University of Nebraska at Omaha
DigitalCommons@UNO

Computer Science Faculty Proceedings & Presentations

Department of Computer Science

1-2008

On the Tradeoff between Speedup and Energy Consumption in High Performance Computing – A Bioinformatics Case Study

Sachin Pawaskar
University of Nebraska at Omaha, spawaskar@unomaha.edu

Hesham Ali
University of Nebraska at Omaha, hali@unomaha.edu

Follow this and additional works at: https://digitalcommons.unomaha.edu/compsicfacproc

Part of the Bioinformatics Commons, and the Computer Sciences Commons

Recommended Citation
Pawaskar, Sachin and Ali, Hesham, "On the Tradeoff between Speedup and Energy Consumption in High Performance Computing – A Bioinformatics Case Study" (2008). *Computer Science Faculty Proceedings & Presentations*. 52.
https://digitalcommons.unomaha.edu/compsicfacproc/52

This Conference Proceeding is brought to you for free and open access by the Department of Computer Science at DigitalCommons@UNO. It has been accepted for inclusion in Computer Science Faculty Proceedings & Presentations by an authorized administrator of DigitalCommons@UNO. For more information, please contact unodigitalcommons@unomaha.edu.

On the Tradeoff between Speedup and Energy Consumption in High Performance Computing – A Bioinformatics Case Study

Sachin Pawaskar and Hesham H. Ali
Department of Computer Science
College of Information Science and Technology
University of Nebraska at Omaha
Omaha, NE 68182, USA
sachinpawaskar@msn.com | hali@unomaha.edu

ABSTRACT

High Performance Computing has been very useful to researchers in the Bioinformatics, Medical and related fields. The bioinformatics domain is rich in applications that require extracting useful information from very large and continuously growing sequence of databases. Automated techniques such as DNA sequencers, DNA microarrays & others are continually growing the dataset that is stored in large public databases such as GenBank and Protein DataBank. Most methods used for analyzing genetic/protein data have been found to be extremely computationally intensive, providing motivation for the use of powerful computers or systems with high throughput characteristics. In this paper, we provide a case study for one such bioinformatics application called BLAT running in a high performance computing environment. We use sequences gathered from researchers and parallelize the runs to study the performance characteristics under three different query and data partitioning models. This research highlights the need to carefully develop a parallel model with energy awareness in mind, based on our understanding of the application and then appropriately designing a parallel model that works well for the specific application and domain. We found that the BLAT program is highly parallelizable and a high degree of speedup is achievable. The experiments suggest that the speed up depends on model used for query and database segmentation..

KEY WORDS

High Performance Computing, Bioinformatics, Energy Awareness, Parallel Processing, Sequence Comparisons

1. Introduction

Bioinformatics can be broadly defined as the creation and development of advanced information and computational techniques for problems in biology/genetics domain. It is the set of computing techniques used to manage and extract useful information from the DNA/RNA/protein sequence data which is continually being generated (at very high volumes) by automated techniques (e.g., DNA sequencers, DNA microarrays) and stored in large public databases (e.g., GenBank, Protein DataBank). Most methods used for analyzing genetic/protein data have been found to be extremely computationally intensive, providing motivation for the use of powerful computers or systems with high throughput characteristics.

High-performance computing describes a set of hardware and software techniques developed for building computer systems capable of quickly performing large amounts of computation. These techniques have generally relied on harnessing the computing power of large numbers of processors working in parallel, either in tightly-coupled shared-memory multiprocessors or loosely-coupled clusters of PCs. Experience has shown a great deal of software support is necessary to support the development and tuning of applications on parallel architectures. The marriage between the bioinformatics domain and high performance computing is a natural one, the problems in this domain tends to be highly parallelizable and deal with large datasets, hence using HPC is a natural fit.

Power consumption has been a critical design constraint in the design and setup of high performance computing systems. An increasing amount of system functionality tends to be realized through software, which is leveraged by the high performance of modern processors. As a consequence, reduction of the power consumption of processors is important for the power-efficient design and operation of such systems. Broadly, there are two kinds of methods to reduce power consumption of processors. The first is to bring a processor into a power-down mode, where only certain parts of the processor such as the clock generation and timer circuits are kept running when the processor is in an idle state. Most power-down modes have a tradeoff between the amount of power saving and the latency incurred during mode change. Therefore, for an application where latency cannot be tolerated, such as for a real-time system, the applicability of power-down may be restricted. Another method is to dynamically change the processor speed by varying the clock frequency along with the supply voltage when the required performance on the processor is lower than the maximum performance. A significant power reduction can be obtained by this method because the dynamic power of a CMOS circuit is quadratically dependent on the supply voltage [3].

Comparing biological sequences is one of the most important Bioinformatics problems because it is critical for recognition and classification of organisms. The software package BLAST (Basic Local Alignment Search Tool) has been the method of choice for many biomedical researchers to measure the degree of similarity among biological sequences. Recently, a modified version, called BLAT (the BLAST-Like Alignment Tool) is quickly becoming a very popular tool for similarity measures using the concept of sequence alignment. BLAT, developed by Jim Kent at UCSC to identify similarities between DNA and protein sequences, is an alignment tool like BLAST, but it is structured differently. On DNA, BLAT works by keeping an index of an entire genome in memory. Thus, the target database of BLAT is not a set of GenBank sequences, but instead an index derived from the assembly of the entire genome. The index which uses less than a gigabyte of RAM consists of all non-overlapping 11-mers except for those heavily involved in repeats [1 – 2].

In this paper we propose an energy aware scheduling (EAS) technique for programs in a cluster environment and apply the EAS technique to the bioinformatics domain and more specifically to the BLAT software package. It is important to note that we can parallelize the BLAT program without losing any biologically significant information relevant to the output of the program. This means that parallelizing the program does not impact the conclusions that bioinformatics researchers may draw from the output of BLAT.

2. Energy Aware Scheduling

Scheduling is a classical field with several interesting problems and results. Due to its wide range of applications, the scheduling problem has been attracting many researchers from a number of fields. A scheduling problem emerges whenever there is a choice. The choice could be the order in which a number of tasks can be performed, and/or in the assignment of tasks for processing.

The problem is to determine some sequences of these operations that are preferred according to certain (e.g. economic) criteria. The problem of discovering these preferred sequences is referred to as the sequencing problem. Over the years, several methods have been used to deal with the sequencing problem such as complete enumeration, heuristic rules, integer programming, and sampling methods. It is clear that complete enumeration is impractical because the problem is exponential, which means that it requires too much time, sometimes years of computation time would be required even for a small number of tasks. Hence optimal solutions cannot be obtained in real time [4, 5]. However, many heuristic methods have been used to deal with most general case of the problem. Such methods include traditional priority-based algorithms [6], task merging techniques [7], critical path heuristics [6, 8]. In addition, distributed algorithms have been designed to address different versions of the scheduling problem [9].

In general, the scheduling problem assumes a set of resources and a set of consumers serviced by these resources according to a certain policy. Based on the nature of and the constraints on the consumers and the resources, the problem is to find an efficient policy (schedule) for managing the access to and the use of the resources by various consumers to optimize some desired performance measure such as the total service time (schedule length).

Energy Aware Scheduling is a special case of the general scheduling problem in which our scheduling policy is the optimization of the energy or power of the battery. Minimizing the battery power utilization becomes the most important consideration in a system that is energy aware, at the same time one must realize that along with this there are certain parameters that must be met such as tasks meeting their deadlines.

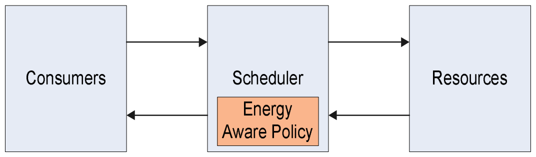

Figure 1: Energy Aware Scheduling System

Simply put an Energy Aware Scheduling System is a scheduling problem which assumes a set of resources and a set of consumers serviced by these resources according to a Energy Aware policy. Based on the nature of and the constraints on the consumers and the resources, the problem is to find an efficient policy (schedule) for managing the access to and the use of the resources by various consumers to optimize the desired performance measure which in this case is minimum amount of battery energy. Accordingly, an Energy Aware scheduling system can be considered as consisting of a set of consumers, a set of resources, and an Energy Aware scheduling policy as shown in the Figure 1 above. Clearly, there is a fundamental similarity to scheduling problems regardless of the difference in the nature of the tasks and the environment.

3. High Performance Computing

In a High Performance Computing (HPC) environment, the objective is to parallelize as much of the program as we can, because of the restrictions placed by Amdahl's Law [10]. Amdahl's law is defined by the formula:

$$\frac{1}{(1 - \quad) + \frac{P}{N}}$$

As $N \rightarrow \infty$, the maximum speedup tends to $1 \ \ (1 - \quad)$. In practice, performance/price falls rapidly as N is increased once there is even a small component of $(1 - P)$ [10 – 13].

A great part of the craft of parallel programming consists of attempting to reduce (1 – P) to the smallest possible value. The figure 2 shows the speedup curves for various values of P.

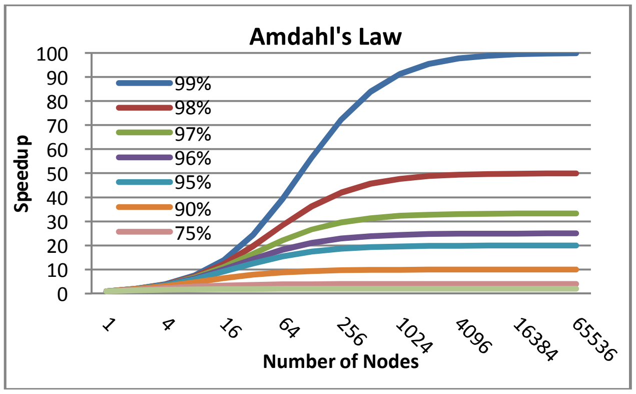

Figure 2: Amdahl's Law

For our experiments we will be using the HPC environments available at UNO (University of Nebraska at Omaha). We initially start out with the Blackforest cluster (16 nodes) [17] and then move to the Firefly cluster, a true commercial strength HPC at Holland Computing Center. The Firefly is a 1,151-node super-computer cluster of Dell SC1435 servers. Each node contains two sockets, and each socket holds a quad-core (four 64-bit AMD Opteron 2.2 GHz processors) [18].

4. Overview of Previous Work

Bioinformatics includes methodologies for processing information characterized by large volume, in order to speedup researches in molecular biology. Sequence analysis, genome sequence comparison, protein structure prediction, pathway research, sequence alignment, phylogeny tree construction, etc. are some of the common operations performed on such biological data [19]. However, bioinformatics applications typically are distributed in different individual projects and they require high performance computational environments.

Most of the previous work done focuses on performance curves that are inherent when one moves a parallelizable application from a single desktop to a HPC cluster environment. Earlier work in parallel sequence search mostly adopts the *query segmentation* method [20, 21], which partitions the sequence query set. This is relatively easy to implement and allows the BLAST search to proceed independently on different processors. However, as databases are growing larger rapidly, this approach will incur higher I/O costs and have limited scalability. Other work follows the more recent trend of pursuing *database segmentation* [22], where databases are partitioned across processors. This approach better utilizes the aggregate memory space and can easily keep up with the growing database sizes. Our approach and experiments uses both these approaches and tries to find which approach is suitable under what circumstances. We use database segmentation approach in the experiment with All query sequences per chromosome, a query merge approach with the experiment of merged query sequences per chromosome (Note here that the query segmentation approach would not have been because BLAT is optimized for running large number of query sequences which are loaded in memory), and finally a combination of the query & database segmentation approach with the experiment of all query files against all chromosome files.

Unlike BLAST, which has been around for a while, the BLAT program which is an alignment tool like BLAST, but it is structured differently is fairly new and there are not a lot of studies on the performance of BLAT in a High Performance Computing environment. We feel this is warranted because BLAT is starting to be more widely used [1 – 2]. Along with this we would like to consider energy utilized as an optimizing criterion and understand its relationship with performance and come up with an energy aware scheduling algorithm that balances the both energy utilized and performance for tasks run in a HPC environment.

5. Proposed Solution

Our main motivation is to move this from a simple speedup to the realm of energy awareness. Now when we speak of energy awareness, a new constraint is placed on the scheduling system. It now has to adopt a scheduling policy which is both traditional performance focused and energy aware. The goal is to find the right harmony between these two, slightly divergent goals. One is focused simply on getting the results as quickly as we can whereas the other is focused on minimizing the energy used in getting the results, which inherently means slowing down if necessary. The crucial question which follows is how one achieves the right balance between these two differing optimization criteria.

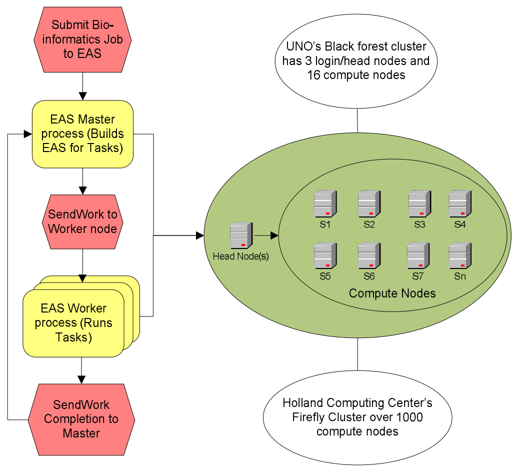
Submit Bio-informatics Job to EAS
EAS Master process (Builds EAS for Tasks)
SendWork to Worker node
EAS Worker process (Runs Tasks)
SendWork Completion to Master
UNO's Black forest cluster has 3 login/head nodes and 16 compute nodes
Head Node(s)
S1
S2
S3
S4
S5
S6
S7
Sn
Compute Nodes
Holland Computing Center's Firefly Cluster over 1000 compute nodes

Figure 3: Process Flow Diagram for EAS Program

This research highlights the need to carefully develop a parallel model with energy awareness in mind, based on our understanding of the application and then appropriately designing a parallel model that works well for the specific application and potentially similar applications within that domain. The figure 3 above describes the general program flow for our implementation of the Energy Aware Scheduler on the HPC cluster (blackforest and firefly). The easblat program is written in C++ and uses MPI (Message Passing Interface) to handle communication between multiple nodes in the cluster [14 – 16]. In general the program consists of a Master and Several worker processes. The program first initializes the MPI environment and then the process with rank=0 is designated as the master process and the rest are designated as worker processes

The Master process builds the work queue and handles all scheduling of work tasks to the respective worker processes. It goes through the work queue and makes scheduling decisions based on performance and energy criteria. Once all the work has been distributed, it then waits and gathers information back from the worker processes. After each worker process replies back the master process sends a terminate message to each worker process/node. The Worker processes simply wait for work from the master process, execute the work given and wait for more work or notification from master to terminate.

6. Implementation and Results

A key contribution of this paper is the importance of data design. We hypothesize that this data design will improve the degree of parallelism, by modifying the why data is structured to maximize the usage of parallelism. In order to support this we design the following experiments.

1) All query sequences per chromosome
2) Merged query sequences per chromosomes, and
3) All query files against all chromosome files.

Our goal is to make energy awareness and scheduling decisions so as to run the BLAT program against given query sequences for a given genome/chromosome file. In most cases researchers today are running this on local desktops and each sequence search is run sequentially and the entire result set may take several hours to days depending on the number of search sequences. Our intention is to first bring some amount of parallelism to this process and then a degree of energy awareness to the scheduling aspects to such tasks from various researchers. With that in mind we had to parallelize the process. Hence we decided to run the following experiments which afforded varying degrees of parallelism and compare them.

The human chromosome files used for these experiments were downloaded from the UCSC Genome bio-informatics website [1]. We used build 36.1 finished human genome assembly (hg18, Mar. 2006). The chromosomal sequences were assembled by the International Human Genome Project sequencing centers. We used the ChromFa.zip file which is the latest dataset as of Dec 2008 [1 – 2]. We used MPI (GNU) to parallelize the runs on multiple nodes, which was a configurable parameter. Our experiments used sequences gathered from researchers at UNMC (University of Nebraska Medical Center) and parallelize the runs to study the performance characteristics under three different conditions. For our tests we used 24 query sequences from a researcher at UNMC. The table below (Table 1) shows some characteristics of these sequences.

Table 1: Query sequences used for analysis

QUERY FILES	.fa size (kb)	.2bit size (kb)	# of lines	# of seqs
MCL_chr1.txt	3311705	1089176	14186	7093
MCL_chr2.txt	2378142	785204	10254	5127
MCL_chr3.txt	1772666	584699	7640	3820
MCL_chr4.txt	1432124	466415	5970	2985
MCL_chr5.txt	1722396	546919	36481	3541
MCL_chr6.txt	1771709	582893	7520	3760
MCL_chr7.txt	1863885	614151	8108	4054
MCL_chr8.txt	1492613	493893	6458	3229
MCL_chr9.txt	1700540	564950	7404	3702
MCL_chr10.txt	1486654	492908	6438	3219
MCL_chr11.txt	2299625	759437	9970	4985
MCL_chr12.txt	1849123	609289	7854	3927
MCL_chr13.txt	703781	231659	2962	1481
MCL_chr14.txt	1302834	430629	5598	2799
MCL_chr15.txt	1024197	338618	4448	2224
MCL_chr16.txt	2320925	763311	10058	5029
MCL_chr17.txt	2863504	943539	12372	6186
MCL_chr18.txt	530863	176476	2376	1188
MCL_chr19.txt	3584718	1193013	15994	7997
MCL_chr20.txt	1297151	430415	5752	2876
MCL_chr21.txt	736972	243709	3202	1601
MCL_chr22.txt	1236062	410443	5464	2732
MCL_chrX.txt	1293959	423823	5438	2719
MCL_chrY.txt	53658	17006	200	100
Total	40029806	13192575	202147	86374

Each query file was a FASTA format text file of sequences with varying number of sequences in each file. Note that the number of nodes 25 comes from the fact that in the human genome we have Chromosome 1 to Chromosome 22 and we have Chromosome X, Chromosome Y and Mitochondrial DNA material. We run the merged query experiment to study the benefits of merging the query files because BLAT is optimized to run large number of sequences in memory.

Firefly Cluster: The firefly cluster is a large commercial strength cluster at the Holland Computing Center which comprises of 1,151-node supercomputer cluster of Dell SC1435 servers. Each node contains two sockets, and each socket holds a quad-core (four 64-bit AMD Opteron 2.2 GHz processors). The computational network utilizes an 800 MB/sec Infiniband interconnect. Each node has its own 8 GB of memory, and 73 GB of disk space [18].

The experiments below were conducted on the Holland Computing Center's firefly cluster.

Experiment 1: The chart below shows the execution time of all query files per chromosome by nodes. When node = 1 it would be the same as running it sequentially on a local desktop. In this case when node is 1 we get a total execution time of 6:16 (hh:mm). When number of nodes = 25 we get a total execution time of 0:28, which is a speedup of 13. Note however that when we vary nodes from 20 – 25, we do not see any additional gains, this is because we have already used the inherent slack in the schedule and there are no additional gains to be made by increasing the number of processors.

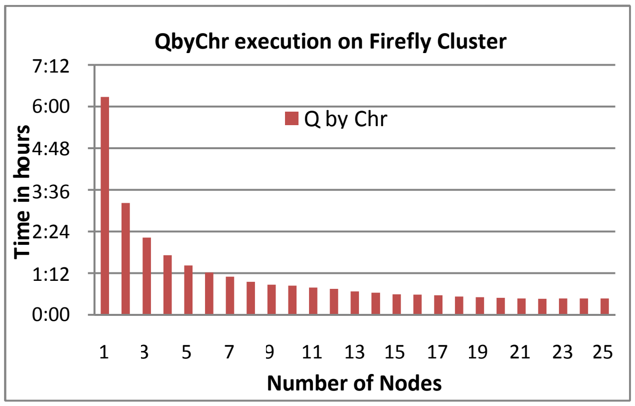

Figure 4: QbyChr execution on Firefly Cluster

Experiment 2: The chart below shows the execution time of a single merged query file per chromosome by nodes. The merged file contains all the query sequences from the submitted files. When node = 1 it would be the same as running it sequentially on a local desktop. In this case when node is 1 we get a total execution time of 4:45 (hh:mm). When nodes = 25 we get a total execution time of 0:22, which is a speedup of 12. Note however that when we vary nodes from 20 – 25, we do not see any additional gains; this is because we have already used the inherent slack in the scheduling and there can be no gains made by increasing the number of processors.

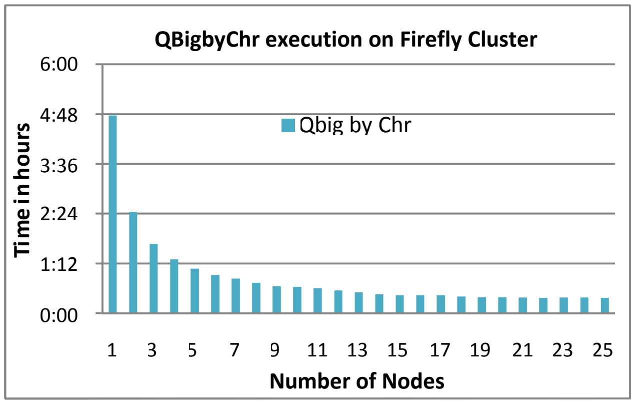

Figure 5: QBigbyChr execution on Firefly Cluster

We also see a certain amount of speedup when we merge query files. This is because BLAT is optimized to handle large number of sequences and we do not have the additional overhead of opening, reading and closing multiple files as all the sequences are loaded upfront into memory since they are in a single file. The average speed up achieved by merging is 1.31 and varies between 1.24 and 1.39 depending on number of processors used.

Experiment 3: The chart below shows the execution time of all query files v/s all chromosome files by nodes. When node = 1 it would be the same as running it sequentially on a local desktop. In this case when node is 1 we get a total execution time of 6:20 (hh:mm). When nodes = 25 we get a total execution time of 0:16, which shows a speedup of 22 compared to the query execution by chromosome method. With nodes = 150 we see an execution time of 0:04 which is a speedup of 86. If we had 1176 processors (24 query files times 49 chromosome files) we would have seen this go down to the max execution for one combination of query file and chromosome file out of the 1176 combinations this is the best we can hope to achieve. Now this can vary depending on the capability of the hardware used.

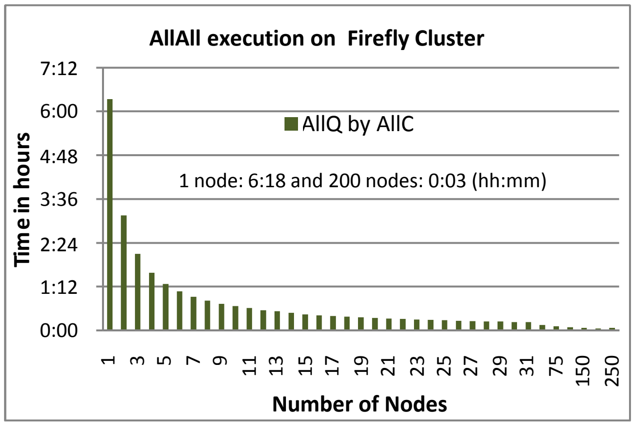

Figure 6: AllAll execution on Firefly Cluster

Comparisons:

The chart below shows a comparison of all the 3 experiments by nodes. When node = 1 it would be the same as running it sequentially on a local desktop. In this case when node is 1 we see that the merged query approach is better than the other two approaches.

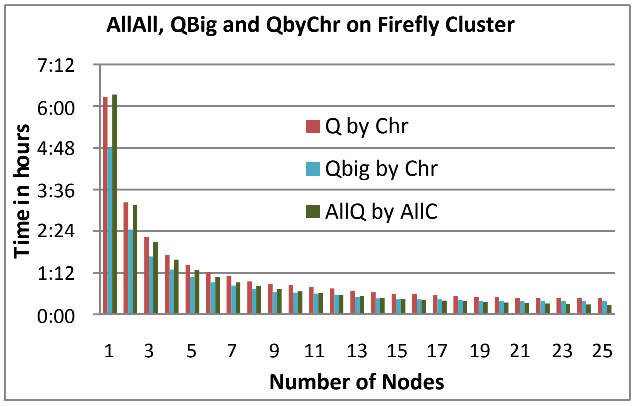

Figure 7: AllAll, QBig & QbyChr on Firefly Cluster

We also note that this true when nodes 1 – 5. After five nodes we see that the "All Query All Chromosome" approach gives us better results. With nodes equal to 25 – 30, we will get twice the speedup with the "All Query All chromosome" approach. One can also note that the Merged Query approach always performs better that the Query by Chromosome approach.

A closer look at the above charts with a focus on nodes from 1 – 10 (Figure 11) and 11 – 25 (Figure 12) is presented below.

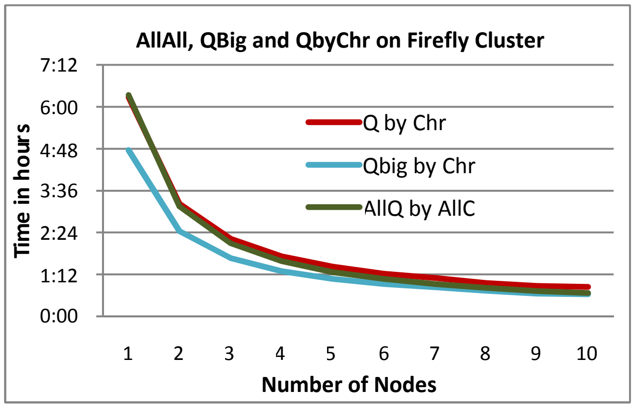

Figure 8: Details on Nodes 1 - 10

The charts suggest that the Merged Query approach and the All Query All Chromosome approach consistently perform better than the Query by Chromosome approach. For nodes 1 – 5, we see that the Merged query approach is better, for nodes 6 – 10 the Merged Query and All Query All Chromosome approach have similar performance and for nodes 11 and beyond the All Query All Chromosome approach out performs the other two approaches.

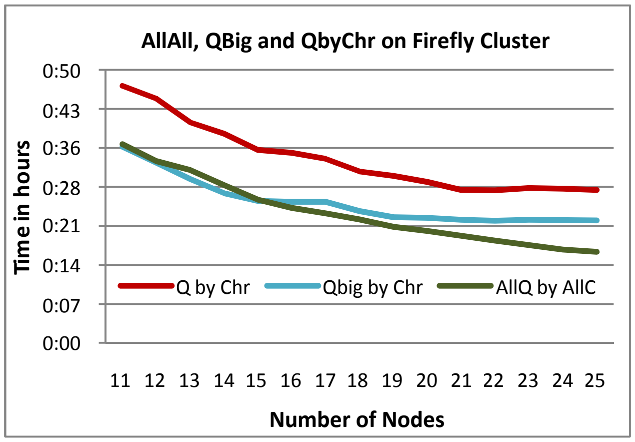

Figure 9: Details on Nodes 11 - 25

Let us try and answer the question how parallelizable is the program? In-order to answer this question we try and plot the speedup for each node and place these by the curves in figure 2. From the figure below we can conclude that the QBigbyChr and QbyChr have a speedup of around 25 times (97% parallelizable) and the AllAll approach has close to 100 times the speedup (99% parallelizable).

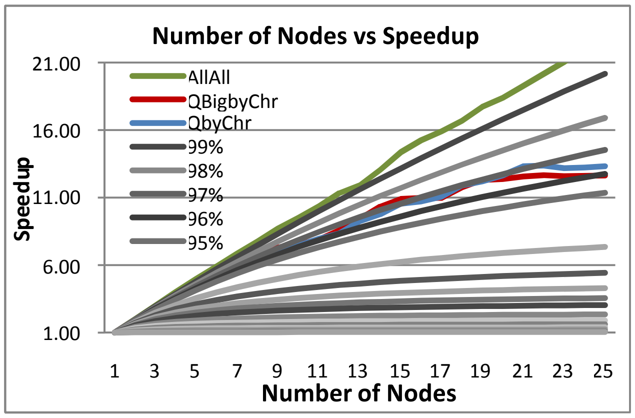

Figure 10: Number of Nodes vs Speedup

7. Scheduling – Energy & Deadline aware

In this section we bring together our understanding of scheduling (section 2), High Performance Computing (section 3) and our specific knowledge about BLAT in HPC (sections 4 – 6). Using our understanding of the speedup profile for the BLAT application, we develop a simple machine learning energy aware scheduling algorithm that takes into account the run profile (figure 6), the number of sequences that were processed, the number of nodes that were used for processing and the time it took to execute. Now when new BLAT queries are submitted along with their desired deadline, the algorithm uses information on the number of sequences that need to be processed, to allocate the least number of nodes needed to meet that deadline, thus managing performance as well as energy to finish the tasks. We used 4 groups of query files each group had 5 files with varying number of sequences as shown in the table below (Table 2).

Table 2: Query groups used for analysis

Groups	Query Files	Total # of Sequences
G1	5	22566
G2	10	40530
G3	15	55946
G4	20	79222

Each group of query sequence files was run against 5 different deadlines (15, 30, 45, 60, and 75 minutes). In each instance we found (Figure 11 below) that the actual execution time (AET) met the given deadline based on the minimum number of nodes assigned for each task group, thus optimizing both performance and energy considerations.

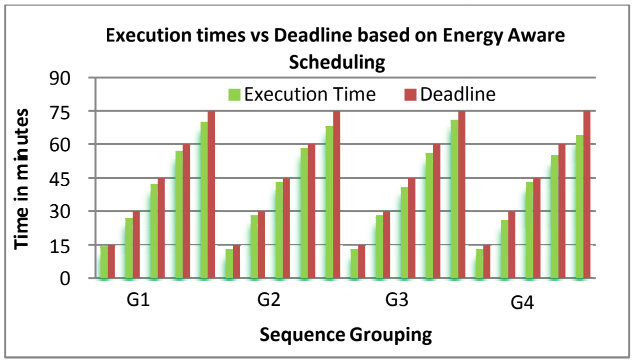

Figure 11: Scheduling – Energy & Deadline aware

The table below (Table 3) shows the actual execution time (AET) being met in each instance given deadlines (15, 30, 45, 60, and 75 minutes). It also shows the number of nodes scheduled to perform the task. One can see that as the deadline increases, we have more time to perform the task enabling us to schedule the least number of nodes (hence managing energy) while still meeting the deadline.

8. Conclusions

In this paper we proposed a HPC based approach to BLAT, implemented the approach and ran multiple experiments for different datasets. We found that the BLAT program is highly parallelizable and has a speedup of 99%. The experiments suggests that the merged query approach and the hybrid approach of all query segmentation and database segmentation consistently performs better that just the database segmentation approach.

Table 3: Least nodes scheduled to meet deadline

Groups	AET (min)	Deadline (min)	Nodes Used
G1	14	15	10
	27	30	6
	42	45	5
	57	60	4
	70	75	3
G2	13	15	15
	28	30	9
	43	45	7
	58	60	5
	68	75	4
G3	13	15	22
	28	30	12
	41	45	8
	56	60	6
	71	75	5
G4	13	15	30
	26	30	15
	43	45	9
	55	60	7
	64	75	6

We also find that we one has only about 5 nodes it is better to use the merged query approach, for number of nodes 6 – 10, we would be better off using the merged query approach, and then beyond 10 nodes we do see a whole lot of performance gains, but this is also the space in which we can do more research to find the right balance between performance and energy utilized by scheduling the BLAT jobs such that they run in a reasonable time yet utilize minimum energy and resources.

This research highlights the need to carefully develop a parallel model with energy awareness in mind, based on our understanding of the data and application. This will help us in designing a parallel model that works well for the specific application and potentially similar applications within that domain. Many of the bioinformatics application follow a similar structure/pattern, where we have a set of input query sequences, which go against an existing set of database genome sequences (such as DNA/RNA/Protein) and output results in a specified output file(s) or directory. These programs also take optional parameters which are used as tuning options for the program itself such as MinScore.

Our future research will focus on moving away from a simple heuristic and explore the use of additional AI techniques such as machine learning algorithms to enhance the modeling, which would allow for a more automated way of dealing with energy utilization and performance of the HPC environment.

References

[1] UCSC Genome Bioinformatics – Sequence and Annotations downloads. Retrieved Dec 2008 from http://hgdownload.cse.ucsc.edu/downloads.html

[2] UCSC Genome Bioinformatics. Retrieved Oct 2008 from http://genome.ucsc.edu/index.html

[3] Kent Informatics – Genome BLAT program. Retrieved Oct 2008 from http://www.kentinformatics.com

[4] J. Ullman, NP-complete scheduling problems, Journal of Computer and System Sciences, 10, 384-393, 1975.S

[5] E. G. Coffman, R. L. Graham, J. L. Bruno, W. H. Kohler, R. Sethi, K. Steiglitz, and J. D. Ullman: Computer and Job-Shop Scheduling Theory, John Wiley & Sons, A Wiley-Inter-Science publication, 1976.

[6] Hesham El-Rewini, Theodore G. Lewis, Hesham H. Ali: Task Scheduling in Parallel and Distributed Systems, PTR Prentice Hall, Inc. Englewood Cliffs, New Jersey 07632. 1994.

[7] Peter Aronsson and Peter Fritzson: Task Merging and Replication using Graph Rewriting, Tenth International Workshop on Compilers for Parallel Computers, Amsterdam, the Netherlands, Jan 8-10, 2003

[8] A. A. Khan, C. L. McCreary and M. S. Jones, A Comparison of Multiprocessor Scheduling Heuristics, International Conference on Parallel Processing, 1994.

[9] Rong Xie, Daniela Rus and Cliff Stein: Scheduling Multi-Task Agents. In Proceedings of the Fifth IEEE International Conference on Mobile Agents, pages 260-276, Atlanta, Georgia, December, 2001.

[10] G. M. Amdahl, “Validity of the single processor approach to achieving large scale computing capabilities” Proc. Am. Federation of Information Processing Societies Conf., AFIPS Press, 1967, pp 483 – 485.

[11] M. D. Hill and M. R. Marty, “Amdahl’s Law in the Multi-core Era”, IEEE Computer, Vol. 41, pages 33 – 38, July 2008.

[12] Wikipedia – Amdahl’s Law: Retrieved Jan 2009 from http://en.wikipedia.org/wiki/Amdahl's_law

[13] S. Cho and R. M. Melhem, Corollaries to Amdahl’s Law of Energy, IEEE Computer Architecture Letters (CAL), 7(1):25 – 28, Jan 2008.

[14] MPICH – A Portable Implementation of MPI. Retrieved Nov 2009, http://www-unix.mcs.anl.gov/mpi

[15] William Gropp, Ewing Lusk and Anthony Skjellum: Using MPI: Portable Parallel Programming with the Message Passing Interface, MIT Press, 55 Hayward St. Cambridge, MA 02142, Oct 1994.

[16] William Gropp, Ewing Lusk and Rajeev Thakur: Using MPI-2: Advanced Features of the Message Passing Interface, MIT Press, 55 Hayward St. Cambridge, MA 02142, Nov 1999.

[17] Blackforest Computing Cluster. Retrieved Oct 2008 from http://blackforest.gds.unomaha.edu/about.php

[18] Holland Computing Center. Retrieved Nov 2008 from http://www.hollandhpc.com/index.shtml

[19] Michel Dayde. et. al: High Performance Computing for Computational Science – VECPAR 2006, Springer-Verlag, Berlin Heidelberg, Germany 2007.

[20] R. Braun, K. Pedretti, T. Casavant, T. Scheetz, C. Birkett, and C. Roberts. Parallelization of local BLAST service on workstation clusters. Future Generation Computer Systems, 17(6), 2001.

[21] E. Chi, E. Shoop, J. Carlis, E. Retzel, and J. Riedl. Efficiency of shared-memory multiprocessors for a genetic sequence similarity search algorithm, 1997.

[22] R. Bjornson, A. Sherman, S. Weston, N. Willard, andJ. Wing. TurboBLAST(r): A parallel implementationof BLAST built on the TurboHub. In Proceedings ofthe International Parallel and Distributed ProcessingSymposium, 2002.

[23] A. Darling, L. Carey, and W. Feng. The design, implementation, and evaluation of mpiBLAST. In Proceedingsof the ClusterWorld Conference and Expo, iconjunction with the 4th International Conference onLinux Clusters: The HPC Revolution, 2003.

[24] D. Mathog. Parallel BLAST on split databases. Bioinformatics, 19(14), 2003.