specifications.

The initial tests were strong scaling tests restricted to one node of an HPC cluster [42], but using the different listed architectures. A full suite of tests was conducted varying parameters including number of database partitions, number of replication groups, number of thread groups, and number of NCBI search threads used. In order to present the trends in a clear fashion, only two configurations are presented: 6 total MPI processes with 2 replication groups and 8 total MPI processes with 2 replication groups; the two combinations that showed greatest performance. This leads to 3 and 4 database partitions, respectively. In the strong scaling tests, the total number of threads is increased. For each of these two configurations, there are several ways to distribute the threads between thread groups and NCBI search threads. For instance, given a total of 16 threads, threads could be allocated as such: 8 for thread groups and allow each to have 2 NCBI search threads. Or, 4 thread groups with 4 NCBI search threads. The allocations simply need to be a factorization of the total desired thread count. For clarity, the thread distribution for a given total thread count that represents the best performance is reported. All reported tests demonstrate the performance of the 1.0 version of HPC-BLAST where, as discussed in the Implementation section, processes aggregate search data before writing to disk. This in situ aggregation imparts minimal runtime cost relative to the 0.5 version of the software. In all observed trials, the performance impact did not exceed 5% of total runtime. All the performance plots show speedup of HPC-BLASTp relative to the runtime of NCBI BLASTp (2.2.31) using a single search thread on the same architecture, unless otherwise noted.

The first two architectures examined were the Intel Xeon Sandy Bridge and Skylake processors. The compute nodes used for each processor had dual sockets. For the Sandy Bridge, a total of 16 physical cores and 32 logical cores were available. For the Skylake platform, each physical CPU socket has 20 physical and 40 logical cores available. The speedup plot for the Sandy Bridge processors is shown in Figure 3. Using all logical cores on the system, HPC-BLAST is able to out-perform NCBI BLAST by 60%. The performance difference on the Skylake processors is even more marked when compared with NCBI 2.2.31 as shown in Figure 4. Here, the performance advantage of HPC-BLAST is nearly double compared to NCBI BLAST, when comparing the 2.2.31 version of the toolkit. NCBI 2.7.1 shows great improvement itself relative to the 2.2.31 release, gaining nearly twice the speedup. However, HPC-BLASTp has greater speedup than the NCBI 2.7.1, even when using the older toolkit.

The impact of the additional parallelization in the traceback phase of the search in the NCBI 2.7.1 release was studied in HPC-BLAST. For this experiment, a configuration of 4 MPI ranks with 2 replication groups was used. Scaling tests were done with 1 NCBI search thread and 4 NCBI search threads, i.e. setting `-num_threads [1,4]`. The speedup performance seen in Figure 5 shows that there is some benefit to using some number of NCBI search threads in the total thread distribution. This is in contrast to HPC-BLAST with the 2.2.31 toolkit which

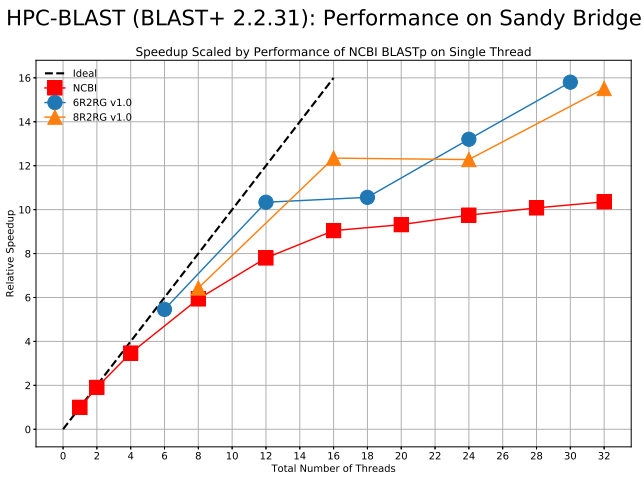

Figure 3: Comparison of strong scaling on a pair of Intel Xeon Sandy Bridge processors.

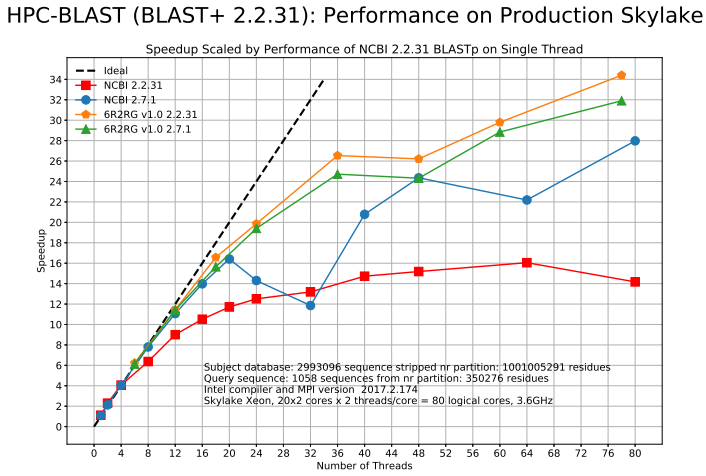

Figure 4: Comparison of strong scaling on a pair of Intel Xeon Skylake processors.

did not see benefit from these serach threads in testing.

The performance of HPC-BLAST was also investigated on the Intel Xeon Phi line of coprocessors and processors. As compared to the Xeon family of processors, the Xeon Phi architecture features many more cores, albeit ones that are generally clocked slower and possesses less transistor real estate for serial processing tasks. The Xeon Phis are intended for highly vectorizable code and parallel computation tasks that are common in traditional HPC applications. Although the BLAST algorithm is not a good candidate for vectorization, the highly parallel nature of the search tasks are suitable for the Xeon Phi architecture. The scaling results for the KNC coprocessor demonstrate a nearly 4 times performance improvement compared to NCBI BLAST as seen in Figure 6. The second generation Intel Xeon Phi, KNL product line, provides

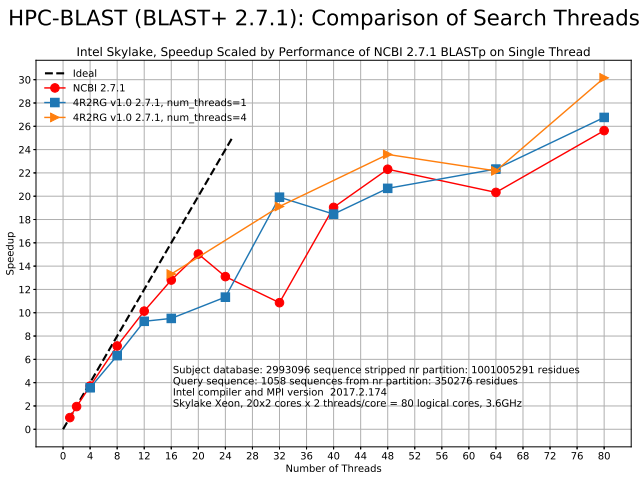

Figure 5: Comparison of speedup on a pair of Intel Xeon Skylake processors relative to search threads..

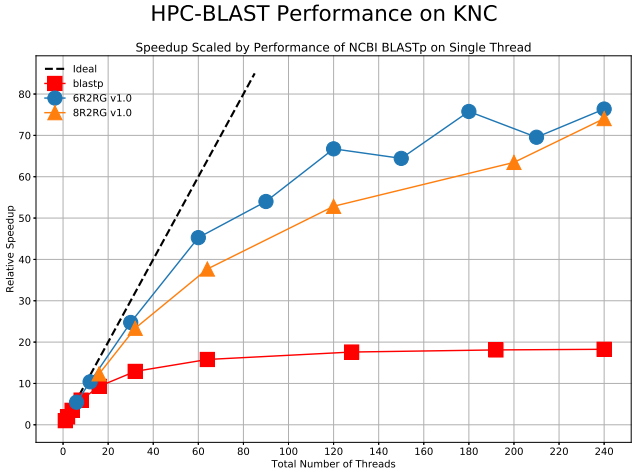

Figure 6: Comparison of strong scaling on an Intel Xeon Phi (KNC) coprocessor.

better serial processing capabilities and faster clock speeds. While these improvements lessen the performance gap as compared to the KNC, HPC-BLASTp is still able to achieve roughly 3 times speedup over NCBI BLAST as demonstrated in Figure 7.

An interesting observation is that HPC-BLAST continues to scale even when the number of threads exceeds the number of physical cores. There are two factors that contribute to the observed scaling. First, NCBI blast searches exhibit more memory-bound behavior than compute-bound behavior. Because of this, when two threads are scheduled on the same physical core but different logical cores, one thread may proceed execution in the pipeline even as the other thread is waiting for a memory access to be fulfilled. A second factor is that the OpenMP

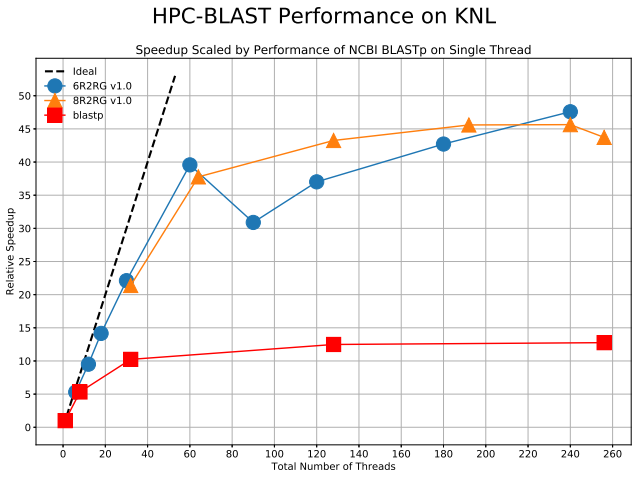

Figure 7: Comparison of strong scaling on an Intel Xeon Phi (KNL) processor.

threads perform independent NCBI BLAST searches while sharing access to the same reference database, providing increased parallelization throughout the search while not scaling memory requirements.

A weak scaling performance test was conducted on the Percival supercomputer [45] which features the Intel Xeon Phi processor (Knights Landing). The weak scaling test was devised as follows. The initial performance was measured on a single node. The same query input and database files from the strong scaling tests are used. A total of 6 MPI processes were launched with the processes being split into 2 replication groups. Each rank then spawned 40 thread groups. Thus, a total of 240 threads were searching concurrently on the node. The search was performed on a variety of node counts. Each node received the same query input file as to ensure that each node was performing the same amount of computational work. Figure 8 shows the parallel efficiency of HPC-BLASTp on the Percival supercomputer. As HPC-BLASTp in its current implementation does not perform result agglomeration between processes, it is expected that the parallel efficiency would remain above 90%, even if the number of nodes increases significantly, as almost no additional costs are incurred from MPI communication.

6 Conclusions

HPC-BLAST is designed to achieve high performance and excellent scaling on modern processor architectures featuring high core counts. This goal has been validated by strong scaling results showing performance improvements relative to NCBI BLAST on a variety of Intel architectures and by a weak scaling test demonstrating capability to run on up to 160 Xeon Phi nodes.

The are several directions for taking HPC-BLAST forward. Only the protein-to-protein (blastp) search variant has been extensively tested. Implementing the HPC-BLAST model in all blast search strategies is of interest.

The 1.0 release of HPC-BLAST uses a static scheme to partition the input queries among the MPI processes and thread groups. Moving forward, the development of a dynamic load balancing strategy would allow for greater performance by parceling out search tasks as pro-

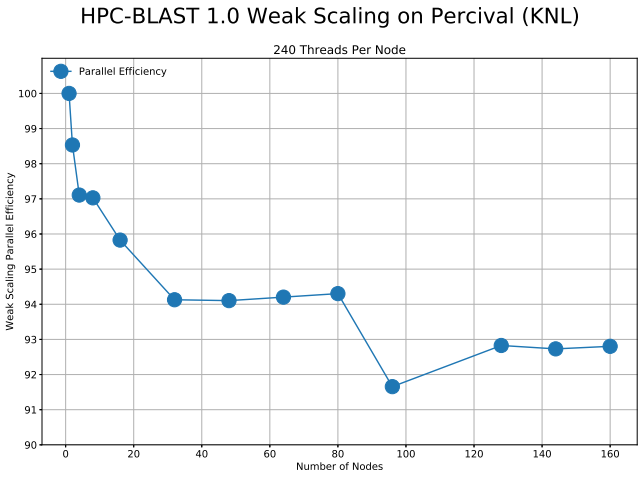

Figure 8: Weak scaling plot on the Percival supercomputer.

cesses become idle. This would also increase usability as end users would not be required to preprocess the input queries to achieve optimal performance.

Currently, the result aggregation has only been tested with the default pairwise alignment output format supported by NCBI BLAST. A strategy worth investigating is to look at an intermediate form of search results that can be communicated among processes so that fewer processes are responsible for writing output. In this manner, dedicated writing processes could operate concurrently with search processes so that both tasks execute in parallel.

7 Availability

HPC-BLAST is released under the Apache Software License 2.0 and is available at the following GitHub repository: https://github.com/UTennessee-JICS/HPC-BLAST.

8 Acknowledgments

This material is based upon work supported by (1) the National Science Foundation under grant numbers 1137907, 0711134 and ACI-1053575, (2) the University of Tennessee through the Beacon Project and the Joint Institute for Computational Sciences, and (3) Intel Corporation through an Intel Parallel Computing Center award to support development of HPC-BLAST.

This research used resources of the Oak Ridge Leadership Computing Facility at the Oak Ridge National Laboratory, which is supported by the Office of Science of the U.S. Department of Energy under Contract No. DE-AC05-00OR22725.

References

[1] Genbank and wgs statistics. https://www.ncbi.nlm.nih.gov/genbank/statistics/. Accessed: 2018-08-21.

[2] Enis Afgan, Pavithran Sathyanarayana, and Purushotham Bangalore. Dynamic task distribution in the grid for blast. In *GrC*, pages 554–557, 2006.

[3] Ankit Agrawal and Xiaoqiu Huang. Psiblast_pairwisestatsig: reordering psi-blast hits using pairwise statistical significance. *Bioinformatics*, 25(8):1082–1083, 2009.

[4] Stephen F Altschul, Warren Gish, Webb Miller, Eugene W Myers, and David J Lipman. Basic local alignment search tool. *Journal of molecular biology*, 215(3):403–410, 1990.

[5] Stephen F Altschul, Thomas L Madden, Alejandro A Schäffer, Jinghui Zhang, Zheng Zhang, Webb Miller, and David J Lipman. Gapped blast and psi-blast: a new generation of protein database search programs. *Nucleic acids research*, 25(17):3389–3402, 1997.

[6] Daniel L Ayres, Aaron Darling, Derrick J Zwickl, Peter Beerli, Mark T Holder, Paul O Lewis, John P Huelsenbeck, Fredrik Ronquist, David L Swofford, Michael P Cummings, et al. BEAGLE: an application programming interface and high-performance computing library for statistical phylogenetics. *Systemic Biology*, 61(1):170–173, 2012.

[7] Robert D Bjornson, AH Sherman, Stephen B Weston, N Willard, and J Wing. Turboblast (r): A parallel implementation of blast built on the turbohub. In *ipdps*, page 0183. IEEE, 2002.

[8] Atul J Butte. Challenges in bioinformatics: infrastructure, models and analytics. *TRENDS in Biotechnology*, 19(5):159–160, 2001.

[9] Christiam Camacho, George Coulouris, Vahram Avagyan, Ning Ma, Jason Papadopoulos, Kevin Bealer, and Thomas L Madden. Blast+: architecture and applications. *BMC bioinformatics*, 10(1):421, 2009.

[10] Michael Cameron, Hugh E Williams, and Adam Cannane. Improved gapped alignment in blast. *Computational Biology and Bioinformatics, IEEE/ACM Transactions on*, 1(3):116–129, 2004.

[11] Chen Chang. Blast implementation on bee2. *Electrical Engineering and Computer Science, Univ. of Cal Berkeley*, 2004.

[12] Ed Huai-Hsin Chi, Elizabeth Shoop, John Carlis, Ernest Retzel, and John Riedl. Efficiency of shared-memory multiprocessors for a genetic sequence similarity search algorithm. 1997.

[13] Aaron Darling, Lucas Carey, and Wu-chun Feng. The design, implementation, and evaluation of mpiblast. *Proceedings of ClusterWorld*, 2003:13–15, 2003.

[14] Rogerio Luis de Carvalho Costa and Sérgio Lifschitz. Database allocation strategies for parallel blast evaluation on clusters. *Distributed and Parallel Databases*, 13(1):99–127, 2003.

[15] Alexei J Drummond, Marc A Suchard, Dong Xie, and Andrew Rambaut. Bayesian phylogenetics with BEAUti and the BEAST 1.7. *Molecular Biology and Evolution*, 29(8):1969–1973, 2012.

[16] Robert C Edgar. Search and clustering orders of magnitude faster than blast. *Bioinformatics*, 26(19):2460–2461, 2010.

[17] Xizhou Feng, Duncan A Buell, John R Rose, and Peter J Waddell. Parallel algorithms for bayesian phylogenetic inference. *Journal of Parallel and Distributed Computing*, 63(7):707–718, 2003.

[18] Xizhou Feng, Kirk W Cameron, Carlos P Sosa, and Brian Smith. Building the tree of life on terascale systems. In *Parallel and Distributed Processing Symposium, 2007. IPDPS 2007. IEEE International*, pages 1–10. IEEE, 2007.

[19] Jeffrey D. Grant, RL Dunbrack, Frank J Manion, and Michael F Ochs. Beoblast: distributed blast and psi-blast on a beowulf cluster. *Bioinformatics*, 18(5):765–766, 2002.

[20] Hannes Hauswedell, Jochen Singer, and Knut Reinert. Lambda: the local aligner for massive biological data. *Bioinformatics*, 30(17):i349–i355, 2014.

[21] Martin C Herbordt, Yongfeng Gu, Bharat Sukhwani, and Tom VanCourt. Single pass, blast-like, approximate string matching on fpgas. In *Field-Programmable Custom Computing Machines, 2006. FCCM'06. 14th Annual IEEE Symposium on*, pages 217–226. IEEE, 2006.

[22] Daniel H Huson, Alexander F Auch, Ji Qi, and Stephan C Schuster. Megan analysis of metagenomic data. *Genome research*, 17(3):377–386, 2007.

[23] Anne Juülich. Implementations of blast for parallel computer. *Computer applications in the*

biosciences: CABIOS, 11(1):3–6, 1995.

[24] Thomas M Keane, Thomas J Naughton, Simon AA Travers, James O McInerney, and Grace P McCormack. DPRml: distributed phylogeny reconstruction by maximum likelihood. *Bioinformatics*, 21(7):969–974, 2005.

[25] Joseph Lancaster, Jeremy Buhler, and Roger D Chamberlain. Acceleration of ungapped extension in mercury blast. *Microprocessors and microsystems*, 33(4):281–289, 2009.

[26] Ben Langmead, Cole Trapnell, Mihai Pop, Steven L Salzberg, et al. Ultrafast and memory-efficient alignment of short dna sequences to the human genome. *Genome biol*, 10(3):R25, 2009.

[27] W. Lathe, J. Williams, M. Mangan, and D. Karolchik. Genomic data resources: Challenges and promises. *Nature Education*, 2(1(3)), 2008.

[28] Dominique Lavenier et al. Plast: parallel local alignment search tool for database comparison. *BMC bioinformatics*, 10(1):329, 2009.

[29] Yuheng Li, Nicholas Chia, Mario Lauria, and Ralf Bundschuh. A performance enhanced psi-blast based on hybrid alignment. *Bioinformatics*, 27(1):31–37, 2011.

[30] Heshan Lin, Pavan Balaji, Ruth Poole, Carlos Sosa, Xiaosong Ma, and Wu-chun Feng. Massively parallel genomic sequence search on the blue gene/p architecture. In *High Performance Computing, Networking, Storage and Analysis, 2008. SC 2008. International Conference for*, pages 1–11. IEEE, 2008.

[31] Heshan Lin, Xiaosong Ma, Praveen Chandramohan, Andrei Geist, and Nagiza Samatova. Efficient data access for parallel blast. In *Parallel and Distributed Processing Symposium, 2005. Proceedings. 19th IEEE International*, pages 72b–72b. IEEE, 2005.

[32] Cheng Ling and Khaled Benkrid. Design and implementation of a cuda-compatible gpu-based core for gapped blast algorithm. *Procedia Computer Science*, 1(1):495–504, 2010.

[33] Weiguo Liu, Bertil Schmidt, and Wolfgang Muller-Wittig. Cuda-blastp: accelerating blastp on cuda-enabled graphics hardware. *IEEE/ACM Transactions on Computational Biology and Bioinformatics (TCBB)*, 8(6):1678–1684, 2011.

[34] Rachel Mackelprang, Mark P Waldrop, Kristen M DeAngelis, Maude M David, Krystle L Chavarria, Steven J Blazewicz, Edward M Rubin, and Janet K Jansson. Metagenomic analysis of a permafrost microbial community reveals a rapid response to thaw. *Nature*, 480(7377):368–371, 2011.

[35] David R Mathog. Parallel blast on split databases. *Bioinformatics*, 19(14):1865–1866, 2003.

[36] Andréa Matsunaga, Maurício Tsugawa, and José Fortes. Cloudblast: Combining mapreduce and virtualization on distributed resources for bioinformatics applications. In *eScience, 2008. eScience'08. IEEE Fourth International Conference on*, pages 222–229. IEEE, 2008.

[37] Scott McGinnis and Thomas L Madden. Blast: at the core of a powerful and diverse set of sequence analysis tools. *Nucleic acids research*, 32(suppl 2):W20–W25, 2004.

[38] Zhen Meng, Jianhui Li, Yunchun Zhou, Qi Liu, Yong Liu, and Wei Cao. bcloudblast: An efficient mapreduce program for bioinformatics applications. In *Biomedical Engineering and Informatics (BMEI), 2011 4th International Conference on*, volume 4, pages 2072–2076. IEEE, 2011.

[39] Bui Quang Minh, Le Sy Vinh, Arndt Von Haeseler, and Heiko A Schmidt. pIQPNNI: parallel reconstruction of large maximum likelihood phylogenies. *Bioinformatics*, 21(19):3794–3796, 2005.

[40] Bernard ME Moret, David A Bader, and Tandy Warnow. High-performance algorithm engineering for computational phylogenetics. *The Journal of Supercomputing*, 22(1):99–111, 2002.

[41] Krishna Muriki, Keith D Underwood, and Ron Sass. Rc-blast: Towards a portable, cost-effective open source hardware implementation. In *Parallel and Distributed Processing Symposium, 2005. Proceedings. 19th IEEE International*, pages 8–pp. IEEE, 2005.

[42] NICS. Web page, 01 2018.

[43] Christopher Oehmen and Jarek Nieplocha. ScalaBLAST: A scalable implementation of BLAST for high-performance data-intensive bioinformatics analysis. *Parallel and Distributed Systems, IEEE*

Transactions on, 17(8):740–749, 2006.

[44] Christopher S Oehmen and Douglas J Baxter. Scalablast 2.0: rapid and robust blast calculations on multiprocessor systems. *Bioinformatics*, 29(6):797–798, 2013.

[45] OLCF. Web page, 01 2018.

[46] OpenMP Architecture Review Board. OpenMP application program interface version 3.0, May 2008.

[47] Huzefa Rangwala, Eric Lantz, Roy Musselman, Kurt Pinnow, Brian Smith, and Brian Wallenfelt. Massively parallel blast for the blue gene/l. In *High Availability and Performance Workshop*. Citeseer, 2005.

[48] Message Passing Interface Forum. *MPI: A Message-passing Interface Standard, Version 3.1 ; June 4, 2015*. High-Performance Computing Center, 2015.

[49] Torbjørn Rognes and Erling Seeberg. Six-fold speed-up of smith–waterman sequence database searches using parallel processing on common microprocessors. *Bioinformatics*, 16(8):699–706, 2000.

[50] Heiko A Schmidt, Korbinian Strimmer, Martin Vingron, and Arndt von Haeseler. Tree-puzzle: maximum likelihood phylogenetic analysis using quartets and parallel computing. *Bioinformatics*, 18(3):502–504, 2002.

[51] Euripides Sotiriades and Apostolos Dollas. Design space exploration for the blast algorithm implementation. In *null*, pages 323–326. IEEE, 2007.

[52] Seung-Jin Sul and Andrey Tovchigrechko. Parallelizing blast and som algorithms with mapreduce-mpi library. In *Parallel and Distributed Processing Workshops and Phd Forum (IPDPSW), 2011 IEEE International Symposium on*, pages 481–489. IEEE, 2011.

[53] Shuji Suzuki, Masanori Kakuta, Takashi Ishida, and Yutaka Akiyama. Faster sequence homology searches by clustering subsequences. *Bioinformatics*, page btu780, 2014.

[54] Joshua Tan, Durga Kuchibhatla, Fernanda L Sirota, Westley A Sherman, Tobias Gattermayer, Chia Yee Kwoh, Frank Eisenhaber, Georg Schneider, and Sebastian Maurer-Stroh. Tachyon search speeds up retrieval of similar sequences by several orders of magnitude. *Bioinformatics*, 28(12):1645–1646, 2012.

[55] Panagiotis D Vouzis and Nikolaos V Sahinidis. Gpu-blast: using graphics processors to accelerate protein sequence alignment. *Bioinformatics*, 27(2):182–188, 2011.

[56] Jiren Wang and Qing Mu. Soap-ht-blast: high throughput blast based on web services. *Bioinformatics*, 19(14):1863–1864, 2003.

[57] David L Wheeler, Deanna M Church, Scott Federhen, Alex E Lash, Thomas L Madden, Joan U Pontius, Gregory D Schuler, Lynn M Schriml, Edwin Sequeira, Tatiana A Tatusova, et al. Database resources of the national center for biotechnology. *Nucleic acids research*, 31(1):28–33, 2003.

[58] Fei Xia, Yong Dou, and Jinbo Xu. Fpga-based accelerators for blast families with multi-seeds detection and parallel extension. In *Bioinformatics and Biomedical Engineering, 2008. ICBBE 2008. The 2nd International Conference on*, pages 58–62. IEEE, 2008.

[59] Shucai Xiao, Heshan Lin, and Wu-chun Feng. Accelerating protein sequence search in a heterogeneous computing system. In *Parallel & Distributed Processing Symposium (IPDPS), 2011 IEEE International*, pages 1212–1222. IEEE, 2011.

[60] Chao-Tung Yang, Tsu-Fen Han, and Heng-Chuan Kan. G-blast: a grid-based solution for mpiblast on computational grids. *Concurrency and Computation: Practice and Experience*, 21(2):225–255, 2009.

[61] WC Yim and JC Cushman. Divide and conquer (dc) blast: Fast and easy blast execution within hpc environments. *PeerJ*, 5(e:3486), 2017.

[62] Huiliang Zhang, Bertil Schmidt, and Wolfgang Müller-Wittig. Accelerating blastp on the cell broadband engine. In *Pattern Recognition in Bioinformatics*, pages 460–470. Springer, 2008.

[63] Kaiyong Zhao and Xiaowen Chu. G-blastn: accelerating nucleotide alignment by graphics proces-

sors. *Bioinformatics*, 30(10):1384–1391, 2014.