Semantics-based Distributed I/O for mpiBLAST *

P. Balaji* W. Feng† J. Archuleta† H. Lin‡ R. Kettimuthu* R. Thakur* X. Ma‡§

Mathematics and Computer Science, Argonne National Laboratory*
Department of Computer Science, Virginia Tech†
Department of Computer Science, North Carolina State University‡
Computer Science and Mathematic Division, Oak Ridge National Laboratory§
balaji@mcs.anl.gov feng@cs.vt.edu jsarch@cs.vt.edu hlin2@ncsu.edu kettimut@mcs.anl.gov
thakur@mcs.anl.gov ma@cs.ncsu.edu

Abstract

BLAST is a widely used software toolkit for genomic sequence search. mpiBLAST is a freely available, open-source parallelization of BLAST that uses database segmentation to allow different worker processes to search (in parallel) unique segments of the database. After searching, the workers write their output to a filesystem. While mpiBLAST has been shown to achieve high performance in clusters with fast *local* filesystems, its I/O processing remains a concern for scalability, especially in systems having limited I/O capabilities such as distributed filesystems spread across a wide-area network. Thus, we present *ParaMEDIC*—a novel environment that uses application-specific semantic information to compress I/O data and improve performance in distributed environments. Specifically, for mpiBLAST, ParaMEDIC partitions worker processes into compute and I/O workers. Compute workers, instead of directly writing the output to the filesystem, the workers process the output using semantic knowledge about the application to generate *metadata* and write the *metadata* to the filesystem. I/O workers, which physically reside closer to the actual storage, then process this metadata to re-create the actual output and write it to the filesystem. This approach allows ParaMEDIC to reduce I/O time, thus accelerating mpiBLAST by as much as 25-fold.

Categories and Subject Descriptors J.3 [*Computer Applications*]: Life and Medical Sciences

General Terms Design, Performance

Keywords mpiBLAST, I/O, distributed filesystem

1. Introduction

Many computational biology tools and applications use nucleotide and protein sequence-searches to find similarities between different species of organisms. Given the importance of sequence searches, researchers have designed a number of tools to perform sequence search in an efficient manner. Among the most widely used sequence-search tools is the Basic Local Alignment Search Tool (BLAST) from the National Center for Biotechnology Information (NCBI). mpiBLAST [1, 2] is a freely available, open-source parallelization of NCBI BLAST. The overall software architecture of mpiBLAST follows a master-worker model. The master fragments the sequence database across multiple nodes so that each fragment fits in memory. Each worker process then searches its unique portion of the database independently of the other workers. Once the search is complete, the results are merged and written out. While this model works well for clusters with a fast *local* filesystem, it works poorly when the filesystem is distributed across a wide-area network [3], e.g., NSF TeraGrid and the distributed and encrypted compute environment between Argonne National Lab (ANL) and Virginia Tech (VT).

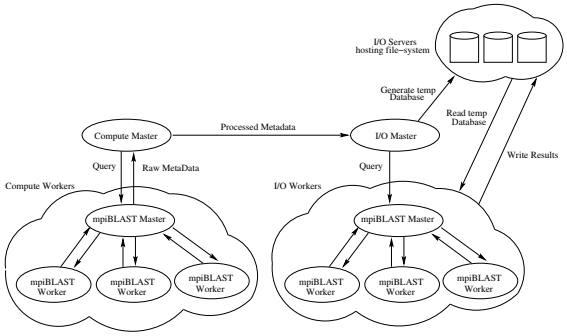

Figure 1. ParaMEDIC Framework

Thus, we propose *ParaMEDIC: **Para**llel **M**etadata **E**nvironment for **D**istributed **I/O** and **C**omputing*, a novel environment that uses application-specific semantic information to compress I/O data and improves performance in distributed environments. Specifically, ParaMEDIC divides the worker processes into two groups: compute workers and I/O workers. The compute workers reside on the compute cluster (as before), while the I/O workers reside physically closer to the actual storage. When a biologist requests that a query be searched, the compute workers act as before, but instead of directly writing the output to the filesystem, the workers process the output using semantic knowledge about the application to generate orders-of-magnitude smaller *metadata* and write this *metadata* to the filesystem, thus significantly reducing the I/O time taken by the compute workers. The I/O workers perform a small amount of additional processing of this metadata to re-generate the final output and write it to the filesystem. Since the I/O workers are physically closer to the actual storage, this model is substantially faster than traditional distributed I/O.

2. ParaMEDIC Design Overview

ParaMEDIC provides a two-tiered hierarchical framework for decoupling computation and I/O in mpiBLAST. The upper tier consists of two processes, *compute manager* and *I/O manager*, while the lower tier consists of two groups of processes – *compute workers* and *I/O workers*. The actual sequence search is handled by the compute workers. Once the output is generated, the compute master *understands* the semantics of the output data and converts this output initially to raw metadata, and then processes and compresses it to form the final metadata. It then writes the final metadata to the filesystem and sends a signal to the I/O master. The I/O master, upon receiving a signal from the compute master, uses the I/O workers to process the metadata and generate the final output. We note that the amount of computation required is higher than what is required by the original application (due to the additional metadata processing). However, such metadata processing potentially allows the the I/O cost to be significantly reduced. In other words, the ParaMEDIC framework aims at trading a small amount of additional computation for reduced I/O cost.

Managing Compute and I/O Worker Processes: Managing the compute and I/O worker processes in ParaMEDIC essentially deter-

* This research is funded in part by the Mathematical, Information, and Computational Sciences Division subprogram of the Office of Advanced Scientific Computing Research, Office of Science, U.S. Department of Energy, under Contract DE-AC02-06CH11357 and the Computer Science Department at Virginia Tech.

PPoPP'08, February 20–23, 2008, Salt Lake City, Utah, USA.
ACM 978-1-59593-960-9/08/0002.

mines the tradeoff in the amount of time spent in computation versus the amount of time saved in I/O. In general, since the I/O worker processes are restricted to the cluster that hosts the filesystem, the number of I/O workers that are available is restricted. For example, in a distributed environment hosting 10,000 processors, only 1000 processors might reside on the same cluster that hosts the filesystem. Thus, in this case, it is ideal to maintain a 10:1 ratio between the number of compute workers and the number of I/O workers. Depending on the ratio, the appropriate metadata processing scheme needs to be picked.

Metadata Processing for mpiBLAST: Several sequence databases use unique identifiers for each sequence in the database. For the nucleotide database, for example, GenBank Identifiers (GIs) are used to represent the different sequences. The output that is generated from mpiBLAST typically consists of the sequences themselves, together with a significant amount of additional information. ParaMEDIC parses through the results generated by the mpiBLAST compute workers to extract the GI information for each matching sequence and compress the output before sending it to the I/O workers. Depending on the query, this compressed GI information can be nearly three to four orders of magnitude smaller than the actual output.

I/O Post-Processing: I/O post-processing consists of two primary components—database creation and query search. For the former, the I/O master creates a new temporary database based on the *matched segments* that were found by the compute workers. This temporary database is *much* smaller than the original database. For example, in the default configuration, if a single query sequence is provided by the user, while the actual database has about 5 million sequences, the temporary database will have at most 500 sequences, i.e., 0.01% of the original size. Once the temporary database is created, the I/O workers recompute the original query sequences against this temporary database to generate the final output. Since the temporary database is very small, this search time is minimal.

3. Performance Evaluation

We first look at the performance of mpiBLAST and ParaMEDIC on a distributed system between ANL and VT connected over Internet2. Next, we illustrate the performance of mpiBLAST and ParaMEDIC on the TeraGrid infrastructure.

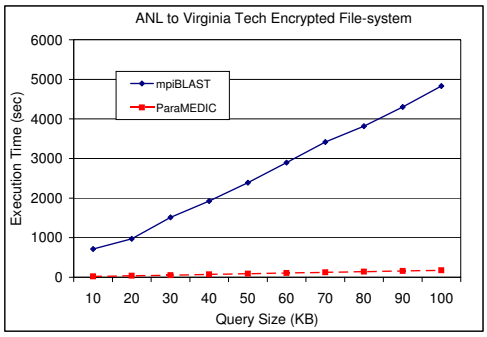

Figure 2. ANL-VT Encrypted Filesystem

Figure 2 shows that ParaMEDIC significantly outperforms mpiBLAST in this environment with the performance difference increasing with query size. For a query size of 100 KB, we observe more than a *25-fold* improvement in performance. This difference is attributed to multiple aspects. First, given the shared network connection between the two sites, the effective network performance achievable is usually lower than within the cluster. Thus, with mpiBLAST transferring the entire output over the network, its performance would be heavily impacted by the network performance. Second, since data is communicated over the Internet, it is locally encrypted before being transmitted; mpiBLAST has to pay the penalty for such encryption. Though ParaMEDIC also pays such data encryption penalty, the amount of data it transfers is significantly lesser, and hence the penalty is less as well. Third, the distance between the two sites causes the communication latency to be high. Thus, filesystem operations tend to take a large amount of time, resulting in further loss of performance.

The TeraGrid infrastructure represents a distributed environment for several compute- and I/O-intensive applications including mpiBLAST. A GPFS-based distributed filesystem is hosted at SDSC, which can be accessed from all facilities and forms a part of the TeraGrid facility. Since TeraGrid is a dedicated facility, it does not utilize any encryption of the data exchanged between sites. For our experiments, we utilized the nodes at the University of Chicago and SDSC. The I/O workers in ParaMEDIC are always scheduled at SDSC because of its close proximity to the actual storage. The compute workers, however, are scheduled at the University of Chicago.

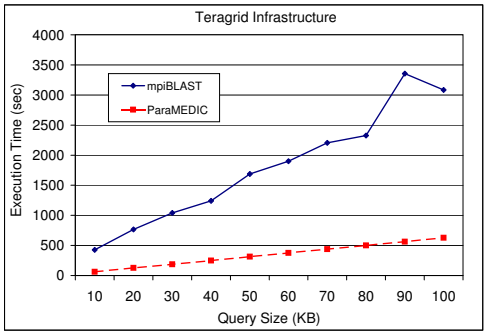

Figure 3. Evaluation on the TeraGrid Infrastructure

Figure 3 shows the performance of mpiBLAST and ParaMEDIC on the TeraGrid infrastructure. While the final output is written to the same global filesystem in both cases, mpiBLAST suffers from the fact that the compute workers are performing the I/O for the output results. Since they reside on a remote cluster as compared to the actual storage, their I/O performance is limited resulting in an overall degradation in execution time. For ParaMEDIC, on the other hand, since the I/O workers are performing the I/O for the output results, the amount of time taken is significantly smaller. For a query file size of 100 KB, ParaMEDIC outperforms mpiBLAST by about *five times*.

4. Concluding Remarks

mpiBLAST is an open-source parallelization of BLAST, a widely used software for genome sequence searching. In spite of recent enhancements, I/O processing in mpiBLAST is still a concern, especially in environments that use distributed filesystems with limited I/O capabilities. In this paper, we presented *ParaMEDIC*—an environment that uses application-specific semantic information to compress I/O data and improve performance in distributed environments. We provided details of the ParaMEDIC design and studied its performance impact on mpiBLAST in two distributed environments: (i) an encrypted file-system between Argonne National Lab (VT) and Virginia Tech (VT) over Internet2 and (ii) NSF TeraGrid. We demonstrated that ParaMEDIC can accelerate mpiBLAST by *25-fold* in the ANL-VT distributed environment and by *5-fold* on TeraGrid.

References

[1] A. Darling, L. Carey, and W. Feng. The Design, Implementation, and Evaluation of mpiBLAST. In *International Conference on Linux Clusters: The HPC Revolution 2003*, 2003.

[2] W. Feng. Green destiny + mpiblast = bioinfomagic. In *International Conference on Parallel Computing (ParCo)*, 2003.

[3] M Gardner, W Feng, J Archuleta, H Lin, and X Ma. Parallel genomic sequence-searching on an ad-hoc grid: Experiences, lessons learned, and implications. In *ACM/IEEE SC2006: The International Conference on High-Performance Computing, Networking, and Storage*, 2006.