7-2013

Energy Awareness and Scheduling in Mobile Devices and High End Computing

Sachin S. Pawaskaw
University of Nebraska at Omaha

Energy Awareness and Scheduling in Mobile Devices and High End Computing

By

Sachin S. Pawaskar

A DISSERTATION

Presented to the Faculty of

The Graduate College at the University of Nebraska

In Partial Fulfillment of Requirements

For the Degree of Doctor of Philosophy

Major: Information Technology

Under the Supervision of Dr. Hesham Ali

Omaha, Nebraska

July, 2013

Supervisory Committee:

Dr. Hesham Ali

Dr. Deepak Khazanchi

Dr. Jon Youn

Dr. Dhundy Bastola

Dr. Hamid Sharif

ProQuest LLC.
789 East Eisenhower Parkway
P.O. Box 1346
Ann Arbor, MI 48106 - 1346

ABSTRACT

Energy Awareness and Scheduling in Mobile Devices and High End Computing

Sachin S. Pawaskar, Ph.D.

University of Nebraska, 2013

Advisor: Dr. Hesham Ali

In the context of the big picture as energy demands rise due to growing economies and growing populations, there will be greater emphasis on sustainable supply, conservation, and efficient usage of this vital resource. Even at a smaller level, the need for minimizing energy consumption continues to be compelling in embedded, mobile, and server systems such as handheld devices, robots, spaceships, laptops, cluster servers, sensors, etc. This is due to the direct impact of constrained energy sources such as battery size and weight, as well as cooling expenses in cluster-based systems to reduce heat dissipation. Energy management therefore plays a paramount role in not only hardware design but also in user-application, middleware and operating system design. At a higher level Datacenters are sprouting everywhere due to the exponential growth of Big Data in every aspect of human life, the buzz word these days is Cloud computing. This dissertation, focuses on techniques, specifically algorithmic ones to scale down energy needs whenever the system performance can be relaxed. We examine the significance and relevance of this research and develop a methodology to study this phenomenon.

Specifically, the research will study energy-aware resource reservations algorithms to satisfy both performance needs and energy constraints. Many energy management schemes focus on a single resource that is dedicated to real-time or non-real-time processing. Unfortunately, in many practical systems the combination of hard

and soft real-time periodic tasks, a-periodic real-time tasks, interactive tasks and batch tasks must be supported. Each task may also require access to multiple resources. Therefore, this research will tackle the NP-hard problem of providing timely and simultaneous access to multiple resources by the use of practical abstractions and near-optimal heuristics aided by cooperative scheduling. We provide an elegant EAS model which works across the spectrum which uses a run-profile based approach to scheduling. We apply this model to significant applications such as BLAT and Assembly of gene sequences in the Bioinformatics domain. We also provide a simulation for extending this model to cloud computing to answers "what if" scenario questions for consumers and operators of cloud resources to help answers questions of deadlines, single v/s distributed cluster use and impact analysis of energy-index and availability against revenue and ROI.

KEYWORDS:

Energy Awareness, Scheduling, High Performance Computing, Bioinformatics, Heuristics, Parallel Processing, Optimal Algorithms, Run-Profile, Cloud Computing, Alignment, Sequencing, Energy Aware Scheduling, Mobile Computing, Simulation.

DEDICATION

I dedicate my dissertation work to my family and many friends. A special gratitude to my loving parents, Sudhakar and Sharmila Pawaskar whose words of encouragement and teachings have always guided me throughout my life.

A special thank you to my wife Kelly, who has put up with me through this endeavor and to my daughter Sihley and son Sam who have driven me to see this through.

To all my friends and well-wishers who have supported me throughout the process. I will always appreciate all they have done.

ACKNOWLEDGEMENT

I would like to express the deepest appreciation to my committee chair Dr. Hesham Ali, who first encouraged me to start on this endeavor, then continually nourished me during this journey and would not give up on me when the going got tough. Without his guidance and persistent help this dissertation would not have been possible.

I would like to thank my committee members, Dr. Deepak Khazanchi for his continuous encouragement whenever he saw me. Dr. Dhundy Bastola was very helpful in his guidance when working on the application aspects of energy awareness in the bio-informatics domain and Dr. Jon Youn in application with mobile and sensor devices. Dr. Hamid Sharif for his engagement and support over the last several years.

In addition, a thank you to all the professors and faculty at the University of Nebraska at Omaha for all the little things I have learned from them and a thank you to my fellow students who were there to inspire me when needed.

Table of Contents

Chapter 1: Introduction

1.1 Research Problem

1.1.1 High End Computing

US Data centers consumed 5 MKW of energy in 2005 (Snyder, 2008), 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 (AMD, 2007) (He, 2008). 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 (Foley, 2008) and despite economic slump; Yahoo plans a new datacenter in La Vista, Nebraska (Yahoo, 2008). Clearly "Energy" is becoming a key business driver.

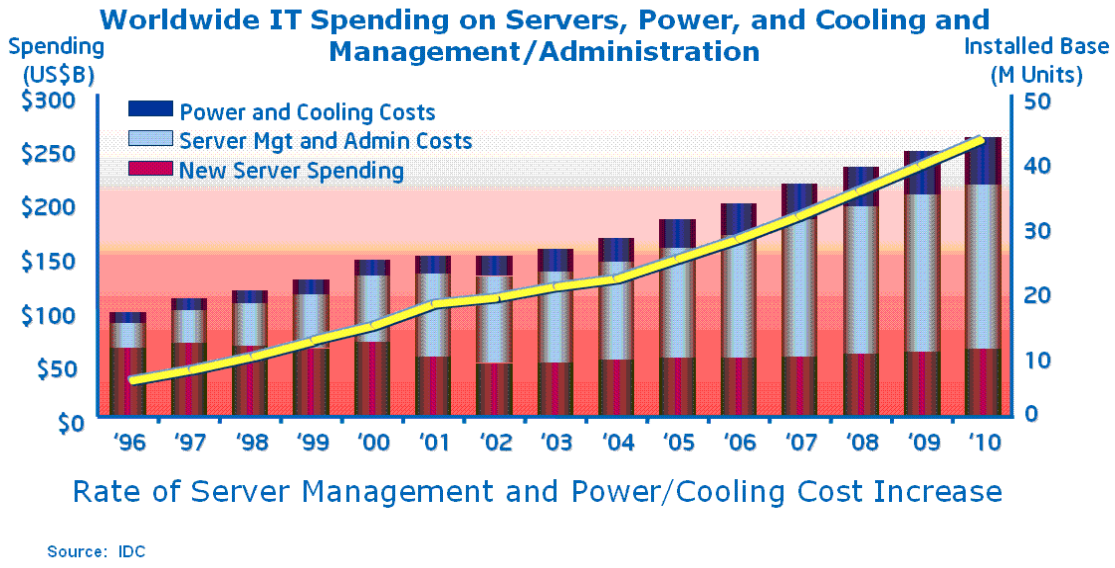

Figure 1: World IT Spending - Energy Cost Increase

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 section we will also focus on

studying energy aware scheduling mechanism in a high end 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 High End Computing (HEC) such as clusters and grids and make intelligent scheduling decisions which will balance energy minimization requirements against performance based upon user needs.

The need for minimizing energy consumption continues to be compelling in embedded, mobile, and server systems such as handheld devices, robots, spaceships, laptops, cluster servers, etc. This is due to the direct impact of constrained energy sources (e.g., battery size and weight), as well as cooling expenses in cluster-based systems to reduce heat dissipation (Rajkumar, 2005). Battery-operated portable devices are now ubiquitous and are widely used in mobile computing and wireless communication applications. Maximizing battery lifetime is the most important design metric for such systems. This problem is quite challenging due to the non-linear behavior of the battery. Since the amount of energy delivered by the battery depends on the discharge current profile (Martin, August 1999) (Battery life challenge, 2004), the battery life can be extended by controlling the discharge current level and shape. In recent years, there has been significant amount of work done in studying battery characteristics (Martin, August 1999) (Battery life challenge, 2004) and using these characteristics to shape the discharge profile (Rakhmatov, Vrudhula, & Chakrabarti, 2002) (Chowdhary & Chakrabarti, 16-18 Oct. 2002). All of the earlier work on battery aware task scheduling has been for static tasks where complete information about the tasks is known a priori. Task scheduling for real-time tasks has been investigated in the context of ideal power sources (Chowdhary &

Chakrabarti, 16-18 Oct. 2002). Some work has also been done on battery aware scheduling for real-time tasks (Ahmed & Chakrabarti, 2004). In my opinion not enough emphasis has been placed on making software applications aware of energy usage to the extent that this knowledge can drive the running of tasks on these devices based on some policy (such as mission criticality, time, etc); which leads us in the direction of how do we best schedule these tasks on battery operated devices such that we maximize the energy property (such as usage, efficiency) of these devices and the related applications that run on them.

1.1.2 Mobile Devices

Mobile computing has become a reality. Through the Wireless Verification Program, Intel® and leading wireless LAN service providers have verified more than 40,000 hotspots around the world, with more cropping up each day (Battery life challenge, 2004). Mobile technology is continually advancing to keep up with the needs of the mobile user. But as we work to make the ideal mobile experience, we find ourselves up against an inherent struggle between extending battery life and improving mobile performance. Power consumption has been a critical design constraint in the design of digital systems due to widely used portable systems such as cellular phones and PDAs, which require low power consumption with high speed and complex functionality. The design of such systems often involves reprogrammable processors such as microprocessors, microcontrollers, and DSPs in the form of off-the-shelf components or cores. Furthermore, 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 of such systems.

Battery operated portable devices are widely used in mobile computing and wireless communication applications. Maximizing battery lifetime is the most important design consideration for such systems. Since the amount of energy delivered by the battery depends on the discharge current profile, the battery life can be extended by controlling the discharge current level and shape (Ahmed & Chakrabarti, 2004) (Shin & Choi, 1999). 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 (Shin & Choi, 1999). In recent years there has been a significant amount of work done on studying battery characteristics and using these characteristics to shape the discharge profile. Most of the earlier work for battery-aware task scheduling has been for static tasks where complete information about the tasks is known apriori (Ahmed & Chakrabarti, 2004).

1.3 Energy Management

Energy management clearly plays a paramount role in not only hardware design but also in user-application, middleware and operating system design. This project focuses on techniques, specifically algorithmic ones to scale down energy needs whenever the system performance can be relaxed. Specifically, the project will study energy-aware resource reservations algorithms to satisfy both performance needs and energy constraints. Many energy management schemes also focus on a single resource that is dedicated to real-time or non-real-time processing. Unfortunately, in many practical systems such as Personal Digital Assistants (PDA), cellular phones, robots and personal computers, the combination of hard and soft real-time periodic tasks, a-periodic real-time tasks, interactive tasks and batch tasks must be supported. Each task may also require access to multiple resources (Rajkumar, 2005). Therefore, we will tackle the NP-hard problem of providing timely and simultaneous access to multiple resources by the use of practical abstractions and near-optimal heuristics aided by cooperative scheduling. Approaches where power management is carried out in different islands separately will also be compared.

1.4 Research Questions

The purpose of this research is to build a model for studying energy aware scheduling in mobile devices and high end computing where energy resources are constrained and heat dissipation is a major concern. Thus, the general research questions are as follows:

1) *Is there a general model for performing energy aware scheduling of tasks in mobile devices and HEC?*

2) *What are some of the general algorithms that we can use to schedule tasks in an energy aware environment?*

3) *Are there special cases of the energy aware scheduling problem and can we come up with specific algorithms that have polynomial runtime for them?*

4) *Are there specific domains where this problem exists and can we apply this some specific solutions to these domains?*

To answer these questions, additional research questions must also be answered:

- Is the consumer energy aware and is society willing to fund research in this area?
  - How likely and how much additional cost is the consumer willing to pay for energy aware devices and/or applications?
  - Is it the responsibility of the consumer or the manufacturers who make mobile devices to be energy aware?
- How can we model the problem using techniques in graph theory?
  - What makes graphs such as pervasive data structure for modeling this problem?
  - Are there general scheduling techniques that can be directly applied to the energy aware scheduling problem?
  - Given that the general scheduling problem is NP-Hard, What approximation algorithms and heuristics can we use for polynomial solutions?
- Can we solve these for special cases of the general problem?

- What are the special cases of the general problem?
- What is the usefulness of the special cases?
- Are there polynomial algorithms for the special cases?

The energy aware problem will be expressed through a conceptual model, which is a means of communicating specifications, and defining the problem space using events, or processes in a graphical format (Wand & Weber, 2002). Algorithmic graph theory will be used to provide polynomial time solution to the problem.

1.5 Research Motivation

1.5.1 Professional Motivation

Humanity has always shown great resolve in finding solutions to problems and grand challenges that go far beyond mere intellectual curiosity. The NITRD program defines a grand challenge as "A Grand Challenge is a long-term science, engineering, or societal advance, whose realization requires innovative breakthroughs in information technology research and development (IT R&D) and which will help address our country's priorities." (Strawn, Howe, & King, November 2006). In the latter half of the twentieth century information technology has amplified our intellectual and physical abilities. Scientific and Engineering marvels such as the internet, the global positioning system (GPS), DNA fingerprinting, facial recognition, and the human genome project have become possible only with advances in information technology. "Today there are eight billion computers in the world. Most are embedded invisibly in products, making goods and services safer, more secure, flexible, and energy-efficient, and less expensive than

ever before. The tremendous advances in productivity that we have witnessed in the past decade rest on this foundation." (Feigenoff & al., 2003).

We are moving beyond stand-alone computers or components to build large, integrated, distributed information systems which mobile and ubiquitous that are in service to society. In the future, we can expect our computational infrastructure to offer an even more impressive range of social and economic benefits as it grows to include billions of people worldwide. Information technologies have the potential to reduce energy consumption, provide improved health care at lower cost, enhance security, reduce pollution, enable further creation of worldwide communities, engender new business models, and contribute to the education of people anywhere in the world. The CRA with funding from NSF, convened a group of researchers, who during a 3 day conference discussed the specific and urgent challenges related to building the systems of the future. As a result of that discourse, the participants selected five grand research challenges that will provide a focus for more directed and immediate relevant research. These are listed below (Feigenoff & al., 2003):

1) Create a Ubiquitous Safety.Net.
2) Build a Team of Your Own.
3) Provide a Teacher for Every Learner.
4) Build Systems You Can Count On.
5) Conquer System Complexity.

1.5.1 A National Priority

The NITRD illustrative grand challenges were formulated to stimulate current and future generations of NITRD and applications researchers. The operative word here is illustrative because there are easily hundreds if not thousands of grand challenges that could be identified. By describing these challenges, NITRD intends to explain, justify, and galvanize the IT R&D community to solve IT hard problems that are important to society. The NITRD Program's illustrative grand challenges are shown in Appendix A.

The national priorities and the IT hard problems are the key pillars on which the grand challenges are structured. By describing the relationship between a grand challenge and national priorities, the grand challenge's significance is connected to the highest aspirations of our country. The IT hard problems, whose solution the grand challenge requires, tie the grand challenge to core elements of information technology research and development and the NITRD Program. The NITRD grand challenges were specifically a call to update the list called for in the High-Performance Computing (HPC) Act of 1991, which formally established the High Performance Computing and Communications (HPCC) Program. Through the HPCC Program, the U.S. Government coordinated multi-agency investments in developing and using high-performance computing systems and advanced networking technologies to meet the mission needs of the participating agencies and larger national goals. The Act's objectives included to:

1) Develop teraops (trillions of operations per second) computing systems.

2) Develop gigabit (billions of bits per second) networks.

3) Develop advanced algorithms and software.

4) Demonstrate innovative solutions to "grand challenge" problems using HPCC technologies.

Relationship of IT Hard Problem Areas and IT Hard Problems: IT hard problems areas are broad categories of topics of interest to the information technology research and development community and the NITRD Program. The Task Force identified 14 IT hard problem areas (Appendix). It was shown in (Strawn, Howe, & King, November 2006) that Algorithms and Applications had direct relationship with most of the illustrative Grand Challenges.

1.5.2 Scholarly Motivation

Our research in Energy Aware Scheduling based on Algorithmic Graph Theory can be used in wireless mobile devices, sensor networks, parallel computing, grid computing, high end computing, etc. This is due to the direct and indirect ability to manage energy consumption of battery operated devices and sensors as well as regulating heat in grid computing and high end computing by scheduling tasks away from over-heated components. Some of the areas identified by the grand challenges that are directly related to this research are:

1) Embedded multimodal sensor/actuator nodes.

2) Self-adaptive systems

3) Network reliability and availability will be key features of all large systems, our research will help address the critical aspect network nodes being able communicate their energy levels and node temperatures so as to avoid catastrophic failure of these nodes due to

draining of battery capacity or over-heating, thus helping improve the overall reliability and availability of the network and system.

1.6 Key Contributions

In this dissertation we take the challenge of coming up with a model that addresses the Energy Awareness question across the spectrum from High End Computing to Mobile & Sensor devices and we propose our EAS model which uses scheduling heuristics to balance opposing criterions of energy minimization and performance across the spectrum. We first provide an overarching EAS model and then focus on each segment of this spectrum and tailor our approach based on the uniqueness of that spectrum and propose solutions that are then incorporated into the EAS model, thus allowing the model to scale along the spectrum yet handle the nuances within each segment. At one end of the spectrum we are challenged with reducing costs and at the other we have to optimize battery and energy utilization Figure 2.

Figure 2: EAS model working across the spectrum

Our other key contributions include the following.

1) We provide a robust EAS Model which works in all High Performance Cluster environments that allow the use of MPI.
2) We apply this EAS Model to the Bioinformatics domain and apply it to specific applications and extend the knowledge gained using the concept of run profiles.
3) We provide a new cloud simulation package as part of our work, which allows us to run simulations across cloud resources to see if task deadlines can be met and simulate more complex scenarios for cloud operators such as impact of cluster availability v/s ROI.
4) We provide a Run-Queue Peek scheduling heuristic at the low end of the spectrum to address devices that run periodic tasks.
5) We also provide an enhanced Expected Execution Task heuristic which builds on the earlier scheduling heuristic to provide additional energy efficiency gains.
6) The need to carefully develop a parallel model based on the importance of understanding of the data within the specific application domain.

Chapter 2: Literature Survey, Basic Terminology & Problem Definition

2.1 Introduction to Scheduling

Scheduling is a classical field with several interesting problems and results. 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 to servers for processing. A problem may involve jobs that need to be processed in a manufacturing plant, bank customers waiting to be served by tellers, aircrafts waiting for landing clearances, or program tasks to be run on a parallel or a distributed computer. Clearly, there is a fundamental similarity to scheduling problems regardless of the difference in the nature of the tasks and the environment.

The scheduling problem has been described in a number of different ways in different fields. The classical problem of job sequencing in production management has influenced most of what has been written about this problem. Most manufacturing processes involve several operations to transform raw material into a finished product. 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 (Ullman, 1975) (Coffman, et al., 1976). However, many heuristic methods have been used to deal with most general case of the

problem. Such methods include traditional priority-based algorithms (Hesham, Lewis, & Hesham, 1994), task merging techniques (Aronsson & Fritzson, Jan 8-10, 2003), critical path heuristics (Hesham, Lewis, & Hesham, 1994) (Khan, McCreary, & Jones, 1994). In addition, distributed algorithms have been designed to address different versions of the scheduling problem (Xie, Rus, & Stein, Dec, 2001).

2.1.1 The Scheduling Problem

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). Accordingly, a scheduling system can be considered as consisting of a set of consumers, a set of resources, and a scheduling policy as shown in Figure 3.

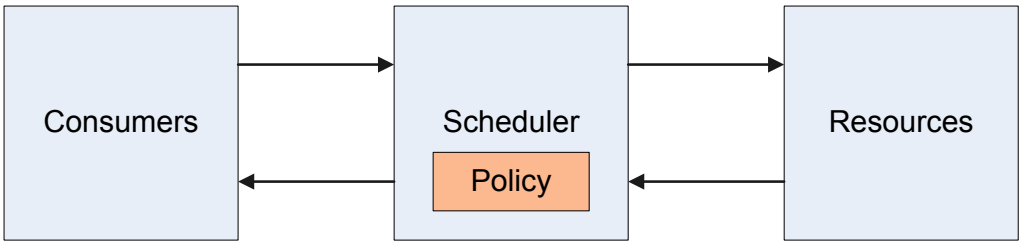

Figure 3: The Scheduling System

Examples of consumers are a task in a program, a job in a factory, or a customer in a bank. Examples of resources are a processing element in a computer system, a machine in a factory, or a teller in a bank. First-come-first-served is an example of a scheduling policy. Scheduling policy performance varies with different circumstances. While first-come-first-served may be appropriate in a bank environment, it may not necessarily be

the best policy to be applied to jobs on a factory floor. Performance and efficiency are two parameters used to evaluate a scheduling system. It's customary to evaluate a scheduling system based on the goodness of the produced schedule and the efficiency of the policy.

In the general scheduling problem, we are concerned with scheduling dependent program tasks on parallel and distributed systems. The tasks are the consumers and will be represented using directed graphs called task graphs. Task graphs are used to represent precedence relationships between tasks. The processing elements are the resources and their interconnection networks will be represented using undirected graphs. The "scheduler" (Figure 4) generates a schedule using a timing diagram called the Gantt chart. The scheduler performs allocation, which means it will tell which tasks go on which processor, but does not give their order. Whereas "scheduling" will perform allocation as well as provide an order for the tasks on the individual processors. The Gantt chart illustrates the allocation of the parallel program tasks onto the target machine processors and their execution order. A Gantt chart consists of a list of all processors in the target machine and, for each processor, a list of all tasks allocated to that processor ordered by their execution time. The term tasks, nodes and jobs will be regarded as equivalent to the term "consumers". Also, resources may be referred to as processors or processing elements.

2.1.2 Task Scheduling Model

The model that we will study in this thesis is deterministic and static in the sense that all information governing the scheduling decisions are assumed to be known in advance. In

particular, the task graph representing the parallel program and the target machine is assumed to be available.

There are four components in any scheduling system: the target machine, the parallel tasks, the generated schedule, and the performance criterion. In our task-scheduling model we will ignore the communication delays and consider all tasks to have the same unit execution time. Also most of the time, we deal with the same machine, i.e. multiple processors on the same machine. Nowadays we have such similar environments that it leads to almost same communication delay times. We will discuss and define the scheduling problem in more detail later in the thesis.

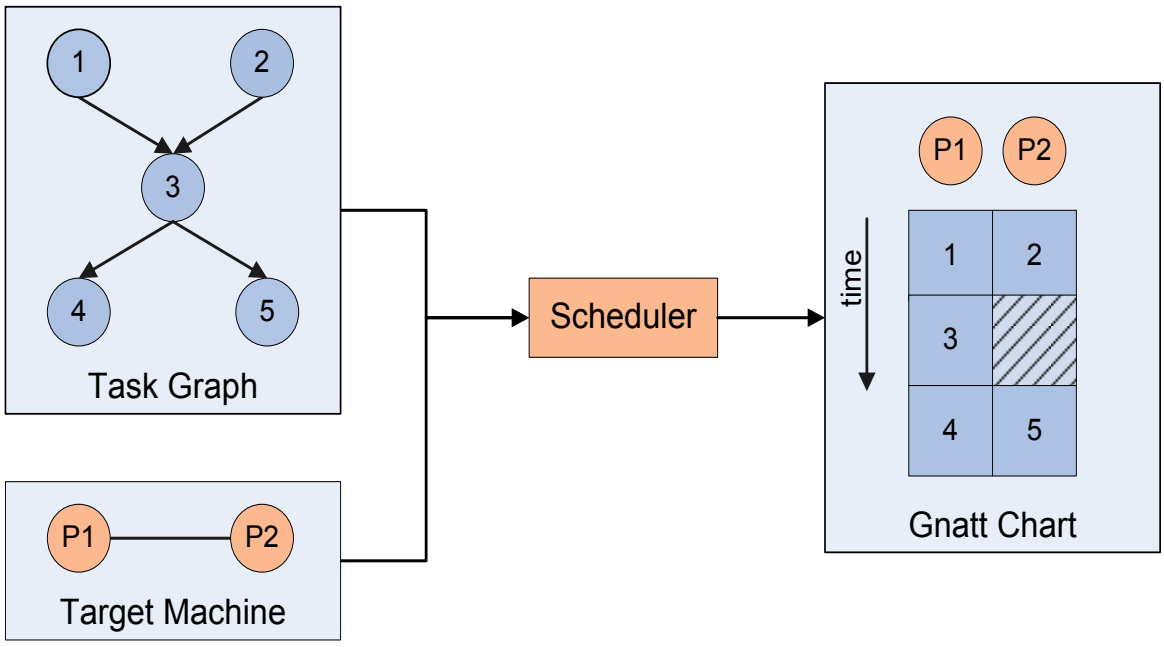

Figure 4: A Scheduler

2.1.3 Energy Aware Scheduling

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.

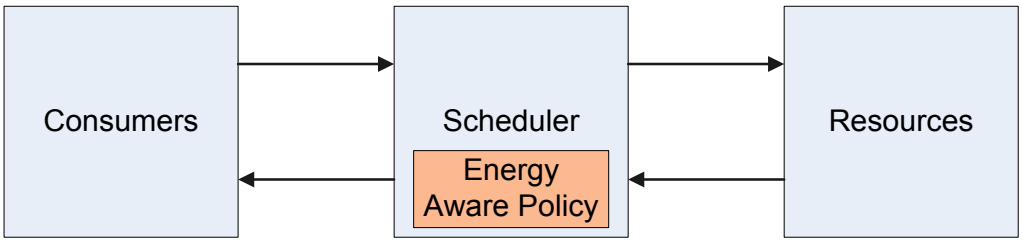

Figure 5: Energy Aware Scheduling System

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 5 above.

2.2 From Static to Dynamic to Dynamic Energy aware scheduling

Our study initially focus on feasibility of the schedule, followed by, honoring the restrictions and meeting the requirements. Our research aims to move scheduling research from the classical static scheduling approaches of the 1910 - 1970, to dynamic scheduling approaches of 1990's to take advantage of the slack generated due to the difference between WCET and AET, and finally take the dynamic approach to the next level with a focus on energy utilization, given the latest advances in DVS technology and the business

driven need for reducing energy costs with dynamic energy aware scheduling. The figure below Figure 6 shows a conceptual model of 2 tasks T_1 and T_2 as they evolve through the various models from static to dynamic to dynamic energy aware scheduling.

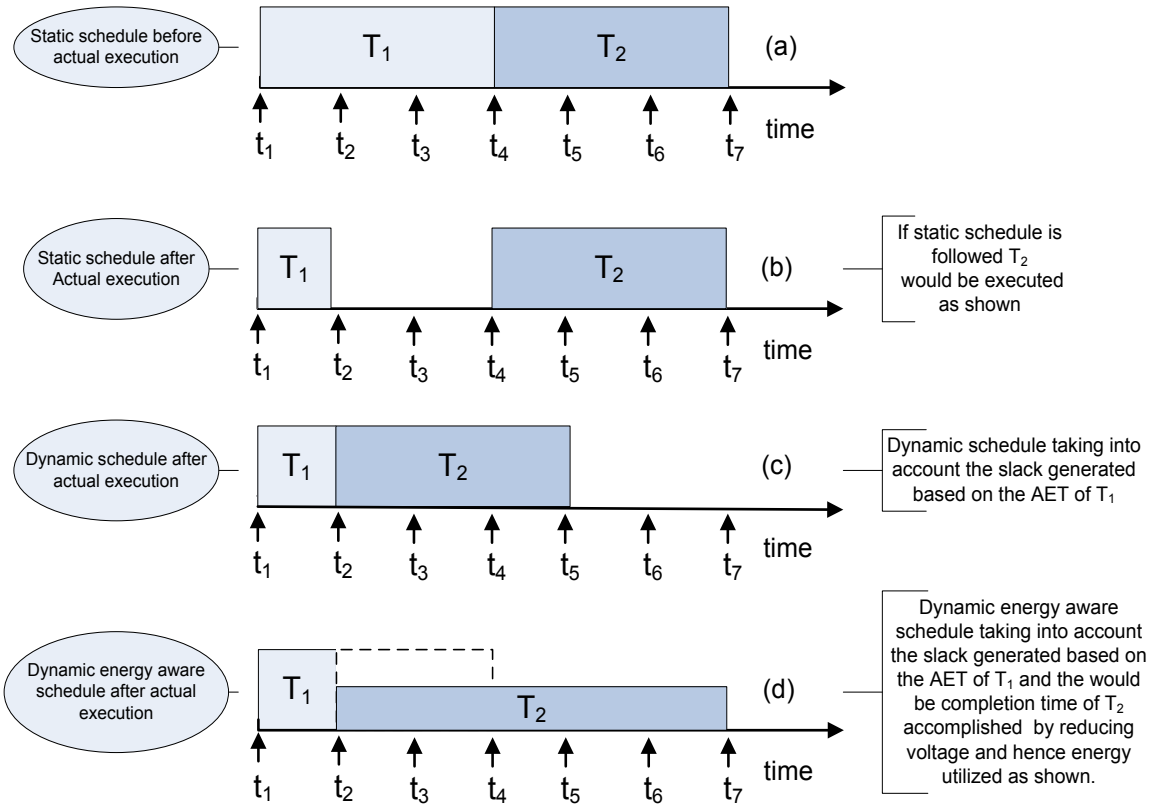

Figure 6: Static to Dynamic to Dynamic Energy Awareness Scheduling

2.3 Basic Terminology & Problem Definition

In this section we define a few terms that will be used in the later sections of this paper. We discuss the NP-completeness of the scheduling problem and present a complexity comparison of the various scheduling problems.

Task Graph: A task graph G = (T, A) is a directed acyclic graph. For a pair of tasks t_i, t_j $\in$T, a directed edge (i, j) $\in$A between the two tasks specifies that t_i must be completed before t_j can begin. Figure 7 shows a task graph.

Density or Sparseness: The density or sparseness of a graph G=(T,A) is computed as a ratio of the number of edges |A| in the graph as a percentage to the maximum number of edges that graph can have which is of order (|T| * |T-1|) / 2. So a graph with density of 0.5 will have half the number of maximum edges possible for that graph.

Task Level: Let the level of a node x in a task graph be the maximum number of nodes (including x) on any path from x to a terminal task. In a tree, there is exactly one such path. A terminal task is at level 1. Given the graph in Figure 7, we can say that nodes 1, 2 and 3 are at level 1, 4 and 5 are at level 2, nodes 6,7,8,9 and 10 are at level 3, and so on.

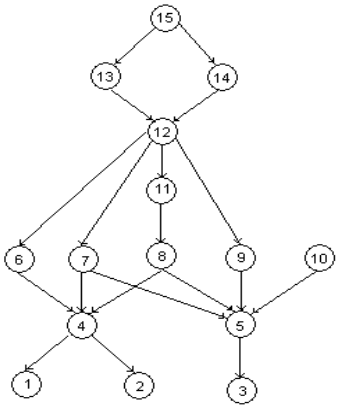

Figure 7: A Task Graph

Schedule Length or Schedule Time: Given a task graph G = (T, A) and its schedule on m processors, f, the length of schedule f of G is the maximum finishing time of any task in G.

2.3.1 NP-Completeness of the Scheduling Problem

In general, the time complexity of an algorithm refers to its execution time as a function of its input. We specify the complexity of a scheduling algorithm as a function of the number of tasks and the number of processors. A scheduling algorithm whose time complexity is bounded by a polynomial is called a polynomial-time algorithm. An optimal algorithm is considered to be efficient if it runs in polynomial time. Inefficient algorithms are those, which require a search of the whole enumerated space and have an exponential time complexity. The problem of scheduling parallel programs tasks on multiprocessor systems is known to be NP-complete in its general form. There are few known polynomial-time scheduling algorithms even when severe restrictions are placed on the task graph representing the program and the parallel processor models. In general we can say classify the known results as follows:

1) The NP-Completeness of several versions of the scheduling problems (Ullman, 1975).
2) Optimal "efficient" algorithms, for solving restricted versions of the scheduling problems (Coffman, et al., 1976), (Hesham, Lewis, & Hesham, 1994).
3) Heuristic algorithms for tackling more general cases of the scheduling problems (Hesham, Lewis, & Hesham, 1994).
4) Table 1 summarizes the complexity of several versions of the scheduling problem when the target machine is fully connected. Note that n is the number of tasks and e is the number of arcs in the task graph. Note also that the results in Table 1 are obtained

when communication costs are not considered. Forest and interval-order are special classes of task graphs. For more detailed definition and the formal discussion of NP-completeness please refer (Ullman, 1975) (Hesham, Lewis, & Hesham, 1994).

Table 1: Complexity comparison of scheduling problem

Task Graph	Task Execution Time	Number of Processors	Complexity
Tree	Identical	Arbitrary	O(n)
Interval order	Identical	Arbitrary	O(n)
Arbitrary	Identical	2	$O(e + n\alpha(n))$
Arbitrary	Identical	Arbitrary	NP-complete
Arbitrary	1 or 2 time units	≥ 2	NP-complete
Opposing forest	Identical	Arbitrary	NP-complete
Interval order	Arbitrary	≥ 2	NP-complete
Arbitrary	Arbitrary	Arbitrary	NP-complete

As mentioned earlier a number of scheduling heuristic have been developed to deal with many versions of the scheduling problem. Among the developed heuristics, List scheduling has been used often due to its simplicity and over all good results. List scheduling is a class of scheduling heuristics in which tasks are assigned priorities and placed in a list ordered in decreasing magnitude of priority. Whenever tasks contend for processors, the selection of tasks to be immediately processed is done on the basis of priority with the higher-priority tasks being assigned processors first. If there is more than one task of a given priority, ties are broken randomly.

2.3.1 Scheduling and the Battery Operated Device Model

Our research will focus in the software – application area and will specifically try to address the question of energy aware scheduling of application tasks. There are several models for which different algorithms have been proposed (see Appendix B). We take look at one such model, discuss the scheduling algorithm proposed for this model (battery

operated devices), its variations and finally present our improvement for scheduling on this model.

Let us understand the basic characteristics of this Battery Operated Model.

1. The model assumes fixed priority scheduling.

2. The model is for a real time system, in which task deadlines must be met.

The system configuration for the battery-operated processor under consideration is described in Figure 8. The system consists of one DVS processor driven by a single battery. The battery is used to power the processor through a DC-DC converter. The DC-DC converter has an efficiency $\eta = I_{proc}*V_{proc}/I_{batt}*V_{batt}$, where V_{batt} and I_{batt} are the battery voltage and current and V_{proc} and I_{proc} are the processor voltage and current.

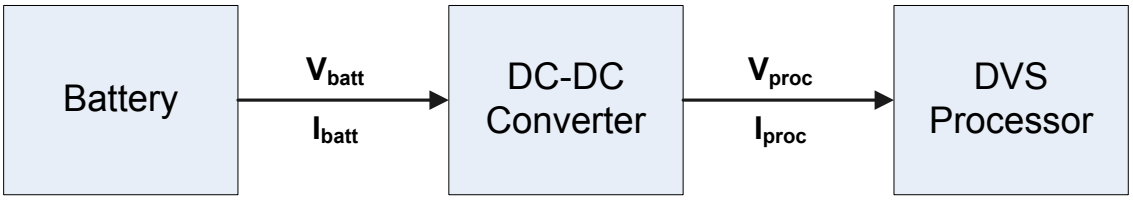

Figure 8: System Level Configuration

Non-linear properties of the battery:

There are several important properties of the battery with respect to voltage scaling that have been derived from the analytical model. We present two of the properties used for developing the real-time scheduling heuristics (Ahmed & Chakrabarti, 2004) (Chowdhary & Chakrabarti, 16-18 Oct. 2002):

Property 1: *For a fixed voltage assignment (only task start times can be changed), sequencing tasks in the non-increasing order of their currents is optimal when the task loads are constant during the execution of the task.*

Property 2: *Given a pair of two identical tasks in the profile and a delay slack to be utilized by voltage down-scaling, it is always better to use the slack on the later task than on an earlier task.*

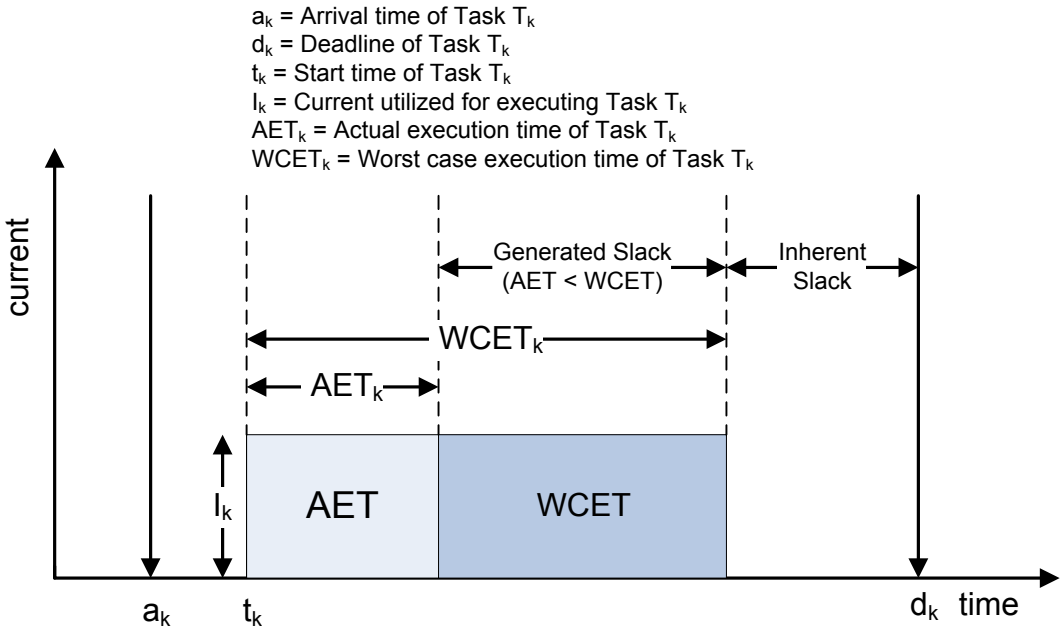

Figure 9: Task Description

Task description: A given task ***k*** is associated with the following parameters: the current $\boldsymbol{I_k}$, the worst case execution time $\boldsymbol{WECT_k}$, the arrival time $\boldsymbol{a_k}$, the start time $\boldsymbol{t_k}$, the actual execution time $\boldsymbol{AET_k}$, the deadline $\boldsymbol{d_k}$ and the period $\boldsymbol{P_k}$. The slack associated with a task is due to two factors: (1) the inherent slack due to the difference between the deadline and the WCET and (2) the slack generated due to the actual execution time being less than the worst case execution time.

Power-Down Modes:

In most embedded systems, a processor often waits for some events from its environment, wasting its power. To reduce the waste, modern processors are often equipped with various levels of power modes. In the case of the PowerPC 603 processor (Gary, et al., 1994), there are four power modes, which can be selected by setting the appropriate control bits in a register. Each mode is associated with a level of power saving and delay overhead. For example, in *sleep mode*, where only the PLL and clock are kept running, power consumption drops to 5% of full power mode with about 10 clock cycles delay to return to full power mode.

For the rest of the dissertation, we assume that the problem is deterministic in the sense that all information governing the scheduling decisions are assumed to be known in advance. In particular, the task graph representing the parallel program and the target machine is assumed to be available before the program starts execution. As in the standard scheduling system, our system has four components: the target machine, the parallel tasks (represented as a task graph), the generated schedule and the performance criterion. The minimization of the schedule length is the performance criterion considered in our scheduling model.

Chapter 3: Our Proposed Model and Research Approach

3.1 The Model

The current state of research is dominated by parallelization of code and how to achieve high degree of speedup. The discussion about tradeoff between performance and energy is limited based mainly on mathematical model with no concrete working model. Even on the applied side there is a feeling that hardware is cheap, so let's take advantage of it. Hence, we felt the need for a robust model that incorporates key information from the application domain is essential to study the tradeoff. We call our model the Energy Aware Scheduling (EAS) Model (Figure 10).

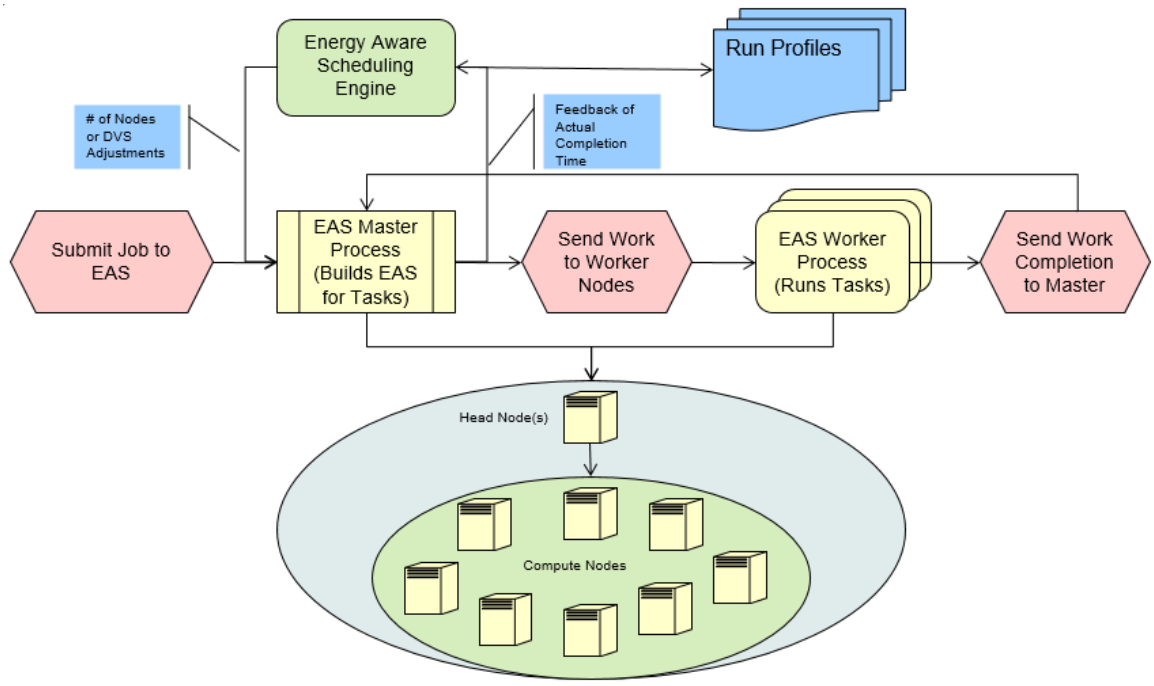

Figure 10: Our EAS Model

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.

3.1.1 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, and time it takes to run the task). Using this we can then first allocate a set of nodes for a given deadline.

3.1.2 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.

3.2 The Model – Logical View

In general the program consists of a Master and Several 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. The feedback mechanism is used as a learning mechanism to refine future decisions made by the EAS Model (Figure 11).

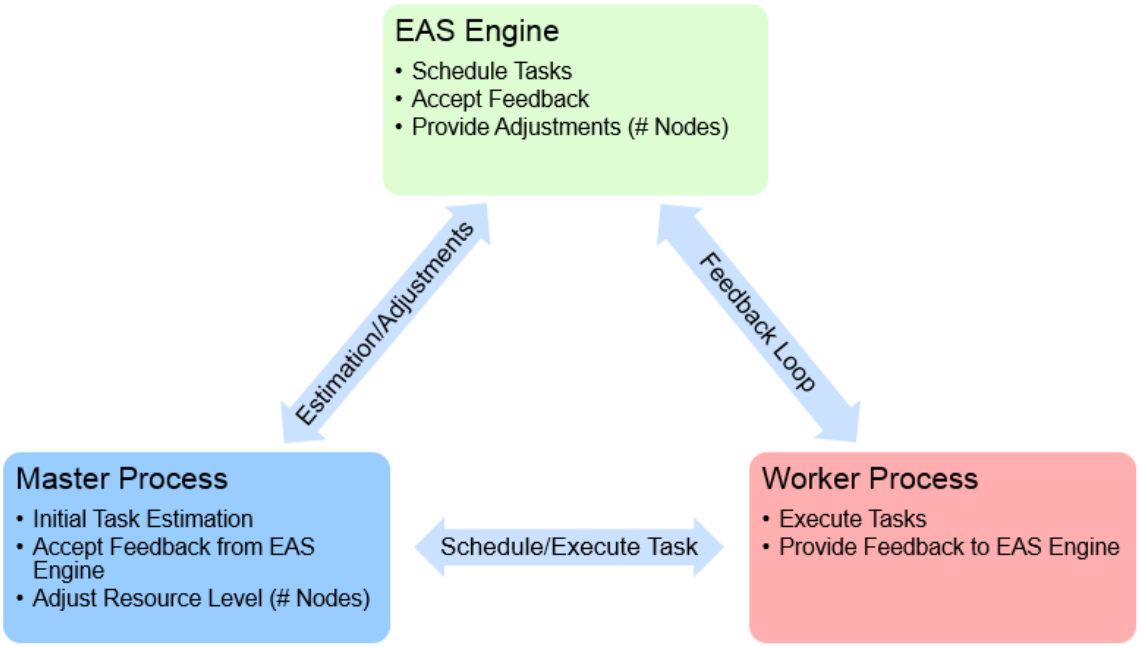

Figure 11: EAS Model - Logical View

3.2 Research Vision for our Model

Our EAS – Model is designed to address the issue of energy aware in using computing resources based on scheduling of tasks/jobs in such a manner that we no longer just focus on performance meaning returning results as fast as we can but also try and minimize energy utilized in the process while still meeting the desired deadline. Our EAS Model is designed to handle this issue across the broad spectrum from larger "Cloud" computing – HPC to small Node – Mobile devices (Figure 12).

Figure 12: Research Vision for our EAS Model

At the highest level we apply our model at the High Performance Computing – Cluster level by adjusting the number of nodes used to complete a set of tasks and by simulation show that this can be extended to the "Cloud" by introducing the concept of energy-index for Cloud datacenters. Finally we extend the model to the node level and show how we can minimize energy and still meet deadlines by using DVS techniques.

3.3 Research Approach

Conducting research in Algorithmic Graph Theory and related areas is no different from say research conducted in the area of Artificial Intelligence, Computer Sciences or Social

Sciences at least in terms of the research methodology that the researcher has to follow. In this article, I will discuss the research methodology or shall I say approach that I intend to take in conducting my research in “Energy Aware Scheduling”.

For the purpose of this research study, I intend to adapt Cohen and Howe’s cyclic multistage process for conducting research (Cohen & Howe, How evaluation guides AI research, 1988) (Cohen & Howe, Toward AI research methodology, 1989). First I need to step back and try and answer the question why do I need to define a methodology for my research? By defining my research methodology, I intend to accomplish two things, first I will be able to objectively answer the questions surrounding evaluation of the research being conducted and secondly it will provide me a framework that will guide me and keep me focused throughout the research. That being said; let us briefly discuss Cohen’s cyclic multistage process for conducting research; which can be viewed as a five-stage cycle (Figure 13).

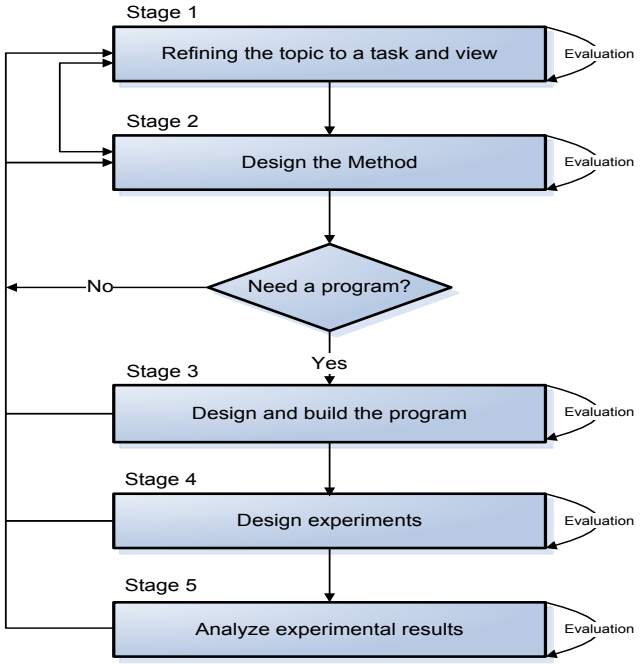

Figure 13: Cycle of Research (Cohen & Howe, How evaluation guides AI research, 1988)

Cohen defines evaluation as a mechanism of progress both within and across research projects. Evaluation can tell us how and why our methods and programs work and so, tell us how our research should proceed. For the community, evaluation expedites the understanding of available methods and so, their integration into further research.

3.4 Research Methodology

3.4.1 Stage 1: Relevance and Refinement

When researchers find particular topics fascinating the first stage of the research cycle involves simultaneously refining the research topic to a task and identifying a view. A task is something we want a computer to do, and a view is a pre-design, a rough idea of how to do it. An important point to note is that this in itself is an iterative process (Cohen & Howe, How evaluation guides AI research, 1988). Cohen provides a list of criteria for evaluating research problems which is presented in Figure 14.

The Big Picture: Energy is fundamental to the quality of our lives. Nowadays, we are totally dependent on an abundant and uninterrupted supply of energy for living and working. It is a key ingredient in all sectors of modern economies. We know that energy demand will increase significantly in the future. How then will we satisfy this huge energy requirement in an environmentally friendly way? (The importance of energy, 2005)

Future directions: Energy supply must be sustainable and diverse, and energy needs to be used more efficiently. A sustainable energy supply, both in the short- and the long-term, is needed for promoting both economic development and people's quality of life, as well as protecting the environment. We also need a greater diversification of energy

resources - if we are largely dependent on one fuel source, we risk price rises and supply disruptions. Energy is a precious resource which must be conserved. Improved energy efficiency, therefore, in our homes, factories, transport and in our day to day activities needs to be strongly encouraged.

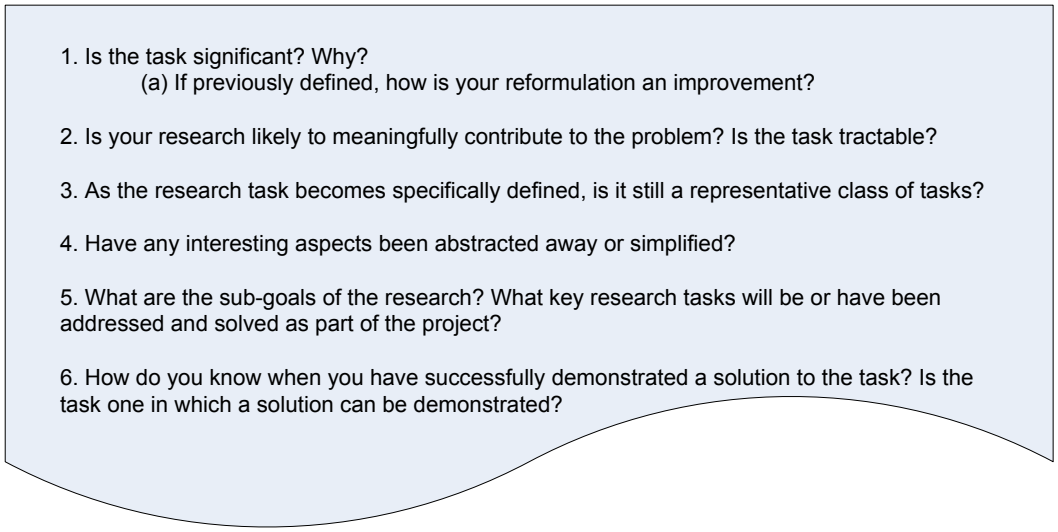

Figure 14: Criteria for Evaluating Research Problems

A sub-goal of this research will be to understand level of energy awareness amongst consumers and consumer attitudes to energy and related issues. I intend to accomplish by preparing a questionnaire dealing with energy awareness and attitudes and then performing statistical analysis on the data collected based on various demographics to identify with variables have significance. In the dissertation, I hypothesize that people are becoming more and more energy aware and want to incorporate energy awareness in various aspects of their lives and since mobile devices are becoming ubiquitous and more prevalent in our lives and their energy constraint, the problem of research in this dissertation namely “Energy aware scheduling” is significant for a social standpoint.

There is a lot of research being done in Europe on intelligent energy usage (Intelligent Energy Europe, 2007). There have also been various survey studies done on Energy conservation and awareness in households sector (Jaber, Mamlook, & Awad, Dec 2003), saving money through energy efficiency (Saving money through Energy Efficiency, Feb 2004), energy home improvements (Ulrich & Flagg, 2003), etc. The Canadian Electricity Association studied the attitudes to Canadians towards energy efficiency (Canadian Electricity Association, 2006). Here in the United States, the U.S Department of Energy (DOE) is spearheading research in several directions related to energy awareness and energy efficiency. Research has been done with regards to home appliance buying trends (U.S. Department of Energy, 1999).

3.4.2 Stage 2: Design the Method

At this stage one's view is refined to a method for solving the task. The method could be a single algorithm such as List Scheduling, Coffman and Graham's 2-P Scheduling algorithm, etc. We maintain this design the method step to remind us that we don't jump immediately into building programs and writing code but first decide how we want to solve the tasks. Cohen presents a list of criteria for evaluating methods; which are listed below in Figure 15 (Cohen & Howe, How evaluation guides AI research, 1988) (Cohen & Howe, Toward AI research methodology, 1989).

1. How is the method an improvement over existing methodology?
    (a) Does it account for more situations (inputs)?
    (b) Does it produce a wider variety of desired behaviors (outputs)?
    (c) Is the method expected to be more efficient (space, solution time, and so on)?
    (d) Does it hold more promise for further development (new paradigm)?

2. Does a recognized metric exist for evaluating the performance of your method?

3. Does it rely on other methods?

4. What are the underlying assumptions?

5. What is the scope of the method?
    (a) How extendible is it? Will it easily scale up to a larger knowledge base?
    (b) Does it exactly address the task? Portions of the task? A class of tasks?
    (c) Could it or parts of it be applied to other problems?
    (d) Does it transfer to complicated problems?

6. When it cannot provide a good solution, does it do nothing or does it provide bad solutions or does it provide the best solution given the available resources?

7. How well is the method understood?
    (a) Why does it work?
    (b) Under what circumstances, won't it work?
    (c) Are the limitations of the method inherent or simply not yet addressed?
    (d) Have the design decisions been justified?

8. What is the relationship between the problem and the method? Why does it work for this task?

Figure 15: Criteria for evaluating methods

3.4.3 Stage 3: Build a Program

After the second stage of "Design the method" we will move on to the next stage which is "Build a Program". Cohen's criteria for evaluating method implementation are presented in Figure 16 below. In this stage we will actually implement our scheduling algorithms and other comparative algorithms. We will be able to set up different energy policy functions and then run these different policies and compare them in terms of how effective were they in effectively utilization of the available energy in a battery and HPC environment using MPI and build a simulation for extending the model to "Cloud" computing.

1. How demonstrative is the program?
    (a) Can we evaluate its external behavior?
    (b) How transparent is it? Can we evaluate its internal behavior?
    (c) Can the task capabilities be demonstrated by a well-defined set of test cases?
    (d) How many test cases does it demonstrate?

2. Is it specially tuned for a particular example?

3. How well does the program implement the method?
    (a) Can you determine the program's limitations?
    (b) Have parts been left out or kludged? Why and to what effect?
    (c) Has implementation forced detailed definition or reevaluation of the method?
    (d) If reevaluation was required, How was this accomplished?

4. Is the program's performance predictable?

Figure 16: Criteria for Evaluating Method Implementation

3.4.4 Stage 4: Design Experiments

After the third stage of "Build a Program" we will move on to the next stage which is "Design Experiments". Cohen's criteria for evaluating the experiments design are presented in Figure 17 below.

1. How many examples can be demonstrated?
    (a) Are they qualitatively different?
    (b) Do the examples illustrate all the claimed capabilities?
    (c) Do the examples illustrate the limitations?
    (d) Is the number of examples sufficient to justify the inductive generalizations?

2. Should the program's performance be compared to a standard such as another program, or experts or novices, or its own tuned performance?

3. What are the criteria for good performance? Who defines the criteria?

4. Does the program purport to be general (domain independent)?
    (a) Can it be tested on several domains?
    (b) Are the domains qualitatively different?
    (c) Do they represent a class of domains?
    (d) Should there be inter-domain performance comparisons?
    (e) Is the set of domains sufficient to justify inductive generalization?

5. Is a series of related programs being evaluated?
    (a) Can differences in programs and their behavioral manifestations be determined?
    (b) Do the implementation differences of programs affect the generalizations?
    (c) Were difficulties encountered in implementing the method in other programs?

Figure 17: Criteria for Evaluating the Experiment Design

Various experiments will be run on the proposed energy management algorithm, other static scheduling algorithms, and the list scheduling heuristic using different graphs. The two most important properties of the graphs that the algorithms will be tested against are:

a) Number of nodes in the graph, and

b) The Density/Sparseness of the graph

3.4.5 Stage 5: Analyze the Experimental Results

After the fourth stage of "Design Experiments" we will move on to the next stage which is "Analyze the Experimental Results". Cohen's criteria for evaluating what the experiments told us are presented in Figure 18 below.

1. How did program performance compare to its selected standard (other programs)?
2. Is the program's performance different from predictions of how the method should perform?
3. How efficient is the program in terms of space and knowledge requirements?
4. Did the program demonstrate good performance?
5. Did you learn what you wanted from the program and experiments?
6. Is is easy for the intended users to understand?
7. Can you define the program's performance limitations?
8. Do you understand why the program work or doesn't work?
   - (a) What is the impact of changing the program even slightly?
   - (b) Does it perform as expected on examples not used for debugging?
   - (c) Can the effect of different control strategies be determined?
   - (d) How does the program respond if input is rearranged, noisy, or missing?
   - (e) What is the relationship of characteristics of the test problems and performance?
   - (f) How generalized is the understanding of the method and its characteristics?

Figure 18: Criteria for Evaluating What the Experiments Told Us

In this stage we will compare the performance results of our proposed energy management policy algorithm with that of the other static scheduling algorithms as well the heuristic list scheduling algorithm. We will also provide the Big-O for our proposed

algorithm and compare it with that of the other algorithms; this will also be accompanied with the space analysis for each program. This will be followed by an understanding of the limitation of the proposed algorithm such as communication costs/delays and how these may or may not affect the generalizations of the findings.

3.5 Application Domains for the Energy Aware Scheduling Problem

Any academic research has to be eventually related to specific application domains in which that research can be applied. Our energy aware scheduling problem is prevalent in several application domains such as mobile technology applications, wireless sensor networks, grid computing, animal field studies, oceanography, space technology, etc. Basically anywhere battery technology is being used or minimizing energy utilization is a key objective function. In this research we will focus on two application domains; one is the mobile devices and the second is the grid/parallel computing domain. In the grid/parallel computing domain we plan on teaming with the bioinformatics group to run several long running programs on a grid computing cluster and simultaneously minimize various objective functions key amongst which will be the minimization of energy.

The rest of the dissertation is organized as follows.

1) In chapter 4 we apply our EAS model to a HPC environment and more specifically to a commonly used application called BLAT (which is similar to BLAST) in the Bioinformatics domain.
2) In chapter 5 we extend our EAS model to incorporate a feedback mechanism. Our EAS Engine uses the concept of "Run-Profiles" to make intelligent scheduling decisions based on previous AET knowledge to adjust the schedule so as to minimize energy utilization while still meeting task/job deadlines.

3) In chapter 6 we extend and apply our EAS model to a more complex application within the Bioinformatics domain namely “Assembling of Short Reads”. Assembling of sequences is a very important and frequently used application by bioinformatics researchers.
4) In chapter 7 we take the next big step and extend our EAS model to the new paradigm of “Cloud” computing. We use simulation techniques which applies our EAS Model to answer important “What if” scenarios for both customers/users and operators of “Cloud” systems/clusters.
5) In chapter 8 we stretch our EAS Model to touch the other end of the spectrum, namely small Node level and Mobile devices and introduce additional algorithms such as Run-Queue peek and use DVS techniques to address the “Energy Awareness” aspects for scheduling purposes.
6) Finally in chapter 9 we present our overall conclusions and future research directions.

Chapter 4: Energy aware scheduling in High End Computing

US Data centers consumed 5 MKW of energy in 2005 (Snyder, 2008), 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 (AMD, 2007) (He, 2008). 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 (Foley, 2008) and despite economic slump; Yahoo plans a new datacenter in La Vista, Nebraska (Yahoo, 2008). 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 section we will also focus on studying energy aware scheduling mechanism in a high end 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 (Figure 19) for HEC such as clusters and grids and make intelligent scheduling decisions which will balance energy minimization requirements against performance based upon user needs.

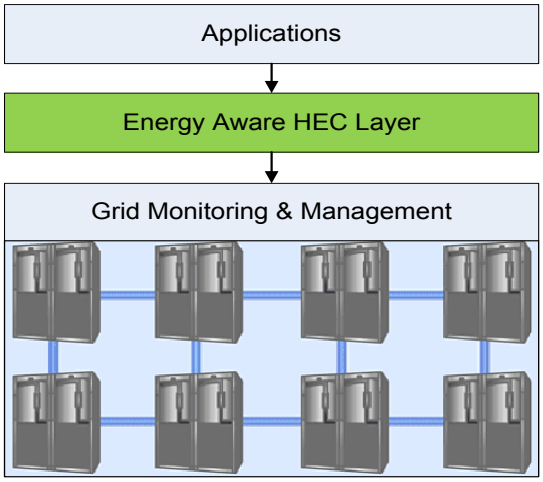

Figure 19: Energy aware scheduling layer for HEC

We will take the following approach to test our solution and proposed energy aware scheduling algorithms.

1) We will begin studying our proposed algorithm using simulation techniques where we will tailor the environment using various parameters and test our proposed scheduling algorithms.
2) Next we will run some simple bio-informatics applications which are inherently parallel, such as running a program to find if a particular gene sequence is present in a particular chromosome. In this problem we can run the program with the given gene sequence and a chromosome from 1 – 23 on 1 – 23 nodes on a cluster, this is because there is no dependency in running the gene sequence against the different chromosomes.
3) Next we will increase the complexity of the problem by introducing dependency within the problem space such that output of one run is input to the following run. This will require the scheduling algorithm to be smart enough to dynamically adjust to the runtime slack and schedule the follow-up task appropriately. Here we may have to deal with communication costs and handle task deadlines.
4) We will also study solutions from the standpoint of feasibility versus performance in the backdrop of energy utilization, the main objective being to understand how these impacts and influence energy utilization.

4.1 High Performance Computing and Amdahl's Law

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 (Amdhal, 1967). Amdahl's law is defined by the formula:

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

As $N \to \infty$, 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 $(1-P)$ (Amdhal, 1967) (Cho & Melhem, 2008) (Amdahl's Law, n.d.) (Hill & Marty, 2008). A great part of the craft of parallel programming consists of attempting to reduce (1 – P) to the smallest possible value. The Figure 20 below shows the speedup curves for various values of P.

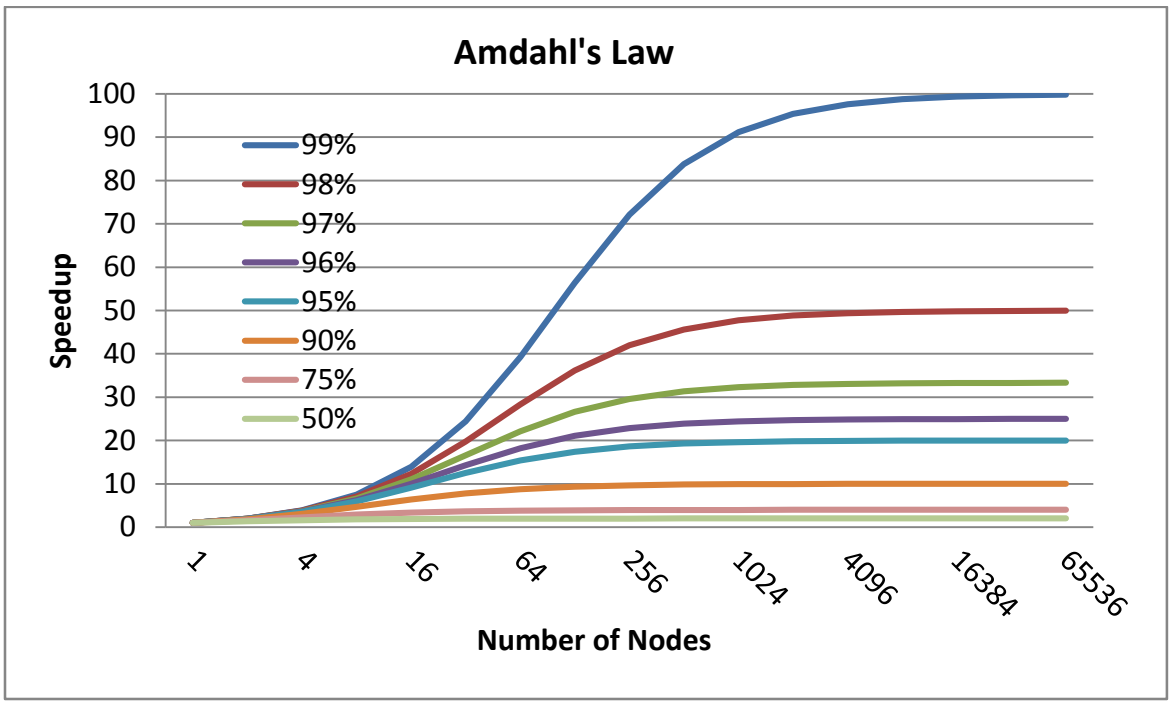

Figure 20: Amdahl's Law

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) (Blackforest Computing Cluster, n.d.), 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) (Holland Computing Center, n.d.).

4.2 Bioinformatics & High End Computing

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.

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 (Sequence and Annotations downloads, n.d.) (Genome Bioinformatics, n.d.).

In this section 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.

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 (Dayde, 2006). 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 (Braun, Pedretti, Casavant, Scheetz, & Roberts, 2001) (Chi, Shoop, Carlis, Retzel, & Riedl, 1997), 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 (Bjornson, Sherman, Weston, Willard, & Wing, 2002), 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 (Sequence and Annotations downloads, n.d.) (Genome Bioinformatics, n.d.). 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.

4.3 From Simple Speedup to the Realm of Energy Awareness

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.

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. The Figure 21 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 (A Portable Implementation of MPI, n.d.) (Gropp, Lusk, & Skjellum, Using MPI: Portable Parallel Programming with the Message Passing Interface, Oct 1994) (Gropp, Lusk, & Thakur, Using MPI-2: Advanced Features of the Message Passing Interface, Nov 1999). 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

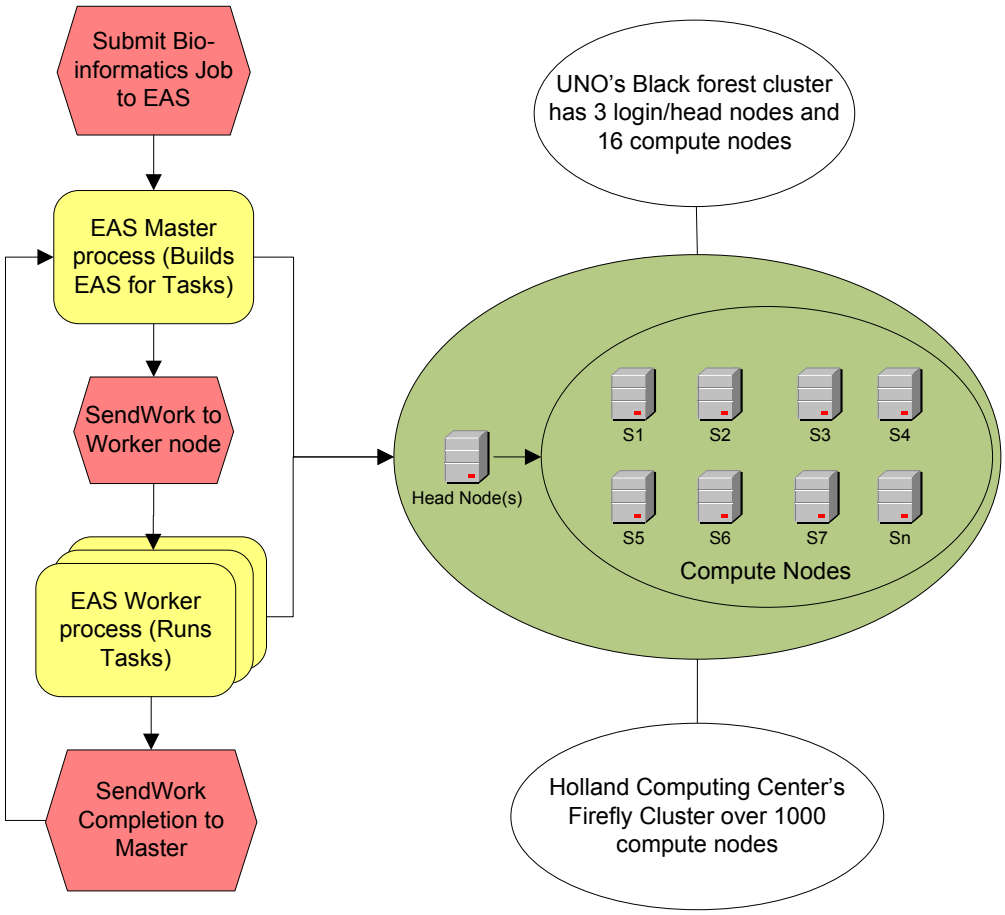

Figure 21: Process Flow Diagram for EAS Program

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.

4.4 Implementation and Results for BLAT in HPC

A key contribution of this dissertation 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 (Sequence and Annotations downloads, n.d.). 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 (Sequence and Annotations downloads, n.d.) (Genome Bioinformatics, n.d.). 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 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. 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 (Holland Computing Center, n.d.).

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

Experiment 1: The chart below (Figure 22) 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.

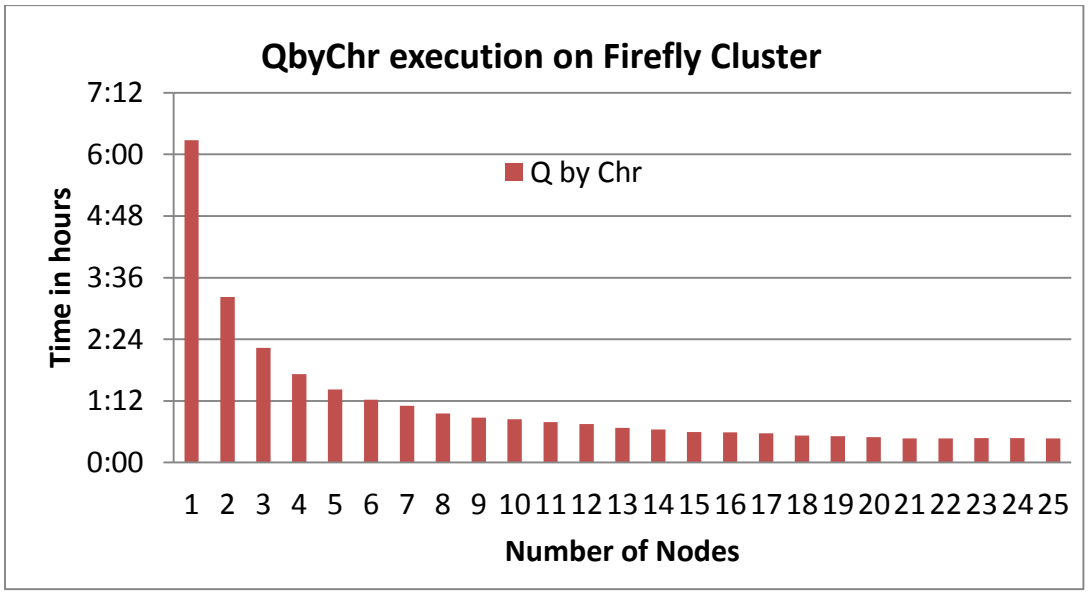

Figure 22: QbyChr execution on Firefly Cluster

Experiment 2: The chart below (Figure 23) 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.

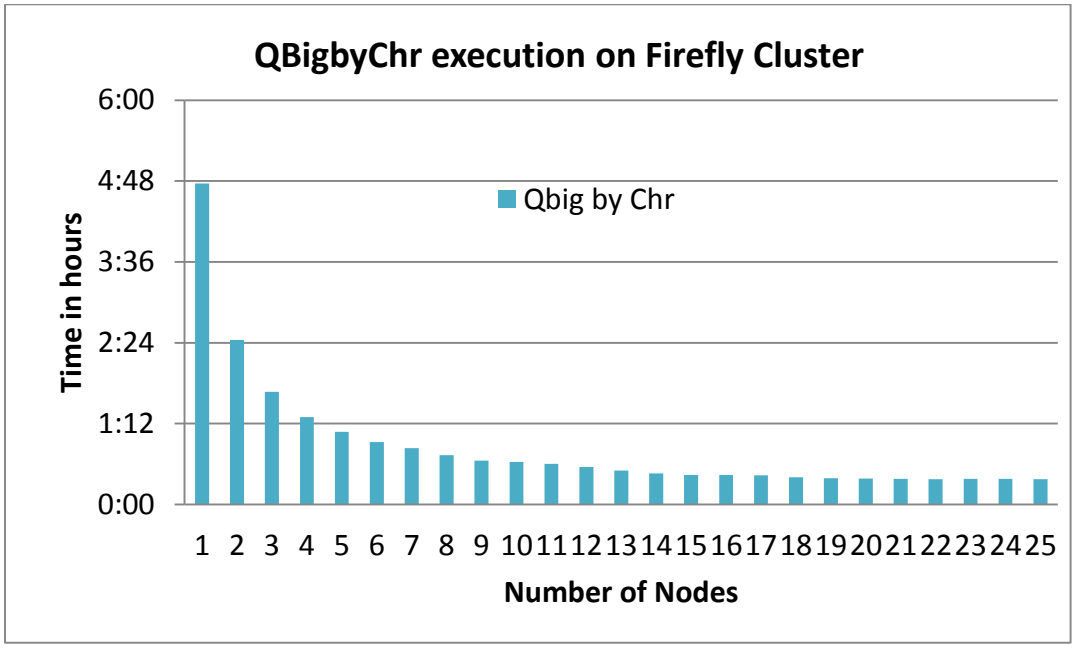

Figure 23: QBigbyChr execution on Firefly Cluster

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 (Figure 24) 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.

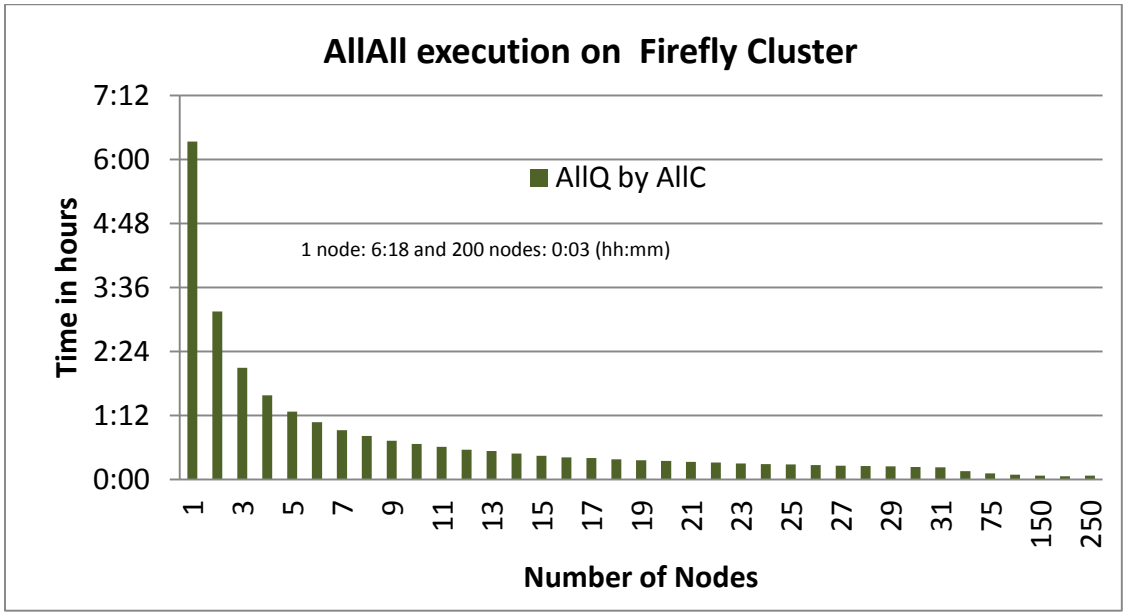

Figure 24: AllAll execution on Firefly Cluster

Comparisons:

The chart below (Figure 25) 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.

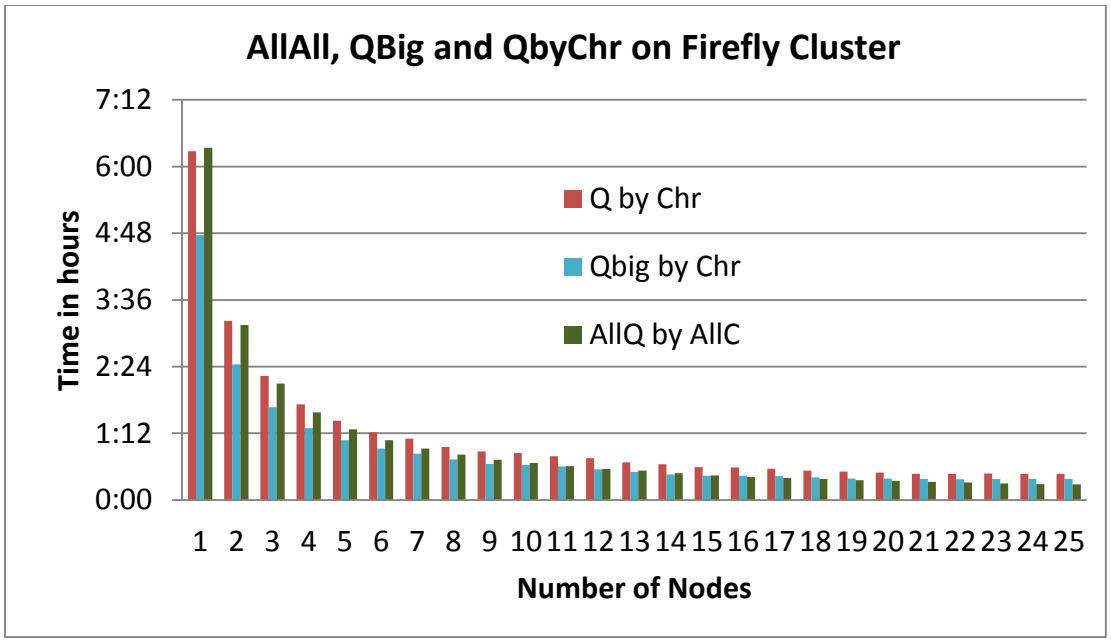

Figure 25: AllAll, QBig & QbyChr on Firefly Cluster

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 26) and 11 – 25 (Figure 27) is presented below.

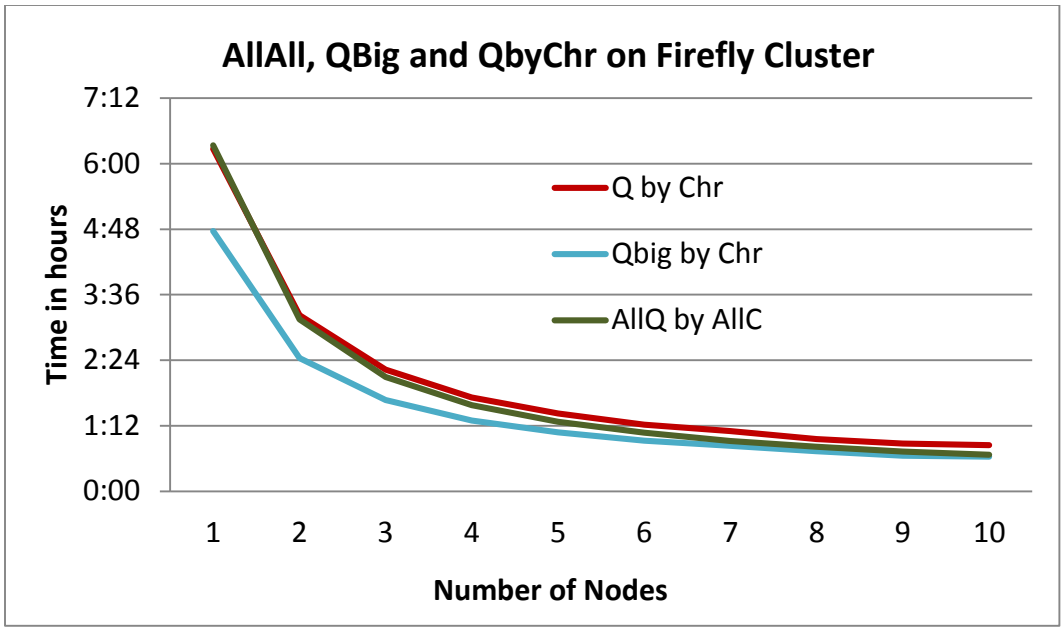

Figure 26: Details on Nodes 1 - 10

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.

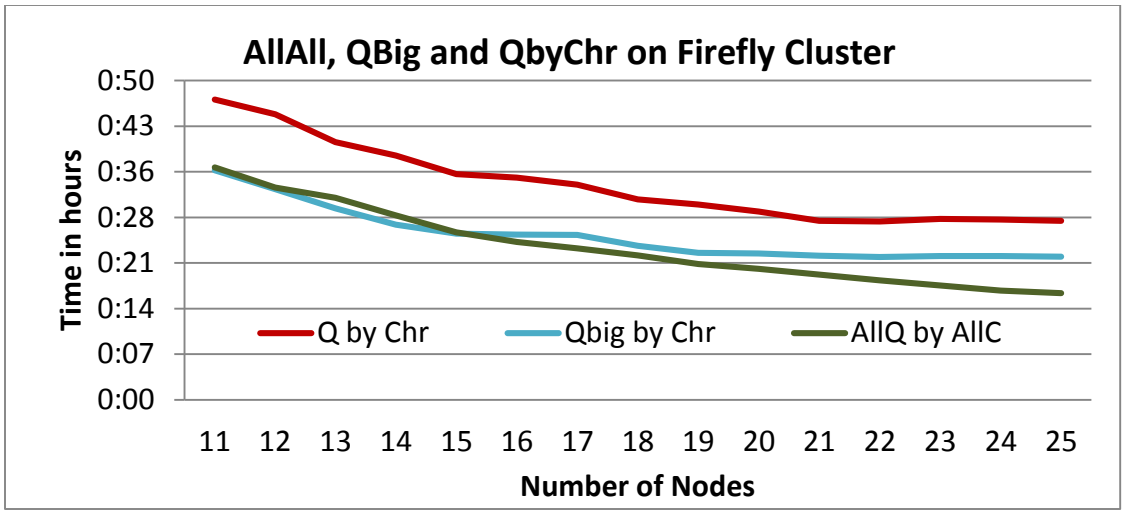

Figure 27: Details on Nodes 11 - 25

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 28. From the curves 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).

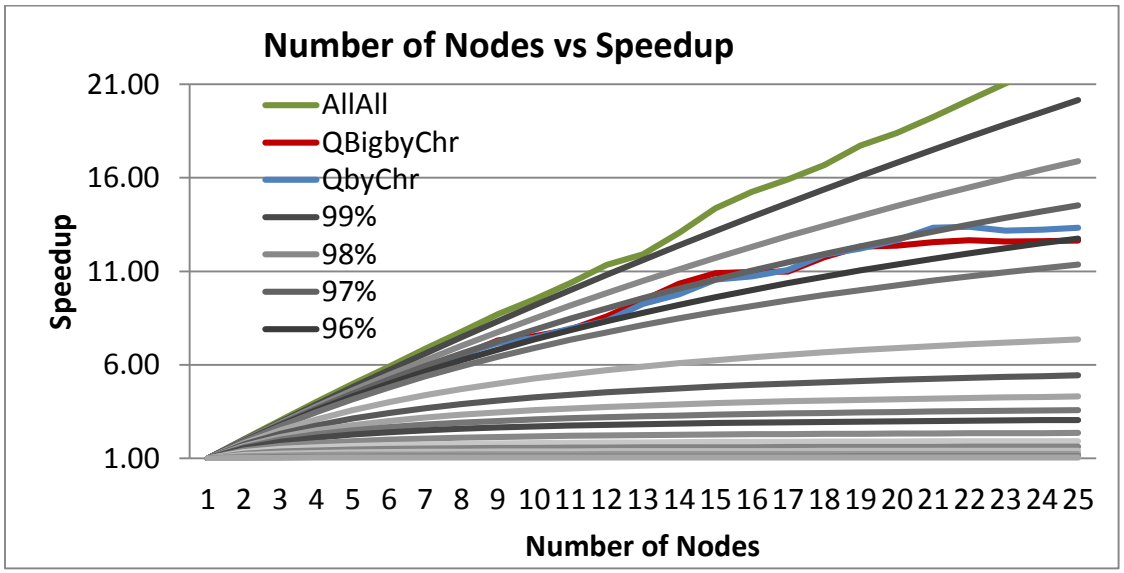

Figure 28: Number of Nodes v/s Speedup

4.5 Scheduling – Energy & Deadline aware

In this section we bring together our understanding of scheduling, High Performance Computing and our specific knowledge about BLAT in HPC. 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, 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 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). In each instance we found (Figure 29 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.

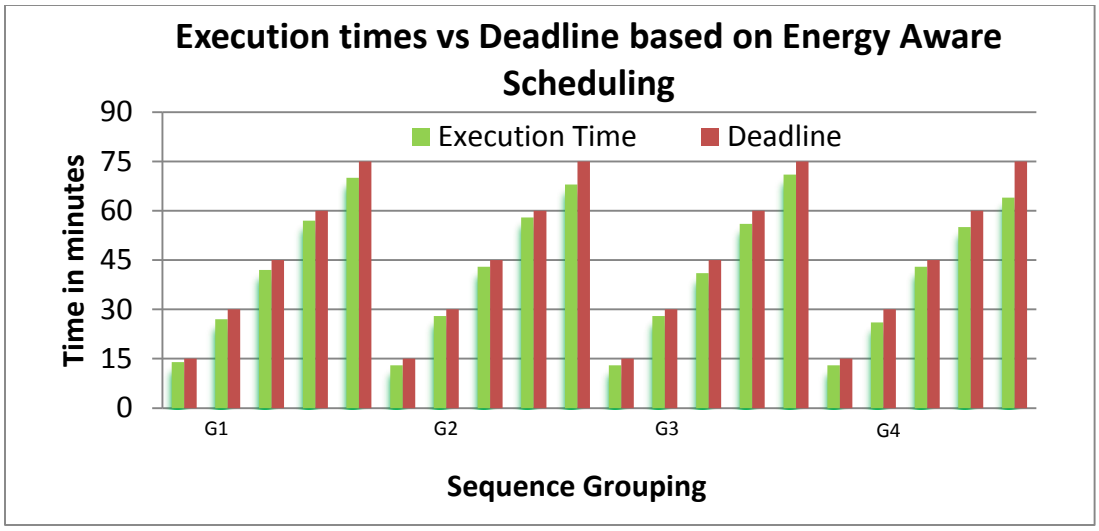

Figure 29: Scheduling – Energy & Deadline aware

The (Table 4) below 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.

Table 4: 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

4.6 Summary of Results

This section provides a research methodology for conducting research in Algorithmic Graph Theory with the focus on Energy Aware Scheduling in the computer science discipline. It addresses the most important question for conducting research, which is its "significance". The why questions are dealt with and supporting arguments are made for the importance of improving energy efficiency in general and in the specific for mobile battery operated devices. It is argued that energy management plays a vital role not only in hardware design but also in user-application, middleware and operating system design. The main research goal is the focus on techniques, specifically algorithmic ones to scale down energy needs whenever the system performance can be relaxed. The dissertation identifies the research strategy that will be followed with clearly defined stages for the

research life cycle. Each stage in the research life-cycle is considered carefully and appropriate evaluation criteria are imposed at every stage before we move to the next stage. This ensures that we build the appropriate internal and external validity factors for conducting this research at every stage of the research life cycle.

In this section we have also proposed an enhanced dynamic task scheduling algorithm using task run-queue peek technique for battery operated DVS systems that further maximize the residual charge and the battery voltage. This algorithm is expected to have a better battery performance compared to the other algorithms. Our future research will focus on using the information regarding expected execution time (EET) instead of WCET because WCET is a very conservation approach used in the Off-line Phase to schedule tasks. We intend to explore both the suggested approaches of computing EET namely conservative and risky and study their performance relative to each other.

In this section 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.

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.

Chapter 5: Run-Profile Approach towards Energy aware scheduling

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.

5.1 Enhancing our Two Step approach

We follow a simple 2-step approach as proposed in (Pawaskar & Ali, 2010). However in Phase 1 we use the different run profiles created based on the experiments conducted above. The run profiles are based on the 3 experimental approaches namely (1) Database segmentation, (2) Query merge and (3) Hybrid. Our goal is to study and examine the behavior of the EAS Model proposed in () when the first phase is seeded with differing run profiles. 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.

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.

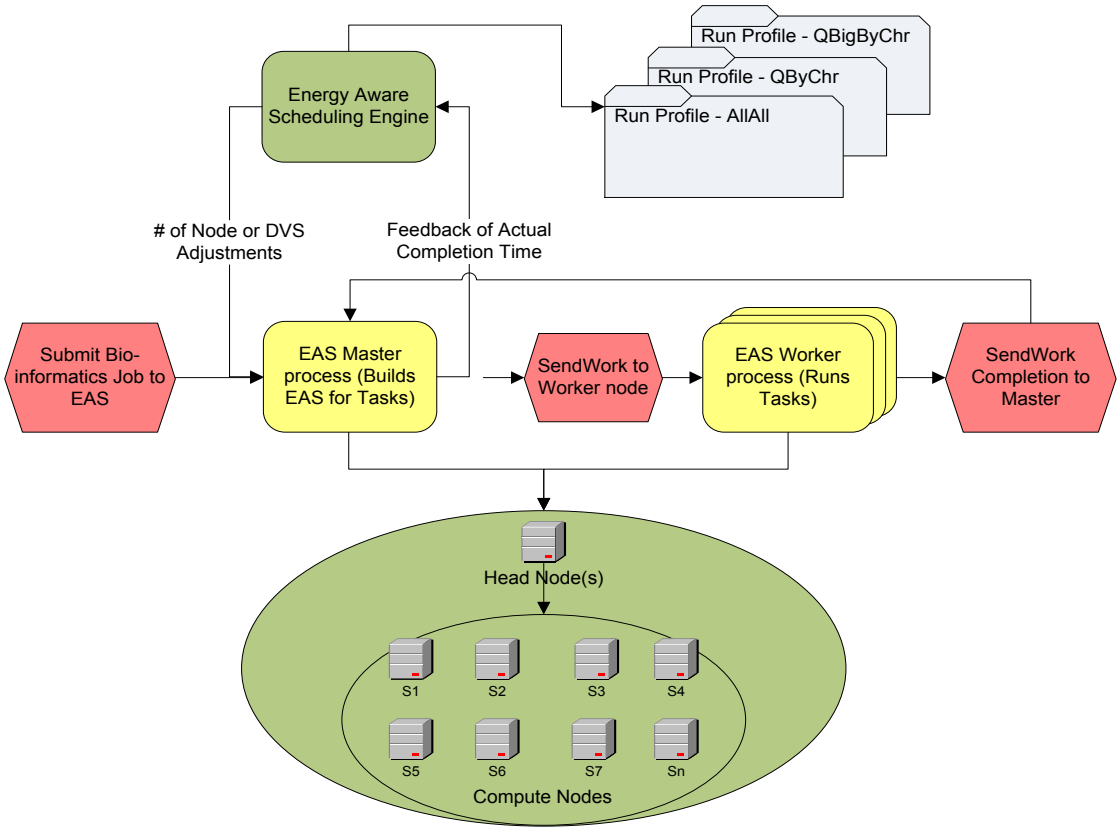

Figure 30: Process Flow Diagram for MPI Program with EAS Engine & Run Profiles

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 30 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 (Gropp, Lusk, & Skjellum, Using MPI: Portable Parallel Programming with the Message Passing Interface, Oct 1994), (Gropp, Lusk, & Thakur, Using MPI-2: Advanced Features of the Message Passing Interface, Nov 1999). 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.

5.1.1 Step 1 Enhancement.

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 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. We used the run profile generated to seed the initial scheduling decision by the EAS Engine

and then compared the results of final node adjustments. 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.

The chart below (Figure 31) 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.

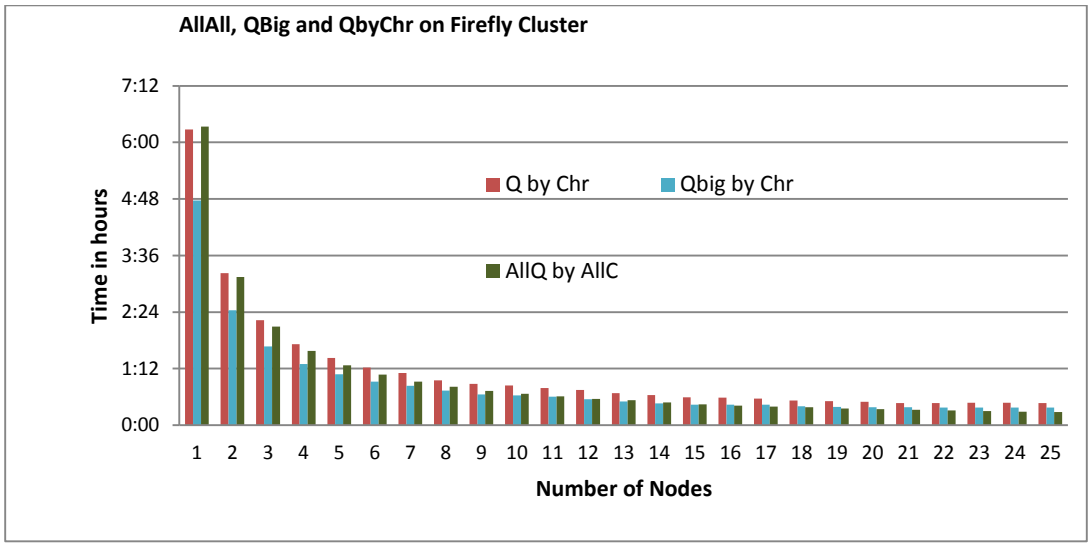

Figure 31: AllAll, QBig & QbyChr on Firefly Cluster

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.

5.1.2 Step 2 Enhancement

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 (Pawaskar & Ali, 2010), each group had 5 files with varying number of sequences as shown in the table below (Table 5).

Table 5: 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.

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 (–).

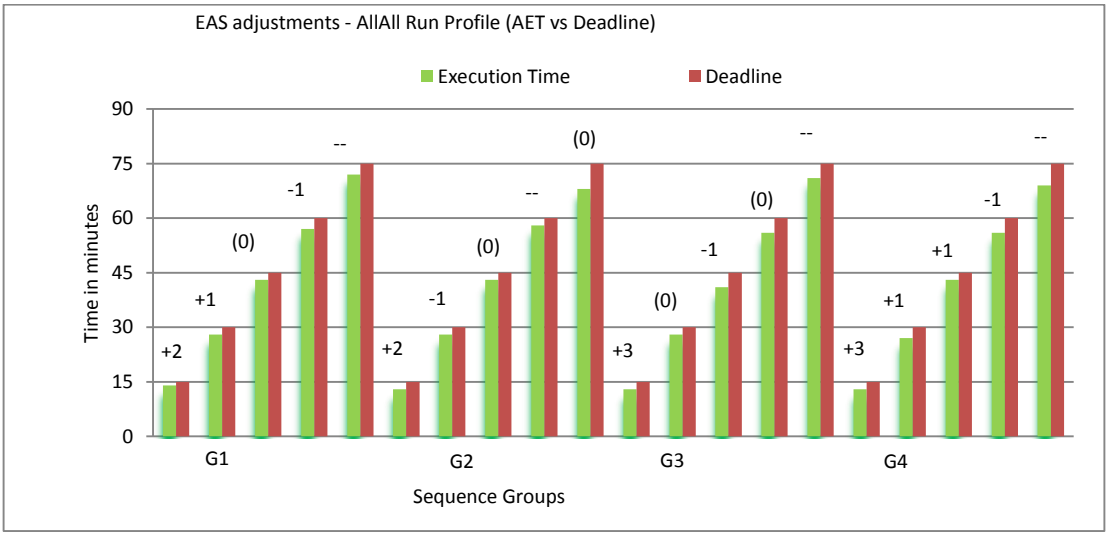

Figure 32: EAS Engine – AllAll Profile Adjustments

We ran the experiments using the three different run profiles in step 1 of the algorithm. When the AllAll run profile was used (Figure 32) in all instances we found 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.

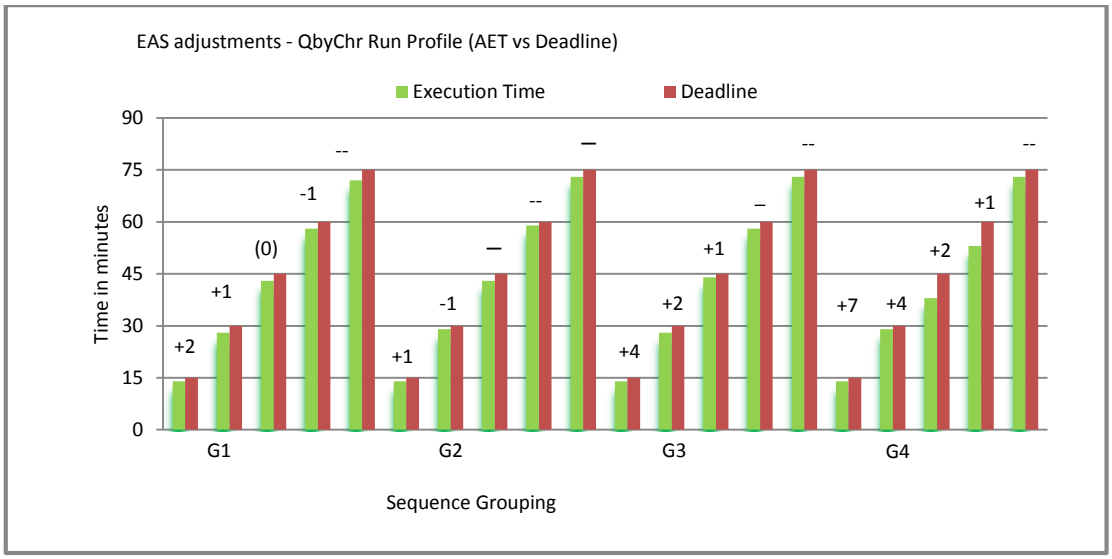

Figure 33: EAS Engine – QByChr Profile Adjustments

The Figure 33 above for the "QbyChr" run profile suggests that for lower deadlines more node adjustments had to be made to meet deadline than what was allocated step 1.

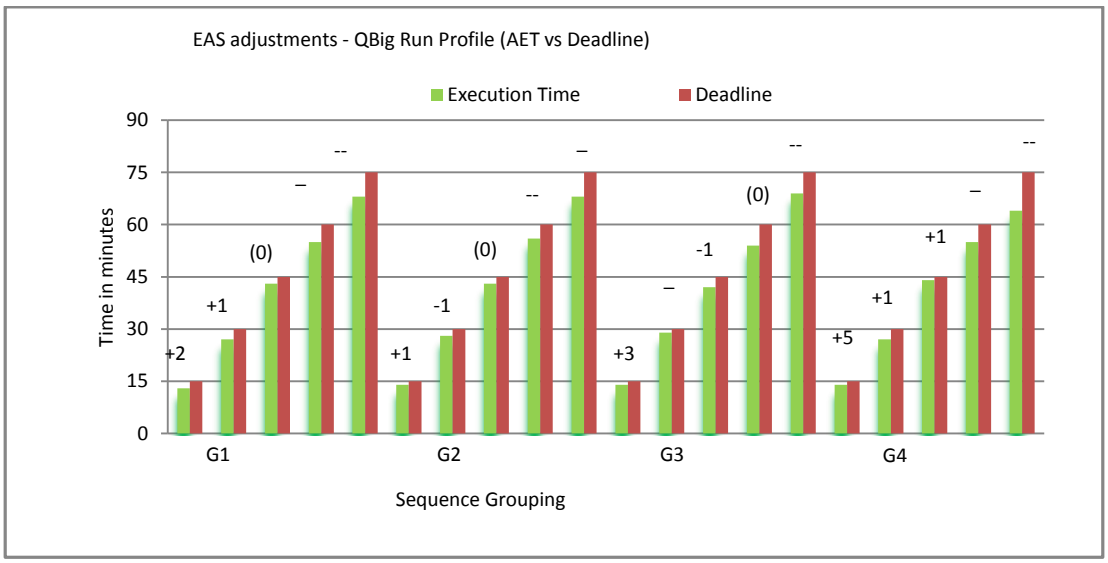

Figure 34: EAS Engine – QBig Profile Adjustments

The Figure 34 above for the "QBigbyChr" run profile also suggests that for lower deadlines more node adjustments had to be made to meet deadline than what was allocated in step 1.

When no run profile is seeded to step 1 the EAS engine defaults to using the WCET schedule. This graph is presented in (Figure 35) below. The graph shows that using WCET schedule we have significantly more node adjustments compared to using a run profile.

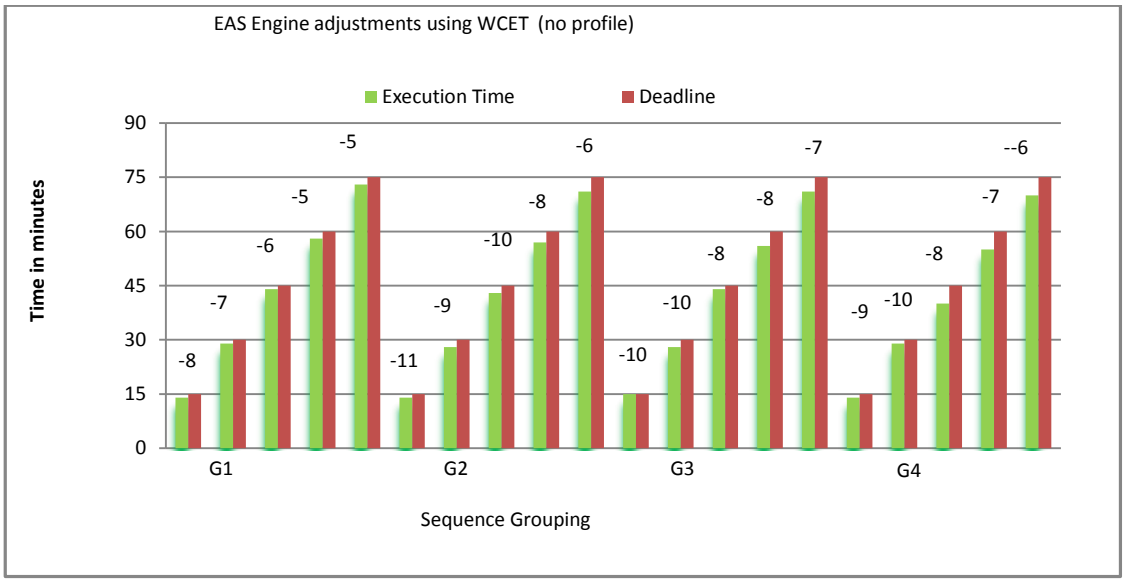

Figure 35: EAS Engine – No Profile Adjustments

The Table 6 below shows the node adjustments made by the EAS Engine to meet the deadline depending upon which run profile was chosen in step 1, 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.

Table 6: Nodes used to meet deadline based on Profile

Groups	AllAll Adjustments	QBigbyChr Adjustments	QbyCht 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)

5.2 Summary of Results

In this section 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. Our experiments suggest that the choice of run profile in step 1 of the algorithm has an impact on the overall performance of the algorithm because it impacts the number of adjustment the algorithm has to make to meet deadlines. Each adjustment has an associated overhead which impacts the energy optimization. Clearly 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.

Chapter 6: An EAS Application for Assembling Short Reads in HPC

6.1 Assembling of Next Generation Sequencing Data

Since its inception in the mid 2000's, next generation sequencing has produced massive amounts of genetic information, making a large impact on numerous research fields. As next generation sequencing systems and centers become more readily available, massively parallel sequencing has become the cornerstone of many diverse research endeavors, including those such as cancer transciptome and gene expression analysis studies (Meyerson, 2010) and microbiomics (Qin, 2010). Next generation sequencing technologies are capable of producing millions to even billions of short reads per run. Individually each read represents only a fraction of the original genome and provides no information in itself. However, sequencing reads are produced at a high coverage of the original genome such that many of these reads overlap with one another. Relationships between overlapping sequence reads assist the identification of fragments that are consecutive within the genome, allowing the recursive merging of these overlapping sequences until long stretches of contiguous genetic data, known as contigs, are recovered.

The assembly of next generation sequencing data still remains a challenging task due to the massive size of read datasets, short read lengths, and underlying target sequence composition such as repeat content. The assembly of short reads produced by these devices is a critical and computationally intensive process. Fortunately, many steps of this process are good candidates for parallel computing. The parallel implementation of the read overlap detection phase of assembly is relatively straightforward. High

performance computing has been successfully applied to help reduce the computational burden of detecting read overlaps in large datasets (Huang, 2003). However, straightforward parallel applications developed for overlap detection could achieve an unnecessary high degree of parallelism at the expense of significant energy consumption.

6.2 Assembly Algorithm Overview

Merge and Traverse assembler follows the traditional overlap-layout-consensus paradigm that has been successfully employed by various assemblers (Huang, 2003) (Sommer, 2007) (Myers, A whole-genome assembly of Drosophila, 2000). Our algorithm assembles reads into contigs in three stages: 1) overlap detection and alignment, 2) graph construction and manipulation, and 3) consensus sequence generation by multiple alignment (Miller, 2010).

6.2.1 Overlap Detection and Alignment

The Merge and Traverse algorithm uses short k-mer words to seed overlaps between reads. These short seed matches are extended into full alignments using dynamic programing. The overlap relationships found during the overlapping phase are placed into two categories by the assembly algorithm. The first type of overlap that the assembly algorithm considers is the dovetail overlap. The dovetail overlap occurs when the reads align such that they form a suffix-prefix relationship as shown in Figure 36.

The second type of overlap that the assembly algorithm considers is the containment overlap. The containment overlap occurs when the sequence of one read is fully contained in another read. For the purpose of simplifying the overlap graph in subsequent assembly phases, our algorithm disregards containment overlap relationships. Each read

that is contained in one or more other reads is mapped to a suitable representative read using a clustering approach detailed in section four.

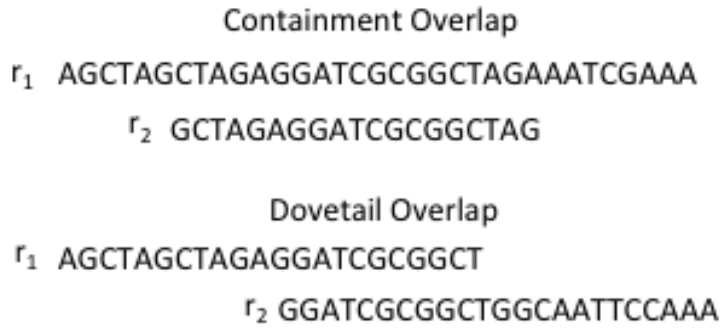

Figure 36: Read Overlaps

6.2.2 Graph Construction and Manipulation

The second phase of the assembly process builds an overlap graph using high quality dovetail overlaps between the remaining representative reads. In this graph theoretic model, each node represents a sequencing read. An edge joins two nodes if their corresponding reads overlap. As shown in example below reads map to nodes and overlaps map to edges; each edge is assigned a weight representing the length of the overlap shared between the reads.

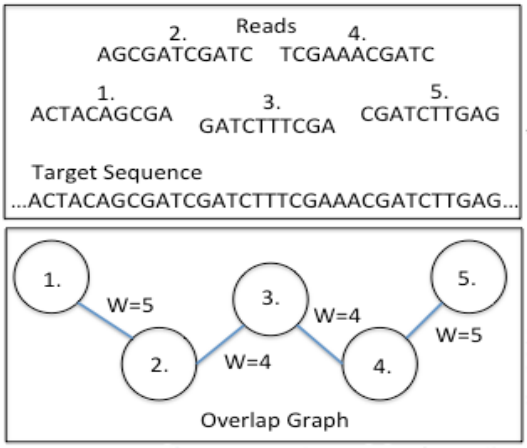

Figure 37: The overlap graph.

After graph construction is complete, the algorithm preforms transitive reduction of the graph (Myers, The Fragment Assembly String Graph, 2005) revealing non-branching paths that likely correspond to unique regions of the target sequence being assembled. The algorithm identifies and merges these non-branching paths into super-nodes in the overlap graph (Figure 37). Remaining graph structural features such as dead-end paths and bubbles, where two paths start and end at a common node, are in many cases caused by sequencing error present in the read data set. The algorithm identifies this noise using a Dijkstra shortest path method. Each dead-end path that is shorter than a user-provided threshold is removed from the overlap graph. For each bubble whose component paths are shorter than the user-provided threshold, the least covered path in the bubble is removed. After graph trimming is complete, the algorithm extracts all maximal non-branching paths from the graph for use in the consensus phase of the assembly process to construct contigs.

6.2.3 Consensus Sequence Generation

In the final consensus phase, progressive multiple alignment guided by the read path layout is used to determine contig consensus sequence.

6.3 Read Overlap Detection

In this section, we provide a description of our three-step approach for read overlap detection. The first step orders a read dataset S in descending read length and partitions it into subsets. The second step maps each read that forms a containment overlap with one or more other reads to a suitable representative read following a hierarchical clustering scheme introduced by CD-Hit (Myers, The Fragment Assembly String Graph, 2005).

After clustering is complete, the final step identifies dovetail overlap relationships among the remaining representative reads.

6.3.1 Read Preprocessing

The containment clustering step of the overlap detection phase requires that the reads are sorted by descending length. First the reverse complements of an input read dataset *R* are generated to form the read set $S = (R, \overline{R})$. It then sorts *S* into descending order of length by a merge sort algorithm, and partitions *S* into *n* subsets = $\{S_0, S_1, \ldots S_{n-1}\}$ of size *m,* where *n* is specified by the user. Each read subset S_k is sorted in descending read length and the subsets are ordered such that $readLengths(S_0) \geq readLengths(S_1) \geq \ldots \geq readLengths(S_{n-1})$.

6.3.2 Containment Clustering

The initial read clustering step follows the greedy hierarchical clustering scheme introduced by the CD-hit algorithm (Li & Godzik, 2006). The longest read becomes the first representative. It is used to search for containment overlaps among the remaining reads using the exact matching and alignment methods described in the section three. If a read forms a containment overlap with the current representative and its alignment meets minimum length and alignment identity requirements, it is mapped to that representative read. The algorithm considers each read in the order of descending length. If a read is not already mapped to an existing representative, it becomes a new representative read and is used to query the remaining reads in the dataset for containment overlaps (Figure 38). In the example below we have reads r_2 and r_4 cluster to r_1 and read r_5 cluster to r_3.

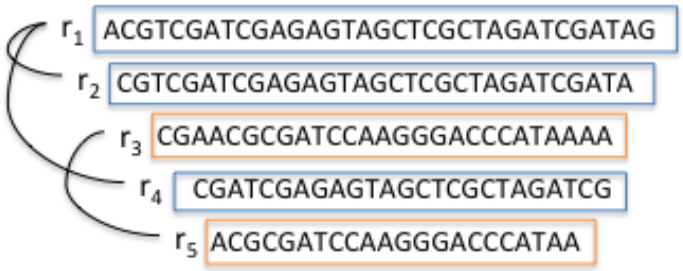

Figure 38: Containment clustering – reads r2 & r4 → r1 and read r5 → r3.

A read that has been mapped to a previous representative read but forms a containment overlap with the current representative is remapped to the current representative if its alignment identity with the current representative is greater than its alignment identity with the previous representative. After this process has completed, all read to representative mappings are recorded for use in the consensus phase of the assembly process.

6.3.3 Dovetail Overlaps

After containment clustering is complete, the remaining representative reads are used to query the read dataset for dovetail overlaps with other representative reads. The exact matching and alignment methods of section three are used to locate dovetail overlap relationships. If a dovetail overlap meets minimum alignment length and alignment identity requirements, it is recorded for use in the graph construction phase of the assembly algorithm.

6.3.4 Implementation Details

The containment clustering and dovetail overlapping steps accept two read subsets S_i and S_j as input. The subset S_i is the query dataset and the subset S_j is the reference dataset, where $i \leq j$.

To facilitate the identification of exact matches between reads, a suffix array constructed by Larsson and Sadakane's algorithm (Larsson & Sadakane, 1999) is used to index the reference dataset. In succession, each read in the query dataset is broken into all of its possible subwords of size k (denoted as k-mers). These k-mers are used to query the suffix array for exact matches. If one or more exact matches are found between the query read and a reference read indexed by the suffix array, then both reads are passed to an alignment algorithm for evaluation. The k-mers shared by the reads are chained (Ohlebusch & Abouelhoda, 2006) and the Needle-Wunsh algorithm (Needleman & Wunsch, 1970) is used to align the regions between k-mers and to align the beginning and end regions of the reads.

After the alignment of the two reads is complete, the computed overlap is evaluated by its alignment length and alignment percent identity. If the overlap does not meet the user-provided minimums for these measurements, it is not included in subsequent steps of the assembly process.

Since the containment clustering step is dependent on the read ordering, each subset S_j must be ran against each S_i as a reference dataset, where $i < j$, before it can be used as a query dataset against any other read subset. The dovetail-overlapping step is not dependent on read ordering and can accept read subsets in any order.

6.4 Parallel Implementation using the EAS Model

The input read dataset S is partitioned into *n* subsets = $\{S_0, S_1, \ldots S_{n-1}\}$ of size *m* during the initial read sorting and preprocessing step. A master thread sends each unique subset combination of size two as input to worker processors running serial versions of the

containment clustering and dovetail overlapping algorithms. The master thread manages the execution order constraints of the containment clustering step.

The EAS engine runs the pre-processor (Figure 39) on the input fasta file, the output of which is the n-split read subsets. Let us assume that the large file has m sequences, and then each of the smaller files will contain (m/n) sequences in sorted order. The files created in the pre-processing step become inputs to the EAS engine. The EAS engine runs the alignment program in a 2-step process. The first step finds the containment overlaps and the second step determines the dovetails overlaps among the remaining representative reads. The containment part of the execution is not naively parallel; the execution of certain pairs of subsets (tasks) has to be done in order, only then can dependent subsets be processed. The main process flow is shown in Figure 40 below.

6.4.1 Containment Execution – Step 1

The execution dependencies are shown in Figure 41 for the following set of containment tasks T = {(0, 0), (0, 1), (0, 2), (0, 3), (0, 4), (1, 1), (1, 2), (1, 3), (1, 4), (2, 2), (2, 3), (2, 4), (3, 3), (3, 4), (4, 4)}, where each integer represents a read subset. The tasks along the diagonal (0, 0), (1, 1) (2, 2), (3, 3) and (4, 4) are considered to be higher priority tasks because they have a greater number of child/dependent tasks.

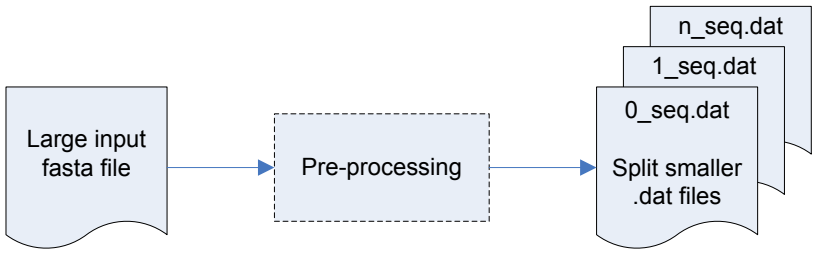

Figure 39: Pre-processing step

All other tasks have a normal priority in terms of execution. After a task gets released, meaning that all of its predecessors have been executed, it is sent to the EAS execution queue. When the task has completed executing, the EAS engine checks to see if any dependent tasks can be released for execution.

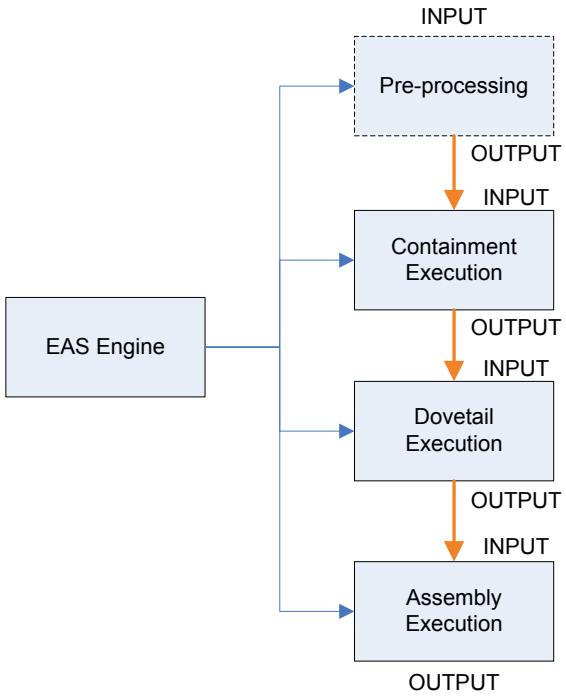

Figure 40: Process Flow Diagram

Now let us take a look at the example where we have five read subsets. When the task (0, 0) is complete, it releases all the tasks in that row which are tasks (0, 1), (0, 2), (0, 3) and (0, 4). It cannot release (1, 1) because task (1, 1) still has another dependency on (0, 1). When (0, 1) is completed, it will release task (1, 1). Completion of task (1, 1) will flag (1, 2), (1, 3), and (1, 4) but they will only be released when both (1, 1) and the tasks above them namely (0, 2), (0, 3), and (0, 4) have completed execution. This will continue until all tasks are executed. The last task to be executed will be task (4, 4) in our example. Note that the total number of tasks executed would be fifteen. This can be calculated

easily using equation one. We would like to point out that the containment phase is bounded by the number of files (in this case five). We cannot use more than five nodes at any given time due to task dependencies even though we have a total of fifteen containment tasks.

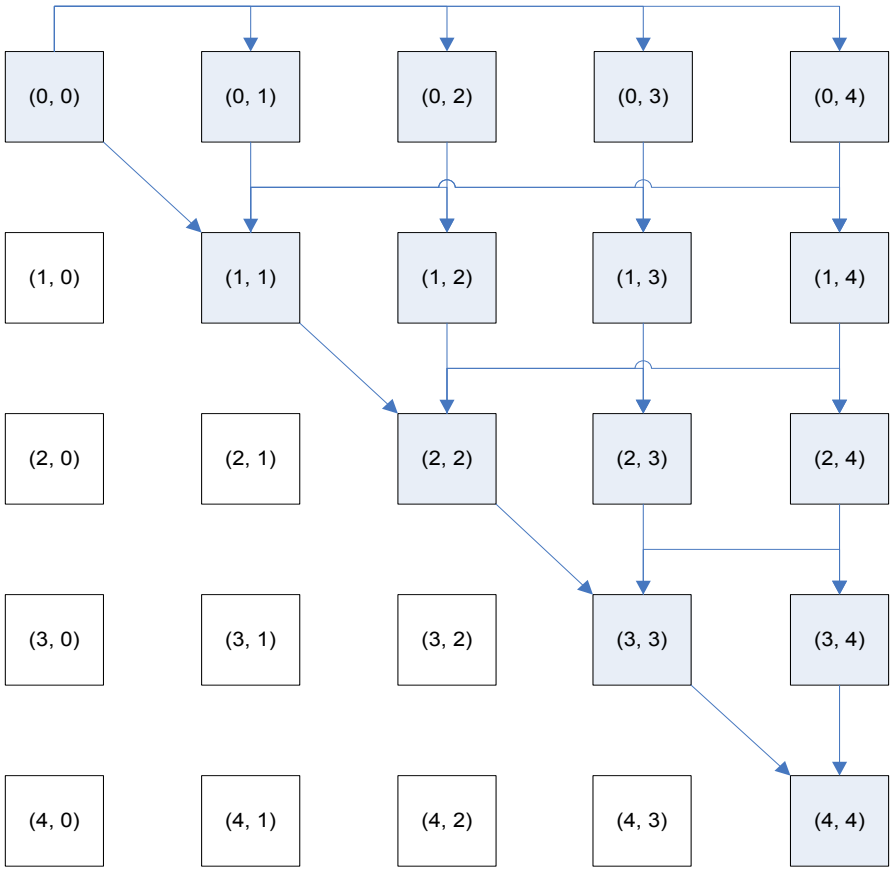

Figure 41: Execution dependencies of containment tasks

6.4.2 Dovetail Execution – Step 2

The execution dependencies of the dovetail tasks are much more straightforward than those for the containment tasks. The dovetail tasks do not have any dependencies on each other and hence can be run in a naively parallel way, allowing us to use as many processors as possible. Continuing with our previous example with fifteen tasks, we

could execute (0, 0), (0, 1), (0, 2), (0, 3), (0, 4), (1, 1), (1, 2), (1, 3), (1, 4), (2, 2), (2, 3), (2, 4), (3, 3), (3, 4), (4, 4) all at the same time during the dovetail phase.

The total number of tasks that need to be executed in each of the above steps (containment and dovetail steps) is given by the equation below, where n is the number of read subsets and T is the total number of tasks.

$$T = \frac{n(n+1)}{2} \qquad (1)$$

6.5 Implementation and Results

We downloaded *Escherichia coli W* reads produced by the 454 Titanium technology from the NCBI (NCBI Database, 2010) sequence read archive (accession no. SRR060736 and SRR060737, made public by JCVI). The sequences were trimmed to remove adaptors. The final result was 337,294 trimmed reads. For our experiment in the pre-processing step we decided to split these into 16,866 sequence reads per file, i.e. read subset (except for the last file which contained 16,814 reads). This resulted in 40 files and a total of 674,588 reads. (The preprocessing step generates the reverse complement of each read.) We then used the EAS engine to run the assembly algorithm using 1 to 31 nodes. For our experiments we used the HPC environments available at UNO (University of Nebraska at Omaha). We initially start out with the Blackforest cluster (16 nodes) (Blackforest Computing Cluster, n.d.), and then move to a true commercial strength HPC named Firefly cluster (1100 nodes) at the Holland Computing Center (Holland Computing Center, n.d.).

Figure 42 shows the execution time of the algorithm in seconds versus the number of nodes used for each run. It shows that after 11 to 12 nodes we do not see any significant performance gain. Along with the total execution time, we captured the average execution time per worker node and the overhead. We find that as we increase the number of nodes the overhead curve follows the execution time curve.

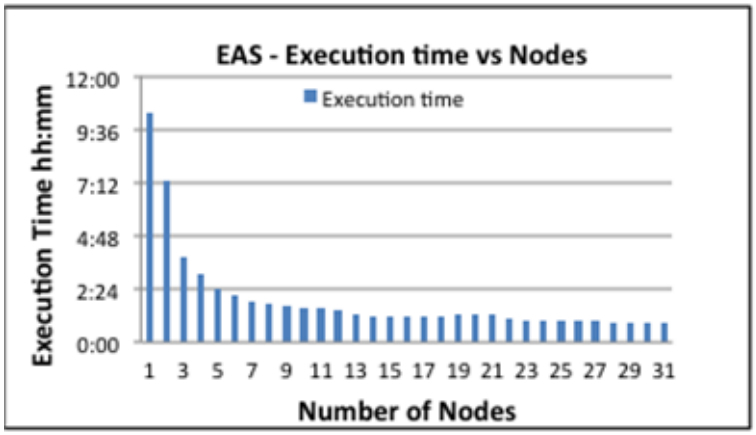

Figure 42: EAS - Execution time v/s Nodes

It is important to note that in a HPC a significant portion of the master process's work is distributing the tasks and managing the task dependency among the worker processes along with handling of the communication between master and worker processes. This is clearly depicted in Figure 43.

It is important to note that given the nature of the task dependencies in the containment phase not all nodes are working all the time, and hence we see a smaller overall curve for the average worker time per node. This leads us to ask the question, "How parallelizable is the program?" For the purpose of answering this question we plotted the program speedup against the number of nodes and integrated this curve with a plot of Amdahl's law in Figure 44. Amdahl's law is defined by the formula:

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

As $N \to \infty$, 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 $(1-P)$. A great part of the craft of parallel programming consists of attempting to reduce $(1-P)$ to the smallest possible value. We can conclude that the overlap detection algorithm of the Merge and Traverse assembler has a speedup between 20 - 25 times (which is between 90% - 95% parallelizable).

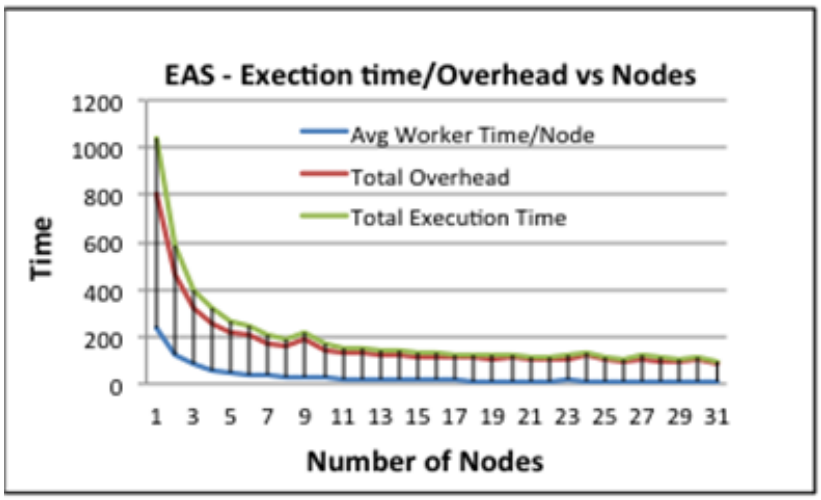

Figure 43: EAS - Execution time/Overhead v/s Nodes

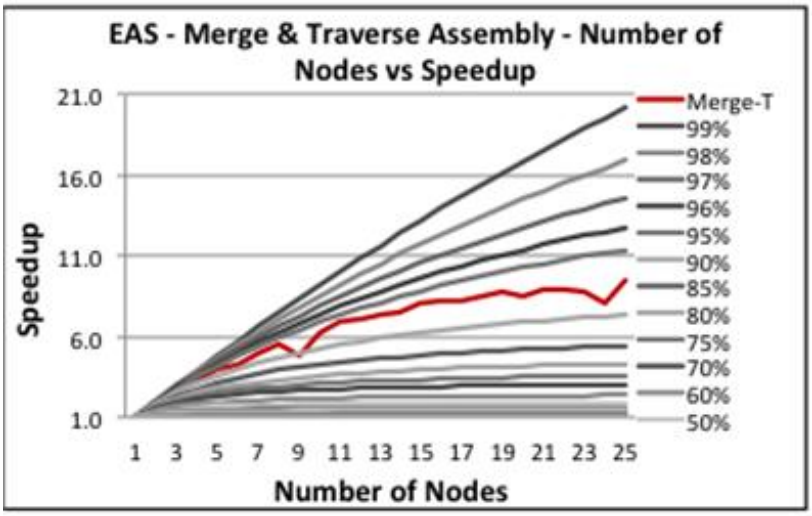

Figure 44: EAS - Merge & Traverse Assembly - Nodes v/s Speedup

Next we set up experiments to see if the EAS engine would be able to dynamically adjust the number of nodes to meet a given deadline. We used four groups of read datasets generated from SRR060736 and SRR060737. Each group was partitioned into a different number of files as shown in Table 7.

Table 7: Read Subset Group used for Analysis

Group	Number of Files	Number of Sequences
G1	5	84330
G2	10	168660
G3	15	337320
G4	20	674588

Each group of files was ran against five different deadlines (30, 60, 90, 120, and 150 minutes). Each of these jobs was assigned a starting number of nodes by the EAS engine based on the run profile/speedup curve. As the tasks were completed, variances between EET (Expected Execution Time) and AET (Actual Execution Time) 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 (–). The experimental results (Figure 45) showed that the EAS engine was able to dynamically adjust nodes to minimize energy utilized while meeting the deadlines.

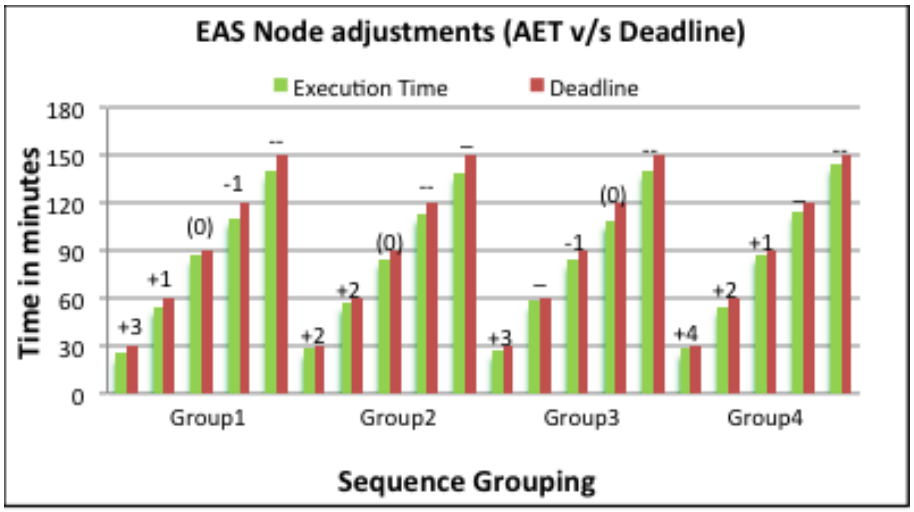

Figure 45: EAS Engine - dynamic node adjustments

6.6 Summary of Results

Based on the results we can clearly observe that given a deadline we can choose the appropriate number of nodes to run the overlap detection phase of the assembler on based on our new understanding of the run-profile we just produced. This will allow us to apportion just enough nodes to meet the deadline thus maximizing the objective of performance with minimum energy utilization. We also observed that with a smaller number of nodes we have larger gains in performance and above a certain number of nodes the performance gain is only modest at best. In fact as we add additional nodes our communication costs and related overhead is higher.

Clearly different bioinformatics applications and algorithms will have different run profiles and understanding each one of them will allow us to best assign the appropriate number of nodes to meet a given deadline. It was also important to see how the number of read subsets impacted the performance/energy criterion. Our experiments suggest a bowl shaped curve when we varied the number of files for the same number of nodes. Clearly there must be some optimum value for the number of files for each set.

This section highlights the importance of understanding the degree of parallelism for the program, which is done by establishing the run profile/speedup curve. The EAS engine uses the knowledge from the run profile to make intelligent and dynamic decisions about number of nodes to use to minimize energy utilization and still provide necessary performance. Clearly it is no longer sufficient to simply run a program in a HPC environment. It is important and essential to understand the data, its characteristics, and the application domain to build a parallel program that is energy aware.

In designing these experiments, we have several parameters we could study and the relationship between them. These parameters are (1) Number of files; (2) Number of sequences per file; (3) Number of nodes used and (4) Average sequence length. In this section we have only looked at number of nodes used as a parameter for our experimental design. In the future we plan to investigate how adjusting the different tuning parameters such as number of files, number of sequences per file, number of nodes impacts the performance and energy efficiency. We also plan on including the pre-processing step and final assembly as part of the EAS processing. Our main motivation is to move this from a simple speedup to the realm of energy awareness. Our EAS model for the purposes of the experiments conducted calculated energy as a function of resources used

in this case number of nodes. The energy function could be made more complex; we leave that for a future study.

Chapter 7: Towards an Energy Aware Cloud (A Simulation)

7.1 Energy Aware Cloud based on Energy index

The obvious next step in the evolution of the EAS Model is to apply this to the cloud. But what is the cloud? It is nothing more than a bunch of Datacenters, Each of these datacenters can be looked upon as a High Performance Computing environment, essentially as a computing resource. We can then apply our profile based approach for a known application such as BLAT and using a known datacenter in our case the Holland Computing Center as a baseline (energy index of 1) we can then run the same application against another datacenter and if the results are returned faster we assign a relative high number depending on how fast we got the result set or a lower number depending on how slow the result set was delivered. Finally by knowing the energy index for a datacenter we can choose to schedule our tasks across datacenters depending on the necessary deadline. We will examine this model by performing simulation experiments on the same dataset we used before.

7.2 Cloud Computing – Lifting the Veil

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, It is **Commoditised -** 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** – They hide 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** - Pay for use and as needed, which allows for scale up and down in capacity and functionalities as needed (Amazon Elastic Compute Cloud (Amazon EC2), 2013).

7.2.1 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 46 (Amazon Elastic Compute Cloud (Amazon EC2), 2013) and (Microsoft on Cloud Computing, 2013).

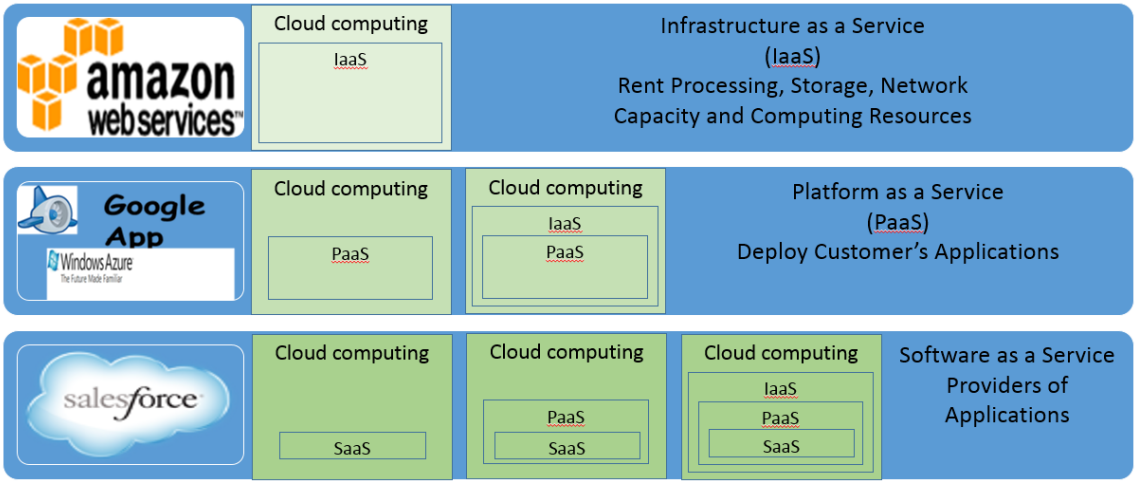

Figure 46: Cloud Computing Models

7.2.2 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 47).

	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 47: Cloud Computing Service Layers

7.2.3 Cloud Deployment Models

Cloud can also be characterized based on how they are deployed and used. The most well-known deployment models are the public cloud and private cloud. The Figure 48 shows the different cloud deployment models in use (Google Cloud Platform, 2013).

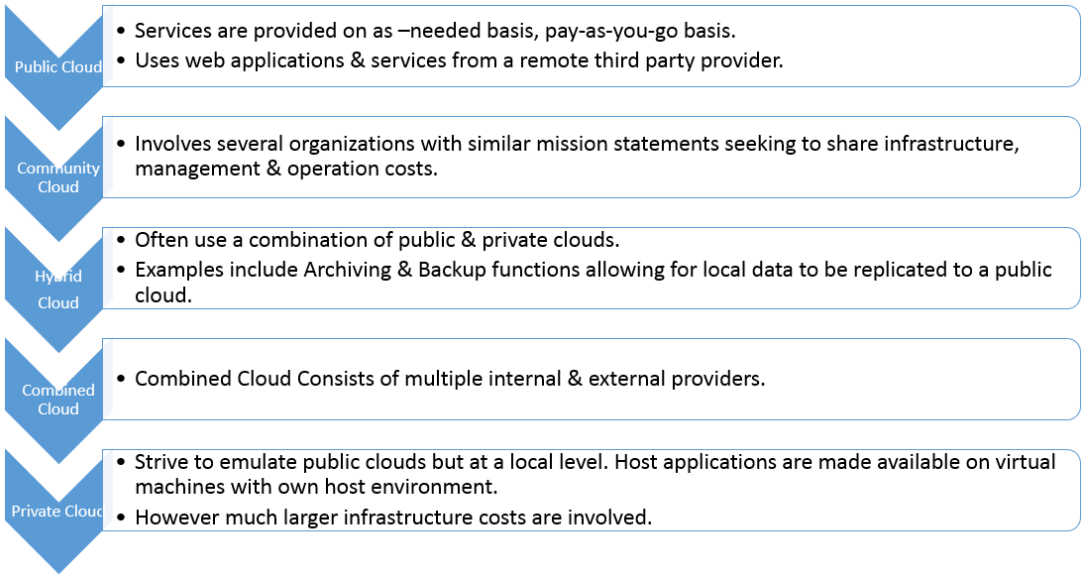

Figure 48: Cloud Deployment Models

7.2.4 Cloud Service – Opportunities and Challenges

Cloud Computing provides us with opportunities and challenges.

Opportunities:

- It enables services to be used without any understanding of their infrastructure.
- Cloud computing works using economies of scale:
- Cost would be by on-demand pricing.
- Data and services are stored remotely but accessible from "anywhere".

Challenges:

- Use of cloud computing means dependence on others and that could possibly limit flexibility and innovation.
- Security could prove to be a big issue. It is still unclear how safe out-sourced data is and when using these services ownership of data is not always clear.
- There are also issues relating to policy and access. If your data is stored abroad whose policy do you adhere to? What happens if the remote server goes down? There have been cases of users being locked out of accounts and losing access to data.

7.3 Why – Simulation Model?

So why did we build this simulation app. The objective was to help answer some questions regarding expensive resources such as Clusters by running simulations which return results quickly to facilitate better decision making on where to send jobs and what

deadline, how many number of nodes to use, what energy level to run at, what costs are associated with a job, etc.

1) In the real world it is difficult to actually run "what if" scenarios on real clusters which are in production because of scarcity of resources and the potential costs and time constraints associated with such activities.
2) Customers can run basic scenarios to see if the deadlines they are providing can be met based on past run profiles.
3) Most customers what to get their results fast, but when these deadlines have costs associated with them, they can make better informed decisions about their deadline settings and relax these if necessary. Our simulation will help provide these type of analytics for better decision making.
4) Customers can check to see if they can run their application within a given deadline on a specific cluster for the given data set.
5) Customers can run their applications on different clusters to see what resource costs they may incur on each specific cluster given the resource and energy usage.
6) Customers can adjust the availability of cluster and make decisions where they want to send their job load based on job completion.
7) The above information can also be used by Cluster operators to run simulations per customer to see how the availability of their cluster impacts their customer's decision making and also measure potential revenue loss or gain.
8) Customers can also run scenarios to see whether they should use a single cluster or distribute the work load across multiple clusters to meet specific deadline requirements.

9) Customers can also check the above scenarios to see which of the above single cluster or distributed cluster option is most cost effective.
10) Customers can also check the above scenarios to see which of the above single cluster or distributed cluster option is most efficient.
11) Cluster operators can also run "what if" scenarios to see if increasing the energy index of their Custer may have a potential impact on revenue based on number of additional jobs they might get at a certain energy index and whether the costs of increasing the energy index are justified based on opportunity costs.
12) Cluster operators can also run "what if" scenarios to see if increasing the availability of their Custer may have a potential impact on revenue based on number of additional jobs they might get at a certain availability level and whether the costs of increasing the availability are justified based on opportunity costs.
13) Customers can run scenarios to see if data split or merge for their application offers any cost and/or efficiency benefits.

7.4 The Simulation Program

Currently there is no mechanism to run tasks across multiple clusters in the cloud. We understand that this is a significant challenge. We also realized that there was no cloud simulation package available that would meet our needs. Hence we decided to write our own cloud simulation package that would allow us to use the EAS Model and also allow us to tailor the simulator to help answers questions such as can deadlines be met on certain clusters, cluster availability, energy-index and ROI. The simulation program is written in Java using the Eclipse IDE. The program itself consists of a random run generator and the main simulation run. The random run generator was used to generate

our test run data for our experiments, which are discussed in detail later in this chapter. In order to run the main simulation you need 2 input files, one is an init.xml which is used to initialize your cloud and the second is the run.xml file in which you define the program you want to run. The program can be run will command line arguments. Running the program with the "help" command will display the program usage as shown below.

```
Usage: Following arguments are for Simulation Run program
     : runType=(Single, Distributed, Mixed)
     : numRuns=(any number)
     : randomNumStart=(any number)
     : randomNumEnd=(any number greater than randomNumStart)
     : runFile=(Run File Name)
     : outputFile=(Output File Name)
     : initFile=(Init file Name for the Cloud)
     : generate=(0 or 1 - if you want to generate a random run file)
     : simulationRun=(0 or 1 - if you want to run simulation)
```

Figure 49: Simulation Program Usage

An example command line to run the random run generator would be something like "**generate=1 numRuns=100 runType=Single randomNumStart=10 randomNumEnd=5000**". An example command line to run the main simulation would be something like "**simulationRun=1 runType=Single initFile=init.xml runFile=run.xml outputFile=runResults.csv**".

The class diagram for the simulation objects is shown in the Figure 50 below.

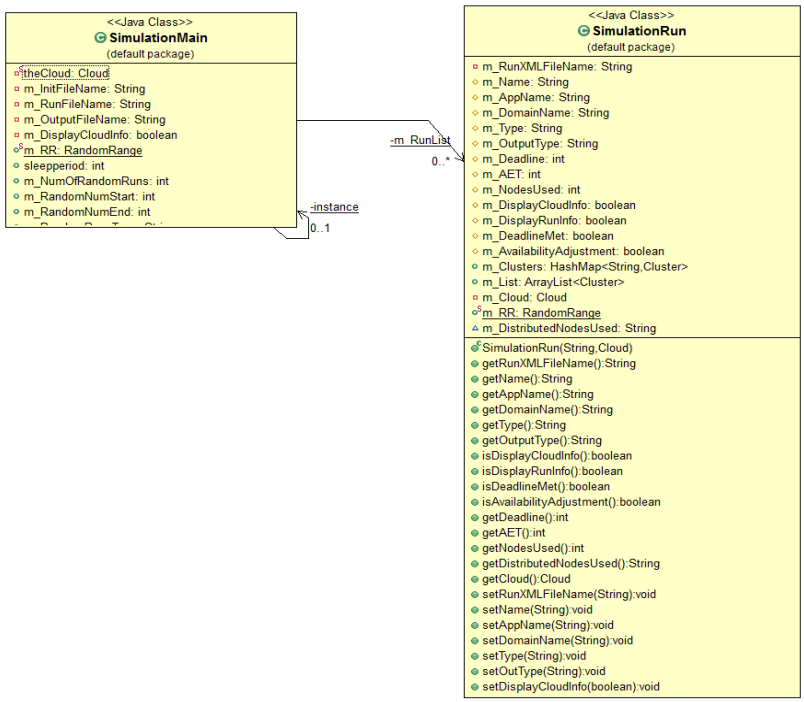

Figure 50: Class diagram for Simulation Main & Run

The class diagram for the Cloud is shown below Figure 51.

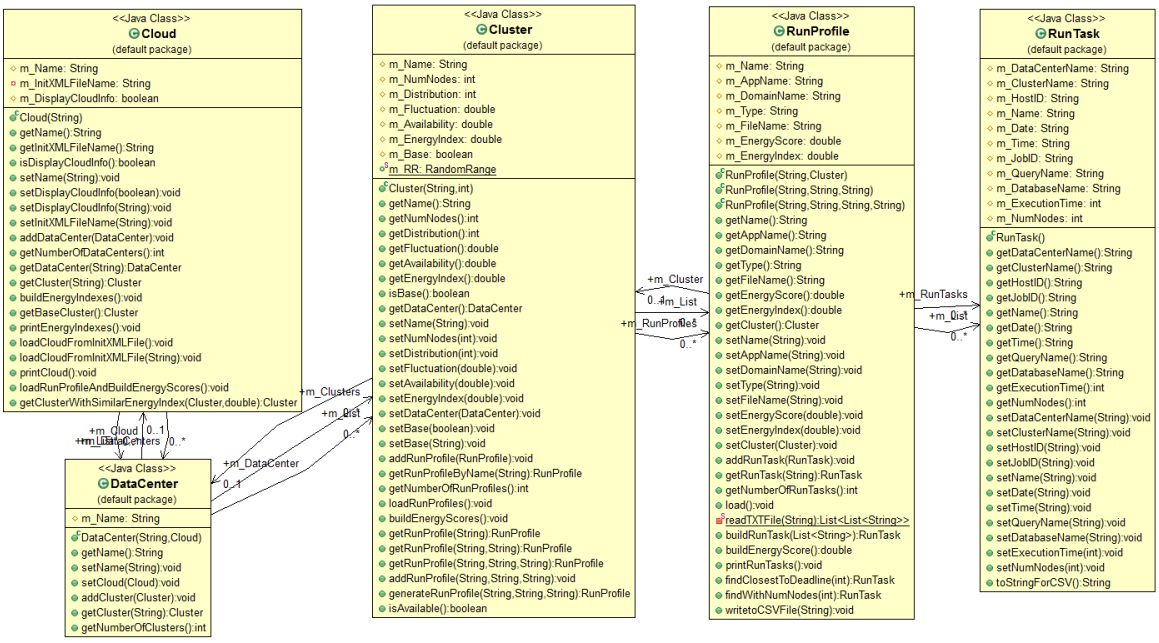

Figure 51: Class diagram for the Cloud & Run profile

The cloud is initialized using an init.xml file which is an XML with an associated XML Schema definition. A sample XML file and the Schema are shown below (Figure 52 and Figure 53).

```
<?xml version="1.0" encoding="UTF-8"?>
<Simulation xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="SimulationInit.xsd"
xsi:noNamespaceSchemaLocation="SimulationInit.xsd">
    <Cloud name="The Simulation Cloud" displaycloudinfo="true">
        <DataCenter name="Holland Computing Center">
            <Cluster name="FireFly" numnodes="4096" energyindex="1.0" availability="100" base="true">
                <RunProfile appname="BLAT" type="QAll" domain="BioInformatics" filename="BioInformatics-BLAT-QAll.csv"></RunProfile>
                <RunProfile appname="KAlign" type="QBig" domain="BioInformatics" filename="BioInformatics-KAlign-QBig.csv"></RunProfile>
            </Cluster>
            <Cluster name="Sapling" numnodes="32" energyindex="0.5" availability="90">
                <RunProfile appname="BLAT" type="QMerge" domain="BioInformatics" filename="BioInformatics-BLAT-QMerge.csv"></RunProfile>
            </Cluster>
        </DataCenter>
        <DataCenter name="Other Computing Center">
            <Cluster name="Cluster1" numnodes="8192" energyindex="3.5" availability="80">
                <RunProfile appname="BLAT" type="QBig" domain="BioInformatics" filename="test.csv"></RunProfile>
            </Cluster>
            <Cluster name="Cluster2" numnodes="2048" energyindex="1.5" availability="50">
            </Cluster>
        </DataCenter>
        <DataCenter name="Hesham's Computing Center">
            <Cluster name="MyCluster1" numnodes="8192" energyindex="3.5" availability="10">
                <RunProfile appname="BLAT" type="QBig" domain="BioInformatics" filename="test.csv"></RunProfile>
            </Cluster>
        </DataCenter>
    </Cloud>
</Simulation>
```

Figure 52: Sample Cloud initialization XML file

```
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema" attributeFormDefault="unqualified" elementFormDefault="qualified">
 <xs:element name="Simulation">
  <xs:complexType>
   <xs:sequence>
    <xs:element name="Cloud">
     <xs:complexType>
      <xs:sequence>
       <xs:element name="DataCenter" maxOccurs="unbounded" minOccurs="0">
        <xs:complexType>
         <xs:sequence>
          <xs:element name="Cluster" maxOccurs="unbounded" minOccurs="0">
           <xs:complexType mixed="true">
            <xs:sequence>
             <xs:element name="RunProfile" maxOccurs="unbounded" minOccurs="0">
              <xs:complexType>
               <xs:simpleContent>
                <xs:extension base="xs:string">
                 <xs:attribute type="xs:string" name="appname" use="optional"/>
                 <xs:attribute type="xs:string" name="type" use="optional"/>
                 <xs:attribute type="xs:string" name="domain" use="optional"/>
                 <xs:attribute type="xs:string" name="filename" use="optional"/>
                </xs:extension>
               </xs:simpleContent>
              </xs:complexType>
             </xs:element>
            </xs:sequence>
            <xs:attribute type="xs:string" name="name" use="optional"/>
            <xs:attribute type="xs:short" name="numnodes" use="optional"/>
            <xs:attribute type="xs:float" name="energyindex" use="optional"/>
            <xs:attribute type="xs:float" name="availability" use="optional"/>
            <xs:attribute type="xs:string" name="base" use="optional"/>
           </xs:complexType>
          </xs:element>
         </xs:sequence>
         <xs:attribute type="xs:string" name="name" use="optional"/>
        </xs:complexType>
       </xs:element>
      </xs:sequence>
      <xs:attribute type="xs:string" name="name"/>
      <xs:attribute type="xs:string" name="displaycloudinfo"/>
     </xs:complexType>
    </xs:element>
   </xs:sequence>
  </xs:complexType>
 </xs:element>
</xs:schema>
```

Figure 53: Cloud initialization XML Schema definition

The cloud consists of Datacenters and each Datacenter consists of Clusters each cluster is properties such as availability of the cluster, energy index, etc. One can assign an energy index to a cluster if it is known. The base cluster is always assigned an energy index of 1 and all other cluster energy index are calculated based their relative performance and energy usage using the various application Run Profiles of these clusters.

The main simulation run is performed using an input simulation file such as the one shown in the Figure 54 below. The simulation run file is also a XML file with an associated Schema definition file (Figure 55).

```xml
<?xml version="1.0" encoding="UTF-8"?>
<Simulation xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="SimulationRun.xsd"
xsi:noNamespaceSchemaLocation="SimulationRun.xsd"
name="Simulation with 2 Runs" displaycloudinfo="true">
    <Run name="SingleRun1" displayruninfo="true" deadline="1000" appname="BLAT" domain="BioInformatics" type="QAll">
        <Cluster name="Cluster1" distribution="100" fluctuation="0.5"></Cluster>
    </Run>
    <Run name="DistributedRun1" displayruninfo="true" deadline="1500" appname="BLAT" domain="BioInformatics" type="QAll" output="file">
        <Cluster name="Cluster1" distribution="25" fluctuation="1.0"></Cluster>
        <Cluster name="FireFly" distribution="75" fluctuation="0.2"></Cluster>
    </Run>
    <Run name="SingleRun2" displayruninfo="true" deadline="2000" appname="BLAT" domain="BioInformatics" type="QAll">
        <Cluster name="MyCluster1" distribution="100" fluctuation="0.5"></Cluster>
    </Run>
    <Run name="SingleRun3" displayruninfo="true" deadline="3000" appname="BLAT" domain="BioInformatics" type="QAll">
        <Cluster name="MyCluster1" distribution="100" fluctuation="0.5"></Cluster>
    </Run>
    <Run name="SingleRun4" displayruninfo="true" deadline="4000" appname="BLAT" domain="BioInformatics" type="QAll">
        <Cluster name="MyCluster1" distribution="100" fluctuation="0.5"></Cluster>
    </Run>
    <Run name="SingleRun5" displayruninfo="true" deadline="5000" appname="BLAT" domain="BioInformatics" type="QAll">
        <Cluster name="MyCluster1" distribution="100" fluctuation="0.5"></Cluster>
    </Run>
</Simulation>
```

Figure 54: Sample Simulation Run XML file

```xml
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema" attributeFormDefault="unqualified" elementFormDefault="qualified">
  <xs:element name="Simulation">
    <xs:complexType>
      <xs:choice maxOccurs="unbounded" minOccurs="0">
        <xs:element name="Run">
          <xs:complexType>
            <xs:sequence>
              <xs:element name="Cluster" maxOccurs="unbounded" minOccurs="0">
                <xs:complexType>
                  <xs:simpleContent>
                    <xs:extension base="xs:string">
                      <xs:attribute type="xs:string" name="name" use="required"/>
                      <xs:attribute type="xs:byte" name="distribution" use="optional"/>
                      <xs:attribute type="xs:float" name="fluctuation" use="optional"/>
                    </xs:extension>
                  </xs:simpleContent>
                </xs:complexType>
              </xs:element>
            </xs:sequence>
            <xs:attribute type="xs:string" name="name" use="optional"/>
            <xs:attribute type="xs:string" name="displayruninfo" use="optional"/>
            <xs:attribute type="xs:short" name="deadline" use="required"/>
            <xs:attribute type="xs:string" name="appname" use="optional"/>
            <xs:attribute type="xs:string" name="domain" use="optional"/>
            <xs:attribute type="xs:string" name="type" use="optional"/>
            <xs:attribute type="xs:string" name="output" use="optional"/>
          </xs:complexType>
        </xs:element>
      </xs:choice>
      <xs:attribute type="xs:string" name="name"/>
      <xs:attribute type="xs:string" name="displaycloudinfo"/>
    </xs:complexType>
  </xs:element>
</xs:schema>
```

Figure 55: Sample Simulation Run XML Schema definition

We chose to pick 3 of the “What if” scenarios discussed above and conducted the experiments below using the simulation program. The datasets were broken into 3 sets based on their deadlines. We have the following 3 deadline based datasets. There were 100 runs in each dataset which were randomly generated.

1) Between 10 sec and 10 minutes – 10 minutes dataset

2) Between 10 minutes and 1 hour – 1 hour dataset

3) Between 1 hour and 1 day – 1 day dataset

7.4.1 Scenario 1 – Meeting Deadlines on specific Cloud Clusters

Customers can run basic scenarios to see if the deadlines they are providing can be met based on past run profiles.

We generated multiple datasets (10 minutes, 1 hour, 1 day) with deadlines as mentioned above and ran our simulation application. The charts below (Figure 56 and Figure 57) show that when deadline and AET of these job runs. For the dataset with deadlines below 10 minutes 70% of the times deadlines were met given the specified cluster. With deadlines on 1 hour and 1 day deadlines were met in all cases. This simple basic scenario can be used by customers to test if their job deadlines will be met on a given cluster and then make the actual run on that cluster instead of a shot in the dark. This would help in wastage of resources due to unmet deadlines and result in higher productive.

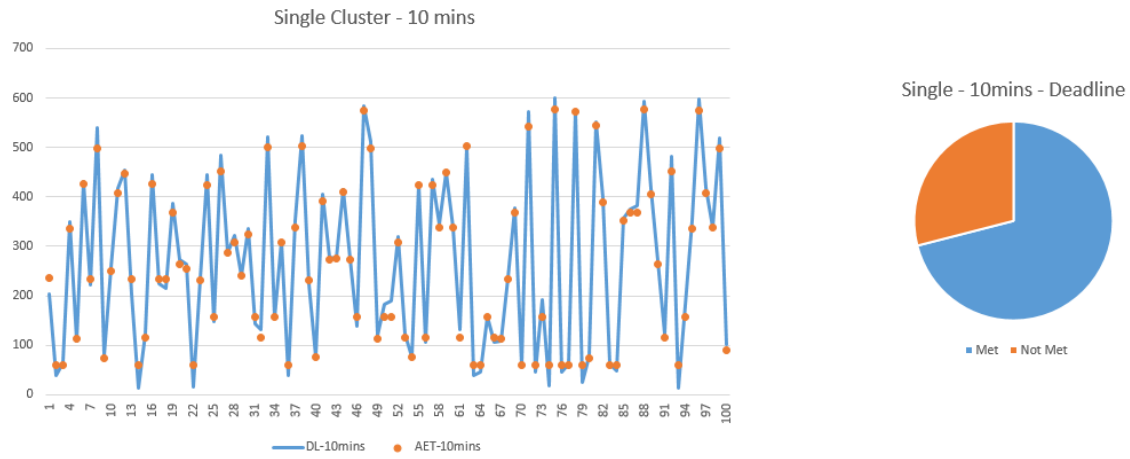

Figure 56: Scenario1 - Meeting deadline (10 minutes)

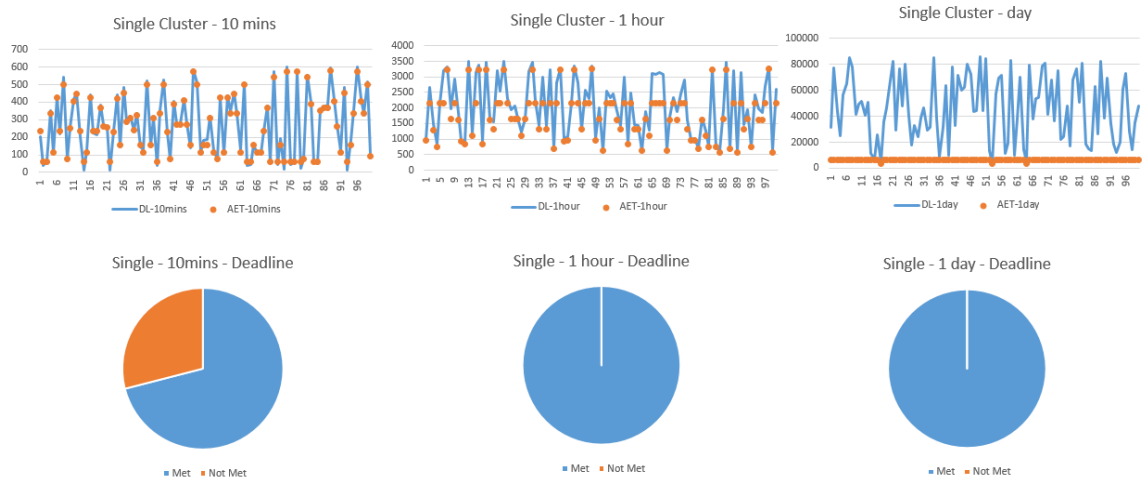

Figure 57: Scenario 1 - Meeting deadlines (10 minutes, 1 hour, 1 day)

7.4.2 Scenario 2 – User Single v/s Distributed Clusters

Customers can also run scenarios to see whether they should use a single cluster or distribute the work load across multiple clusters to meet specific deadline requirements.

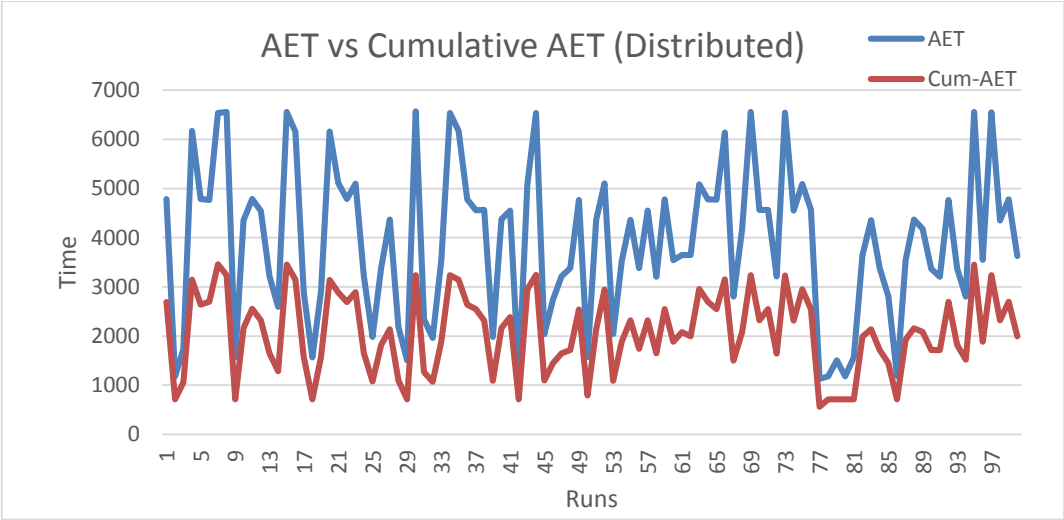

Figure 58: Scenario 2 - Single v/s Distributed Cluster runs

We generated different cloud initializations and run sets based on distributed cluster runs and single cluster and ran our simulation application to see the impact of how deadlines were met. The Figure 58 clearly shows that when a job fails to meet a deadline on a single

cluster the customers can choose to run them against distributed clusters and meet deadlines. The chart also shows we get better cumulative AET from using distributed clusters compared to using single clusters.

7.4.3 Scenario 3 – Analyze impact of Cluster Availability

Cluster operators can also run "what if" scenarios to see if increasing the availability of their Custer may have a potential impact on revenue based on number of additional jobs they might get at a certain availability level and whether the costs of increasing the availability are justified based on opportunity costs.

We generated cloud clusters with varying availability from 10% - 100% with 10% increments and ran our simulation application to see how it impacted deadline met and node adjustments needed to complete execution within the given deadline. The resulting chart Figure 59 shows that close to 29% of jobs failed to meet deadline as availability fell and 71% succeeded in meeting the given deadline.

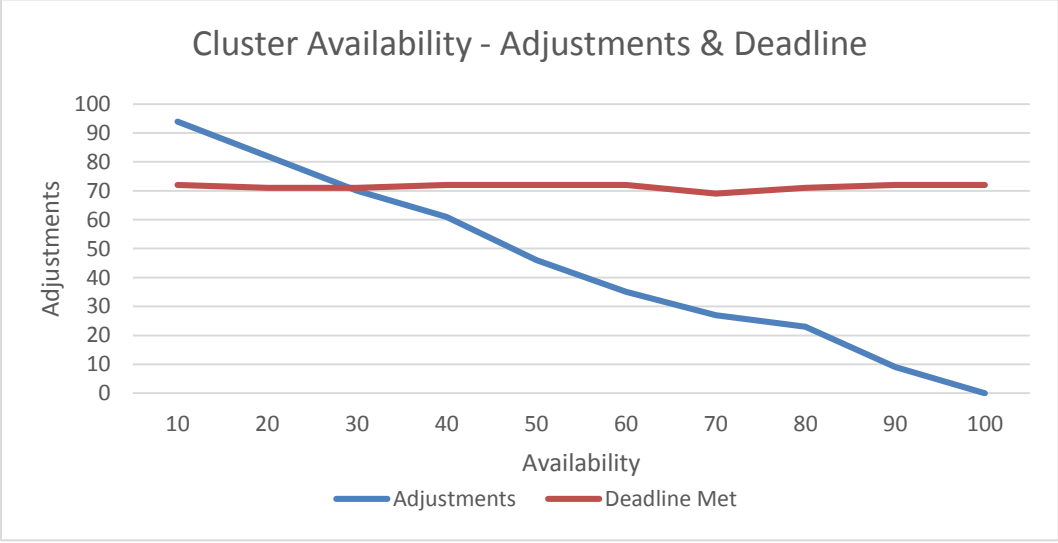

Figure 59: Scenario 3 - Availability v/s Node Adjustments

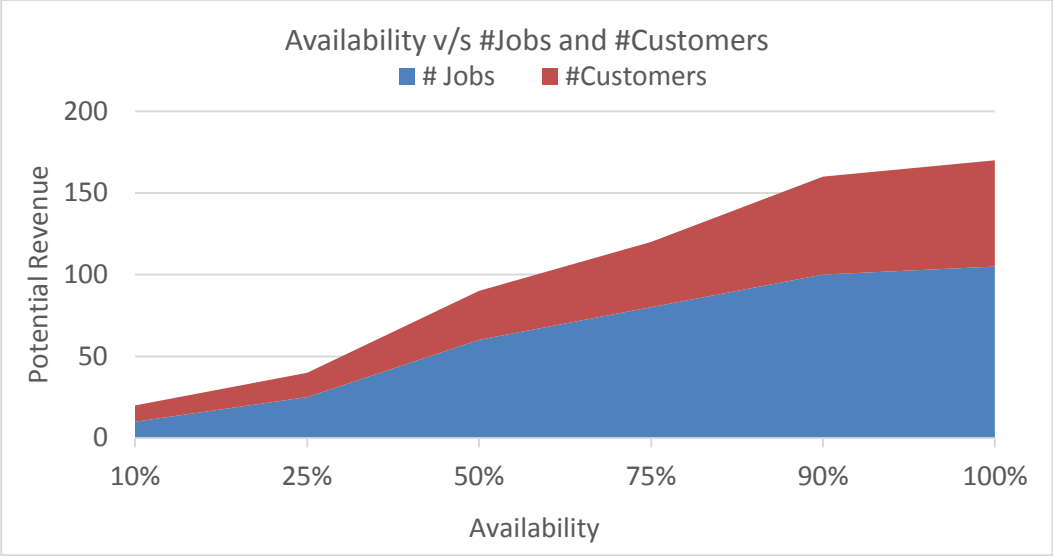

Figure 60: Scenario 3 – Impact of Availability on # of Jobs/Customers

We also found that as we adjusted availability and availability increased availability of a given cluster more jobs were bound to be sent to that cluster than not and more customers would be likely to send jobs to that cluster, meaning that availability has a direct impact on revenue generated from that cluster (Figure 60).

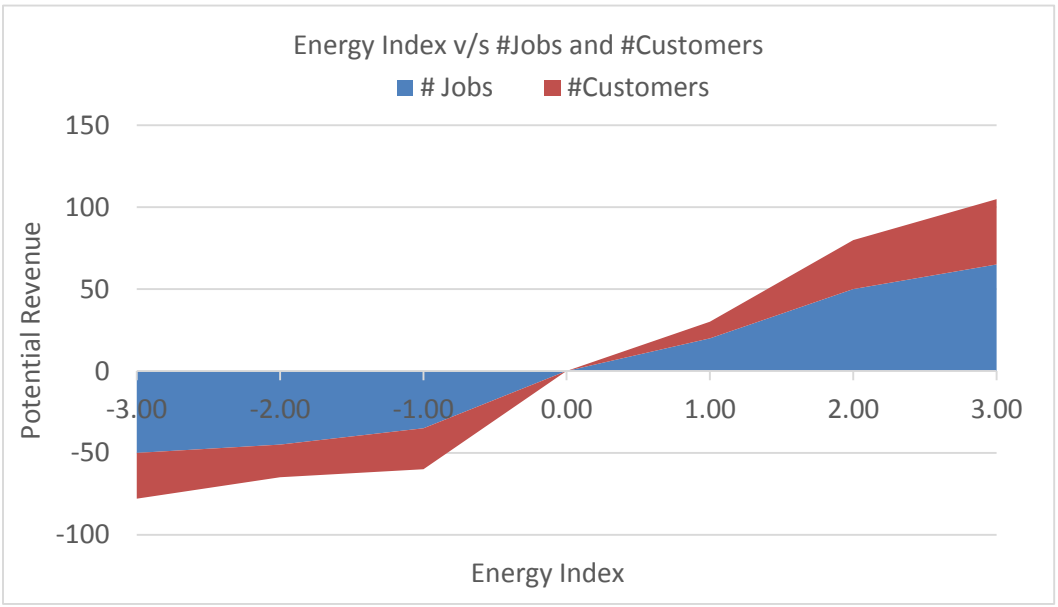

Figure 61: Scenario 3 – Impact of Energy Index on # of Jobs/Customers

We also find that as we increase the energy index for a given cluster more jobs were likely to be sent to that cluster by more customers and for a cluster with lower energy index less jobs would be sent by fewer customers again impacting revenue (Figure 61).

These experiments could be used by Cluster operators to determine if investing in upgrading their infrastructure would result in enough increased revenues to justify the ROI.

Chapter 8: Energy aware scheduling in Mobile Devices

8.1 Development: Creation of a Conceptual Model

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 62 below illustrates this approach.

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

Figure 62: 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 (Intel, 2006).

Table 8: Variables influencing the energy-efficiency equation

Hardware			Software	
Silicon Process Technology	Chip Technology	Power Management	Operating System	Applications
• Second generation strained silicon • Improved interconnects	• Dynamic sleep transistor • Demand based switching • On-die voltage regulation • Multi-core and clustered micro-architecture • Power Gating, Macro Fusion.	• Voltage Regulation Technology • Improved display power specs • Thermal design for advanced heat-sync technology	• 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.	• 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. The Table 8 illustrates the various complimentary areas of research being pursued to address the overall energy efficiency question.

In the conventional approach employed in most portable computers, a processor enters power-down mode after it stays in an idle state for a predefined time interval. Since the processor still wastes its energy while in the idle state, this approach fails to obtain a large reduction in energy when the idle interval occurs intermittently and its length is short. In (Srivastava, Chandrakasan, & Brodersen, 1996) (Hwang & Wu, 1997), the length of the next idle period is predicted based on a history of processor usage. The predicted value becomes the metric to determine whether it is beneficial to enter power-

down modes or not. This method focuses on event driven applications such as user-interfaces because latency, which arises when the predicted value does not match the actual value, can be tolerated. However, we need an exact value instead of a predicted value for the next idle period when we are to apply the power-down modes in a hard real-time system, which is possible in the LPFPS.

8.2 Previous Work on this Model

(Ahmed & Chakrabarti, 2004), enhanced the algorithm proposed by (Shin & Choi, 1999), by extending the algorithm to account for the slack generated at runtime due to the difference between WECT and AET (Actual Execution Time). They proposed an algorithm which had 2 Phases. The basic idea of the algorithms in this model is to exploit the slacks generated to reduce the voltage levels of the tasks, so that the battery charge consumed or the drop in voltage is minimized. The algorithm operates in two phases.

1. **Phase I:** Off-line task scheduling algorithm using WCET.
2. **Phase II:** On-line algorithm using AET.

In Phase I the tasks are assumed to be executed at their WCETs. A schedule is determined for one hyper-period (defined as the least common multiple of the periods of all the tasks in the task set). In Phase II (on-line), the slack generated due to the AET being less than the WCET, is used to further scale the voltage levels of the tasks.

Phase I: The off-line scheduling algorithm is based on a paper presented by the same co-authors (Chowdhary & Chakrabarti, 16-18 Oct. 2002); it determines the task ordering and the voltage level of each instance of a task in a hyper-period. Applying WCETs in this phase guarantees that the tasks meet their deadline. This is done in two steps.

Step 1: Obtain a feasible schedule by using the earliest deadline first algorithm.

Step 2: Utilize the available slack by voltage down scaling as much as possible starting from the end of the profile.

Phase II: During operation of the system, the AET of a task could be a lot smaller than its WCET. It is suggested that it is best to use the slack as late as possible; which is achieved by a process called as slack forwarding. Slack forwarding is based on the observation that slack generated by early completion of a task can be made available to a later task if the later task is released prior to the time at which the slack originated.

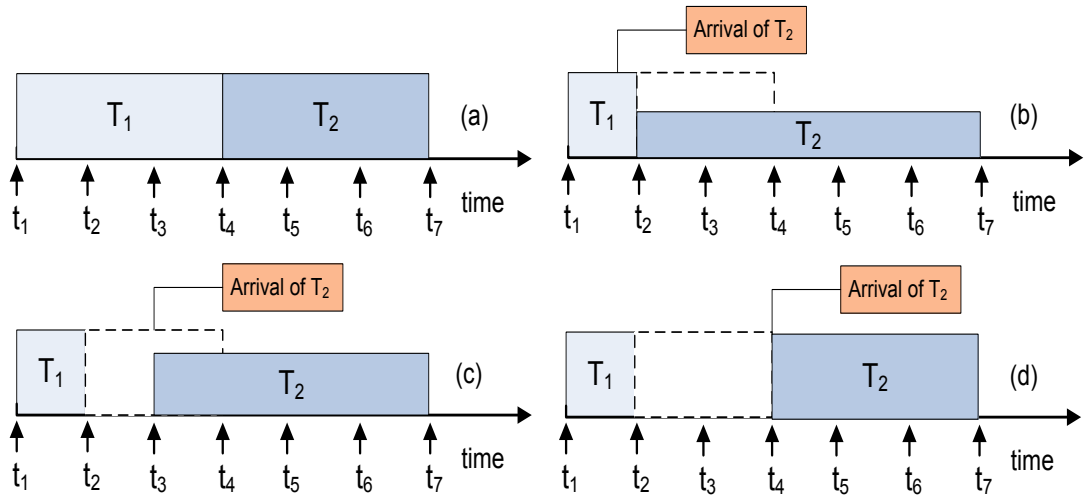

Figure 63: (a) WCET Schedule. (b) WCET Schedule with full slack forwarding. (c) WCET Schedule with partial slack forwarding. (d) WCET Schedule with no partial slack forwarding.

Consider two tasks T_1 and T_2 and let us assume WCET for the tasks. Task T_1 starts at t_1 and finishes at t_4 and T_2 starts at t_4 and finishes at t_7, as shown in Figure 63(a). Suppose T_1 actually finishes earlier at time t_2, generating a slack of (t_4-t_2). All of this slack is available to T_2 if its arrival time is at t_2 or before, as depicted in Figure 63(b). If the task T_2 was released at t_3, only a part of the generated slack is available to T_2, as shown in Figure 63(c). If the task T_2 was released at t_4 none of the generated slack is available to T_2 as

shown in the Figure 63(d). Thus the decision of slack forwarding can be made by inspecting the arrival time of the subsequent task to be executed.

Figure 64 provides the pseudo-code for the on-line algorithm. The input to the online-algorithm consists of ordering of tasks as well as their voltage levels based WCET. The purpose of this algorithm is to readjust the voltage level of the task based on additional slack. The basic steps are as follows. After the completion of a task, the scheduler gets the next task from the run queue. The finish time of the task is estimated based on the voltage level determined in Phase I. If the finish time is before the release time of the next task in the queue, the voltage level of the task is readjusted.

```
Input: Phase I schedule based on EDF algorithm
Repeat for Every Task
Get the scaling level of the next task Ti
If the task is not available (Current time < Task start time)
{
        Wait
}
Else
{
        If (finish time of task Ti < release time of task Ti+1)
                Update the scaling level to absorb the slack
}
Execute the task
```

Figure 64: Pseudo-Code for On-line Phase II

Example:

Consider the three tasks given in Table 9 which is reproduced below. Rate monotonic priority assignment is a natural choice because periods (*Pi*) are equal to deadlines (*Di*). Priorities are assigned in row order as shown in the fifth column of the Table 9. Note that this is the same example from the original algorithm 1 by (Shin & Choi, 1999); which is

being adapted to show the incremental improvement done by (Ahmed & Chakrabarti, 2004).

Table 9: Example Task Set

	P_i	D_i	C_i	Priority
T_1	50	50	10	1
T_2	80	80	20	2
T_3	100	100	40	3

Let us consider the task set in (Shin & Choi, 1999) represented by the Table above. There are three tasks with periods 50, 80 and 100 minutes. The hyper-period is 400 minutes (L.C.M of 50, 80 and 100).The set of operating voltages considered during voltage scaling is S_v = {3.3, 3.0, 2.7, 2.5, 2.0} volts. Figure 65(c) shows the final task profile with the improved algorithm after each phase as well as that generated with the low power fixed priority algorithm in (Shin & Choi, 1999).

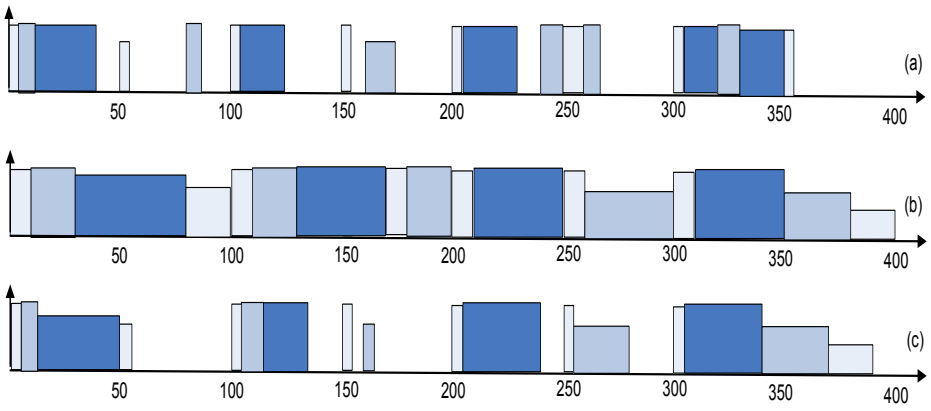

Figure 65: Task scheduling using LPFPS algorithm versus enhancements

8.3 Proposed Solution

We realized that some online slack could be potentially wasted in the algorithm proposed by (Ahmed & Chakrabarti, 2004) due to the fact that even though some tasks become available based on the actual periodicity of a task they are not executed because the run is

determined by the schedule generated in Phase I; which uses the EDF algorithm based on the WCET of the tasks and we already know that this is a very safe yet conservative approach.

Motivation: Our solution exploits the fact that even though some tasks become available based on the actual periodicity of a task they are not executed because the run queue is determined by the schedule generated in the offline phase I of the algorithm using the conservative EDF (Earliest Deadline First) algorithm. We peek at the task run-queue to find such tasks and schedule them for execution if possible based on the knowledge of the available slack and the arrival on the next task. This helps in minimizing the wastage of the generated slack.

Considering the same set of tasks as described in (Ahmed & Chakrabarti, 2004) (Shin & Choi, 1999) and shown here in Table 9, this waste of slack can be observed at time t=80 even though T_2 becomes available as per the periodicity of the task it is not executed because the run queue determined by the Offline phase has T_1 as the next task. We also notice that T_2 can be easily completed before T_1 whose next earliest start time is t=100; because T_2 has WCET execution time of 20 and since it starts at time t=80 we have a timeframe of (100 – 80) = 20 available for execution.

A similar yet slightly different situation occurs at time t=240, where even though T_2 becomes available as per the periodicity of the task it is not executed in (Ahmed & Chakrabarti, 2004) because the run queue determined by the Offline phase has T_1 as the next task at t=250. We also notice that T_2 cannot be easily completed before T_1 whose next earliest start time is t=250; because T_2 has WCET execution time of 20 and since it starts at time t=240 we have a timeframe of (250 – 240) = 10 available for execution. But

a simple task look –ahead shows that to execute both T_1 and T_2 we have a total time of (240-300) = 60 and the WCET for each is 10 and 20 respectively; a total duration WCET of 30; which tells us that scheduling T_2 now will not cause us to miss the deadline for T_1 and that both tasks can be executed within the available time of 60.

To avoid this waste, we enhance the algorithm such that the original start time for each periodic task is fed to the algorithm as input. Figure 66 shows the final task profile with our algorithm as well as those generated by (Ahmed & Chakrabarti, 2004) and with the low power fixed priority algorithm (Shin & Choi, 1999). Since we further scale down the voltage and make more use of online slack we expect our algorithm to perform better compared to (Ahmed & Chakrabarti, 2004) (Shin & Choi, 1999). (This will be proven later by simulation experiments).

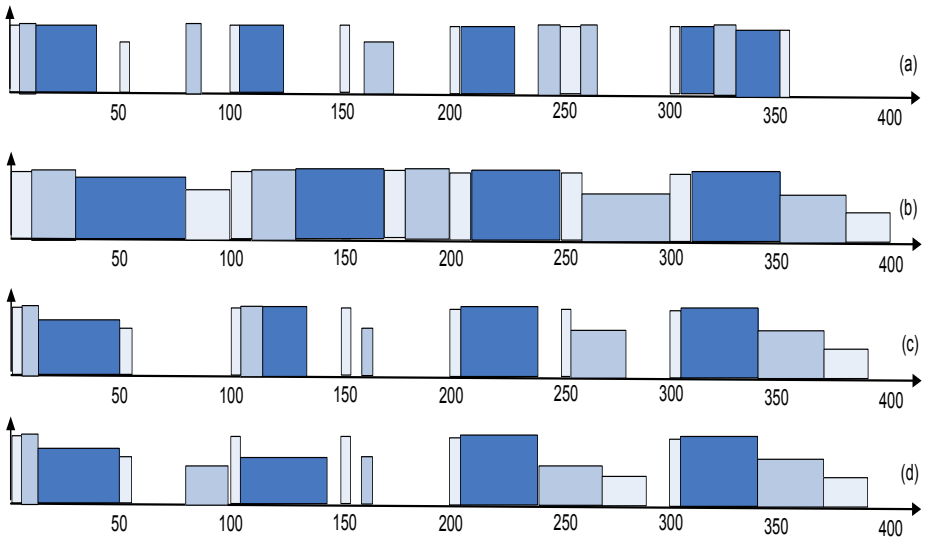

Figure 66: Task scheduling using proposed algorithm, LPFPS and 2-Phase algorithm

The pseudo-code for the proposed algorithm is shown in Figure 67. It is important to keep the time complexity of the online phase of the algorithm to a minimum for obvious reasons. It should be noted here that we do not add any additional time complexity to the

online phase in the proposed algorithm as we only check to see if tasks are available based on their original periodicity and if these can be scheduled in the slack time before the next task becomes available based on the Phase I EDF schedule. This can easily be accomplished in constant time which is O(1) and hence there is no increase in the time complexity of the algorithm. Another important point to note if we do schedule a task earlier based on the reasoning above, we have to remove this task from the Phase I schedule or mark it as complete to make sure that we do not re-execute the task again.

```
Input: Phase I schedule and original task periodicity
Repeat for Every Task
Get the scaling level of the next task Ti based on Phase I schedule
If the task is not available (Current time < Task Ti start time)
{
        if ( (original task periodicity shows a task To is available earlier) and
            (start time of Ti – To >= WCET of To ) or (Ti+2 - To >= WCET Ti + WCET To) )
                Schedule task To and remove it from Phase I schedule
        else
                Wait
}
Else
{
        If (finish time of task Ti < release time of task Ti+1)
                Update the scaling level to absorb the slack
}
Execute the task
```

Figure 67: Pseudo-Code for proposed algorithm

8.4 Results of the proposed solution

We calculated the average energy utilized for all the test cycles and the plot below (Figure 68) clearly suggests that the enhanced algorithm performs better than the algorithms in (Ahmed & Chakrabarti, 2004). We get an average reduction of approximately 9.29% as

compared with the algorithm in (Ahmed & Chakrabarti, 2004). Note that we use a similar technique as in (Ahmed & Chakrabarti, 2004), (Shin & Choi, 1999)to generate our tasks, to have a high degree of confidence in our conclusions.

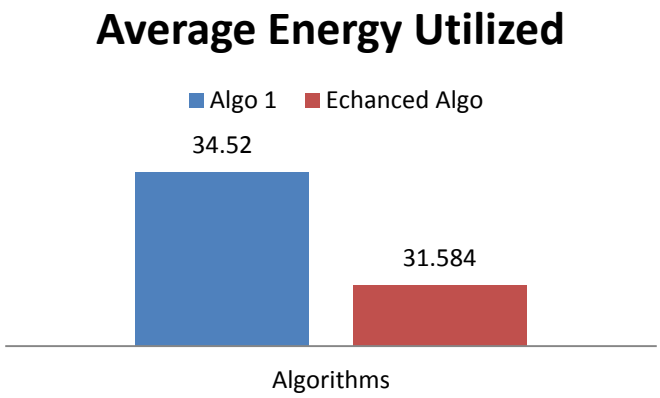

Figure 68: Enhanced Algorithm - Average Energy Utilized

8.5 Expected Contributions and Limitations

Most of the Energy Aware Scheduling Algorithms designed so far use WCET to compute the workloads in the offline phase. In general most tasks complete between BCET and WCET. In fact; it is a well-known that most tasks complete well before WCET. We propose to exploit this knowledge to our advantage and propose that instead of computing workload at WCET, we use information regarding expected execution time (EET).

Expected Execution Time (EET) may be computed in several ways; one way to compute this would be based on Actual Execution Time (AET) in the previous hyper-period, another approach could be average of all previous AET for that task, so on and so forth. An important aspect of this approach is that at runtime depending on AET we may have some tasks completing in time greater than EET and some less than EET. This could potentially lead to deadline violations; which we need to resolve.

Approaches to compute Expected Execution Time

1. **Conservative Approach:** Expected Execution Time is computed conservatively so that it is closer to WCET. This approach has a lower propensity for deadline violations; which need to be resolved.
2. **Risky Approach:** Expected Execution Time is computed quite generously so that it is closer to BCET. This approach has a higher propensity for deadline violations; which need to be resolved.

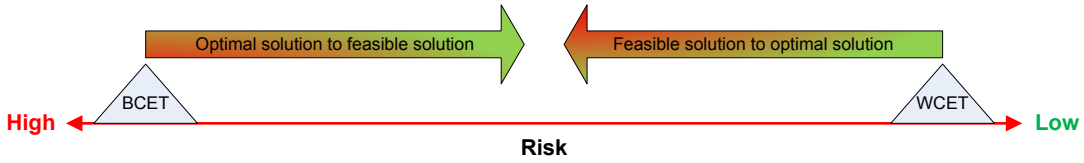

Figure 69: Understanding Risk w.r.t Optimal v/s Feasible solutions

Our research aims to contribute a model for energy aware scheduling and come up with a few algorithms for that model. It will also attempt to explore the approaches stated above to compute Expected Execution Time which will help in better scheduling tasks based on their energy profile. When running our experiments we have several overheads that are inherent to the systems itself such as energy utilized by the network card, or the energy utilized by the graphics card, etc. For the purpose of our study we are focused on the energy utilized to actually execute the tasks and not these other overheads. Also at times the scheduling algorithm itself has an execution overhead which is considered only where it is significant.

Chapter 9: Overall Conclusions and Future Research

In chapter 4 we used our proposed EAS model in an HPC environment in the Bioinformatics domain to the BLAT application, 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. 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.

In chapter 5 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. 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. Our experiments suggest that the choice of run profile in step 1 of the algorithm has an impact on the overall performance of the algorithm because it impacts the number of adjustment the algorithm has to make to meet deadlines. Each adjustment has an associated overhead which impacts the energy optimization. Clearly 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.

In chapter 6, we applied our run-profile based approach on another bioinformatics application for assembling short reads. Based on the results we can clearly observe that given a deadline we can choose the appropriate number of nodes to run the overlap detection phase of the assembler on based on our new understanding of the run-profile we just produced. This will allow us to apportion just enough nodes to meet the deadline thus maximizing the objective of performance with minimum energy utilization. We also observed that with a smaller number of nodes we have larger gains in performance and above a certain number of nodes the performance gain is only modest at best. In fact as we add additional nodes our communication costs and related overhead is higher. Clearly different bioinformatics applications and algorithms will have different run profiles and understanding each one of them will allow us to best assign the appropriate number of nodes to meet a given deadline. It was also important to see how the number of read subsets impacted the performance/energy criterion. Our experiments suggest a bowl shaped curve when we varied the number of files for the same number of nodes. Clearly there must be some optimum value for the number of files for each set. This highlights the importance of understanding the degree of parallelism for the program, which is done by establishing the run profile/speedup curve. The EAS engine uses the knowledge from

the run profile to make intelligent and dynamic decisions about number of nodes to use to minimize energy utilization and still provide necessary performance. Clearly it is no longer sufficient to simply run a program in a HPC environment. It is important and essential to understand the data, its characteristics, and the application domain to build a parallel program that is energy aware. We have several parameters we could study and the relationship between them. These parameters are (1) Number of files; (2) Number of sequences per file; (3) Number of nodes used and (4) Average sequence length. In this section we have only looked at number of nodes used as a parameter for our experimental design. In the future we plan to investigate how adjusting the different tuning parameters such as number of files, number of sequences per file, number of nodes impacts the performance and energy efficiency. We also plan on including the pre-processing step and final assembly as part of the EAS processing. Our main motivation is to move this from a simple speedup to the realm of energy awareness. Our EAS model for the purposes of the experiments conducted calculated energy as a function of resources used in this case number of nodes. The energy function could be made more complex; we leave that for a future study.

In chapter 7, we took the next logical step to applying the run-profile based approach to the “Cloud”, as cloud computing in gaining more and more importance. We took the approach to write a Cloud Simulation based on our EAS run-profile model so that we can as customers and operators of Cluster resources ask the “what if” questions, run them on the cloud simulator, to help make better informed decisions. We provided several “What if” scenarios and chose to design experiments for 3 of these scenarios demonstrating the

value of such a cloud simulation program to help answers questions such as the following before actually utilizing cloud resources.

1) Can the given deadlines be met based on the past run profile.
2) Should we use a single cluster or distribute the work load across multiple clusters to meet specific deadline requirements.
3) Analyze impact of cluster availability/energy index on revenue and ROI.

In this dissertation we proposed several algorithmic approaches to address energy awareness across the spectrum from small mobile devices to large high performance clusters to Cloud computing. We also pose questions for further research & study in the very important area of "Energy Awareness & Scheduling".

Finally in chapter 8 we presented an energy aware algorithm for mobile devices for scheduling purposes where the average energy utilized for all the various job cycles provided an average reduction of approximately 9.29% as compared with previous algorithms. Most of the Energy Aware Scheduling Algorithms designed so far use WCET to compute the workloads in the offline phase. In general most tasks complete between BCET and WCET. In fact; it is a well-known that most tasks complete well before WCET. We exploited this knowledge to our advantage and proposed the Run-queue peek algorithm which provided additional energy savings.

We also proposed an enhanced dynamic task scheduling algorithm using task run-queue peek technique for battery operated (mobile devices) DVS systems that further maximize the residual charge and the battery voltage. Our future research focused on using the information regarding expected execution time (EET) instead of WCET because WCET

is a very conservation approach used in the Off-line Phase to schedule tasks. We explored both the suggested approaches of computing EET namely conservative and risky and study their performance relative to each other.

In the future we plan on applying our EAS model to a much complex problem in the bioinformatics domain of clustering and networks. We also plan on taking the Department of Energy's "**Better Building Challenge**"– "**to reduce the energy used across their building portfolios by 20 percent or more by 2020**". Our goal is to use our EAS model along with scheduling heuristics and apply them to HVAC and other building sensor data to perform real-time analytics and address the issue of "**finding what matters in a timely matter**" to save energy costs.

Bibliography

A Portable Implementation of MPI. (n.d.). Retrieved Nov 2009, from MPICH: http://www-unix.mcs.anl.gov/mpi

Ahmed, J., & Chakrabarti, C. (2004). A Dynamic Task Scheduling Algorithm for Battery Powered DVS Systems. *ISCAS - International Symposium on Circuits and Systems*, *II*, pp. 813-816.

Amazon Elastic Compute Cloud (Amazon EC2). (2013). Retrieved from Amazon: http://aws.amazon.com/ec2/

AMD. (2007, Feb). *AMD Report Pegs Global Data Center Energy Costs at $7.2 Billion*. Retrieved Sep 2008, from http://www.environmentalleader.com/2007/02/16/amd-report-pegs-global-data-ceneter-energy-costs-at-72-billion/

Amdahl's Law. (n.d.). Retrieved Jan 2009, from Wikipedia: http://en.wikipedia.org/wiki/Amdahl's_law

Amdhal, G. M. (1967). Validity of the single processor approach to achieving large scale computing capabilities. *Proc. Am. Federation of Information Processing Societies Conf.*, 483 – 485.

Aronsson, P., & Fritzson, P. (Jan 8-10, 2003). Task Merging and Replication using Graph Rewriting. *Tenth International Workshop on Compilers for Parallel Computers.* Amsterdam, the Netherlands.

Benini, L., Bogliolo, A., Paleologo, G. A., & De Micheli, G. (1999, June). Policy optimization for dynamic power management. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, 18*(6), 813 - 833.

Bjornson, R., Sherman, A., Weston, S., Willard, N., & Wing, J. (2002). TurboBLAST(r): A parallel implementationof BLAST built on the TurboHub. *International Parallel and Distributed Processing Symposium.*

Blackforest Computing Cluster. (n.d.). Retrieved Oct 2008, from UNO: http://blackforest.gds.unomaha.edu/about.php

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

Canadian Electricity Association. (2006). *Canadian Attitudes Towards Energy Efficiency.* Retrieved Feb 15, 2007, from Canadian Electricity Association: http://www.canelect.ca/en/Pdfs/2973_Fact_Sheet_EN_noDate.pdf

Chi, E., Shoop, E., Carlis, J., Retzel, E., & Riedl, J. (1997). Efficiency of shared-memory multiprocessors for a genetic sequence similarity search algorithm.

Cho, S., & Melhem, R. M. (2008, Jan). Corollaries to Amdahl's Law of Energy. *IEEE Computer Architecture Letters (CAL), 7*(1), 25 – 28.

Chowdhary, P., & Chakrabarti, C. (16-18 Oct. 2002). Battery aware task scheduling for a system-on-a-chip using voltage/clock scaling. *IEEE Workshop on Signal Processing Systems, (SIPS 2002).*, (pp. 201-206).

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

Cohen, P. R., & Howe, A. E. (1988, Dec). How evaluation guides AI research. *AI Magazine, 9*(4), pp. 35-43.

Cohen, P. R., & Howe, A. E. (1989, May/June). Toward AI research methodology: three case studies in evaluation. *IEEE Transactions on Systems, Man and Cybernetics, 19*(3), 634-646.

Dayde, M. (2006). High Performance Computing for Computational Science. *VECPAR.* Berlin Heidelberg, Germany: Springer-Verlag.

Eui-Young, C., Benini, L., & De Micheli, G. (1999). Dynamic power management using adaptive learning tree. *IEEE/ACM International Conference on Computer-Aided Design*, 274 - 279.

Feigenoff, C., & al., e. (2003). *Grand Research Challenges in Information Systems (NSF Grant No. 0137943).* Computing Research Association. Washington, DC 20036-4632: Computing Research Association. Retrieved March 21, 2006, from Computing Research Association: http://www.cra.org/reports/gc.systems.pdf

Foley, J. (2008, Mar). *Google's Iowa Data Center Emerges*. Retrieved Nov 2008, from Information week: http://www.informationweek.com/blog/main/archives/2008/03/googles_iowa_da.html

Gary, S., Ippolito, P., Gerosa, G., Dietz, C., Eno, J., & Sanchez, H. (1994, Oct). PowerPC: A microprocessor for portable computers. *IEEE Design & Test of Computers, 11*(4), 14-23.

Genome Bioinformatics. (n.d.). Retrieved Oct 2008, from UCSC: http://genome.ucsc.edu/index.html

Google Cloud Platform. (2013). Retrieved from https://cloud.google.com/products/compute-engine

Greenawalt, P. M. (1994, Jan 31). Modeling power management for hard disks. *Proceedings of the Second International Workshop on Modeling, Analysis, and Simulation of Computer and Telecommunication Systems, MASCOTS '94.*, 62-66.

Gropp, W., Lusk, E., & Skjellum, A. (Oct 1994). *Using MPI: Portable Parallel Programming with the Message Passing Interface.* 55 Hayward St. Cambridge, MA 02142: MIT Press.

Gropp, W., Lusk, E., & Thakur, R. (Nov 1999). *Using MPI-2: Advanced Features of the Message Passing Interface.* 55 Hayward St. Cambridge, MA 02142: MIT Press.

He, J. (2008, Feb). *Datacenter Power Management: Power Consumption Trend*. Retrieved Sep 04, 2008, from Intel: http://communities.intel.com/openport/blogs/server/2008/02

Hesham, E.-R., Lewis, T. G., & Hesham, A. H. (1994). *Task Scheduling in Parallel and Distributed Systems.* Englewood Cliffs, New Jersey 07632: PTR Prentice Hall, Inc.

Hill, M. D., & Marty, M. R. (2008, July). Amdahl's Law in the Multi-core Era. *IEEE Computer, 41*, 33 – 38.

Holland Computing Center. (n.d.). Retrieved Nov 2008, from UNO: http://www.hollandhpc.com/index.shtml

Hu, J., & Marculescu, R. (2004). Energy-Aware Communication and Task Scheduling for Network-on-Chip Architectures under Real-Time Constraints. *Proceedings of the conference on Design, automation and test in Europe. 1*, pp. 234 - 239. IEEE Computer Society, Washington, DC, USA.

Huang, X. (2003, Sept). PCAP: A Whole-Genome Assembly Program. *Genome Res., 13*(9), 2164 - 2170.

Hwang, C.-H., & Wu, A. (1997, Nov). A predictive system shutdown method for energy saving of event-driven computation. *Proceedings of the 1997 IEEE/ACM international conference on Computer-aided design*, 28-32.

Intel. (2006, Oct). Intel is Leading the Way in Designing Energy-Efficient Platforms. *Technology@Intel Magazine*.

Intelligent Energy Europe - Publications and Documents. (2007, Jan 15). Retrieved Jan 15, 2007, from Intelligent Energy - Europe: http://ec.europa.eu/energy/intelligent/library/publications_en.htm

Jaber, J. O., Mamlook, R., & Awad, W. (Dec 2003). Assessment of Energy Conservation and Awareness program in Household sector in Jordan. Rio de Janerio, Brazil: World Climate and Energy.

Jejurikar, R., & Gupta, R. (2002). Energy Aware Task Scheduling with Task Synchronization for Embedded Real Time Systems. *Proceedings of the 2002 international conference on Compilers, architecture, and synthesis for embedded systems* (pp. 164 - 169). Grenoble, France: ACM Press New York, NY, USA.

Jha, N. K. (2001). Low Power System Scheduling and Synthesis. *Proceedings of the 2001 IEEE/ACM international conference on Computer-aided design* (pp. 259 - 263). San Jose, California: IEEE Press Piscataway, NJ, USA.

Khan, A. A., McCreary, C. L., & Jones, M. S. (1994). A Comparison of Multiprocessor Scheduling Heuristics. *International Conference on Parallel Processing*, *2*, pp. 243-250.

Kim, W., Kim, J., & Min, S. L. (2002). Dynamic voltage scaling algorithm for dynamic-priority hard real-time systems using slack time analysis. *Proceedings of the conference on Design, automation and test in Europe* (pp. 788 - 794). IEEE Computer Society Washington, DC, USA.

Larsson, N. J., & Sadakane, K. (1999). *Faster suffix sorting.* Lund University. Sweden: Lund.

Li, W., & Godzik, A. (2006, July). Cd-hit: A Fast Program for Clustering and Comparing Large Sets of Protein or Nucleotide Sequences. *Bioinformatics, 22*(13), 1658-659.

Lu, Y.-H., Benini, L., & De Micheli, G. (2000). Low-power task scheduling for multiple devices. *Proceedings of the Eighth International Workshop on Hardware/Software Codesign, CODES 2000.*, (pp. 39 - 43).

Martin, T. (August 1999). *Balancing batteries, power, and performance: System issues in CPU speed-setting for mobile computing.* Ph.D Dissertation.

Meyerson, M. (2010, Oct). Advances in Understanding Cancer Genomes through Second-generation Sequencing. *Nat. Rev. Genet., 11*(10), pp. 685 - 696.

Microsoft on Cloud Computing. (2013). Retrieved from http://www.microsoft.com/en-us/news/presskits/cloud/

Miller, J. (2010, June). Assembly Algorithms for next-generation sequencing data. *Genomics, 95*(6), 315 - 327.

Mishra, R., Rastogi, N., Zhu, D., Mossé, D., & Melhem, R. (2003). Energy Aware Scheduling for Distributed Real-Time Systems. *Proceedings of the 17th International Symposium on Parallel and Distributed Processing. IPDPS* (p. 21.2). IEEE Computer Society, Washington, DC, USA.

Myers, E. (2000, Mar). A whole-genome assembly of Drosophila. *Science 2000, 287*(5461), 196–204.

Myers, E. (2005, Sept). The Fragment Assembly String Graph. *Bioinformatics, 21*(2), 79 - 85.

NCBI Database. (2010, Nov). Retrieved from NCBI: http://www.ncbi.nlm.nih.gov/sra

Needleman, S. B., & Wunsch, C. D. (1970). A general method applicable to the search for similarities in the amino acid sequences of two proteins. *Molecular Biolology, 48*(3), 443 – 453.

Ohlebusch, E., & Abouelhoda, M. I. (2006). *Chaining Algorithms and Applications in Comparative Genomics* (Vol. 15). Boca Raton, FL: Chapman and Hall/CRC Computer and Information Science Series.

Pawaskar, S. S., & Ali, H. H. (2010). On the Tradeoff between Speedup and Energy Consumption in High Performance Computing – A Bioinformatics Case Study. *Parallel and Distributed Computing and Networks.* Innsbruck, Austria.

Pawaskar, S., & Ali, H. (2010). A Dynamic Energy-Aware Model for Scheduling Computationally Intensive Bioinformatics Applications. *Proc. Int'l Conf. on High Performance Computing and Simulation.*

Qin, J. (2010, Mar). A Human Gut Microbial Gene Catalogue Established by Metagenomic Sequencing. *Nature, 464*(7285), 59 - 65.

Raghunathan, V., Pereira, C. L., Srivastava, M. B., & Gupta, R. K. (2005, Feb). Energy-aware wireless systems with adaptive power-fidelity tradeoffs. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems, 13*(2), 211- 225.

Rajkumar, R. (2005, Nov 01). *NSF Award Abstract - #0509305 CSR-EHS: Power-Aware Real-Time Systems*. Retrieved Nov 29, 2005, from National Science Foundation (NSF): http://www.nsf.gov/awardsearch/showAward.do?AwardNumber=0509305

Rakhmatov, D., Vrudhula, S., & Chakrabarti, C. (2002). Battery-conscious task sequencing for portable devices including voltage/clock scaling. *Design Automation Conference, 2002. Proceedings. 39th*, (pp. 189-194).

Rao, V., Singhal, G., Kumar, A., Visweswaran, G., & Navet, N. (2004). Battery Aware Scheduling for Embedded Systems. *17th International Conference on VLSI Design, 2004*, (pp. 105 - 110).

Rusu, C., Melhem, R., & Mossé, D. (2005, April). Multi version scheduling in Rechargeable Energy aware Real time Systems. *Journal of Embedded Computing, 1*(2), 271 - 283.

(Feb 2004, Feb). *Saving money through Energy Efficiency: A Guide to implementing an energy efficiency awareness program.* Natural Resources Canada, Office of Energy Efficiency. Ottawa ON Canada: Energy Innovators Initiative Office of Energy Efficiency Natural Resources Canada.

Sequence and Annotations downloads. (n.d.). Retrieved Dec 2008, from UCSC Genome Bioinformatics: http://hgdownload.cse.ucsc.edu/downloads.html

Shin, D., Kim, J., & Lee, S. (2001, March). Intra Task Voltage Scheduling for Low Energy Hard Real Time Applications. *IEEE Design & Test, 18*(2), 20 - 30.

Shin, Y., & Choi, K. (1999). Power Conscious Fixed Priority Scheduling for Hard Real-Time Systems. *Annual ACM IEEE Design Automation Conference* (pp. 134-139). New Orleans, Louisiana, United States: ACM Press New York, NY, USA.

Simunic, T., Benini, L., & De Micheli, G. (1999). Event-driven power management of portable systems. *Proceedings. 12th International Symposium on System Synthesis.*, (pp. 18-23). San Jose, CA, USA.

Snyder, L. (2008, Jan). *Slash Data Center Energy Cost*. Retrieved Sep 2008, from http://www.facilitiesnet.com/bom/article.asp?id=8068

Sommer, D. (2007, Feb). A Fast, Lightweight Genome Assembler. *BMC Bioinformatics, 8*(1).

Srivastava, M. B., Chandrakasan, A. P., & Brodersen, R. W. (1996, Mar). Predictive system shutdown and other architectural techniques for energy efficient programmable computation. *IEEE Trans. on VLSI Systems, 4*(1), 42-55.

Strawn, G. O., Howe, S. E., & King, F. D. (November 2006). *Grand Challenges: Science, Engineering, and Societal Advances requiring Networking and IT Research and Development.* National Coordination Office for Information Technology Research and Development (NITRD). Arlington, Virginia: NITRD.

Swaminathan, V., & Chakrabarty, K. (2001). Investigating the effect of voltage-switching on low-energy task scheduling in hard real-time systems. *Proceedings of the ASP-DAC 2001. Asia and South Pacific Design Automation Conference* (pp. 251 - 254). Yokohama, Japan: ACM Press, New York, NY, USA.

Swaminathan, V., & Chakrabarty, K. (2001, Sept). Real time task scheduling for energy aware embedded systems. *Journal of The Franklin Institute, 338*(6), 729-750.

Swaminathan, V., & Chakrabarty, K. (2002). Pruning-based energy-optimal device scheduling for hard real-time systems. *Proceedings of the tenth international symposium on Hardware/software Codesign* (pp. 175 - 180). Estes Park, Colorado: ACM Press New York, NY, USA.

Swaminathan, V., Chakrabarty, K., & Iyengar, S. S. (2001). Dynamic I/O power management for hard real-time systems. *Proceedings of the Ninth International Symposium on Hardware/Software Codesign, CODES 2001.*, (pp. 237 - 242).

The battery life challenge – balancing performance and power. (2004, Jan 1). (Intel) Retrieved Apr 8, 2007, from Intel: http://onlinetoolkit.intel.com/docs/busi/ExtendingBatteryLifeWP.pdf

The importance of energy. (2005). Retrieved Dec 3, 2005, from Europa Energy Research: http://europa.eu.int/comm/research/energy/gp/gp_imp/article_1081_en.htm

U.S. Department of Energy. (1999). *Home Appliance Buying Trends Survey.* DOE. D&R International, Ltd.

Ullman, J. (1975). NP-complete scheduling problems. *Journal of Computer and System Sciences, 10*, 384-393.

Ulrich, N. C., & Flagg, J. (2003, March). *Consumer Energy Awareness and Attitude Study.* Retrieved Jan 15, 2007, from Energy Home Improvements: www.myenergystar.com/documents/PressReleases/2004/2004Facts_Energy2.ppt

Wand, Y., & Weber, R. (2002, Dec). Information Systems and Conceptual Modeling - A Research Agenda. *Information Systems Research, 13*(4), 363-376.

Wang, M.-f. (1992). A dynamic task scheduling method for multiprocessor system. *Proceedings of the 1992 ACM/SIGAPP symposium on Applied computing: technological challenges of the 1990's* (pp. 840 - 845). Kansas City, Missouri, United States: ACM Press New York, NY, USA.

Xie, R., Rus, D., & Stein, C. (Dec, 2001). Scheduling Multi-Task Agents. *Proceedings of the 5th International Conference on Mobile Agents* (pp. 260-276). Atlanta, Georgia: Springer-Verlag London, UK.

Yahoo. (2008, Oct). *Despite slump, Yahoo plans new operations in Neb*. Retrieved Nov 2008, from http://www.usatoday.com/tech/products/2008-10-24-733224501_x.htm

Yang, P., Wong, C., Marchal, P., Catthoor, F., Desmet, D., Verkest, D., & Lauwereins, R. (2001, Sept). Energy Aware Runtime Scheduling for Embedded Multiprocessor SOCs. *IEEE Design & Test, 18*(5), 46 - 58.

Zhuo, J., & Chakrabarti, C. (2005). An efficient dynamic task scheduling algorithm for battery powered DVS systems. *Proceedings of the 2005 conference on Asia South Pacific design automation* (pp. 846 - 849). Shanghai, China: ACM Press New York, NY, USA.

Zhuo, J., & Chakrabarti, C. (2005). System-level energy-efficient dynamic task scheduling. *Proceedings of the 42nd annual conference on Design automation* (pp. 628 - 631). San Diego, California, USA: ACM Press New York, NY, USA.

Appendix A-1: The NITRD program's illustrative grand challenge.

No.	Grand Challenges
1	Knowledge Environments for Science and Engineering
2	Clean Energy Production through Improved Combustion
3	High Confidence Infrastructure Control Systems
4	Improved Patient Safety and Health Quality
5	Informed Strategic Planning for Long-Term Regional Climate Change
6	Nanoscale Science and Technology: Explore and Exploit the Behavior of Ensembles of Atoms and Molecules
7	Predicting Pathways and Health Effects of Pollutants
8	Real-Time Detection, Assessment, and Response to Natural or Man-Made Threats
9	Safer, More Secure, More Efficient, Higher-Capacity, Multi-Modal Transportation System
10	Anticipate Consequences of Universal Participation in a Digital Society
11	Collaborative Intelligence: Integrating Humans with Intelligent Technologies
12	Generating Insights from Information at Your Fingertips
13	Managing Knowledge-Intensive Dynamic Systems
14	Rapidly Acquiring Proficiency in Natural Languages

Appendix A-2: IT hard problem areas identified based on grand challenges.

No.	IT hard problem areas
1	Algorithms and Applications
2	Complex Heterogeneous Systems
3	Hardware Technologies
4	High Confidence IT
5	High-End Computing Systems
6	Human Augmentation IT
7	Information Management
8	Intelligent Systems
9	IT System Design
10	IT Usability
11	IT Workforce
12	Management of IT
13	Networks
14	Software Technologies

Appendix B: Survey of work done with approaches taken

Researchers	Year	Approach	Method Description
(Greenawalt, 1994)	1994	Statistical	Equation modeling for hard disk power management
(Hwang & Wu, 1997)	1997	Probabilistic	Exponential average method in conjunction with prediction miss correction and pre-wakeup mechanism
(Benini, Bogliolo, Paleologo, & De Micheli, 1999)	1999	Finite machines	Finite-state, model based on Markov decision process
(Simunic, Benini, & De Micheli, 1999)	1999	Finite machines	Modifications to Benini et al's method
(Eui-Young, Benini, & De Micheli, 1999)	1999	Probabilistic and Statistical	Adaptive Learning Tree Data structure
(Lu, Benini, & De Micheli, 2000)	2000	Exact solution [non probabilistic and non statistical]	Rearrange tasks executions to prolong device idle periods
(Swaminathan, Chakrabarty, & Iyengar, 2001)	2001	Exact solution	LEDES algorithm – Rearranges task executions (online)
(Swaminathan & Chakrabarty, Pruning-based energy-optimal device scheduling, 2002)	2002	Exact solution	EDS algorithm (online) – rearranges task execution (offline)
(Wang, 1992)	1992	Exact solution	A dynamic task scheduling method which extends the Round-Robin policy task scheduling.
(Ahmed & Chakrabarti, 2004)	2004	Exact solution	The authors propose a two phase algorithm with the objective of maximizing the residual charge and the battery voltage after the execution of the tasks. In phase 1 (offline) a battery aware algorithm schedules the tasks assuming WCET. In Phase 2 (online) the algorithm reassigns the voltage levels based on the additional slack generated because AET < WCET.
(Chowdhary & Chakrabarti, 16-18 Oct. 2002)	2002	Heuristic solution	The proposed algorithm maximizes battery life by shaping the current load profile. The shaping algorithm makes extensive use of voltage/clock scaling and is guided by heuristics that are derived from the properties of the battery model. This is for a-periodic task scheduling.
(Zhuo & Chakrabarti, Dynamic Task Scheduling Algorithm, 2005)	2005	Exact solution	A new battery aware dynamic task scheduling algorithm, darEDF, based on an efficient slack utilization scheme that employs dynamic speed setting of tasks in run queue. Comparison with lpfpsEDF, lppsEDF, lpSEH energy efficient algorithms is performed.
(Kim, Kim, & Min, 2002)	2002	Exact solution (*)	Energy efficiency of a DVS algorithm largely depends on the performance of the slack estimation method used. The proposed algorithm takes full advantage of the periodic characteristics of the task under priority-driven scheduling such as EDF.

(Zhuo & Chakrabarti, System-level energy-efficient dynamic task scheduling, 2005)	2005	Exact solution (*)	In a DVS system with multiple devices, slowing down the processor increases the device energy consumption. A dynamic task scheduling algorithms for periodic tasks that minimize system level energy (CPU + Device standby). The algorithm uses (1) optimal speed setting, which is the speed that minimizes the system energy for a specific task, and (2) limited preemption which reduces the number of possible preemptions.
(Jejurikar & Gupta, 2002)	2002	Heuristic solution	DVS based on slowdown factors can lead to considerable energy savings. An algorithm is proposed to compute static slow down factors for a periodic task set. It takes into consideration effects of blocking that arise due to task synchronization.
(Rao, Singhal, Kumar, Visweswaran, & Navet, 2004)	2004	Heuristic solution	Addresses the issues of making real-time DVS algorithms battery aware by using heuristics instead of computation-intensive battery models for making runtime scheduling decisions.
(Swaminathan & Chakrabarty, Real time task scheduling, 2001)	2001	Mixed Integer Linear Programming (MILP)	The proposed approach (for periodic tasks in real-time systems) minimizes energy consumed by the task set meets deadlines. The approach used is MILP.
(Rusu, Melhem, & Mossé, 2005)	2005	Heuristic solution	To achieve a variety of QoS-aware trade-offs the authors propose (a) a static solution that maximizes the system value assuming WCET and (b) a dynamic scheme that takes advantage of the extra energy in the system when worst-case scenarios do not happen. Three dynamic policies are shown. Algorithm is call MV-Pack
(Yang, et al., 2001)	2001	Combination – Genetic algorithm and MILP.	This task-scheduling method combines the low runtime complexity of a design-time scheduling phase with the flexibility of a runtime scheduling phase. The design time phase uses a genetic algorithm for scheduling where as the runtime phase uses a MILP algorithm
(Swaminathan & Chakrabarty, Effects of voltage-switching, 2001)	2001	Mixed Integer Linear Programming (MILP)	For workloads containing periodic tasks, the authors propose a mixed-integer linear programming model for the complete scheduling problem. For larger tasks sets, a extended-low-energy earliest-deadline-first (E-LEDF) scheduling algorithm is given.
(Shin, Kim, & Lee, 2001)	2001	Exact solution	An intra-task voltage scheduling algorithm is proposed which controls the supply voltage within an individual task boundary. It exploits the slack time to achieve a high-energy reduction. First it automatically selects appropriate program location for performing voltage scaling. Second, it inserts voltage-scaling code to the selected locations.
(Raghunathan, Pereira, Srivastava, & Gupta, 2005)	2005	Exact solution	Authors show how operating system directed DVS and DPM can provide tradeoff. A real-time scheduling algorithm is proposed that uses runtime feedback about application behavior to provide adaptive power-fidelity tradeoffs. Demonstration in the context of a static priority based preemptive task scheduler.
(Mishra, Rastogi, Zhu, Mossé, & Melhem, 2003)	2003	Exact solution	A new static and dynamic power management scheme. The new static scheme uses the static slack (if any) based on the degree of parallelism in the schedule. An online DPM technique is proposed to consider run-time behavior of tasks which exploits the idle periods of processors.
(Hu & Marculescu, 2004)	2004	Heuristic solution	Algorithm considers communication delays in parallel. Main contribution is formulation of the problem for concurrent communication and task scheduling and a heuristic to solve it.

Appendix C: Abbreviations

Abbreviations	Description
LPFPS	Low Power Fixed Priority Scheduling
WCET	Worst Case Execution Time
BCET	Best Case Execution Time
AET	Actual Execution Time
EET	Expected Execution Time
DVS	Dynamic Voltage Scaling
DPM	Dynamic Power Management
LCM	Least Common Multiple
AVR	Average Rate
WCEP	Worst Case Execution Path
ACEP	Average Case Execution Path
MILP	Multiple Integer Linear Programming
PDA	Personal Digital Assistant
IPC	Instructions per Second
EPI	Energy per Instruction
HPC	High Performance Computing
HPCC	High Performance Computing and Communications
HEC	High End Computing
PDA	Personal Digital Assistant
LAN	Local Area Network
GPS	Global Positioning System
NSF	National Science Foundation
BLAST	Basic Local Alignment Search Tool
BLAT	BLAST-Like Alignment Tool
EAS	Energy Aware Scheduling
MPI	Message Passing Interface
UNO	University of Nebraska at Omaha
UNMC	University of Nebraska Medical Center
IaaS	Infrastructure as a Service
PaaS	Platform as a Service
SaaS	Software as a Service
XML	eXtensible Markup Language
MILP	Mixed Integer Linear Programming

Appendix D: Listing of Figures

Appendix E: Listing of Tables