2015

NCBI-BLAST programs optimization on XSEDE resources for sustainable aquaculture

Arun S. Seetharam
Iowa State University, arnstrm@iastate.edu

Antonio Gomez
Texas Advanced Computing Center

Catherine M. Purcell
NOAA Fisheries

John R. Hyde
NOAA Fisheries

Philip D. Blood
Carnegie Mellon University

See next page for additional authors

Authors
Arun S. Seetharam, Antonio Gomez, Catherine M. Purcell, John R. Hyde, Philip D. Blood, and Andrew J. Severin

NCBI-BLAST programs optimization on XSEDE resources for sustainable aquaculture

Arun Seetharam
Genome Informatics Facility, Iowa State University
228 Science I
Ames, IA 50014, USA
+1 (515) 294-6407
arnstrm@iastate.edu

Antonio Gomez
Texas Advanced Computing Center
10100 Burnet Road (R8700)
Austin, TX 78758
+1 (512) 232-7794
agomez@tacc.utexas.edu

Catherine M. Purcell
NOAA Fisheries, Southwest Fisheries Science Center
8901 La Jolla Shores Drive
La Jolla, California 92037
+1 (858) 546-7189
Catherine.Purcell@noaa.gov

John R. Hyde
NOAA Fisheries, Southwest Fisheries Science Center
8901 La Jolla Shores Drive
La Jolla, California 92037
+1 (858) 546-7086
John.Hyde@noaa.gov

Philip D. Blood
Pittsburgh Supercomputing Center, Carnegie Mellon University
300 S.Craig St.
Pittsburgh, PA 15213
+1 (412) 268-9329
blood@psc.edu

Andrew J. Severin
Genome Informatics Facility, Iowa State University
207 Science I
Ames, IA 50014, USA
+1 (515) 294-1320
severin@iastate.edu

ABSTRACT

The development of genomic resources of non-model organisms is now becoming commonplace as the cost of sequencing continues to decrease. The Genome Informatics Facility in collaboration with the Southwest Fisheries Science Center (SWFSC), NOAA is creating these resources for sustainable aquaculture in *Seriola lalandi*. Gene prediction and annotation are common steps in the pipeline to generate genomic resources, which are computationally intense and time consuming. In our steps to create genomic resources for *Seriola lalandi,* we found BLAST to be one of our most rate limiting steps. Therefore, we took advantage of our XSEDE Extended Collaborative Support Services (ECSS) to reduce the amount of time required to process our transcriptome data by 300 percent. In this paper, we describe an optimized method for the BLAST tool on the Stampede cluster, which works with any existing datasets or database, without any modification. At modest core counts, our results are similar to the MPI-enabled BLAST algorithm (mpiBLAST), but also allow the much needed and improved flexibility of output formats that the latest versions of BLAST provide. Reducing this time-consuming bottleneck in BLAST will be broadly applicable to the annotation of large sequencing datasets for any organism.

Categories and Subject Descriptors

J.3 [Life and Medical Sciences]: Biology and Genetics.

General Terms

Algorithms, Performance, Experimentation.

Keywords

NCBI-BLAST; mpi-BLAST; optimization; Stampede.

XSEDE '15, July 26 - 30, 2015, St. Louis, MO, USA

© 2015 ACM. ISBN 978-1-4503-3720-5/15/07$15.00

DOI: http://dx.doi.org/10.1145/2792745.2792749

1. INTRODUCTION

It is now commonplace for researchers to use high throughput genome sequencing technologies to generate terabytes of short read DNA/RNA sequencing data in order to explore a particular scientific question. Our primary research focus is the development of genomic resources that lead to environmentally friendly and economically sustainable production in crops, livestock, and aquaculture.

In our current project, we are creating genomic resources for a fish species, *Seriola lalandi*. This research will advance aquaculture practices for the U.S. *Seriola* industry and can be used to help insure viability and success of domestic finfish aquaculture programs. We are in the process of developing a toolkit that will not only provide an immediate benefit to aquaculture facilities to improve production capacity for *Seriola* species, but the genetic resources we are developing will also be powerful tools for further studies of economically important traits in the long-term. One critical step in our pipeline involves searching large sequence databases to putatively assign function to assembled transcripts and gene models.

One of the most popular tools for sequence similarity searches is the Basic Local Alignment Search Tool (BLAST) by NCBI [1]. Unfortunately, using BLAST to assign functions to transcript and gene models is a computationally intensive process that requires many cpu hours and cannot be achieved in a reasonable amount of time on our local infrastructure. To overcome this limiting step, we took advantage of the XSEDE Extended Collaborative Support Services (ECSS) and created an optimized method for the BLAST tool on the Stampede cluster. The large life sciences community that uses Stampede can benefit from this work. An efficient usage of this cluster can result in a very valuable resource for this community. Also, the lessons learned might be applied to other computational resources that do not need to be specially designed to address the requirements of this specific community.

BLAST uses a heuristic method, to find matches between the sequences. Rather than using the entire sequence for finding a match, it uses short segments (seeds) of the sequence of interest (query) to find a perfect match in the sequence database (target). Once it finds the match, local alignment is performed and alignment is extended. The match is determined to be statistically significant based on the score and e-value calculated based on the similarity matrix and the database size. Recent surge in the sequencing projects due to the ever-decreasing cost to generate sequencing data has resulted in an exponential growth of these databases and thus has increased the computational time required for the sequence similarity finding algorithms. In addition, the BLAST algorithm reads the query sequence sequentially and performs a search against the database, thereby consuming larger amounts of time with the increasing number of query sequences.

Although, several optimizations have been performed over the original BLAST program, including the MPI-enabled BLAST, they have limitations. Some optimizations follow a nested-sequence approach, wherein smaller sequences are fused together to make a large query sequence for comparison to the database, reducing the total number of query sequences (megablast) [7]. Another popular approach is to trivially parallelize the problem by splitting the input sequences into to smaller files and performing BLAST against the database separately. There are also several GUI based front-ends developed for this purpose that handle the input splitting, sequence searching and merging output files automatically. GPU-acceleration has also been implemented in an attempt to speed up the BLAST algorithm but it has not taken off due to the Moore's law increase in cpu speed every year [5]. A parallel implementation of the BLAST program called mpiBLAST is available that can perform BLAST searches more efficiently using large number of processors [4]. All of these implementations result in an increase in the processing speed of large sequencing queries. However, each have their limitations: 1) nested BLAST changes statistical scores for the query sequences and requires further parsing for obtaining readable results, 2) splitting input files creates a huge burden on I/O, and reduces the performance, 3) GUI based front-ends are not open source, 4) not all compute clusters provide GPU acceleration., and finally 5) mpiBLAST provides very little flexibility on the output configurations. Not all NCBI-BLAST options are available in mpiBLAST. Output options are limited to tabular (with or without headers), xml and regular BLAST output, whereas regular BLAST allows 11 output formats. Certain tasks such as identifying the contaminant sequences using BLAST require re-configuring tabular output (by including taxonomical IDs as a field), or in some cases listing only subject sequence ids (without description) are necessary; these options are not allowed in mpiBLAST. Taxonomic classification is becoming increasingly important in the field of metagenomics where hundreds to tens of thousands of organisms are being sequenced simultaneously.

In this study, we have optimized NCBI-BLAST to efficiently run on our dataset using Stampede high performance computing cluster. We present a set of studies on how to efficiently run BLAST on Stampede. Because of the wide use of BLAST in the community, the impact that a set of optimal settings for running BLAST on this XSEDE resource is of high relevance for many users. We have fine-tuned the performance of the BLAST algorithm and have automated splitting of input files, job submission and merging of results in a manner that has a tolerable load on I/O.

2. Algorithm

NCBI-BLAST suite of programs provide tools for searching query sequences against various databases. Among the most popular algorithms: *blastn*, searches nucleotide query sequence against the nucleotide database; *blastp*, searches protein query sequence against the proteins sequence database; *blastx*, searches translated nucleotide sequence against the protein database; *tblastn*, uses protein query sequence to search against translated nucleotide database and *tblastx* uses translated nucleotide query against translated nucleotide database. All of these programs are identical except input query and/or database; protein or nucleotide.

All of our BLAST optimizations were performed using the NCBI-BLAST (version 2.2.28+). The basic BLAST algorithm is simple and robust, meaning that it can be used in many ways to inform on a particular biological question of interest by utilizing the available sequence databases. Here is a brief overview of some of the more common uses.

- Gene model identification: An unknown sequence from a biological sample of an organism that has a database of known gene models.
- Motif/domain searches: A subset of genes that contain regulatory elements or specific protein domains can be identified in a database.
- Gene function prediction: The function of gene models/transcripts from new species can be predicted based on similarity to sequences with known functions in existing databases.
- Taxonomic classification: Using sequence similarity scores, sequences of unknown origin can be placed into taxonomic groups. This is especially helpful for metagenomic samples where many species are sequenced simultaneously.
- Contamination removal: Samples sometimes contain unwanted sequences. Using similarity and taxonomy, these can be removed prior to deposition into a database. This is vital to ensure the integrity of our sequence databases as these databases are used to assign function and clade to unknown sequences.
- Synteny: By combining gene model order contained in a sequence with the sequence similarity of gene models between organisms stretches of sequences can be identified that originated from a common ancestor.

For comparative purposes, we tested our optimizations against the mpiBLAST (version 1.6.0) tool developed at Synergy Lab; Virginia Tech. The mpiBLAST program uses database segmentation approach for running the basic BLAST programs in parallel. Like BLAST, mpiBLAST can also be used across different databases using either nucleotide or protein sequence as query. The statistical values (score and evalue) are comparable with the basic BLAST. Although this program speeds up linearly with the huge sequence databases, it provides limited configurations with respect to output as compared to basic BLAST, limiting the usefulness of mpiBLAST for some of the uses described above.

3. Research Problem

Seriola species (*S. dumerili, S. lalandi, S. rivoliana, S. quinqueradiata*), collectively known as amberjacks, are fish of particular interest to the growing aquaculture industry due to their high value, forming a billion dollar plus component of the sashimi industry. In many locations, one or more of these species comprise a large percentage of the total marine finfish in culture (e.g., over 60% of total mariculture in Japan). Culture of these species has traditionally relied heavily on harvesting and growout of wild juveniles, which can be unpredictable in supply and puts excessive pressure on natural populations. In the U.S. and elsewhere, hatchery production of *Seriola lalandi* is rapidly growing but has been hindered by a propensity for deformities and growth heterogeneity developed during larval and early juvenile stages that limit the production capacity and efficiency.

In order to develop environmentally friendly and economically sustainable *Seriola lalandi* aquaculture, an understanding of the genetic basis of traits that currently limit/enhance the development and progress of domestic aquaculture is needed. To this end, approximately 1 billion reads were generated from an Illumina HiSeq 2500 instrument. These reads were assembled using Trinity [3], and resulted in 240,022 transcripts (82,782 translated proteins) These transcripts were BLASTed against the curated protein database (UniRef 90) [2], to putatively assign function as well as remove any potential contaminants. While the Trinity was successfully run on our local clusters, we required more computational power to run our BLAST jobs and therefore created this optimization for BLAST on Stampede. We used both nucleotide transcripts as well as transcoded protein sequences as queries (*blastx* and *blastp*). Gene model annotation is an important step in the creation of genomic resources for non-model organisms, which is becoming increasing popular as the cost of sequencing decreases and the quality of the assemblies increases.

3.1 Database

UniProt Reference Clusters (UniRef) provides clustered sets of sequences from the UniProt Knowledgebase (including isoforms) and selected UniParc records. A subset of these, that have at least 90% sequence identity (referred as UniRef90 proteins) were used as our target database for BLAST. With over 32 million sequences, all with rich functional annotation, they can be readily used to assign function for the unknown sequences.

4. Running BLAST on Stampede

We needed an efficient parallel implementation of BLAST on Stampede for processing these increasingly common and larger datasets. This type of application represents a challenge for the distributed parallel file system due to the continuous requests to the metadata server (MDS). This produces high loads in the MDS and can lead to instabilities in the file system and even crashes. This is particularly critical when large numbers of instances of BLAST run at the same time. Stampede has two different modules for BLAST available for all the users: BLAST (versions 2.28 and 2.29) and mpiBLAST 1.6.0.

Stampede contains 6400 dual socket eight-core Sandy-Bridge E5-2680 server nodes with 32 GB of memory. The nodes are interconnected by InfiniBand HCAs in FDR mode and the operating system used is CentOS 6.4 with kernel 2.6.32-358.el6. Each node has its own local disk with 72 GB of disk available for the users to write temporary files using the /tmp folder.

4.1 Computational challenges

As mentioned, the considerable impact of these applications on the file system necessitate finding an optimal approach for efficiently running large numbers of BLAST instances in parallel without leading to instabilities in the file system. We present three different alternatives for using BLAST on Stampede: sequential BLAST with Launcher, mpiBLAST, and threaded BLAST.

4.1.1 Sequential BLAST with Launcher

A typical scenario for running BLAST on Stampede is to submit many independent jobs where each one job runs a single sequential instance of BLAST. Since each BLAST instance uses only one core, the other 15 cores on each node are still available as resources. Stampede provides a tool called launcher [6] for this type of serial applications. This tool enables the user to utilize all available cores on each node, so that each core runs a serial application. While this scenario is typically ideal for these applications, for other applications that are I/O demanding, this can represent a problem especially when large numbers of these jobs are submitted. For example, when 20 jobs were running at the same time (320 cores), then the number of I/O requests per second were greater than 2000000. This already high load may be additionally taxed by several hundred users also accessing the file system during the same period.

In our experience with Stampede, this high load can create instabilities in the file system if different users require so many accesses at the same time. Therefore this is not an approach that can be recommended.

4.1.2 mpiBLAST

The first step to run mpiBLAST is to reformat the database since this code uses its own format. The database only needs to be reformatted once, so we do not include the time required for formatting in the results in the next section. It is recommended to set the stripes for the file system to at least 60.

BLAST and mpiBLAST typically require a set of variables to be defined in a .ncbirc file. In our experience with Stampede, using this file with large number of cores can lead to timeouts that will also lead to MPI aborts since all the processes will try to access the file at the same time. The solution is to define these variables as environment variables.

The mpiBLAST allows the use of a very large numbers of cores (as shown in Sec. 5) and its MPI-based implementation achieves a relatively low impact on the distributed file system.

4.1.3 Threaded BLAST

The other approach is to run BLAST with threads. BLAST provides a threaded implementation using OpenMP that allows the user to use all the available cores in one node. The achieved speedup is not optimal due to the nature of the code, but it does improve the performance of the sequential version of BLAST while also reducing the impact on the file system when compared with having one sequential instance per core on the node. In this case, as each Stampede node has 16 cores, so up to 16 threads per node is possible. In Stampede we can allocate several nodes, and run one instance of BLAST per node. Each of those instances will use the 16 cores available at the node using threads. However, if all the instances use the same input and the same database, they will all perform the same computations. So the approach is to split the input into as many parts as there are allocated nodes (or BLAST instances). Thus, each instance will work on a subset of the input data. Once all the BLAST instances have finished, the results are joined together. Initially we launched a set of jobs with this configuration, and the number of I/O requests per second quickly rose to over 600000. While this value is also very large, it was not high enough to create a problem in the system by itself, however it could become a problem if other users have heavy I/O at the same time.

With these results, and with the aim of reducing the impact on the file system, we decided to move all the files from the parallel file system to the local disk that each node has. This requires a larger change in the script used to submit the job:

- Instead of running BLAST directly, the job runs a script.
- The script copies all files required from the parallel file system to the local disk of the node
- Since every time a job is submitted it will likely be allocated to a different node, and since the local node disk is emptied once the job has finished, this copy must be performed with each job run. Because the outputs of the different nodes have to be combined at the end of the execution, the script retrieves the rank of each individual node. This rank is used to create unique names for the output files.

By using this method, the impact of running BLAST on large number of cores in the parallel file system is almost negligible. In fact, when increasing the number of cores, the number of requests to the MDS remained almost constant. However, in this case the bandwidth of the local disk becomes a bottleneck. For large files, it might take several minutes to transfer the files. And it also introduces a limitation, as there are only 72GB available on the local disk of each node in Stampede. If the files are larger than the available space, this technique will not work.

5. Results

With *blastx*, for 244,022 sequences, more than 62% (152,882) of the total query sequences had a match in the UniRef90 database. Among these matches, 97,735 had e-value less than 0.01. Similarly with *blastp* (translated proteins), for 87,782 sequences, more than 98% (81,292) had matches in the UniRef90 database (with 76,353 hits of e-value less than 0.01).

Regarding the performance of the different approaches for running BLAST that we previously presented, Figure 1 shows the time required by the described dataset when using mpiBLAST and BLAST with OpenMP and the local disk. It shows that for small numbers of cores, BLAST is able to run faster than mpiBLAST. However, as the number of cores increases, mpiBLAST outperforms BLAST. This is due to the time required by the BLAST execution to copy all the required files from the distributed file system to the local disk. This is an almost constant time considering that we are always using the same dataset. The results of running BLAST (without OpenMP) with the launcher tool are not included here since the impact on the file system performance of this method is so high that it does not represent a valid alternative in Stampede.

With these results, it is clear that the best approach is to use mpiBLAST when possible. However, if subsequent steps in the analyses require custom output formats of the traditional BLAST, our method will be the solution that is most applicable. For smaller databases, the proposed model using the local disk should provide much better scalability and its results should be closer to those achieved with mpiBLAST.

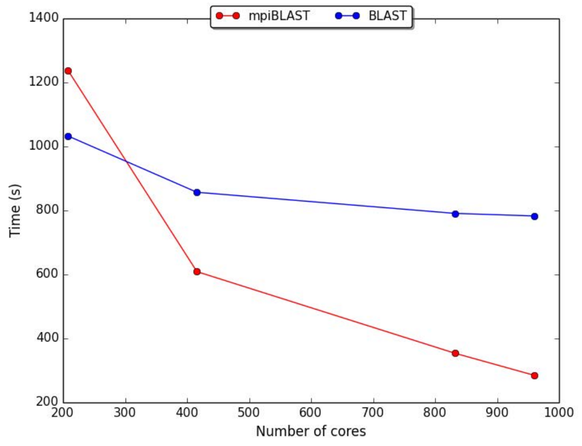

Fig 1. Time required by our dataset when using mpiBLAST or BLAST with OpenMP and local disk

6. Conclusion and future outlook

Our approach provides a convenient way to speed up the basic BLAST algorithm on Stampede. By design, NCBI-BLAST cannot take advantage of distributed infrastructures available in most of the high performance compute clusters. But the program can be trivially parallelized by splitting the input file into smaller files. Although, running several BLAST jobs generates heavy burden on I/O, it can be mitigated by using the local disk of the compute nodes to store the database files. Another alternative for the BLAST is using MPI-enabled BLAST program such as mpiBLAST. Such programs can provide huge improvements in in the run time, but with limited parameter configurations for BLAST. If this limitation does not represent a problem, its simplicity and performance make it the best approach. When mpiBLAST is not an option, the approach that we have presented in this paper using the local disk of each node still achieves excellent results. Based on our results, we strongly recommend to members of the biology community working with these large data sets to use one of the two above approaches rather than sequential BLAST.

6.1.1 Significance

In this study, we used the XSEDE resource, Stampede, to assign function to Trinity transcripts from the *Seriola lalandi,* from curated proteins database (UniRef90) in a more reasonable amount of time than our local infrastructure could provide. Using both nucleotide sequences as well translated sequences (with open reading frames), we found over 62% and 98% of the sequences with matches in the database, respectively. High percent of matches with the protein sequences are as expected, because the translation step eliminates most of the sequences without open reading frames (spurious sequences). Protein sequences are also able to find deeper evolutionary relationship among the sequences as compared to nucleotide sequences. With this method, we were able to eliminate non-fish transcripts to obtain high quality *Seriola lalandi* transcripts. These transcripts and assigned functions will eventually find their way into the annotation of functions for the predicted gene models of the *Seriola* genome project currently underway. A website that includes a genome browser with these gene models and their corresponding annotations will eventually be created. The utilization of XSEDE resources will significantly speed up the most time consuming steps in the process of creating these type of genomic resources for new species.

7. Availability

All the scripts used in this study for running optimized BLAST on Stampede along with the documentation are available on GitHub: https://github.com/ISUgenomics/StampedeBLAST.

8. ACKNOWLEDGMENTS

This work was made possible by the XSEDE research allocation (Grant ID MCB140217) and Extended Collaborative Support Service (ECSS). We are also thankful for Gary Polking and Michael Baker in ISU DNA facility, for helping with our sequencing needs.

9. REFERENCES

[1] ALTSCHUL, S.F., GISH, W., MILLER, W., MYERS, E.W., and LIPMAN, D.J., 1990. Basic local alignment search tool. *J Mol Biol 215*, 3 (Oct 5), 403-410. DOI= http://dx.doi.org/10.1016/S0022-2836(05)80360-2.

[2] CHAN, W.M. and CONSORTIUM, U., 2010. The UniProt Knowledgebase (UniProtKB): a freely accessible, comprehensive and expertly curated protein sequence database. *Genetics Research 92*, 1 (Feb), 78-79.

[3] GRABHERR, M.G., HAAS, B.J., YASSOUR, M., LEVIN, J.Z., THOMPSON, D.A., AMIT, I., ADICONIS, X., FAN, L., RAYCHOWDHURY, R., ZENG, Q.D., CHEN, Z.H., MAUCELI, E., HACOHEN, N., GNIRKE, A., RHIND, N., DI PALMA, F., BIRREN, B.W., NUSBAUM, C., LINDBLAD-TOH, K., FRIEDMAN, N., and REGEV, A., 2011. Full-length transcriptome assembly from RNA-Seq data without a reference genome. *Nature Biotechnology 29*, 7 (Jul), 644-U130. DOI= http://dx.doi.org/DOI 10.1038/nbt.1883.

[4] LIN, H.S., MA, X.S., FENG, W.C., and SAMATOVA, N.F., 2011. Coordinating Computation and I/O in Massively Parallel Sequence Search. *Ieee Transactions on Parallel and Distributed Systems 22*, 4 (Apr), 529-543. DOI= http://dx.doi.org/Doi 10.1109/Tpds.2010.101.

[5] VOUZIS, P.D. and SAHINIDIS, N.V., 2011. GPU-BLAST: using graphics processors to accelerate protein sequence alignment. *Bioinformatics 27*, 2 (Jan 15), 182-188. DOI= http://dx.doi.org/10.1093/bioinformatics/btq644.

[6] WILSON, L.A. and FONNER, J.M., 2014. Launcher: A Shell-based Framework for Rapid Development of Parallel Parametric Studies. In *Proceedings of the Proceedings of the 2014 Annual Conference on Extreme Science and Engineering Discovery Environment* (Atlanta, GA, USA2014), ACM, 2616534, 1-8. DOI= http://dx.doi.org/10.1145/2616498.2616534.

[7] ZHANG, Z., SCHWARTZ, S., WAGNER, L., and MILLER, W., 2000. A greedy algorithm for aligning DNA sequences. *J Comput Biol 7*, 1-2 (Feb-Apr), 203-214. DOI= http://dx.doi.org/10.1089/10665270050081478.