



University of Nebraska at Omaha
DigitalCommons@UNO

Computer Science Faculty Proceedings &
Presentations

Department of Computer Science

7-2016

A Dynamic Run-Profile Energy-Aware Approach 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

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, "A Dynamic Run-Profile Energy-Aware Approach for Scheduling Computationally Intensive Bioinformatics Applications" (2016). *Computer Science Faculty Proceedings & Presentations*. 53.
<https://digitalcommons.unomaha.edu/compsicfacproc/53>

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.



A Dynamic Run-Profile Energy-Aware Approach 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
spawaskar@unomaha.edu | hali@unomaha.edu

Abstract— High Performance Computing (HPC) resources are housed in large datacenters, which consume exorbitant 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 an earlier work, we introduced a dynamic model for energy aware scheduling (EAS) in a HPC environment; the model is domain agnostic and incorporates both the deadline parameter as well as energy parameters for computationally intensive applications. Our proposed EAS model incorporates 2-phases. In the Offline Phase, we use a run profile based approach 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, actual execution times are used to adjust the resources required for scheduling remaining tasks using the least number of nodes while meeting a given deadline. In this paper we study the impact of the quality of initial schedule using different run profiles which is critical to the overall energy optimization as every adjustment made has an overhead. The conducted experiments show that the proposed approach succeeded in meeting preset deadlines while minimizing the number of nodes; thus reducing overall energy utilized and that choosing the right profile in the Offline phase has an impact on the energy optimization achieved by the EAS algorithm.

Keywords: Energy Awareness, Scheduling, High Performance Computing, Bioinformatics, Algorithms, Parallel Computing, Run Profile

I. INTRODUCTION

Cloud Computing is an exciting new trend which many of us in the IT field are, simply put, a “little cloudy about”. It is a general term used to describe a new class of network based computing that takes place over the Internet. Cloud computing is **Commoditized** (basically a step on from Utility Computing) and can be considered to be a collection/group of integrated and networked hardware, software and Internet infrastructure (called a **platform**), which uses the Internet for communication and transport provides hardware, software and networking services to clients. The cloud allows for **Abstraction**, it hides

the complexity and details of the underlying infrastructure from users and applications by providing very simple graphical interface or API. The cloud is **Ubiquitous**, on demand services that are always on, anywhere, anytime, any-place and finally the cloud is **Elastic**, as it supports Pay for use and as needed, which allows for scale up and down in capacity and functionalities as needed [1, 19].

A. Cloud Computing Models

There are 3 main types of cloud computing models, the Infrastructure as a Service (IaaS), Platform as a Service (PaaS) and Software as a Service (SaaS) models which are described in the Figure 1 below.

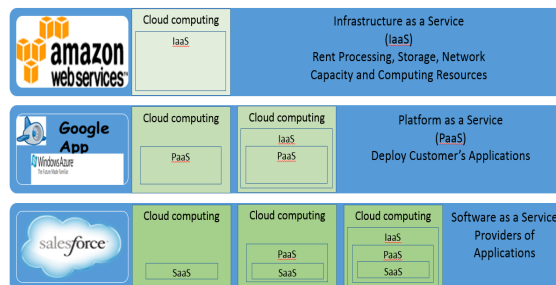


Figure 1: Cloud Computing Models [1, 19]

B. Cloud Service Layers

Another classification for these clouds is based on the type of services layer they provide such as hosting, storage, platform, development, application and services layer Figure 2 [1].

	Services	Description
Application Focused	Services	Complete business services such as PayPal, OpenID, OAuth, Google Maps, Alexa
	Application	Cloud based software that eliminates the need for local installation such as Google Apps, Microsoft Online
	Development	Software development platforms used to build custom cloud based applications (PaaS & SaaS) such as Salesforce
Infrastructure Focused	Platform	Cloud based platforms, typically provided using virtualization, such as Amazon ECC, Sun Grid
	Storage	Data storage or cloud based NAS such as CTERA, iDisk, CloudNAS
	Hosting	Physical data centers such as those run by IBM, HP, Navisite, etc.

Figure 2: Cloud Computing Service Layers

C. HPC, Bioinformatics and the Cloud

More and more companies are starting to realize the importance of making the high performance computing resources available via the cloud. High-performance computing describes a set of hardware and software techniques developed for building computer systems capable of quickly performing large amounts of computation. 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.

Bioinformatics can be broadly defined as set of computing techniques used to manage and extract useful information from the DNA/RNA/protein sequence data. 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. 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. More 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 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 [2 – 3]. HPC has been successfully applied to help reduce the computational burden of large datasets. But, naively parallelizing the applications developed for BLAT could achieve an unnecessary high degree of parallelism at the expense of significant energy consumption.

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-efficient design and operation of such systems critically depends on reduction of the power consumption of processors. There are two kinds of methods to reduce power consumption of processors.

- 1) First is power-down mode, by putting the processor into this mode, only certain parts of the processor such as the clock generation and timer circuits are kept running (idle state). The tradeoff here is between the amount of power saving and the latency incurred during mode change.
- 2) The second is Dynamic Voltage Scaling (DVS), where processor speed is changed by varying the clock frequency along with the supply voltage when the required performance on the processor is lower than the maximum performance [4].

D. What is a Run Profile?

This work focuses on the concept of Run Profiles and assesses its value in the context of Dynamic Energy-Aware Scheduling (EAS). A “Run Profile” for a program provides us with important information about how many nodes/resources were used and how long (time) it took to complete the task, based on certain parameters/characteristics of the data set. In most scientific domains, such as Bioinformatics, the need for High Performance Computing usually follows the 90/10 pattern. Typically, 90% of the tasks involve about 10% of the known software application. These applications can benefit from a better understanding of their Run Profile and the dynamic adjustment to these Profiles. In this paper, we examine multiple initial Run Profiles and the impact of these on the subsequent adjustment made by our EAS Engine. Each profile results in the EAS Engine having to make different number of adjustments to still meet the deadline, which in turn has an impact of the energy utilized. Examining different Run Profiles is critical as application programs in specific domains can be run using different parameters and in turn would influence how long the program takes to run on a specific dataset. The profiles incorporate certain domain-specific parameters such as sequence length and number of sequences which also determine how long the application program (such as BLAT) will run. The Profile would then get adjusted or fine-tuned every time it runs on a given configuration resulting in a more refined Run Profile to be used in future runs. 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.

II. KEY FACTOR – ENERGY

Energy requirements for datacenters has grown massively over the last several years, suggesting that rising energy costs and stricter regulations are not helping to limit datacenter power use and cut carbon emissions. Between 2011 and 2012, power requirements grew by 63% globally to 38GW (gigawatts), up from 24GW in 2011. Energy use in datacenters was at 12GW in 2007 and has been on the rise since. In the four years to 2011, it doubled to 24GW, but in the last year alone it increased to 38GW amid data explosion and business expansion. The census estimated a further rise of 17% to 43GW in 2013 [5]. Major 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 [6] and despite economic slump; Yahoo plans a new datacenter in La Vista, Nebraska [7].

Most data centers, by design, consume vast amounts of energy in an incongruously wasteful manner, interviews and documents show. Online companies typically run their facilities at maximum capacity around the clock, whatever the demand. As a result, data centers can waste 90 percent or more of the electricity they pull off the grid, The Times found [8]. Clearly “Energy” is becoming a key business driver. Given these facts it has become imperative for us to consider the efficient usage of energy in all aspects of data center management.

We believe that there is a place and need for an Energy Aware scheduling layer between Applications and the actual HPC Grid Management layer to control the assignment of number of nodes for an application and manage this dynamically at runtime to manage energy utilization within a datacenter. In this paper we will focus on the impact of the quality of initial schedule using different run profiles which is the starting point for the EAS algorithm on the number of node adjustments which is critical to the overall energy optimization as every adjustment made has an overhead.

III. OVERVIEW OF PREVIOUS WORK

Bioinformatics includes techniques & methods for processing large volumes of information to help speedup research. Some of the common algorithms run on this data are genome sequence comparison, protein structure prediction, sequence alignment, phylogeny tree construction, pathway research, etc. [9].

Most of the previous work done focuses on performance curves that are inherent when one transforms a serial application running on a single desktop to a parallel application running in a HPC environment. Work using BLAST search can be characterized by the data approach used in the search which is of the following 2 main types.

- 1) *Query segmentation method* – partitions the sequence query set. This allows the BLAST search to proceed independently on different processors. However, as databases keep growing, this approach will incur higher I/O costs and has limited scalability [10, 11].
- 2) *Database segmentation* – databases are partitioned across processors which better utilizes the aggregate memory space and can easily keep up with the growing database sizes [12, 13].

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 our previous work [13 – 14] to propose a more generic model to tackle the energy awareness problem. Unlike BLAST, the BLAT program which is an alignment tool like BLAST, is organized uniquely, is relatively new and hence does not have a lot of studies related to its performance in a HPC environment. This is clearly warranted as BLAT is starting to be more widely used [2 – 3]. The main goal of our study is the minimization of energy in a HPC environment and its

relationship with performance. Our main goal is to come up with an energy aware scheduling (EAS) model/algorithm that balances the both energy utilized and performance for such tasks. In [13 – 14, 20] the importance of data design was studied which improved the degree of parallelism, by modifying the way data is structured to maximize the usage of parallelism. The following experiments were designed and compared to study varying degrees of parallelism.

- 1) All query sequences/chromosome (DB segmentation)
- 2) Merged query sequences/chromosome (Query Merge),
- 3) All query files against all chromosome files (Hybrid).

IV. RUN-PROFILE SCHEDULING

Our main goal is to study and examine the behavior of the EAS Model proposed when the online phase of the algorithm is seeded with differing run profiles. The run profiles are based on the 3 experimental approaches namely (1) Database segmentation, (2) Query merge and (3) Hybrid. Obviously each of these run profiles will result in varying schedules during the initial runs, but can the EAS Model adjust appropriately over time and how long (number of runs) does it take for the EAS Model to return comparable results. 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.

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.

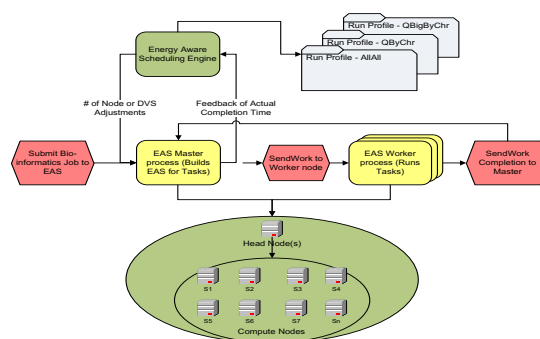


Figure 3: Process Diagram for EAS using Run Profiles

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 feedback loop to the EAS Engine, which looks at the tasks expected execution time (EET), its actual execution time and then takes steps 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 the two divergent goals of minimizing energy utilization and performance.

Figure 3 which is an enhanced version of the EAS Model first described in [13 – 14] shows the program flow for our Run-Profiles based EAS implementation. The EAS Engine assimilates information about the EET based on run-profile and AET of the task from the worker process, and then makes node level and/or DVS adjustments as needed and sends appropriate feedback information back to the Master process.

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 3 describes the general program flow for our implementation of the Energy Aware Scheduling (EAS) Engine on the HPC cluster (firefly). The EAS program is written in C++ and uses MPI (Message Passing Interface) to handle communication between multiple nodes in the cluster [16 – 18]. 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.

A. Implementation of Step 1 – Offline Phase

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 [13, 19] 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

TABLE I. 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

We parallelized the runs to study the performance characteristics under three different conditions. The table (Table 1) shows some characteristics of these query sequences [13 – 14]. We ran the merged query experiment to study the benefits of merging the query files as BLAT is memory optimized to run large number of sequences. 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.

A key question we tried to answer then was “How parallelizable is the program?” In-order to answer this question we charted the performance curve for each experiment type and super impose these by the standard speedup curves based on Amdahl’s Law. The figure 4 below shows 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).

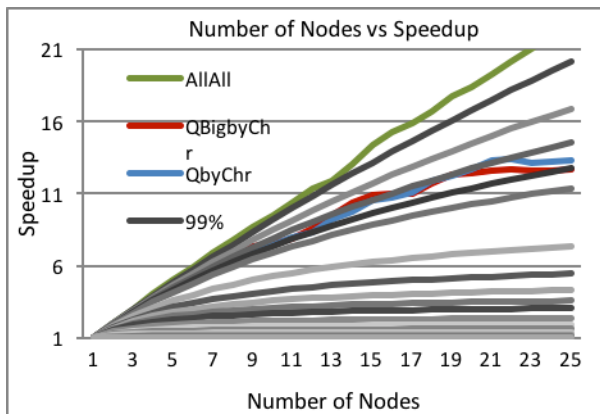


Figure 4: Number of Nodes v/s Speedup based on Amdahl’s Law

Firefly Cluster: We used the firefly cluster at the Holland Computing Center to conduct our experiments. It is a 1,151-node cluster of Dell SC1435 servers. Each node consists of 2 sockets, and each socket holds a 64-bit quad-core AMD Opteron 2.2 GHz processors with 8 GB of memory. An 800 MB/sec Infiniband interconnect forms the computational network [15].

1) Build Run Profile Implementation

We seeded 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. This knowledge was used to generate the initial schedule used to first allocate a set of nodes for any input sequences based on the number of sequences and given deadline.

We used the 3 different approaches discussed above namely (1) All query sequences per chromosome, (2) Merged query sequences per chromosomes, and (3) All query files against all chromosome files. If no run profile is used the initial schedule defaults to WCET (worst case execution time) schedule This will allow us to see if using different run profiles has an impact on the performance of the EAS Engine

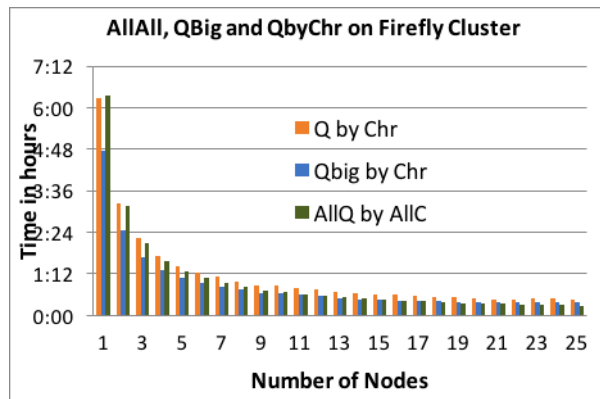


Figure 5: Different Run Profiles on Firefly Cluster

B. Implementation of Step 2 – Online Phase

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. 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 used for future allocation of HPC resources. 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.

TABLE II. QUERY GROUPS USED FOR ANALYSIS

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

We used the same 4 groups of query files as in [13, 19], each group had 5 files with varying number of sequences as shown in the table (Table 2). Each query sequence file group was run against deadlines of 15, 30, 45, 60, and 75 minutes. Each job was assigned an initial number of nodes based on the run profile used in the offline phase. During the online phase, as the tasks were completed, 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 the scenario of no adjustments being made (–).

V. EXPERIMENTAL RESULTS

We used the 3 different approaches discussed above namely (1) All query sequences per chromosome, (2) Merged query sequences per chromosomes, (3) All query files against all chromosome files, and (4) No Initial Profile Adjustment

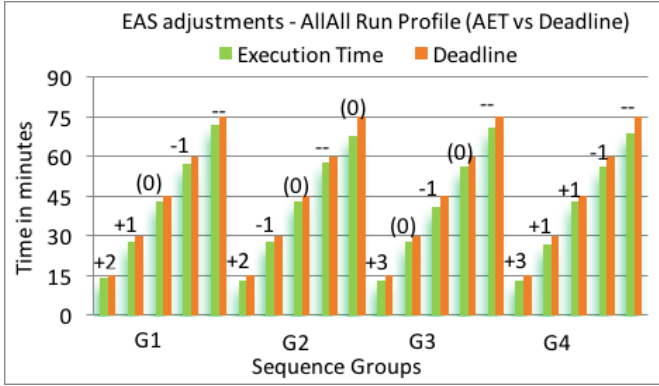


Figure 6: AllAll Run-Profile Adjustments

When the AllAll run profile was used (Figure 6) in all instances we found that the (AET) met the given deadline based on the minimum number of nodes assigned for each task group, thus optimizing both performance and energy considerations.

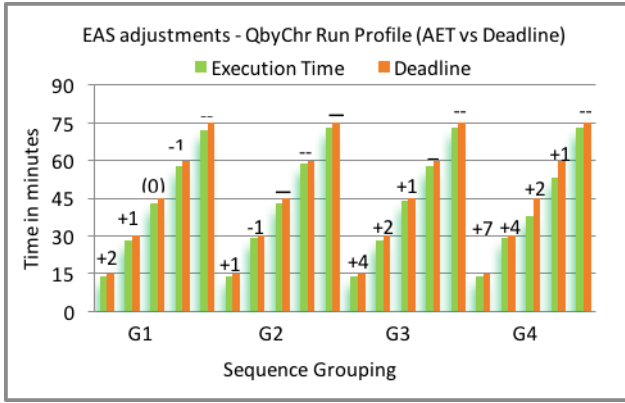


Figure 7: QbyChr Run-Profile Adjustments

The above “QbyChr” run profile graph suggests that for lower deadlines more node adjustments had to be made to meet deadline than what was allocated in the offline phase.

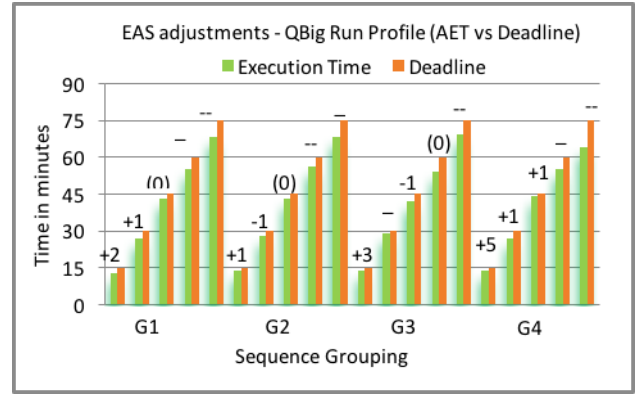


Figure 8: QBig Run-Profile Adjustments

The above “QBigbyChr” run profile graph suggests that for lower deadlines more node adjustments had to be made to meet deadline than what was allocated in the offline phase.

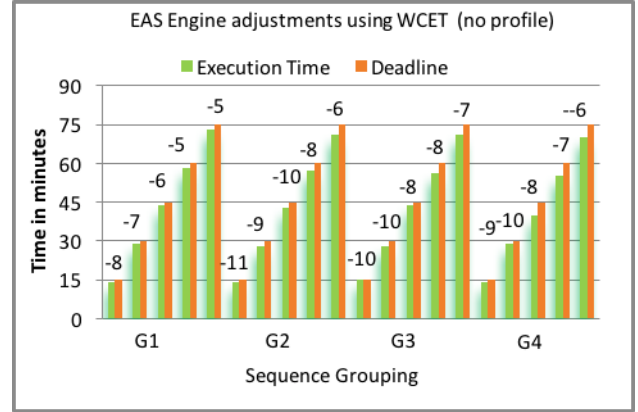


Figure 9: No Profile Adjustments

When no run profile is seeded to the offline phase, the EAS engine defaults to using the WCET schedule. This graph is presented above. The graph shows that using WCET schedule we have significantly more node adjustments compared to using a run profile.

TABLE III. ADJUSTMENTS BASED ON RUN-PROFILE USED

Groups	AllAll Adjustments	QBigbyChr Adjustments	QbyChr Adjustments	WCET (no profile)
G1	(+2)	(+2)	(+2)	(-8)
	(+1)	(+1)	(+1)	(-7)
	(0)	(0)	(0)	(-6)
	(-1)	-	(-1)	(-5)
	-	-	-	(-5)
G2	(+2)	(+1)	(+1)	(-11)
	(-1)	(-1)	(-1)	(-9)
	(0)	(0)	-	(-10)
	-	-	-	(-8)
	(0)	-	-	(-6)
G3	(+3)	(+3)	(+4)	(-10)

	(0)	-	(+2)	(-10)
	(-1)	(-1)	(+1)	(-8)
	(0)	(0)	-	(-8)
	-	-	-	(-7)
G4	(+3)	(+5)	(+7)	(-9)
	(+1)	(+1)	(+4)	(-10)
	(+1)	(+1)	(+2)	(-8)
	(-1)	-	(+1)	(-7)
	-	-	-	(-6)

The table above shows the node adjustments made by the EAS Engine to meet the deadline depending upon which run profile was chosen in offline phase, meaning the run profile used in the initial scheduling of the tasks. It suggests that for large number of sequences and lower deadline thresholds it is better to use the AllAll run profile as the other two run profiles were both unable to meet the lower deadlines (15 min.). For higher deadline and smaller number of sequences, the AllAll and QBigbyChr run profile approaches are mostly comparable. The experiments also show that “QbyChr” run profile approach results in the most node adjustments.

VI. CONCLUSIONS

In this paper we built upon our previous work and enhanced our energy aware scheduling model which is a 2-phase approach using run profiles. 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 bioinformatics domain. We conclude that the BLAT program is highly parallelizable and has a speedup of 99%. We also conclude that the EAS Engine was able to dynamically 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 theorized that using different data modeling mechanisms will result in different run profile curves and the choice of initial schedule has an impact on the number of resource adjustments needed to meet the deadline & overall energy efficiency goals. The conducted experiments clearly show that the proposed dynamic run profile EAS approach succeeded in meeting preset deadlines while minimizing the number of nodes; thus reducing overall energy utilized. While using the AllAll (Hybrid approach) run profile provided the least overall adjustments necessary, followed by the QBigbyChr (Merge approach), and then the QbyChr (Database segmentation approach). A close look shows that the higher the degree of parallelism of the approach the better the EAS Engine performs in terms of node adjustments and overall efficiency. It is also important to note that not having a run profile for the initial schedule and simply using the WCET results in the most node adjustments and hence least energy savings. Clearly

using a known run profile for a given application with the EAS Engine will produce better results.

The focus of our future research will be on automating the enhanced Run-Profile based EAS Engine to accommodate other programs in the same domain or similar domains. We believe that eventually hardware and software OS capabilities will evolve, allowing existing hardware DVS capabilities to be controlled at a program level, thus allowing application programs to have more control and flexibility in handling energy considerations. This will allow software written with intimate knowledge about a specific domain. Understanding of deadline needs of the user will allow us to scale the application in such a way that resources can be added on-demand, and processor speed controlled to either speedup or slowdown the application to manage the divergent goals of performance and energy. We will also examine the usefulness of using simulations using the concept of run profiles for cloud computing to predict deadline completions for jobs, etc. without actually having to use cloud computing resources to run these jobs.

REFERENCES

- [1] Amazon Elastic Compute Cloud (Amazon EC2). (2013). Retrieved from Amazon: <http://aws.amazon.com/ec2/>
- [2] “UCSC Genome Bioinformatics” UCSC [website], Dec 2008, at: <http://hgdownload.cse.ucsc.edu/downloads.html>
- [3] J. Kent, “Kent Informatics – Genome BLAT” [website], Oct 2008, Available: <http://www.kentinformatics.com>
- [4] Y. Shin and K. Choi, “Power Conscious Fixed Priority Scheduling for Hard Real-Time Systems”, 2005
- [5] A. Ventakraman, “Global census shows datacenter power demand grew 63% in 2012”, Datacenter Efficiency, Computer Weekly [web], Oct 2013, <http://www.computerweekly.com/>
- [6] J. Foley, “Google’s Iowa Data Center Emerges”, Information Week [website], Nov 2008, Available: <http://www.informationweek.com>
- [7] “Despite slump, Yahoo plans new operations in Nebraska” USA Today [website], Nov 2008, Available: <http://www.usatoday.com>
- [8] J. Glanz, “Power, Pollution and the Internet”, The Cloud Factories, NYTimes [website], Oct 2013, <http://www.nytimes.com/2012/09/23/technology/data-centers-waste-vast-amounts-of-energy-belying-industry-image.html>
- [9] M. Dayde, E. Pacitti, J. Lopes, “High Performance Computing for Computational Science”, VECPAR, Berlin Heidelberg, Germany 2007.
- [10] 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.
- [11] E. Chi, E. Shoop, J. Carlis, E. Retzel, and J. Riedl. “Efficiency of shared-memory multiprocessors for a genetic sequence similarity search algorithm”, 1997.
- [12] 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.
- [13] 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.
- [14] S. Pawaskar and H. Ali, “A Dynamic Energy-Aware Model for Scheduling Computationally Intensive Bioinformatics Applications” in Proc. Int’l Conf. on High Performance Computing and Simulation, Jun. 2010.

- [15] "Holland Computing Center", [website], Nov 2008, Available: <http://www.hollandhpc.com/index.shtml>
- [16] "MPICH – A Portable Implementation of MPI", MPICH [website], 2009, Available: <http://www-unix.mcs.anl.gov/mpi>
- [17] W. Gropp, E. Lusk and A. Skjellum, "USING MPI: PORTABLE PARALLEL PROGRAMMING WITH THE MESSAGE PASSING INTERFACE", MIT Press, Cambridge, MA, 1994.
- [18] W. Gropp, E. Lusk and R. Thakur, "USING MPI-2: ADVANCED FEATURES OF THE MESSAGE PASSING INTERFACE", MIT Press, Cambridge, MA, 1999.
- [19] The NIST Definition of Cloud Computing (SP 800-145) published at .NIST Tech Beat: Published October 25, 2011, <http://csrc.nist.gov/publications/PubsSPs.html#800-145>
- [20] J. Warnke, S. Pawaskar and H. Ali, "An Energy-Aware Bioinformatics Application for Assembling short reads in High Performance Computing Systems" in "Int'l Conf. on High Performance Computing & Simulation" HPCS Madrid, Spain, July 2 – 6, 2012. <http://hpcs2012.cisedu.info/>