7-2010

A Dynamic Energy-Aware Model for Scheduling Computationally Intensive Bioinformatics Applications

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

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

A Dynamic Energy-Aware Model for Scheduling Computationally Intensive Bioinformatics Applications

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 (HPC) resources are housed in large datacenters, which consume huge amounts of energy and are quickly demanding attention from businesses as they result in high operating costs. On the other hand HPC environments have been very useful to researchers in many emerging areas in life sciences such as Bioinformatics and Medical Informatics. In this paper, we provide a dynamic model for energy aware scheduling (EAS) in a HPC environment; we use a widely used bioinformatics tool named BLAT (BLAST-like alignment tool) running in a HPC environment as our case study. Our proposed EAS model incorporates 2-Phases: an Offline phase and an Online one. In the Offline Phase, we use sequences gathered from researchers and parallelize the runs to understand the run (speedup) profile of the program. The EAS Engine then utilizes such information to generate the initial schedule. In the Online Phase a feedback mechanism is incorporated between the EAS Engine and the master scheduling process. As scheduled tasks are completed, their actual execution time (AET) is used to adjust the resources required for scheduling remaining tasks using the least number of nodes while meeting a given deadline. The conducted experiments show that the proposed approach succeeded in meeting preset deadlines while minimizing the number of nodes; thus reducing overall energy utilized.

KEYWORDS: High Performance Computing, Energy Awareness, Scheduling, Bioinformatics, Algorithms, Parallel Processing

1. INTRODUCTION.

The Bioinformatics domain is rich in applications that require extracting useful information from very large and continuously growing sequence of databases. 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 in various biomedical applications and stored in massive databases. Most methods used for analyzing DNA/Protein sequences are known to be computationally intensive, providing motivation for the use of powerful computational 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. Energy aware scheduling (EAS) which has an understanding of the application domain in a HPC environment can be a game changer in terms of controlling energy costs at datacenters which house these HPC systems. 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) model 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 BLAT does not impact the conclusions that bioinformatics researchers may draw from the output of BLAT.

2. ENERGY AS A KEY DRIVER

US Data centers consumed 5 MKW of energy in 2005 [26], which is equivalent to five 1000 MW power plants. The total energy utility bills in the US alone amount to $2.7 billion annually and world consumption is estimated to cost $7.2 billion annually [27, 28]. Major California companies are being forced to relocate due to high energy costs, e.g. Google has opened a new datacenter in the Midwest in Council Bluffs [29] and despite economic slump; Yahoo plans a new datacenter in La Vista, Nebraska [30]. Clearly "Energy" is becoming a key business driver. Given these facts it has become imperative for us to consider the efficient usage of energy is all aspects of data center management. In this paper we will also focus on studying energy aware scheduling mechanism in a high performance computing environment such as a grid cluster. We will use applications in the bio-informatics domain which will be scheduled on the Holland Computing Center (HCC) grid. This study will come up with an Energy Aware Scheduling layer for HPC such as clusters and grids (Figure 1) and make intelligent scheduling decisions which will balance energy minimization requirements against performance based upon user needs.

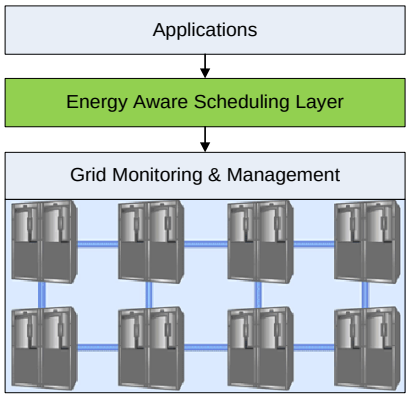

Figure 1. Energy Aware Scheduling Layer for HPC

3. STAGES IN ENERGY EFFICIENCY

The hardware and software industries have realized that in-order to truly address the energy-efficiency question; it has to be tackled at various levels across multiple industries. The first step in this direction is the identification of the variables within the various design, manufacturing, and use of computing and communications devices, operating systems and applications that influence the energy equation. The main goal is to maximize energy efficiency while simultaneously maintaining or increasing performance. This can be achieved by a combination of improvements in micro-architecture, silicon process technology, software at the operating systems level and application level, and platform technologies. The Figure 2 below illustrates this approach.

Hardware	Silicon Process Technology
	Chip Technology
	Power Management
Software	Operating System
	Applications

Figure 2. Different Stages in Accomplishing Energy-Efficiency Objectives

Obviously, processor power is an important consideration in the energy equation, but processors are hardly the only component drawing power. Total energy consumption, for example, is also dependent on memory DIMMs, chipsets, fans, hard disk drives, peripherals, power supply efficiency, and other components. Working with each one of these components can significantly reduce overall energy consumption. For instance, Intel's use of DDR2 memory improves performance up to 11 percent with a 30 percent reduction in memory power consumption. Combining Intel processors with Intel chipsets featuring integrated graphics saves the need for a separate, power-consuming graphics card [31].

Table 1. Variables Influencing Energy-Efficiency

Hardware	Silicon Process Technology	• Second generation strained silicon • Improved interconnects
	Chip Technology	• Dynamic sleep transistor • Demand based switching • On-die voltage regulation • Multi-core and clustered micro-architecture • Power Gating, Macro Fusion.
	Power Management	• Voltage Regulation Technology • Improved display power specs • Thermal design for advanced heat-sync technology
Software	Operating System	• Developing power conscious device drivers. • Tuning OS for less interference with a processor's low-power states. • Energy Aware Scheduling of Applications based on benchmarks.
	Applications	• Application code multi-threaded and multi-core ready. • Power monitoring and analysis tools. • Optimizing code for reducing CPU clock cycles. • Energy Aware Scheduling of Applications tasks.

Within the hardware and software industries there is further breakup depending on where the question of energy efficiency is addressed. Furthermore at each level there are multiple complimentary approaches and areas of research which together become part of the solution in reducing energy utilization. Table 1 illustrates the various complementary areas of research being pursued to address the overall energy efficiency question. (Model can be achieved using one or more of these solution approaches).

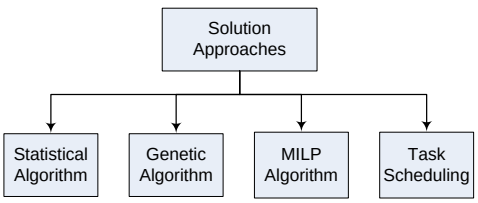

Figure 3. Solution Approaches

There are many solution approaches that can be used to address this problem in the software – application layer as illustrated in Figure 3. Our research focuses on the software – application area and specifically tries to address the question of energy aware scheduling of application tasks. We propose a model for energy aware scheduling and discuss an algorithm proposed for this model.

4. 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, 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. 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 power utilization becomes the most important consideration in a system that is energy aware, at the same time there are certain parameters that must be met such as tasks meeting their deadlines [25].

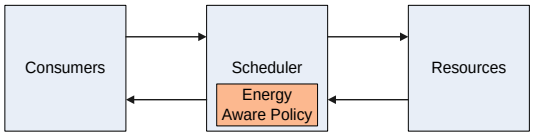

Figure 4. Energy Aware Scheduling System

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 an 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 Figure 4. Clearly, there is a fundamental similarity to scheduling problems regardless of the difference in the nature of the tasks and the environment.

5. 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:

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

As N $\rightarrow \infty$, the maximum speedup tends to $1/(1-P)$. 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 speedup curves for various values of P are shown in Figure 5.

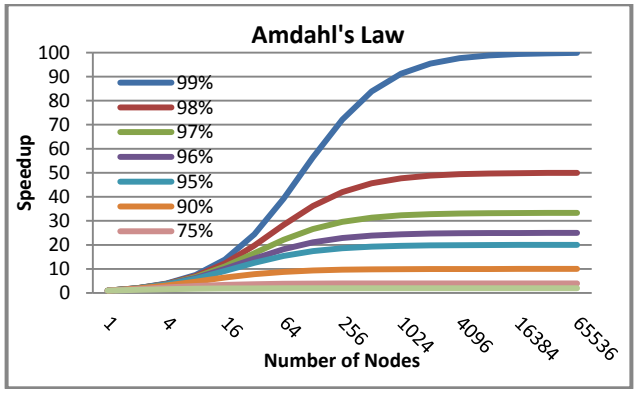

Figure 5. Amdahl's Law

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.

6. HPC FOR BIOINFORMATICS

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. A comparative study of these approaches for BLAT is done in [25]. Our approach and experiments uses a combination of the query & database segmentation approach with the experiment of all query files against all chromosome files. We build on some of the work done in [25] to propose a more generic model to tackle the energy awareness problem. 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]. Of course our main consideration is energy utilized and its minimization in a HPC environment and understanding its relationship with performance. Our goal is to come up with an energy aware scheduling model and algorithm that balances the both energy utilized and performance for tasks run in a HPC environment.

7. PROPOSED SCHEDULING APPROACH

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. We follow a simple 2-step approach.

Step 1: Offline Phase – Build Run Profile, we perform some runs to understand the degree of parallelization (also called run profile) of a program. Based on this we seed our energy aware scheduling (EAS) algorithm in the EAS Engine with the run profile (meaning understanding of the number of nodes required, sequence size and time it takes for the program (BLAT) to run. Using this we can then first allocate a set of nodes for any input sequences based on the number of sequences and given deadline.

Step 2: Online Phase – Dynamic Resource Adjustment Here we dynamically adjust the number of nodes either up or down based upon actual execution time (AET). This then becomes a continuous feedback loop to the EAS Engine, which looks at the tasks expected execution time (EET), its actual execution time and then takes measures to adjust the schedule by adjusting the overall nodes assigned or in future the Dynamic Voltage Scaling (DVS) of each node to meet the overall deadline. This allows us to meet two the two divergent goals of minimizing energy utilization and performance.

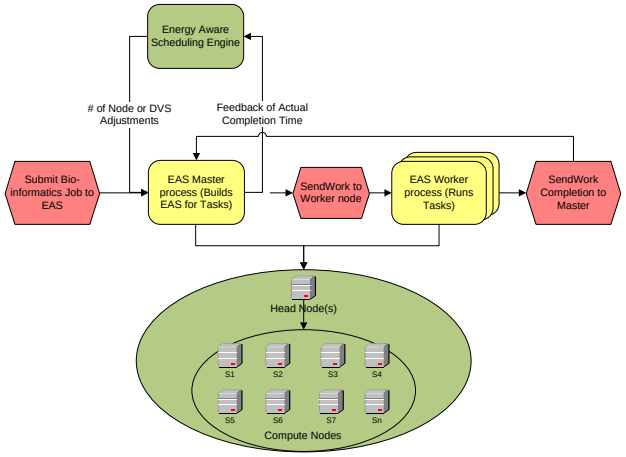

Figure 6. Process Flow Diagram for MPI Program with EAS Engine

This research also 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. Figure 6 describes the general program flow for our implementation of the Energy Aware Scheduling (EAS) Engine on the HPC clusters (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 it calls the Energy Aware Scheduling (EAS) Engine and 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. The EAS Engine takes information about the EET and AET of the task, makes decisions if any node level adjustments need to be made (and/or DVS adjustments) and sends an appropriate feedback message back to the Master process.

7.1. Implementation of Step 1.

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. With that in mind we parallelized the process using the "All query sequences per chromosome" approach used in [25] to understand the degree of parallelism in the BLAT program.

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 different conditions. For our tests we used 24 query

sequences from a researcher at UNMC. The table below (Table 2) shows some characteristics of these sequences.

Table 2. 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.

Experiment: "All query sequences per chromosome" The chart in Figure 7 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.

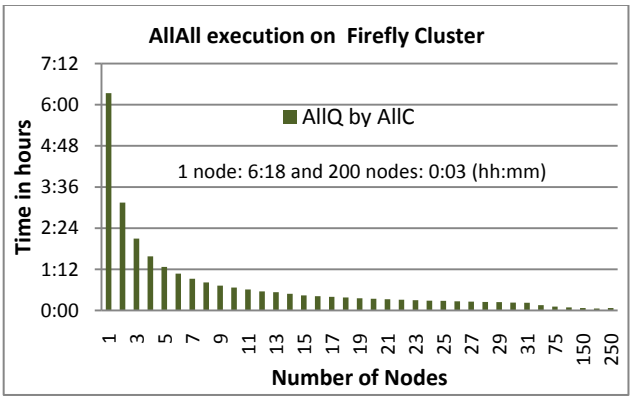

Figure 7. AllAll Execution on Firefly Cluster

7.2. Implementation of Step 2.

In Step 2 of the process, which is the Online Phase of the algorithm we dynamical adjust resource levels. The EAS Engine adjusts the number of nodes either up or down based upon the difference between EET and AET to meet the overall deadline. We maintain a continuous feedback loop between the EAS Engine and the Master process. The energy aware scheduling algorithm within the EAS Engine uses our understanding of the run profile from Step 1 and then adjusts to realities during the actual execution of tasks using information such as the number of sequences that were processed, the number of nodes that were used for processing, the EET and the AET for that task. The information gathered from these new runs is then transformed into knowledge to update the existing run profile allowing the EAS Engine to build a knowledge map that is used for future allocation of HPC resources. Now when new BLAT queries are submitted along with their desired deadline, the algorithm uses this information 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 the same 4 groups of query files as in [25], each group had 5 files with varying number of sequences as shown in the table below (Table 3).

Table 3. 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). Each

of these jobs was assigned a starting number of nodes based on the run profile according to Step 1.

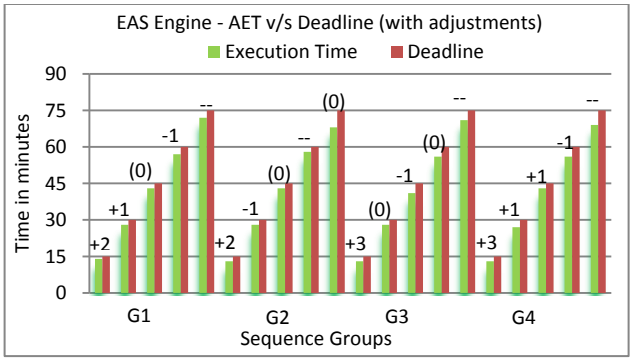

Figure 8. EAS Engine – AET v/s Deadline (Adjustments)

As the tasks were completed, in accordance to Step 2, variances between EET and AET resulted in the EAS engine adjusting the number of nodes up (+N) or down (–N), if there were equal number of (+N) and (–N) adjustments it resulted in a net (0) adjustment and finally the scenario of no adjustments being made (–). In each instance we found (Figure 8) 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.

Table 4. Node Adjustments to meet Deadline

Groups	AET (min)	Deadline (min)	Nodes Assigned	Nodes Used	Adjustments
G1	14	15	8	10	(+2)
	27	30	5	6	(+1)
	42	45	5	5	(0)
	57	60	5	4	(-1)
	70	75	3	3	-
G2	13	15	13	15	(+2)
	28	30	10	9	(-1)
	43	45	7	7	(0)
	58	60	5	5	-
	68	75	4	4	(0)
G3	13	15	20	23	(+3)
	28	30	12	12	(0)
	41	45	9	8	(-1)
	56	60	6	6	(0)
	71	75	5	5	-
G4	13	15	28	31	(+3)
	26	30	14	15	(+1)
	43	45	8	9	(+1)
	55	60	7	6	(-1)
	68	75	6	6	-

Table 4 shows the AET, in each instance meeting the given deadlines. It also shows the starting number of nodes assigned, the final number of nodes scheduled for the tasks and the number of adjustments made by the EAS Engine.

8. CONCLUSIONS

In this paper we proposed an energy aware scheduling model in a HPC environment based on a 2-step approach. The Off-line Phase uses the knowledge of the run-profile of the program based on previous runs and the On-line Phase used a dynamic feedback loop to adjust the resources (# of nodes) to minimize energy utilized while still meeting the deadline. The run-profile and experiments were done for the BLAT program in the bio-informatics domain. We found that the BLAT program is highly parallelizable and has a speedup of 99%. We also found that the EAS Engine was able to dynamically take react to the difference between EET and AET and adjust the number of nodes up or down to balance the minimization of energy and performance criteria for all our experimental datasets. We used a rather conservative approach in our initial allocation of node resources, there are various strategies one could use in the conservative to risk spectrum, but this is also the space in which we can do more research to find the right balance.

Our future research will focus on further automation of the EAS Engine to accommodate other programs in the same domain or similar domains. We would also like to explore the nuances between conservative and risky approaches to the Off-line scheduling of node resources. We believe that eventually OS capabilities will evolve, allowing existing hardware DVS capabilities to be controlled at a program level, thus enabling software programs to have more control and flexibility in handling energy considerations. This will allow programs written with intimate knowledge about a specific domain and an understanding of deadline needs of the user for result sets to scale the application in such a way that resources can be added on-demand, and processor speed controlled (hence controlling energy) to either speedup or slowdown the application to manage the divergent goals of performance and energy. Another key focus of our future research will be to incorporate the ability to incorporate Dynamic Voltage Scaling (DVS) at the node level. This will allow us to add another level of granularity to the EAS algorithm's ability to adjust energy at the node level.

REFERENCES

[1] "UCSC Genome Bioinformatics" UCSC [website], Dec 2008, Available: http://hgdownload.cse.ucsc.edu/downloads.html

[2] J. Kent, "Kent Informatics – Genome BLAT" [website], Oct 2008, Available: http://www.kentinformatics.com

[3] Y. Shin and K. Choi, "Power Conscious Fixed Priority Scheduling for Hard Real-Time Systems", 2005

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

[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, 1976.

[6] H. El-Rewini, T. G. Lewis, H. Ali, TASK SCHEDULING IN PARALLEL AND DISTRIBUTED SYSTEMS, PTR Prentice Hall, Inc. Englewood Cliffs, New Jersey 07632. 1994.

[7] P. Aronsson and P. Fritzson, "Task Merging and Replication using Graph Rewriting", International Workshop on Compilers for Parallel Computers, Amsterdam, 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] R. Xie, D. Rus and C. Stein, "Scheduling Multi-Task Agents" 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] "Amdahl's Law", Wikipedia [website], Jan 2009, Available" 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", MPICH [website], 2009, Available: http://www-unix.mcs.anl.gov/mpi

[15] W. Gropp, E. Lusk and A. Skjellum, "USING MPI: PORTABLE PARALLEL PROGRAMMING WITH THE MESSAGE PASSING INTERFACE", MIT Press, Cambridge, MA, 1994.

[16] W. Gropp, E. Lusk and R. Thakur, "USING MPI-2: ADVANCED FEATURES OF THE MESSAGE PASSING INTERFACE", MIT Press, Cambridge, MA, 1999.

[17] "Blackforest Computing Cluster", UNO [website], Oct 2008, Available: http://blackforest.gds.unomaha.edu/about.php

[18] "Holland Computing Center", [website], Nov 2008, Available: http://www.hollandhpc.com/index.shtml

[19] M. Dayde, E. Pacitti, J. Lopes, "High Performance Computing for Computational Science", VECPAR, Berlin Heidelberg, Germany 2007.

[20] R. Braun, K. Pedretti, T. Casavant, T. Scheetz, C. Birkett, & 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, and J. Wing. "TurboBLAST(r): A parallel implementation of BLAST built on the TurboHub", International Parallel and Distributed Processing Symposium, 2002.

[23] A. Darling, L. Carey, and W. Feng. "The design, implementation, and evaluation of mpiBLAST", ClusterWorld Conference and Expo, in conjunction with the 4th International Conference on Linux Clusters: The HPC Revolution, 2003.

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

[25] S. Pawaskar and H. Ali, "On the Tradeoff between Speedup and Energy Consumption in High Performance Computing" in Proc. Int'l Conf. on Parallel and Distributed Computing and Networks", Feb. 2010.

[26] L. Snyder, "Slash Data Center Energy Cost",[website], 2008 Available: http://www.facilitiesnet.com/bom/article.asp?id=8068

[27] "AMD Report Pegs Global Data Center Energy Costs at $7.2 Billion" AMD [website], Feb 2007. Available: http://www.environmentalleader.com/2007/02/16/amd-report-pegs-global-data-ceneter-energy-costs-at-72-billion/

[28] He, J. "Datacenter Power Management: Power Consumption Trend", [website], Feb 2008, Available: http://communities.intel.com/openport/blogs/server/2008/02

[29] J. Foley, "Google's Iowa Data Center Emerges", Information Week [website], Nov 2008, Available: http://www.informationweek.com

[30] "Despite slump, Yahoo plans new operations in Nebraska" USAToday [website], Nov 2008, Available: http://www.usatoday.com

[31] "Intel is leading the Way in Designing Energy-Efficient Platforms", Intel Oct 2006. Technology@Intel Magazine