ENERGY AWARE SIZE INTERVAL TASK BASED ASSIGNMENT

ENERGY AWARE SIZE INTERVAL TASK BASED ASSIGNMENT.

BY

MAXWELL MOORE, B.Eng.

A THESIS
SUBMITTED TO THE DEPARTMENT OF DEPARTMENT OF COMPUTING AND SOFTWARE
AND THE SCHOOL OF GRADUATE STUDIES
OF MCMASTER UNIVERSITY
IN PARTIAL FULFILMENT OF THE REQUIREMENTS
FOR THE DEGREE OF
MASTER OF SCIENCE

Master of Science (2022) McMaster University
(Department of Computing and Software) Hamilton, Ontario, Canada

TITLE: Energy Aware Size Interval Task Based Assignment.

AUTHOR: Maxwell Moore
B.Eng. (Software Engineering),
McMaster University, Hamilton, Canada

SUPERVISOR: Dr. Douglas Down

NUMBER OF PAGES: xi, 62

Lay Abstract

The intention of this research is to be able to improve on existing size interval task-based assignment policies. We try to improve by turning servers off at key times to save energy costs, while not sacrificing too greatly in terms of mean response time of the servers, and in some cases even improving the mean response time through an intelligent re-balancing of the server loads.

Abstract

In this thesis we consider the impacts of energy costs as they relate to Size Interval Task Assignment Equally–loaded (SITA-E) systems. We find that given systems which have small and large jobs being processed (high variance systems) we could in some cases find savings in terms of energy costs and in terms of lowering the mean response times of the system. How we achieve this is by first working from SITA-E, wherein servers are always on to Electrically Aware SITA-E (EA-SITA-E) by seeing if it is beneficial to make any of our servers rotate between being on and being off as needed. When most beneficial to do so we will turn off some of the servers in question, after this is completed we reallocate some of the jobs that are on the servers that we decide will be cycling to servers that will remain on indefinitely to better use their idle time. This also lowers the mean response time below what we originally saw with SITA-E, by lowering the variance in the sizes of jobs seen by the servers with the longest jobs. These long–job servers are by far the most impacted by the variance of the sizes of the jobs, so it is very desirable to lower this variance. The algorithm contained here can provide benefits in terms of both energy costs and mean response time under some specific conditions. Later we discuss the effect of errors in our assumed knowledge of task sizes. This research contributes methodology that may be used to expand on EA-SITA-E system design and analysis in the future.

To all those people who helped me along the way

Friends, Mom, Dad, Family, Roommates, Teachers, and many more.

Acknowledgements

I have many people to thank for this thesis. Firstly I would like to thank Dr Down, who has been amazingly patient, as well as informative and supportive throughout the entirety of the process. I couldn't have asked for a better supervisor. Secondly I would like to thank my family, who always have been supportive but especially throughout my time writing this thesis were an integral part of maintaining my motivation and happiness. Thank you Dan, Chelsea, Mom and Dad. Finally I would like to thank my friends, roommates, and girlfriend who have proven to me time and time again they are willing to move mountains to make my life a happier one and support me in all I do. I really do mean it from the bottom of my heart when I say this wouldn't have been possible without each and every one of you. Thank you.

Contents

List of Figures

Notation and Abbreviations

Notation

j	Number of servers in the system, $\|M\|$.
M	M is the set of all servers, static or non-static.
N	N is a subset of M containing only the non-static servers.
S	S is a subset of M containing only the static servers.
λ	λ is the arrival rate to the entire system, λ_i is the arrival rate to server i.
t	t is a per-queue decision variable describing the allowable length of queue before the server will go from an OFF state to a SETUP state. This is irrelevant for static servers.
γ	γ is the rate at which servers move from SETUP to BUSY. This is irrelevant for static servers.
k	k is the lower limit on the bounded Pareto distribution.
p	p is the upper limit on the bounded Pareto distribution.

α The shape parameter for the bounded Pareto distribution. Lower α implies a higher variance – as α approaches one, the variance becomes infinitely large.

ρ ρ_i is the utilization of a server i, ρ is the average utilization of a server in the system.

β_1 β_1 describes the scaling factor of electrical costs against response time.

β_2 β_2 describes the scaling factor of cycling costs against response time.

E_{state} this describes the electricity costs associated with each state: BUSY, OFF, IDLE, and SETUP.

r_{state} this describes the ratio between the costs in a state (one of OFF, SETUP, or IDLE) and the cost of state $BUSY$.

ϕ This is the rate at which a server begins to shut down when it has no jobs, typically ∞ or 0 for non-static and static servers respectively.

Abbreviations

BP	bounded Pareto
SITA	Size Interval Task-Based Assignment
EA-SITA-E	Electrically Aware Size Interval Task-Based Assignment with Equal loads
PDF	Probability Distribution Function

Chapter 1

Introduction

1.1 Background

In recent years we have seen both a rise in energy costs and an increase in the amount of electricity used by data centers. Further, the amount of media attention as well as the amount of research funds for climate change have been trending upward. This has created an area of research focused on how to minimize energy costs while still hoping to see only small sacrifices made in terms of the systems that are being altered and their original goals. This research seeks to find ways to lower the energy impacts of data centers both monetarily and in terms of climate impact while still maintaining certain integral metrics of the system, in our case mean response time.

To study the effects of different control strategies we perform a multi–step process. The first step will be constructing a method by which to mathematically represent these complex systems. In this process we will be leveraging a combination of queueing theory and probability, as well as some key findings from previous works which are discussed in more detail in the literature review in Chapter 3.

The second step will be to ensure we are properly comparing and considering different cost metrics when measured against each other, so that we will not disproportionately favour savings for any single cost metric, such as energy costs, which will help us to analyze and compare these systems. For this purpose we create a cost function. Here it will be some linear combination of electricity costs, mean response time – the costs of unsatisfactory system performance – and cycling costs. Cycling costs are a measure of the impact of physically turning devices on and off – especially many times. This has been shown to impact the lifetime of a server, see Chen et al. [22], making it an important metric to consider when working with data centers and servers. At many times we consider cycling costs to be negligible and observe the amount of electricity savings created while only raising our mean response time by an acceptable amount. (What is considered an acceptable mean response time is described in more detail in Section 2.4).

The third step will help us to alter the system in order to reduce the overall costs. Firstly we will explain how we move from a Size Interval Task Assignment Equally-loaded (SITA-E) method to an Electrically Aware Size Interval Task Assignment Equally-loaded (EA-SITA-E) method. At a high level the only difference between EA-SITA-E and SITA-E is that we consider that some of the previously equally loaded always on servers should be cycled on and off under certain conditions to save on energy costs. At this point we will consider if we can further benefit by creating two separate classes of server which are equally loaded within their own class. One class will be the servers that always remain on, and the other will be the class of servers that will turn off when needed. Within these two classes we will still maintain an equal load between the individual servers, but not necessarily between the two classes of servers. Our goal with this method is to see if we can shift some of the weight from the systems that we decided would be turning off at times to the servers which will always remain on to better utilize the idle time of the servers which are always on.

We will see that for a wide range of systems we can see some level of energy savings, with only small sacrifices to mean response time; and in some specific cases we can actually see an improvement in both energy costs and mean response time.

Finally we will briefly examine the effects of error in our *a priori* knowledge, and see that with intelligent foresight they can be largely mitigated so long as an accurate error model is developed for the system in question.

Chapter 2

Preliminary Knowledge

In this chapter we will be discussing information that we feel is pertinent that the reader should be aware of before trying to understand the remainder of the thesis. This is intended as a ground level foundation for individuals who might not be well versed in the processes/methodology we use.

2.1 Queueing Theory

Queueing theory is an approach that mathematically captures real-world phenomena of queues of tasks. This can model a wide range of applications anywhere from wait times in a store where there is queueing to purchase groceries, to the amount of time it will take a process to complete all necessary sub-processes on different parts of a computer. All of these may be described in terms of which resources are being used by other processes when the process in question 'arrives' or more accurately is placed in the queue. Queueing theory is used heavily throughout the remainder of this thesis.

The notion of a random variable is the basic tool from probability theory that is used to describe quantities of interests in queueing systems. A random variable has a corresponding Probability Distribution Function (PDF) relating to the likelihood of certain outcomes. In our case it is typically the probability of a task finishing at a given time. Random variables go hand-in-hand with queueing theory and will also be heavily utilized throughout this thesis. That is to say: we will be referring to the task size or task size distribution throughout this thesis as a random variable. This particular random variable is an attempt to mathematically capture the performance of a single task that arrives to our system with a probability of taking different lengths of time using a server to finish. Random variables are a convenient means to model the statistical variation that can occur in a phenomenon. They are useful as a means to perform analysis. Of course when considering a queue it is important to be able to assess the response time of an individual task, or the average response time for a class of tasks given certain conditions. Basic knowledge of random variables is assumed at this point, but in the next section we discuss some specific concepts from queueing theory that inform our work.

2.2 Motivation

What we are trying to work towards for the majority of this thesis is a routing strategy. A routing strategy is a method by which to decide where a task arriving to a set of possible queues (or servers) will be routed to. These routing strategies are often used to try to lower the response time seen by a task, increase the fairness in how long a task might be delayed or even attempt to lower electrical costs incurred by the servers handling the task in question. When the size of a task is known *a priori* the Shortest Remaining Processing Time (SRPT) policy is an attractive strategy for scheduling [21]. In terms of response time this method has been shown to be optimal for a queue with a single server. SRPT dynamically allocates the task with the lowest amount of work to be done to the (usually single) server in question. This optimizes response time but at the potential cost of fairness, because large tasks can get stuck behind smaller ones. For modern computer systems, it has been shown that workloads do have the feature that they are a mixture of (very) large tasks and (very) small tasks. The Bounded Pareto distribution is often used as a model for task sizes and with this in mind, we focus on this model [6,23]. As a result the tendency for SRPT to be unfair is amplified greatly by the fact that we have such large variance in the tasks that arrive to the system, or even individual servers. Because long tasks tend to be very long it could be the case that many smaller tasks would interrupt a single large task repeatedly over a long period of time. In addition we wanted to consider a system that did not rely heavily on perfect preemption, which often is not possible in real world applications. If we were to consider SRPT without perfect preemption there could be significant performance degradation wherein larger tasks being preempted repeatedly by smaller tasks could result in a significant reduction in a server's useful capacity. With this reasoning in mind we decided to consider a basic M/G/1 model, with first come first served tasks that are never preempted. We now proceed to provide further details about the notion of an M/G/1 queue.

An M/G/1 queue relies on three concepts. The first is Markovian arrivals (the M), which means that for the server in question the system sees tasks arrive such that the time between arrivals follows a probability distribution that is exponential. The second portion the G (or general) means that the processing times probability distribution is any general distribution. Throughout this thesis we will be working with a bounded Pareto distribution as it exemplifies a high-variance distribution quite well. However the methodology outlined in this thesis still holds under any general distribution. Finally the most important piece is also the simplest. The "1" states that we are considering a single server without any other contributing servers. We will see that the systems that we study can be modelled as multiple independent M/G/1 queues. The associated formula for an M/G/1 queue's expected response time $E[R]$ is as follows:

$$E[R] = \frac{1}{\mu} + \frac{\rho^2 + \lambda^2\sigma_S^2}{2\lambda(1-\rho)} \tag{2.2.1}$$

where λ is the arrival rate, μ is the service rate, $\rho = \lambda/\mu$ is the utilization and σ_S^2 is the variance of the task size distribution.

As can be seen clearly in (2.2.1) variance is a significant contributor to the response time of a server, meaning that if we could find a way to lower it significantly we could see a non-negligible decrease in response time. For a system of parallel queues with single servers, Harchol-Balter et al.'s findings [6] show that the variance seen by an individual server can be lowered considerably by selectively sending tasks of certain sizes to specific servers; this is in contrast with other routing algorithms that base their decisions on quantities other than the size of the task in question, for example the queue lengths. Improving on this approach is the inspiration for this thesis. This class of routing strategies is called Size Interval Task-based Assignment (SITA) and will be explained in more detail in the literature review. We saw an opportunity where we would be able to produce possible system benefits, not to mention that we felt this coupled nicely with the idea of selectively increasing and decreasing the load on specific servers, another strategy to lower response time and/or electrical costs without significant costs to the system designer. In Section 4 we will explain the implementation.

When we bring all of this together we see a system of individual servers, each of which has had their variance lowered considerably when compared to methods that do not consider task size when making the decision for where tasks should be routed. This means they will have faster response times and lower overall costs. This also implies the possibility for more electrical savings by cycling servers off during downtimes, and back on when needed. From here onward we will describe the servers we decide should be turned off and on as needed as 'non-static' and any servers that we decide to leave on permanently as 'static'. This method is already used for individual servers by Maccio and Down [1], and will be discussed in greater detail later in Chapter 3. Finally we see that in some cases this leads to the ability to save a great deal of electrical costs while managing to lower mean response time, a win-win situation.

2.3 Problem Formulation

The model that this thesis addresses is a set, M, of parallel servers with size j. Each of these servers can be in one of four states: BUSY, OFF, IDLE, or SETUP. These servers will fall into either a subset of M called N containing non-static servers, or S containing static servers. Between these two sets of servers all of the servers must be included, but either of these sets could be empty. Of course no server can belong to both N and S. Formally:

$$N \subseteq M, S \subseteq M, (S \cup N) \equiv M, (S \cap N) \equiv \emptyset \tag{2.3.1}$$

Static servers will always remain on, while non-static servers will switch off as soon as they have no tasks en-queued and are not working on a task, and back on as soon as a threshold number of waiting tasks denoted t is reached (there may be a different threshold for each server in N). The justification for the choice of this policy for switching non-static servers on and off is discussed in more depth in Section 3.2.

The problem at hand is that the non-static servers will be turning on and off to conserve energy, however, they will use extra energy during the time it takes for them to turn on. To capture this effect we will be adapting a model from Maccio and Down [1]. A server moves from OFF to SETUP instantly. This will happen when the number of waiting tasks at server i exceeds a threshold value t_i – which we will discuss how to determine in more detail in Section 4.1. It will then move to BUSY after an amount of time that is exponentially distributed with known rate γ.

Note that a server does not enter SETUP without having a task queued, and as such cannot move from SETUP to IDLE. A server moves from BUSY to IDLE if it clears all of the tasks that it is processing. Finally, a server instantaneously moves from IDLE to BUSY if a task arrives or from IDLE to OFF if we wish to shut it down (this is assumed to happen instantly). Each of the states has an associated energy cost represented with E_{state}. It is useful to define the ratio between a given state and E_{BUSY}, denoted as r_{state}.

Tasks arrive to the system according to a Poisson process with a known arrival rate λ. Each server uses its own first in first out (FIFO) queue, and task sizes follow a known bounded Pareto distribution which is characterized by a probability density function given by:

$$f(x) = \frac{\alpha k^{\alpha}}{1-(\frac{k}{p})^{\alpha}} x^{-\alpha-1} \quad k \le x \le p, \text{and } 0 \text{ otherwise} \tag{2.3.2}$$

As with the Pareto distribution the bounded Pareto (BP) distribution has a single shape parameter, α, but has been modified to have a lower and upper bound, here representing the smallest and largest size of a task, k, and p respectively. This restricts what was originally an unbounded distribution.

To examine the efficacy of Size Interval Task-based Assignment (SITA) algorithms in the above situation, three problems must be solved to be able to fully describe the situation at hand. First we must decide on a set of intervals so that each server can have an upper and lower bound on the size of task it will serve. Any given server M_i will be assigned tasks from size x_i to x_{i+1}. For the BP distribution, these x_i satisfy the following:

$$x_i > x_{i-1}, \quad x_1 = k, \quad x_{j+1} = p \tag{2.3.3}$$

Secondly we must decide on which servers from M will have static behaviours and which will have non-static behaviours, in other words which will belong to set S and which will belong to set N.

For servers belonging to S we must decide on the threshold, the number of tasks that are allowed to queue before entering the SETUP state from the OFF state. Maccio and Down [1] provided a method to find the optimal t for any server given the system parameters.

Finally we must assess the efficacy of a given solution. For this reason we must formulate a metric or set of metrics. Maccio and Down [1] proposed a suitable set of metrics in the form of a set of cost functions tailored such that they can usefully show the impact of a solution in terms of the following metrics:

- Expected response time: denoted $E[R]$. This metric is meant to capture the level of customer satisfaction.
- Expected Normalized Energy Costs: denoted $E[E^N]$. This metric is meant to capture the typical running costs of a solution so we can attempt to optimize in terms of electrically aware servers.
- Expected Cycling Rate: denoted $E[C]$. This is a (typically optional) metric, denoting the rate at which servers turn on and off. The inclusion of a term involving $E[C]$ in the cost function can be used to model the physical wear and tear a system could experience in being turned ON and OFF repeatedly.

The cost function that Maccio and Down [1] proposed is a weighted sum of these three metrics, formally:

$$E[R] + \beta_1 E[E^N] + \beta_2 E[C] \tag{2.3.4}$$

This cost function allows for the relative importance of the three metrics to be weighed against each other using the β scaling factors.

Chapter 3

Literature Review

In this chapter we will be reviewing relevant research that will help the reader to understand the research sphere in which this research is grounded. It will also provide some reasoning and insight as to why we made particular choices in terms of presentation, process, or topic.

3.1 Task Size Aware Routing

An important assumption has been made in this research that the size of the task is known upon arrival, allowing different decisions to be made about which server it will be routed to within the system. This area has been explored by researchers quite thoroughly. In the case where task processing times are not known upon arrival, it is well known as referenced by Aalto [5] and originally found by Winston [8] and Weber [9] that a Join the Shortest Queue (JSQ) policy is optimal for response times in homogeneous server systems. However, this only holds true if the tasks' processing times are exponentially distributed [8] or have a non-decreasing hazard rate in general [9]. Harchol-Balter et al. [6] claim that heavy-tailed distributions such as the Pareto distribution model computer applications with much better accuracy than the exponential distribution, the latter being an assumption that is made for its mathematical convenience (analysis is much simpler). These claims are backed by studies on UNIX processes [10,11], I/O processes in general [12], and even file transfer speeds over the Internet [13]. Another feature of these types of tasks is that they can often be estimated quite well in terms of time or number of operations needed. This line of thought lends to the importance of studying situations in which solutions like JSQ – though eloquent – might perform far from optimally.

Harchol-Balter et al. [6] was instrumental in providing context and a base from which to start working towards a solution for this problem. They had multiple important insights that guided our problem formulation.

The first important insight made in [6] was that SITA-based policies worked very well in high variance situations. The intuition underlying this finding is that if one is to limit the variance seen by a single server by limiting the size of tasks that can possibly be seen by that server it will be highly beneficial in systems that see variance as a dominating term. It was shown as beneficial to limit task size variance in single server (M/G/1) systems as early as the 1930's by Pollaczek [19], who showed that the length of the queue of a server (positively linked with wait time, mean response time) was given by the expression (2.2.1). As one can clearly see from this expression, if everything else remains the same and the variance of the task-size distribution is limited, as is intended with SITA policies, we can achieve lower mean response times. When compared against Random, and Round Robin routing policies they found that SITA-E frequently performed better in a variety of situations [6]. Anselmi et al. [7] went on to prove that SITA-E is asymptotically optimal for mean wait time as the number of (homogeneous) servers grows infinitely large. This suggests that in situations with a large number of servers an equally loaded SITA system performs near-optimally. It is also shown in [7] that finding a SITA policy that minimizes wait time in applications with a finite number of servers is an intractable problem in that a specific algorithm cannot hold to be optimal across a generalized setting, meaning that if we take an arbitrary set of servers no one SITA policy can be shown to be best across all cases [7]. This means that equally loaded SITA can be suboptimal, and a method with which to balance the load between servers unequally that necessarily provides better performance metrics, or even regularly improves individual non-nominal metrics such as mean response time is difficult to provide [7]. Despite all of this Mor–Harchol–Balter et al. [6] show that when compared against similar dynamic methods like Least-remaining Processing Time (LRT), which was previously thought to be optimal in many cases, SITA-E can still outperform LRT by significant margins in situations where the variance is sufficiently high. Each of these insights is individually useful. First, it is important that SITA-E is asymptotically optimal because the intent of this research is to work in situations with large numbers of relatively consistent (in our case homogeneous) servers. In these types of situations we expect our solutions to be near optimal. Second, it is important to note that equally loading these servers is often not the best case [6,20], which is what led us to use a sort of modified binary-search/descent method with respect to the loads on the two sets of servers – N for non static and S for static – which are amongst themselves, equally balanced to find a load that improves the overall costs seen by our servers.

Harchol-Balter et al. [6] showed that a set of servers following a SITA policy reduces to independent FIFO M/G/1 queues, which helps us to mathematically manipulate them in a few interesting ways. Finally they also gave closed-form expressions which are relied upon heavily to develop subsequent findings, including:

- Equations to determine the correct boundaries to produce an equally utilized set of servers
- Equations to determine the functional attributes of a given server – variance, wait time, expected response time, etc.

- Equations to solve the proportion of tasks seen by a server

3.2 Electrically Aware Servers

Our research involves the incorporation of electrically aware servers. Many studies approach a similar set of questions as we do discussing the trade-off in performance with the energy costs of a system. Many of these works posit that instant-off servers are optimal under any task size distribution [1,2,3,4] and a wide variety of cost functions [1], meaning that under any task size distribution it is optimal to turn a server off as soon as it has become idle if the server in question is a non-static (on-off) server. As with our research these works approach the problem from a perspective that any server can be turned off instantly, but it would take time to turn any server back on. This helps to ground this area of study in reality, as most computer systems face some sort of startup time, lending to the reasoning that an instant-on system would not be as applicable (in which case analysis would be much simpler). There are some key differences between works of this type and our contributions to this research. Aalto and Lassila [5] discuss the same energy versus performance trade-off problem under similar conditions/assumptions. However they assume that the size of an individual task is unknown upon arrival, meaning that Size Interval Task-based Assignment (SITA) policies are not applicable. Other works such as Anselmi and Doncel [7] inform us that though some policies can be near–optimal or even optimal in cases where task sizes are not known upon arrival they can still perform much worse than SITA policies in cases where the task size is known, which implies that the knowledge of the size of jobs arriving to the system is a non-negligible piece of information, which we will manipulate to our benefit. Many works in the research sphere simply discuss optimal timing for turning individual servers off in M/X/1* (* means that it could be a processor sharing situation) situations such as Maccio and Down [1], and several works by Gebrehiwot et al. [2,3,4]. Some of these are in contexts of processor sharing, shortest remaining processing time (task size known dynamic applications) or other single queue assumptions. We present findings above and beyond these in multi-queue applications to show further optimizations based on the additional aspect of deciding between static servers and non-static servers. These are called *InstantOff* and *NeverOff* respectively to better communicate the optimality of instantly turning servers off when they are in the IDLE state. Having said all of this, as far as we are aware there has been little research done on multiple homogeneous queues of electrically aware servers where the task size (or a reasonable approximation thereof) is known upon arrival, as in this thesis.

Maccio and Down [1] had many findings that are useful for the results in this thesis. In particular, this thesis adopts the model that they used for single servers that are electrically aware, with four energy states IDLE, OFF, BUSY, and SETUP. Each of the states has separate associated energy levels meant to capture the different amounts of electricity that would be used by servers in different states. This is discussed in more detail in Section 2.4.

Maccio and Down [1] found in a single server case what threshold value t is optimal for the queue length before servers move from the OFF state to the SETUP state. This optimal value of t is a function of the underlying parameters. After t is found it is a simple matter of calculating whether it is more effective to leave a server static (in the set S) or have it be non static (in the set N).

Maccio and Down [1] provided a set of metrics that were appropriate for measuring the efficacy of a proposed solution in the form of a set of cost functions described in detail in Section 2.4.

Finally Maccio and Down [1] provided a set of observations about the behaviour of energy-aware servers. This includes proving that it is optimal in a single server setting for a non-static server to move to the OFF state from the BUSY state instead of ever entering the IDLE state because no tasks are present in the system. This observation helped to simplify some of the state-based mathematics because it implies that a non-static server will spend time in only three states: OFF, SETUP, and BUSY – it is important to note it will never enter the IDLE state. A server will be in the BUSY state for the same proportion of time as the utilization, ρ. This fact coupled with the total law of probability implies that the portions of time which the server is in the states OFF and SETUP must sum to $(1-\rho)$. Since static servers only exist in the two states BUSY and IDLE they are in those two states for a proportion of time equal to ρ and $(1-\rho)$, respectively.

3.3 Errors in *a priori* Task Size Estimation

For the bulk of this thesis some key assumptions are made. These assumptions include that we are aware of the distribution of task sizes *a priori*. We also know the size (processing time) of any individual task when it arrives to the system. This allows for us to make intelligent size-based decisions as to where the task will be routed in our system, as discussed in Section 3.1. However, with this extra information that we can leverage for improved performance there comes some additional complications. For an array of reasons it is problematic to assume that exact processing times are known. Firstly even in favourable cases such as web server systems where the size of file downloads, etc. may be known exactly (and their distributions estimated reasonably well based upon metadata information) it becomes problematic to model the processing time from this knowledge due to inherent latency of the data networking, and overhead such as sending and receiving acknowledgments [14,15]. This issue arises with even suitably conditioned systems, other systems can suffer from this same class of problems in even more dramatic ways.

This has led to an area of research that is based around whether or not partial information about a task size can be helpful. A way of representing this is as task processing times being given *a priori* with a certain error. Scheduling with processing times with error was first studied by Lu et al. [16], examining the performance of Shortest Remaining Processing Time (SRPT) – which has been shown to be optimal for a single server system by Schrage [17]. They also consider the Fair Sojourn Protocol, which is a modification of the processor sharing discipline. These are considered in single-server contexts, with the goal being to find out how severely these – in some aspects optimal – policies are affected by varying levels of error in estimation. Many works have attempted to lessen the affects of error in *a priori* task size estimation. A common approach to this is through a set of policies called Comparison Splitting, originally proposed by Jelenkovic et al. [18]. This is a class-based approximation to SRPT, which has multiple positive effects. Firstly it helps to lower the administrative storage overhead of the system, but more applicably it helps to make the system more robust to estimation error by having it be less likely for the error itself to be impactful. Mailach and Down [14] study the effects of multiplying the processing time by a random value to simulate some level of error. They also experimented with modifying the error distribution to represent mostly under or over-estimating. We use a similar method to generate errors but consider it in a different multi-server context in which we can hopefully modify our boundaries for the SITA policies to compensate for the errors in estimation found at the server level. Combining the idea of SRPT classes with the natural class-based SITA policies one can imagine that if the bounds were chosen intelligently one could heavily mitigate the negative effects of error in *a priori* estimation, which is what this research aims to show.

Chapter 4

Main Results

In this chapter we discuss the analytic findings based on our previously described systems, and cost metrics, and why some of those findings might be relevant to real–world decisions.

4.1 Electrically-Aware SITA and Rebalancing.

Electrically Aware Size Interval Task Assignment with Equal loads (EA-SITA-E) is an algorithm that combines the variance lowering aspects of Size Interval Task Assignment (SITA) policies (specifically here SITA-E) with the energy efficiency of the electrically-aware servers proposed in a single server environment by Maccio and Down [1]. We will describe the algorithm in the following paragraphs.

Firstly SITA-E will be implemented in the regular fashion. This will be achieved by splitting the task size probability density function $f(x)$ into a number of sections equal to the number of servers, j. For this to happen there must be $j + 1$ boundaries decided upon using $f(x)$. For our purposes we use the bounded Pareto function discussed in Section 2.3. In the remainder of the thesis, we assume that the parameter α for the bounded Pareto distribution is strictly greater than one.

Harchol-Balter [6] provided a means to find this set of cutoffs, which we adopt for our algorithm:

$$x_i = \left(\frac{j-i}{j} k^{1-\alpha} + \frac{i}{j} p^{1-\alpha} \right)^{\left(\frac{1}{1-\alpha}\right)} \tag{4.1.1}$$

These x_i are chosen such that all servers have utilizations equal to the utilization of the system as a whole – or, formally:

$$\rho_k = \rho_i = \rho \quad \forall \quad k, i \tag{4.1.2}$$

What this guarantees is that in steady–state, all servers will be equally loaded. Next we find the proportion of arrivals that are assigned to each server, P_i. Note that while any two servers i and k have equivalent loads they in general do not see the same proportions of arrivals. The intuition behind this is that since they are split by size into larger jobs and smaller jobs to ensure an equal load, one must ensure that the servers which handle smaller jobs handle a proportionally larger number of jobs.

We adapted this method for a generalized task size density $f(x)$ by using an iterative method in which we would find the first segment by leveraging knowledge of the lower bound of $f(x)$ and ensuring the integral of $f(x)$ from the lower limit $k = x_0$ to the first limit x_1 is equal to the value of ρ for the system, which is known ahead of time. Formally solving the following function for x_1 where X is the random variable representing the task size distribution:

$$\int_k^{x_1} f(x)x \ dx \ = \ \frac{\rho}{\lambda} = \frac{E[X]}{j} \tag{4.1.3}$$

After this we can use the second bound x_1 to find the next bound x_2, and so on as follows:

$$\int_{x_i}^{x_{i+1}} f(x)x \ dx \ = \ \frac{\rho}{\lambda} = \frac{E[X]}{j} \tag{4.1.4}$$

Now that we have the boundaries we can try to study if it makes sense for any of these servers as they stand to be non static, so that we will cycle them off and on or if it is still better overall to leave them as static servers. To do so we must first finish fully describing our system by determining the variable t_i, which is in general different for each server. This was shown by Maccio and Down [1] to provide the optimal outcome based upon the cost functions outlined in Section 2.3; recall that the cost function is:

$$E[R] + \beta_1 E[E^N] + \beta_2 E[C] \tag{4.1.5}$$

The relation that provides the t_i value is:

$$0 = t_i^2 \frac{\gamma}{2\lambda_i^2} + t_i \frac{1}{\lambda_i} - \left[\frac{1}{2\lambda_i} + (1-\rho)(\beta_1 r_{Setup} + \beta_2 \gamma) \right] \tag{4.1.6}$$

From here if t_i is not between zero and one both the floor and ceiling of t_i must be tested for the lowest cost in the later steps. If it is between zero and one then one will be used as zero does not make sense in this context. From here we will discuss what one would do with only one case, either floor or ceiling. It is assumed that the valuations are taken with every combination of floor and ceiling between each t_i for a total of $2^{|j|}$ different combinations to be evaluated in the end.

After the value of t_i has been found one must substitute all of the values in question for the server into the following expression, also given in [1]:

$$\beta_1(1-\rho)\lambda_i \gamma r_{idle}(t_i\gamma + \lambda_i) \leq \lambda_i(\lambda_i + t_i\gamma) + \gamma t_i(t_i - 1) + \beta_1(1-\rho)\lambda_i^2 \gamma r_{setup} + \beta_2(1-\rho)\lambda_i^2 \gamma^2 \tag{4.1.7}$$

If the inequality holds a server will be static, and if it does not hold the server in question will be non-static. One can see that if the idle costs were 0 it would necessarily be the case that the server would be static. This makes sense because idling of the server would cost nothing. Conversely if setup costs were sufficiently large the same would hold because it would cost the system too much in terms of startup costs.

One can start seeing the nuance with which this method aims to save costs, taking all energy costs into account, as well as the mean response time and how the mean response time might be harmed by different server policies.

This decision rule on making servers non-static is myopic in that it looks at each server in isolation and as such is suboptimal in the case of our parallel server system. Looking at the system as a whole can yield further adjustments that move closer to (system) optimality. We do so by directly applying the findings in [1] to consider further optimizations to fully realize EA-SITA-E. This form of optimization can only take place if it is the case that some of the servers are static, and some of the servers are not static, or formally :

$$S \neq \emptyset, N \neq \emptyset \tag{4.1.8}$$

From here one has the intuition that since these servers are operating under different circumstances we should try to shift the balance of the workload to potentially improve overall performance. We do this by creating two new load variables ρ_N and ρ_S for the non-static servers and the static servers, respectively. Of course it must still hold that $j * \rho = |N| * \rho_N + |S| * \rho_S$, that is to say that we have to assign all of the arriving jobs, and that we are simply shifting x_i to the left or right to increase or decrease the utilitization of the servers, while still holding the loads on static and non-static servers constant within their own class. We maintain the loads within the different classes of servers to preserve some of the benefits of SITA-E's ability to provide equal loads between servers while still leveraging the different behaviours between classes of servers.

To realize this method we choose a step size Δ_ρ which will represent the total amount of load that we want to shift from static servers to non-static servers, which must be small enough to maintain that each ρ_i must be less than one and greater than zero. There are two methods that we consider.

Method One "Post Check"

Recall that ρ_S is the utilization of a static server and ρ_N is the utilization of a non-static server. In this method we will initially pick a value that will satisfy our best case scenario, that is a Δ_ρ such that if we were to take load evenly from $j-1$ servers (as would be the case if there were only a single static server) and spread it across $j-1$ servers (as would be the case if there were only one non-static server). Of course both of these are unlikely to simultaneously be the case, but it provides a rough choice for the highest value that Δ_ρ can be without invalidating one of the constraints of the value of any ρ_i. In summary, Δ_ρ must be chosen to satisfy:

$$\frac{\Delta_\rho}{j-1} + \rho_S < 1, -\frac{\Delta_\rho}{j-1} + \rho_N \geq 0, \Delta_\rho > 0 \tag{4.1.9}$$

For every iteration done with this method the ρ_i values must be considered, and if any of them are outside of the zone defined by (4.1.9), one must repeat the process and choose a smaller Δ_ρ.

Method Two "Pre–Guarantee"

The motivation for this method is that it will guarantee that neither ρ_N nor ρ_S will become greater than one. For this guarantee to hold we must consider the worst case scenario from the perspective of both non-static and static servers. This means we consider what would happen if we put all of this additional load on a single standalone static server, and what would happen in the case where all of this load is taken off of a single standalone non-static server. Formally the choice must be made as follows:

$$\Delta_\rho + \rho_S < 1, -\Delta_\rho + \rho_N > 0, \Delta_\rho > 0 \tag{4.1.10}$$

Since this method guarantees that even if all of the load is distributed in the most problematic way there are still no issues, the servers cannot be made unstable or have negative utilization. This means that this can be used as a general method for guaranteeing Δ_ρ's validity.

After this the method is relatively simple to complete the rest of the algorithm. We simply shift the desired amount of load based upon the step size from the non-static servers to the static servers, and each set, static or non-static, works with their own independent SITA-E policies. After each balancing we consider the full cost equation again, if the cost has improved then we take another step of size Δ_ρ, if not we revert to the previous iteration of ρ_N and ρ_S, and we decrease Δ_ρ by a factor of $0 < b < 1$, and retry. Recall, however that one must consider the cost of every combination of $t_i^{+/-}$ (both the floor and ceiling for each t_i) and choose the lowest. The terminating conditions can be based on minimum or maximum loads on servers, a limitation on how small Δ_ρ can be, a maximum number of iterations, or any combination thereof.

By using this method we work in a methodical way through possible load values in terms of ρ_N (and ρ_S) and attempt to find a local minimum in terms of our cost function that is an improvement on the original solution. In our tests there were no instances in which we reached a local minimum that was not the global minimum. This leads us to conclude that the time-savings and simplicity of this method coupled with the unlikelihood of missed savings make it a desirable approach.

EA-SITA-E w/ Rebalancing Summary:

1. Find cutoffs x_i such that ρ_i for each server is equal, see (4.1.2)
2. Find P_i the proportion of arrivals seen by each server, i
3. Find t_i, the threshold queue length to turn on each server, i
4. if $t_i \leq 1$, set $t_i = 1$
5. else if $t_i > 1$ record t_i^-,t_i^+, the floor and ceiling of t_i respectively
6. Repeat the following with every combination of floor/ceiling t_i possible:
7. if (4.1.7) holds for i assign server i to S
8. if (4.1.7) does not hold for i assign server i to N
9. choose arbitrary Δ_ρ such that (4.1.10) is satisfied
10. subtract $\frac{\Delta_\rho}{j}$ from ρ_i where $i \in S$ and add it where $i \in N$
11. evaluate all cost equations per step 5
12. if any cost is lower, repeat from step 2
13. if all cost are higher, lower Δ_ρ by a factor between 0 and 1, if $\Delta_\rho < \epsilon$,(a predefined stopping criteria) terminate, otherwise repeat from step 2

4.2 Analysis

To find the full cost of our system we need to be able to find the energy costs, mean response times and cycle rate, which we will detail in the following paragraphs.

Energy Costs

Maccio and Down [1] provided a method by which to assess the cost of our system in terms of energy with the caveat that there was only a single M/G/1 server.We are already familiar with most of these different variables with one exception, ϕ, which is the shutdown rate of servers, and will be zero in the case of static servers (who will never shutdown, and as such have a zero shutdown rate) and will be infinity in the case of non-static servers, which will shut down instantly given an empty queue/server. The cost that they gave is as follows:

$$\rho(E_{BUSY}) + (1-\rho)\frac{\phi\lambda\gamma}{t\phi\gamma+\phi\lambda+\lambda\gamma}\frac{1}{\phi}(E_{IDLE}) + (1-\rho)\frac{\phi\lambda\gamma}{t\phi\gamma+\phi\lambda+\lambda\gamma}\frac{1}{t}(E_{SETUP}) \tag{4.2.1}$$

For our purposes for a given static server the expected cost would be:

$$\rho(E_{BUSY}) + (1-\rho)(E_{IDLE}) \tag{4.2.2}$$

and for a given non-static server the expected cost would be:

$$\rho(E_{BUSY}) + (1-\rho)\frac{\lambda\gamma}{t\gamma+\lambda}\frac{1}{t}(E_{SETUP}) \tag{4.2.3}$$

The above equations match our initial assumptions about the server. A static server only incurs busy and idle costs, at a rate of ρ and $1-\rho$ respectively. Non-static servers are slightly more complex, with the same busy cost but a setup cost that is determined by the proportion of time that a server is in setup.

Since for our policy we simply have individual FIFO queues we can simply add the costs of each individual server together, so the final total energy cost is as follows:

$$\sum_{i=1}^{j} \rho_i(E_{BUSY})+(1-\rho_i)\frac{\phi_i\lambda_i\gamma}{t_i\phi_i\gamma+\phi_i\lambda_i+\lambda_i\gamma}*\frac{1}{\phi_i}(E_{IDLE})+(1-\rho_i)\frac{\phi_i\lambda_i\gamma}{t_i\phi_i\gamma+\phi_i\lambda_i+\lambda_i\gamma}*\frac{1}{t_i}(E_{SETUP}) \tag{4.2.4}$$

The thresholds are found by solving (4.1.6) and ϕ_i by substituting t_i into (4.1.7), and examining the inequality as described at the beginning of this section. The utilization of server i, ρ_i is easily determined with some algebra, if all of the rules have been followed in the setup of the servers in EA-SITA-E fashion then server(s) $n \subseteq N$ will have $\rho_i \equiv \rho_N$ and server(s) $s \subseteq S$ will have $\rho_i \equiv \rho_S$. The constants γ, E_{BUSY}, E_{IDLE}, and E_{SETUP} are known and j is simply the number of servers in the system. Given this expression and the process of the algorithm EA-SITA-E itself as a road map one can fully define the (relative) energy costs seen by the system.

Response Times

The expected number of tasks at server i, $E[N_i]$, was given by Maccio and Down [1] as follows:

$$\begin{aligned} E[N_i] &= E[N_i^{M/G/1}] + \tfrac{\phi_i}{\phi_i+\lambda_i}(\rho_i\tfrac{(t_i-1)\gamma+\lambda_i}{\gamma}\tfrac{1}{2}\tau) \\ &+\phi_i\tfrac{(t_i-1)\gamma+\lambda_i}{t_i\phi_i\gamma+\phi_i\lambda_i+\lambda_i\gamma}(\tfrac{1}{2}-\rho_i-\rho_i\tfrac{\phi_i}{\phi_i+\lambda_i}(\tfrac{(t_i-1)\gamma+\lambda_i}{\gamma})-\tfrac{1}{2}\tfrac{\phi_i}{\phi_i+\lambda_i}\tau) \end{aligned} \tag{4.2.5}$$

$$\tau = (t_i-1)^2+(2t_i-1)\frac{\lambda_i}{\gamma}+\lambda_i^2\sigma_i^2 \tag{4.2.6}$$

where σ_i^2 is the variance of the processing times at server i.

Here, $E[N_{M/G/1}]_i$ refers to the typical formulation for the expected number of users in the system for an M/G/1 queue. From here an application of Little's Law provides $E[R_i]$ by multiplying $E[N_i]$ by $\frac{1}{\lambda_i}$ of course after this similar to (4.2.4) you need to sum over every server. Here, each term in the sum must be weighted by the proportion of arrivals that are sent to the corresponding server.

Cycling Costs

The cycling costs can be found easily through a variable meant to capture the rate of cycling as given in [1]. This takes the form of w_{rate}, a measurement for a single M/G/1 server's rate of cycling. This can be found with the following formula:

$$w_{rate} = (1 - \rho)\frac{\phi_i \lambda_i \gamma}{t_i \phi_i \gamma + \phi_i \lambda_i + \lambda_i \gamma} \tag{4.2.7}$$

Formally this is equivalent to the work-cycle rate i.e., the rate at which a server turns off and then back on. Later we will see that there are two interesting cases here, where ϕ is zero, and as such w_{rate} is zero, and where $\phi = \infty$, in which case it will simplify to (4.2.13) below.

Observation 1

There is a nonlinear trade-off in energy savings with respect to the change of load from static to non-static servers even as the number of servers grows to infinity, so long as the change in λ_i is non-zero. However, for our tests this always appeared to be close to linear. This assumes that there is no changes to N or S, i.e, static servers remain static and non-static remain non-static.

$$\text{define } x = (|S| * \rho_S) - (|N| * \rho_N)$$
$$\text{define } f(x) = E[E^N] \text{ given } x$$
$$\lambda_i \neq \lambda_i'$$
$$f(x) \not\propto x$$

In this setting, we additionally assume only non-zero service rates and a constant t. Consider the initial cost and we will derive savings based on the change in ρ_S, ρ_N, There are two sets of costs, those incurred by static servers and those incurred by non-static servers, formally:

$$E[E^N] = (|S| * \rho_S) * (E_{Busy}) + (|N| * \rho_N) * (E_{Busy}) + (|S| * (1 - \rho_S)) * (E_{IDLE}) + \quad (4.2.8)$$

$$(|N|) * (a * (E_{OFF}) + b * (E_{SETUP}))$$

Here $a + b$ must be equal to $(1 - \rho_N)$ by the law of total probability. So where are there possible savings and where are there added costs? As ρ_S increases and ρ_N decreases proportionately there is no change in the amount of 'busyness' (server time a task needs) in the system - that is to say every task still has to be served, whether by a static server or a non-static one. As such we can ignore these terms, knowing that there will be no net-change between them. This leaves us with only three terms.

$$(|S|) * (1 - \rho_s) * (E_{IDLE_s}) + (|N|) * (a * (E_{OFF_n}) + b * (E_{SETUP_n})) \quad (4.2.9)$$

Note that since each different non-static server will have a different proportion of time spent off or in setup - meaning each non-static server has a different value for a and b we cannot simply drop $|N|$ and $|S|$. We see that as we increase the load on the static servers (as x increases) we linearly decrease the amount of idle electricity costs seen by that server set. Maccio and Down [1]. provided a method for finding a and b through w_{rate}

$$a = \frac{t w_{rate}}{\lambda} \quad (4.2.10)$$

$$b = \frac{w_{rate}}{\gamma} \quad (4.2.11)$$

$$w_{rate} = (1 - \rho_N) \frac{\phi_i \lambda_i \gamma}{t_i \phi_i \gamma + \phi_i \lambda_i + \lambda_i \gamma} \quad (4.2.12)$$

As the servers in N are non-static, $\phi_i = \infty$ and w_{rate} simplifies to:

$$w_{rate,i} = (1-\rho_N)\frac{\lambda_i\gamma}{t\gamma+\lambda_i} \tag{4.2.13}$$

and substituting this we can find:

$$a_i = (1-\rho_N)\frac{t\lambda_i\gamma}{(t\gamma+\lambda_i)\lambda_i} \tag{4.2.14}$$

$$b_i = (1-\rho_N)\frac{\lambda_i\gamma}{(t\gamma+\lambda_i)\gamma} \tag{4.2.15}$$

Simplifying:

$$a_i = (1-\rho_N)\frac{t\gamma}{t\gamma+\lambda_i} \tag{4.2.16}$$

$$b_i = (1-\rho_N)\frac{\lambda_i}{t\gamma+\lambda_i} \tag{4.2.17}$$

This makes a great deal of sense – they are the two pieces of a whole that add up to $(1-\rho_N)$ which is what we were looking for. If we consider only the static server side it seems like we would end up with a linear trade-off, however we have to consider the fact that as we lower the number of jobs seen by the non-static servers there will be an increase or decrease in the number of times we see it switch from OFF to ON, and thus a disproportionate increase in the amount of time spent in OFF, conversely for some servers we may see a disproportionate decrease in OFF time versus SETUP time, this depends on whether λ_i is increasing or decreasing for server i. This will result in either super-linear or sub-linear benefits depending on the situation. Since λ_i in (4.2.16) appears only in the denominator, versus (4.2.17) where it appears in the numerator and the denominator, if λ_i is increasing for server i we will see that a_i lowers proportionally more than b_i will, leaving us with sub-linear profits as we see a larger proportion of SETUP time on server i, all other things held constant. Conversely if

λ_i is decreasing we see super linear profits as the non-static servers spend a smaller portion in the SETUP state versus the OFF state. Later we will see that in our tests the tradeoff is very close to linear. This indicates to us that this nonlinear effect is not overly pronounced. The only case wherein the tradeoff is actually linear is if λ_i does not change - the small change in λ_i that we see in the experiments supports the change being near linear. In this case we would see a perfectly proportional lowering of both a and b, with a linear trade off in terms of change in ρ_N versus energy savings.

The next result shows that in a two-server system with high task size variance, significant savings can be achieved by our method.

Lemma 1

Let σ_i^2 be the variance of the task size distribution at server i, and $f = \frac{\sigma_2^2}{\sigma_1^2}$. Consider a two server system following SITA-E where server one is static and server two is non-static. Also, let $E[R']$ be the expected response time after EA-SITA-E transforms SITA-E and$E[R]$ be the expected response time before the transformation takes place. We then have for some $0 \leq r < 1$

$$\lim_{f \to \infty} E[R'] = r * E[R], 0 \leq r \leq 1 \tag{4.2.18}$$

Proof:

$$E[R_1] = \frac{1}{\mu_1} + \frac{\rho_1 + \lambda_1^2 \sigma_1^2}{2\lambda_1(1-\rho_1)} \tag{4.2.19}$$

$$E[R_2] = \frac{\lambda_2^2 \sigma_2^2}{2\lambda_2(1-\rho_2)} + \Delta$$

Computing the overall mean response time where Δ groups together terms that are independent of σ_2^2:

$$E[R] = p_2 * E[R_2] + (1-p_2) * E[R_1] \tag{4.2.20}$$

Again, gathering terms that are independent of σ_2^2 in Δ, we can conclude that for some $r < 1$, The expression (4.2.20) can then be rewritten as:

$$E[R] = \frac{p_2 \lambda_2^2 \sigma_2^2}{2\lambda_2(1-\rho_2)} + \Delta$$

Cancel λ_2 in the denominator:

$$E[R] = \frac{p_2 \lambda_2 \sigma_2^2}{2(1-\rho_2)} + \Delta$$

A similar expression can be derived after rebalancing:

$$E[R'] = \sigma_2^2 * \frac{p_2' \lambda_2'}{2(1-\rho_2')} + \Delta \tag{4.2.21}$$

Note that we have:
$0 \leq p_2' \leq p_2, 0 \leq \lambda_2' \leq \lambda_2, 0 \leq \rho_2' \leq \rho_2,$
From these relations it follows that: $0 \leq \frac{p_2' \lambda_2'}{2(1-\rho_2')} \leq \frac{p_2 \lambda_2}{2(1-\rho_2)}$
which allows us to write

$$E[R'] = \sigma_2^2 r \frac{p_2 \lambda_2}{2(1-\rho_2)} + \Delta, 0 \leq r \leq 1 \tag{4.2.22}$$

We then have:

$$lim_{f \to \infty} \frac{E[R']}{E[R]} = r, 0 \leq r \leq 1$$

□

When the service time variance is dominant in the expected response time of a system the equations that describe the system become inherently much more simple, and we can see that under certain circumstances 're-balancing' a system can be very worthwhile. Consider what happens to the above equations when a re-balancing takes place. Since the variance is still normalized in these equations both σ_1^2 and σ_2^2 can increase by some amount.The arrival rate λ_1 will rise, and λ_2 will lower. The service rates μ_1 and μ_2 may raise or lower, but again are not dominant terms here; similarly with ρ_1 and ρ_2. So the important terms here are $\lambda_2^2\sigma_2^2$ and $\lambda_1^2\sigma_1^2$ when λ_2 is lowered the expected response time is lowered by some factor, while raising σ_1^2, σ_2^2 and λ_1 each result in small increases in the expected response time, resulting in a large decrease in mean response time. Of course this result is only for two servers, but having said that it shows under which circumstances this type of re-balancing could possibly be beneficial both in terms of energy costs and mean response time, such as with very heavy-tailed systems like the bounded Pareto system. Lemma 1 demonstrates that there is nearly linear energy savings if a re-balancing takes place. This suggests that not only will our algorithm lower energy costs it also implies that it can simultaneously lower mean response times. Note that while one would think that this proof could be extended to an arbitrary number of servers it cannot, because when considering an arbitrary number of servers both the endpoints of the intervals defining the jobs that a server will see can move, where as in this particular example only the lower or upper bound can be shifted. This guarantees that when the utilization of a server is lowered or raised it is necessarily because it sees more or less jobs respectively (the jobs from the other server), as opposed to the possibility that perhaps they are seeing less job arrivals overall, and just larger jobs – making their utilization higher, as can be the case when both x_{i-1} and x_i can vary.

Chapter 5

Numerical Results

In this chapter we will be discussing the findings in a numerical sense, where we will try to draw meaning from several sets of example systems. We will be including both systems with a negligible error in the expected time for a task to complete, and some amount of error that we add to the *a priori* approximations.

5.1 Systems with Negligible Error

Here we provide insights by studying a range of specific instances of our problem. The series of systems was constructed with the intention of producing certain effects to demonstrate the ways in which the system is capable of saving energy, or cost overall. For these experiments a range of values was produced for each of the separate variables, that while demonstrating the benefits of our research keep us grounded in reality. These values are listed in the following table:

Variable Name	Values in tests
β_1	Low ($\beta_1 < 0.05$), Medium ($0.05 < \beta_1 < 1$), high($1 < \beta_1$)
γ	0.01,0.0666667, 0.2,1,100
ρ	0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9
j	8
k	0.0001
p	10^6
α	1.2
β_2	0

The value β_2 was chosen to be zero so that the cycling of the system and the wear associated with that would not be a focus of the results, and to simplify the trade-offs that were being examined. Note also that here the β_1 values are split into categories, low, medium and high. The reasoning behind this is that since $E[R]$ and $E[E^N]$ can have varying units associated with them they lose tangible meaning when compared to each other. For this reason we modify β_1 to be a value such that their relative cost contributions are similar by performing a modified search over β_1 while holding the other experimental values constant until a valuation of β_1 is found such that:

$$\mid (E[R] - (\beta_1 * E[E^N])) \mid < \epsilon_{\beta_1}, \epsilon_{\beta_1} = 10^{-8} \tag{5.1.1}$$

It should be noted that the values of β_1 are calculated using regular SITA-E – how one would envision a system before it has been modified by any of the contributions in this thesis. This gives us a reasonable starting point, picking a β_1 that scales the $E[E^N]$ so that the weighted contribution is equivalent to $E[R]$. From here β_1 is scaled by a factor of 16 and 1/16 to create "high" and "low" valuations. Thus this splits the system into categories of “relatively high”, “relatively equal”, and “relatively low” energy costs.

5.2 Impact of Increasing System Utilization

When ρ is small we see an increased ability to save electricity by making energy–aware decisions, as such the trend of using energy–aware servers shows higher savings at lower ρ. This stands to reason because the costs that we are decreasing are IDLE costs of the static servers when we decide to make them non-static, of course the IDLE costs are greatest when there are the fewest tasks and the servers are idling for the longest periods of time. This couples with another fact that when the utilization of a specific server is lower it typically sees lower setup costs due to the fact that it cycles less frequently, and as such spends less time in that state, and more time in the OFF state (as we know that the same amount of time must be spent in the busy state, and no time will be spent idling). This leads to the near linear trade–off in electrical costs that were discussed in Observation 1:

$$w_{rate} = (1-\rho)\frac{\phi_i\lambda_i\gamma}{t_i\phi_i\gamma + \phi_i\lambda_i + \lambda_i\gamma} \tag{5.2.1}$$

If we are comparing two non–static servers we set $\phi_i = \infty$ – as we know shutdown rate being infinite is optimal in non-static servers. In this case w_{rate} is once again simplified as follows:

$$w_{rate} = \frac{(1-\rho)\lambda_i\gamma}{t_i\gamma + \lambda_i} \tag{5.2.2}$$

Since ρ decreasing to zero implies λ_i will do the same and t_i is one or above then w_{rate} necessarily approaches 0 as ρ approaches zero, meaning that we would expect to see smaller increases in time spent in setup for lesser loaded servers than we would see in moderately loaded servers. What this means for our system is that the cost of making a server non-static is somewhat mitigated, making the benefits even more obvious in this case.

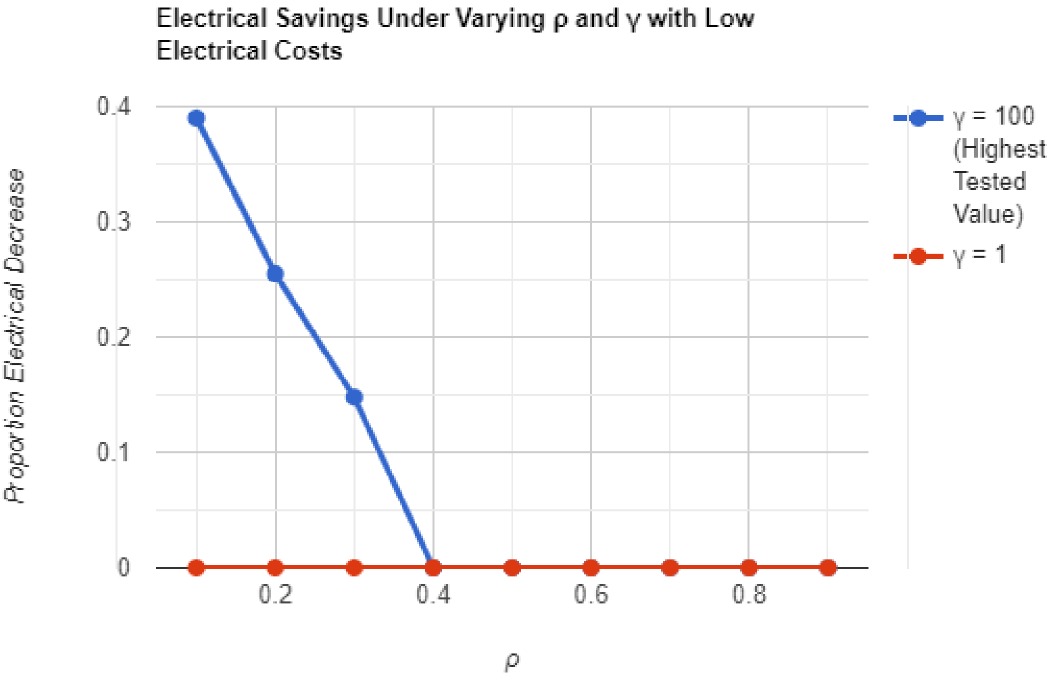

Figure 5.1: Electrical Savings with Low Electrical Costs

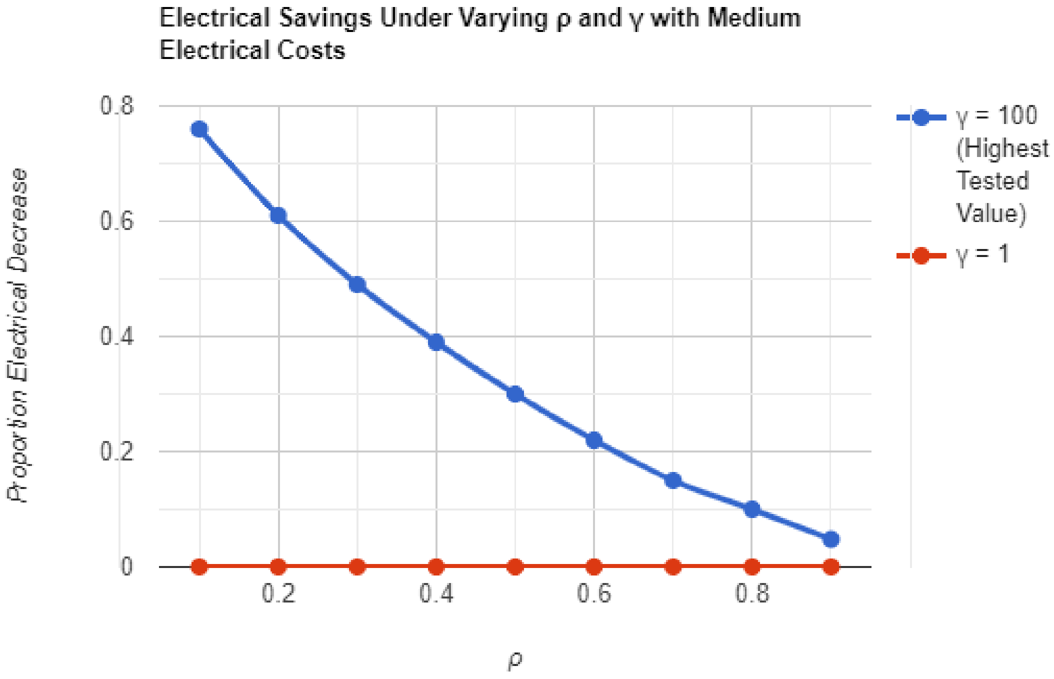

Figure 5.2: Electrical Savings with Medium Electrical Costs

Many trends can be seen in the adjacent graphs. Firstly and most clearly one can see that there is more idle time present in the lower ρ values and by this same token more energy savings to be had by eliminating this idle time. One can also see that typically higher energy valuations come with more energy savings, unsurprisingly so. Finally, and perhaps most interestingly we see that even low energy valuations can still see savings, and high energy valuations with low ρ can still see no savings. This is due to the effect of γ. This means that in the case where γ is 100 the startup rate is fast enough for most of the systems that we considered, and as such we will see savings even in such circumstances. Conversely speaking when γ is $\frac{1}{100}$ our startup rate is so slow that we will never see savings, even in cases where there appear to be such opportunities.

In short there is more opportunity provided for energy savings by shifting from

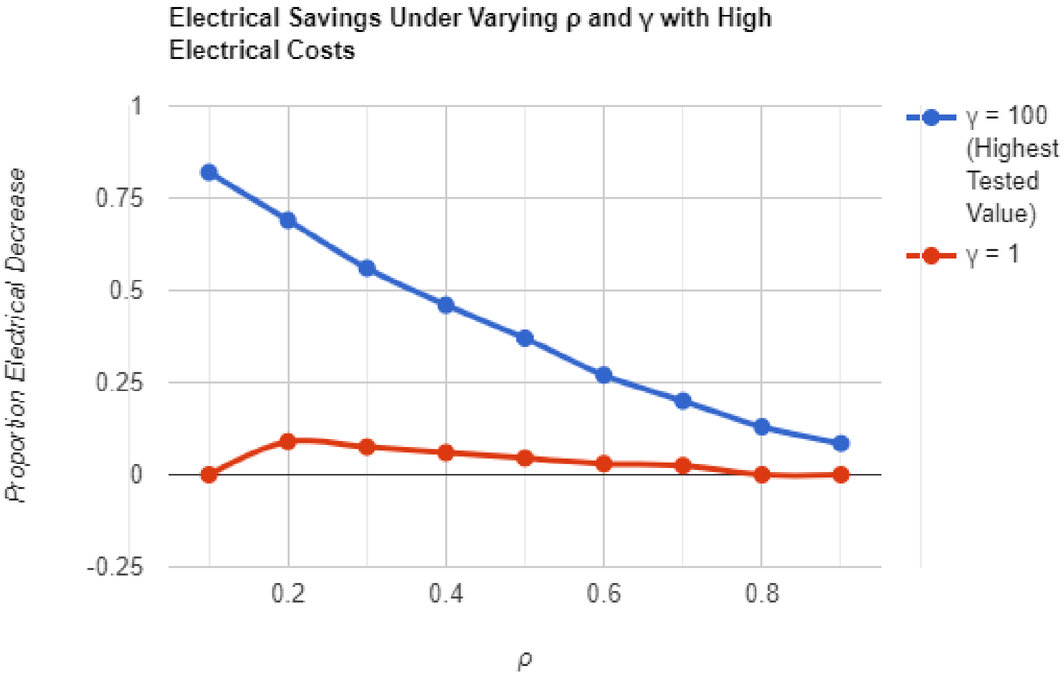

Figure 5.3: Electrical Savings with High Electrical Costs

SITA-E to EA-SITA-E policies in lower loaded systems, and in systems where we more heavily value energy savings. However there is always a dependence on the startup rate of the system, which should likely be a primary consideration when approaching this type of system.

Of course if we look only at the energy costs we are only being told half the story the mean response time is important too, and here is where we see the inherent trade-off in our approach.

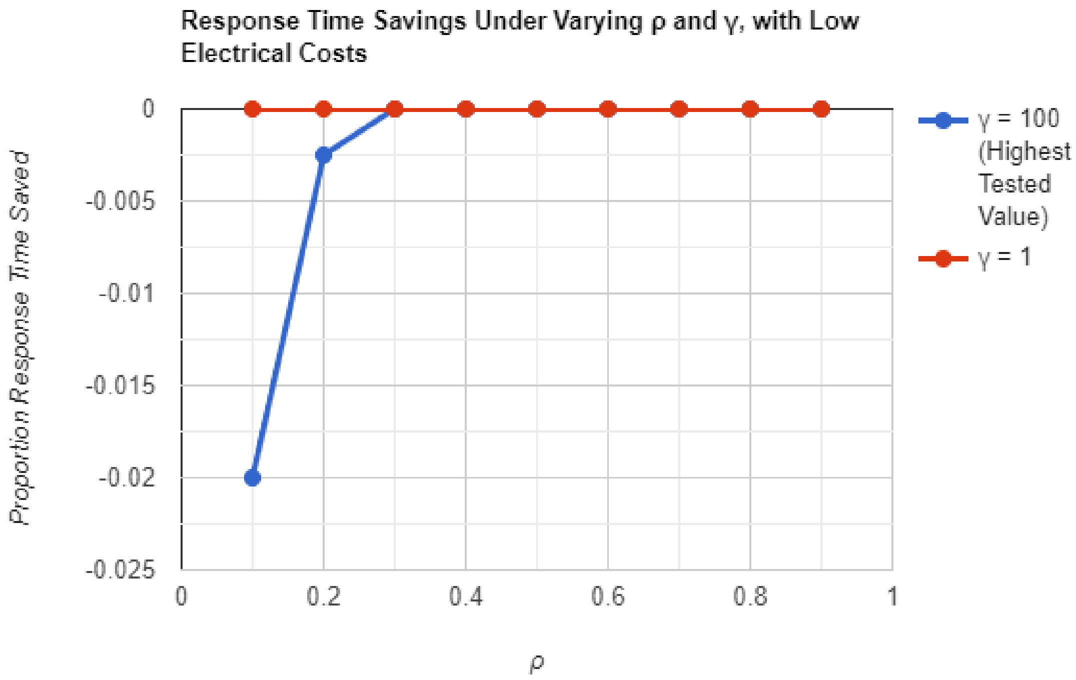

Figure 5.4: Mean Response time Savings with Low Electrical Costs

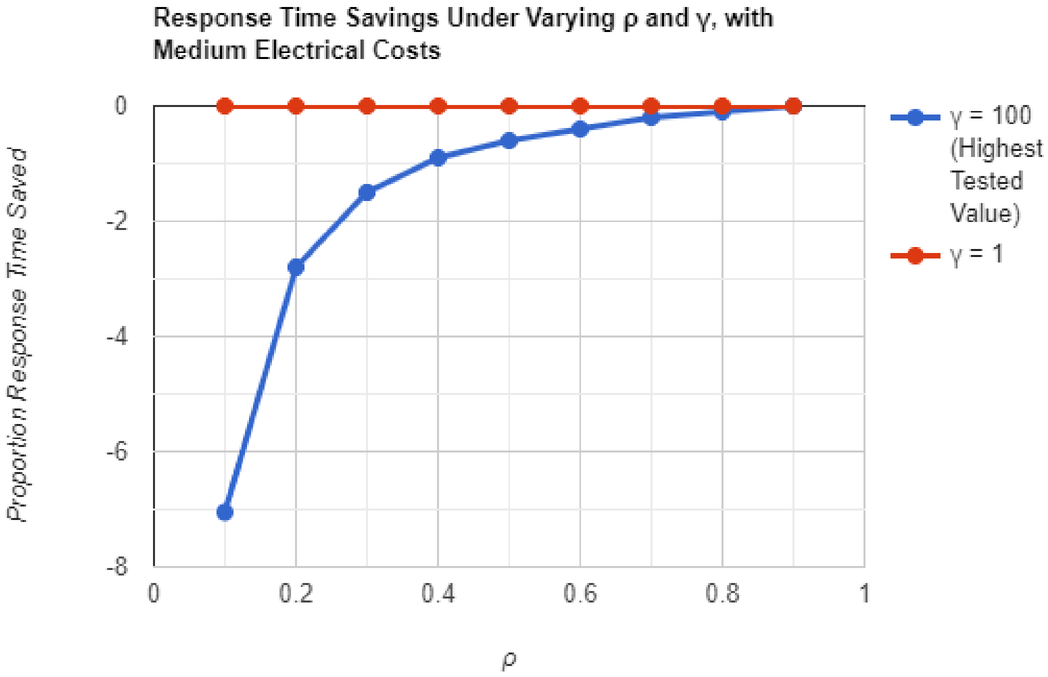

Figure 5.5: Mean Response time Savings with Medium Electrical Costs

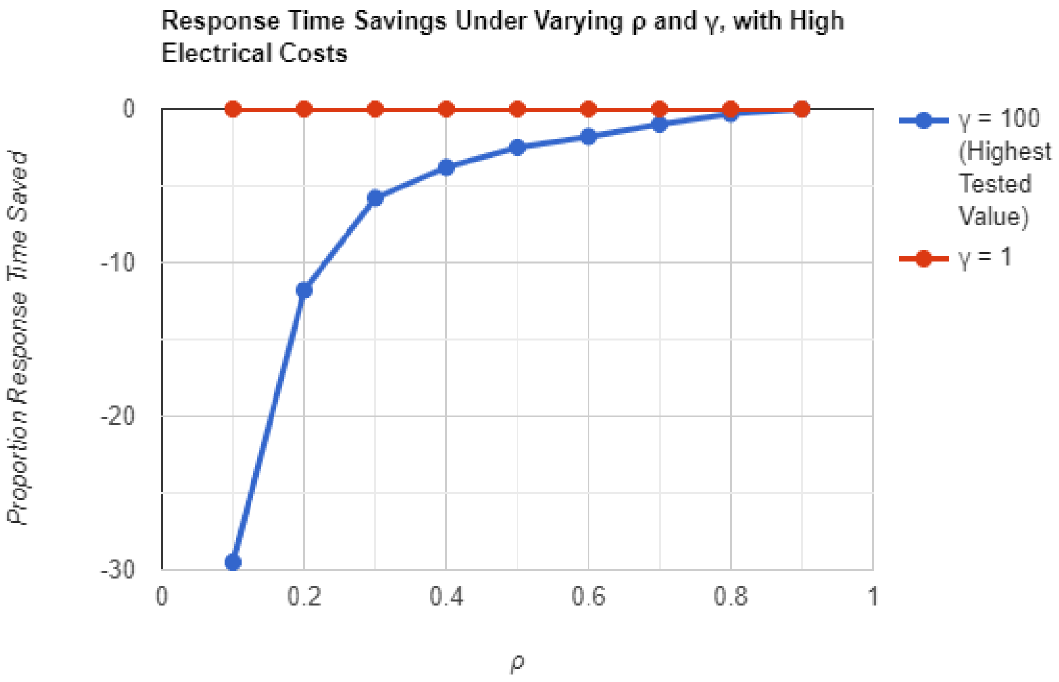

Figure 5.6: Mean Response time Savings with High Electrical Costs

As seen in the graphs above the response time can be hurt quite badly. Note that when the value is -30 this means that the expected response time became 30 times worse. At first glance this might not seem like a promising result but one needs to contextualize the losses in mean response time with the gains in energy savings. For instance the occurrence that is 30 times worse in Figure 5.4 corresponds to a savings of nearly 85 percent of our energy costs, as seen in Figure 5.1. Of course if you are considering the cost benefit analysis between the two cost metrics the lower that you weigh energy costs the less likely you are to see marginal energy savings paired with huge expected response time costs. One can see this validated as with low energy costs there are still savings at low ρ despite there being almost no difference in expected response times. The expected response times in these cases and some others are similar to such a degree that the values are hard to see in the above graphs, and as such any values with an increase in expected response times below a factor of 1 of the previous mean response time have been summarized below.

The Mean Response Time Increase as a Proportion of the Mean Response Time	Energy Reduction in Terms of Proportion of the Original Energy Costs	Energy Valuation (Low, Medium, or High)
0.02027	0.38792	Low
0.00192	0.25334	Low
0.00012	0.14442	Low
0.85107	0.38699	Medium
0.52284	0.29999	Medium
0.32131	0.22198	Medium
0.18788	0.15402	Medium
0.09741	0.094526	Medium
0.035792	0.04260	Medium
0.00040	0.08600	High
0.00024	0.07016	High
0.00015	0.05625	High
0.00009	0.04397	High
0.00006	0.03308	High
0.00004	0.19127	High
0.40198	0.12089	High
0.14876	0.05709	High

The above table shows the response time costs associated with the energy savings we see as in the above graphs in Figure 5.1, noting the energy cost (β_1) valuations.

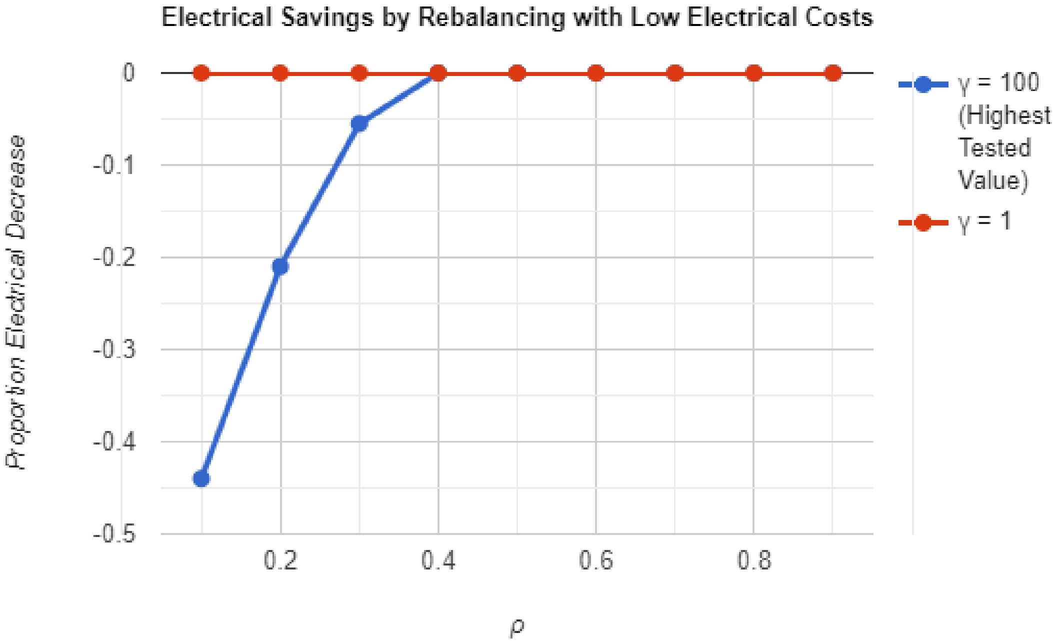

Figure 5.7: Rebalanced Electrical Savings with Low Electrical Costs

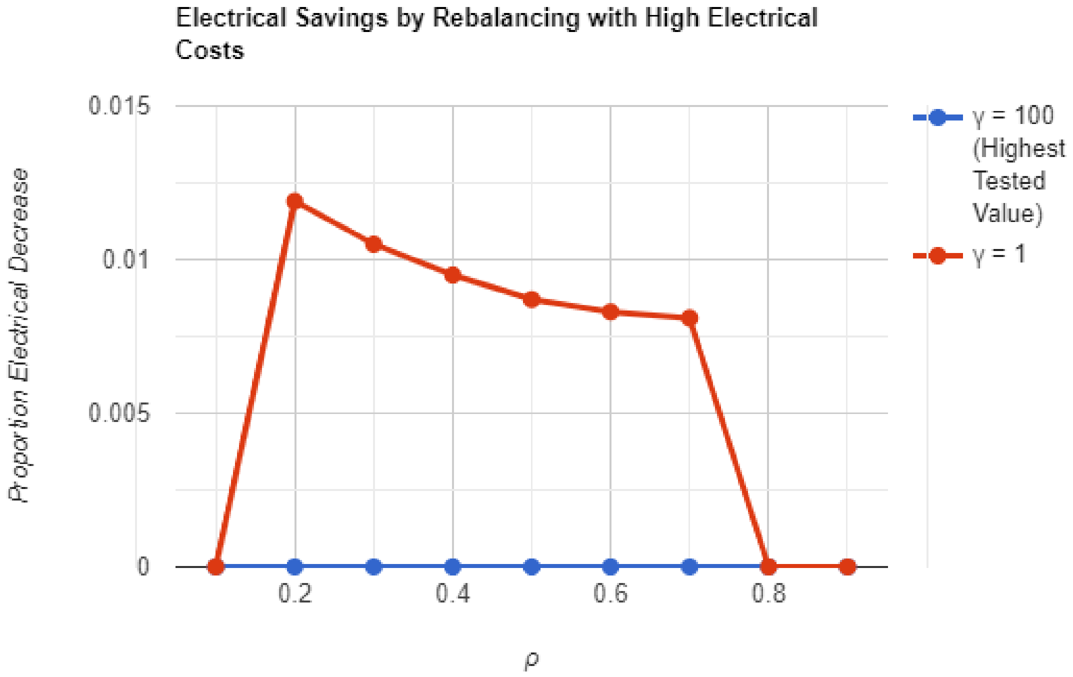

Figure 5.8: Rebalanced Electrical Savings with High Electrical Costs

Above we see the amount of savings that we can further achieve by switching from EA-SITA-E to re-balanced EA-SITA-E in terms of energy costs. Here we see that as one might expect there are actually energy costs to re–balancing given a low energy valuation, as opposed to savings. We see in Figure 5.9 this is because of an opportunity to gain in terms of expected response time, a trade–off that can be seen to be worthwhile because of the relatively high expected response time valuation. However the more interesting case is in Figures 5.8 and 5.11 where there are high energy valuations. One can see that we have saved both in terms of energy costs as well as in terms of mean response time. This stands to reason as given such high energy valuations we will see many non-static servers, and as such the benefit of off–loading tasks from the non-static servers to the static servers and the correspondingly less time spent in the SETUP state is greatly exaggerated. Note here as well that the higher start–up rate system does not experience any of these same benefits, because the start–up rate was not negatively impacting our system enough in the first place for the re–balancing to have been worthwhile. Not only this but that same reasoning has an effect on the expected response time of the system too, so we can simultaneously see positive results in terms of both. As the system utilization approaches one one can see that this reasoning ceases to be relevant, as in all likelihood increasing the load on some static servers begins to outweigh the importance of lowering the number of start–ups seen by the non–static servers.

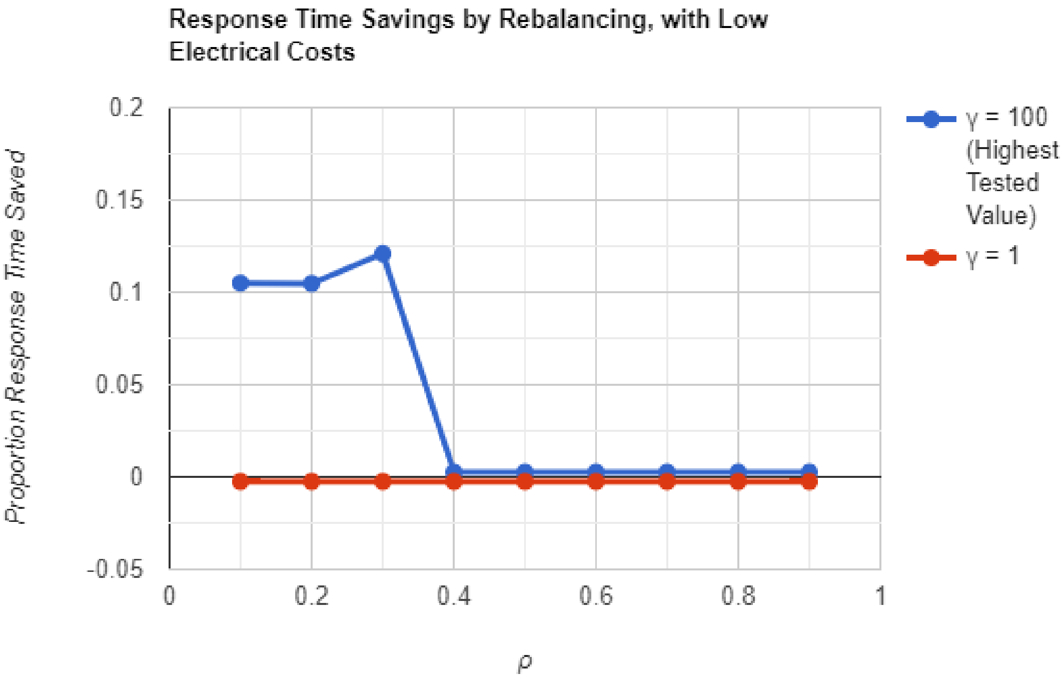

Figure 5.9: Rebalanced Mean Response time Savings with Low Electrical Costs

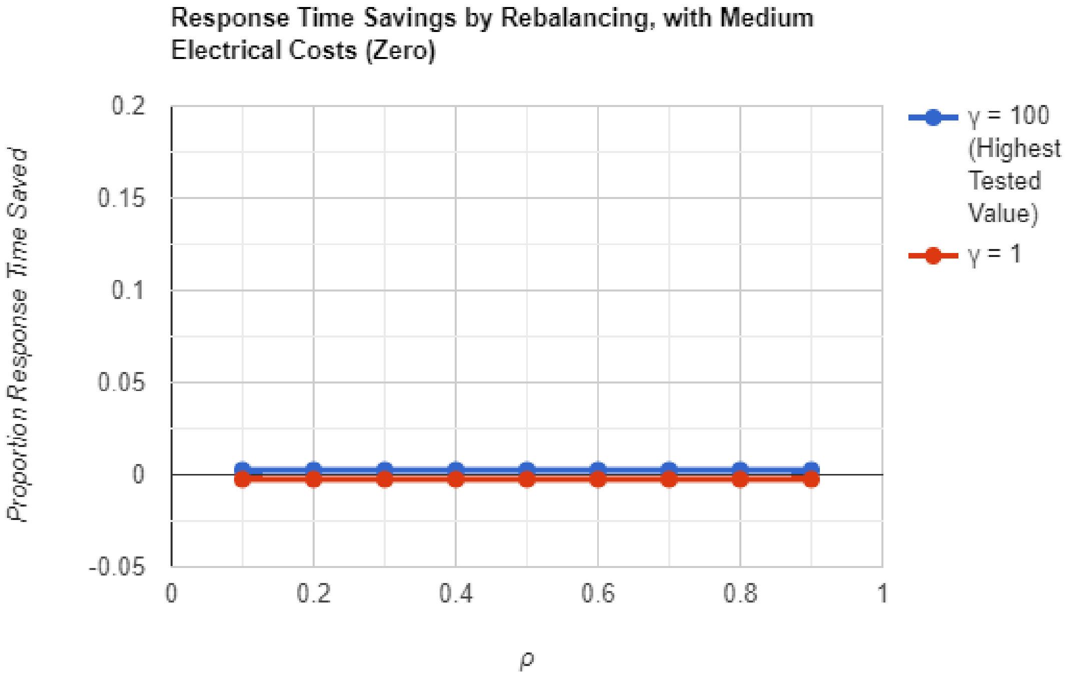

Figure 5.10: Rebalanced Mean Response time Savings with Medium Electrical Costs

As we can see particularly in Figure 5.11, where re-balancing is really effective is in its ability to lower the mean response times a great deal when dealing with high electrical costs from the start (especially at higher ρ). All of this takes place while still maintaining that energy costs are lowered overall. Once again we see how pronounced the unlikeliness of re-balancing being possible is in the first place, but for the first time we see large savings, in at least some forms. As one can see these savings increase super-linearly with increasing ρ, the reason for this is of course that the system sees such high variance for servers assigned longer tasks, which we are decreasing the load on and comparatively negligible variance on short-task servers, which we increase the load on to compensate for, we would expect to see this case because of our findings in Observation 1. Of course this is coupled with the aforementioned decrease in start–ups.

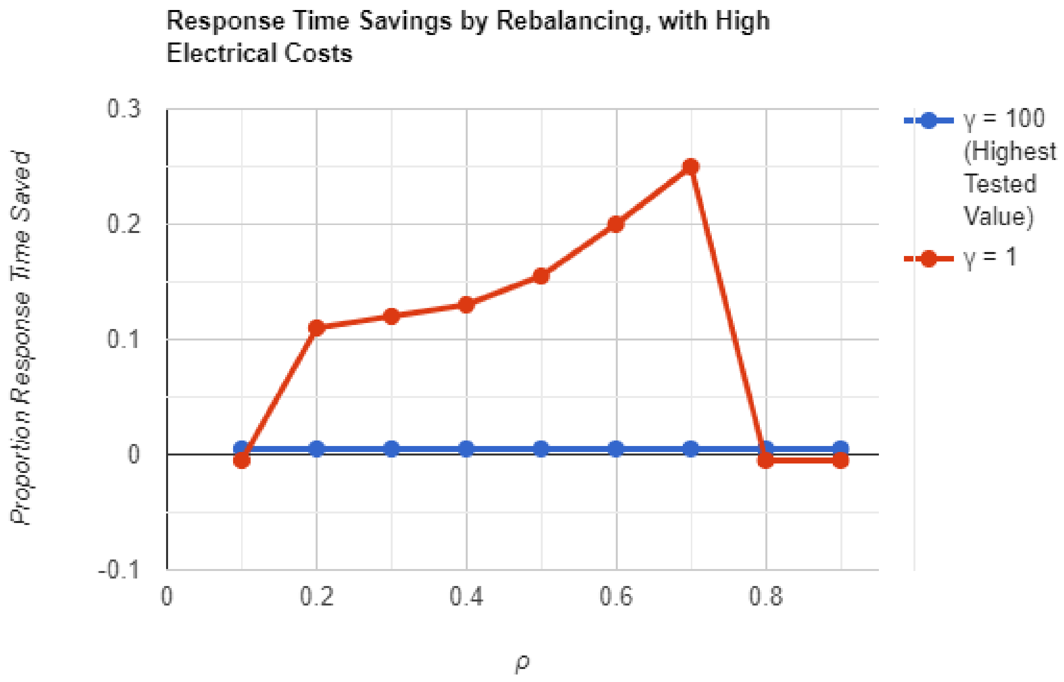

Figure 5.11: Rebalanced Mean Response time Savings with High Electrical Costs

These findings are crucial to our studies because they show that if one wants to assign a very high level of importance to energy costs, and as such use many non-static servers that they should also consider re–balancing especially if none of the servers in question are too heavily loaded. Also important from these findings is that it is key to note that there is no one–size–fits–all solution to these sorts of problems, because γ is so crucial to our system meaning that in–depth analysis and considerations must be made for the re–balancing algorithm to be worthwhile, and one must be careful in its application.

5.3 Incorporating estimation errors

5.3.1 Modified Policy

We began with the assumption that we have perfect knowledge of the sizes of arriving tasks. For the next section we will be introducing some level of uniformly distributed error into the size estimates. That means for an arriving task of size x, we will now equally likely see any value between $(1-n)x$ and $(1+n)x$, where n is the maximum (proportional) error. It is the equivalent of multiplying a random value drawn from a bounded Pareto distribution by a random value from a uniform distribution to create a third random value that would follow another PDF entirely: $\hat{S} = SU$ where S follows $BP(k, p, \alpha)$ and U follows $U[1-n, 1+n]$, or bounded Pareto distribution with uniform error. Note that findings here are presented from $n = 0.005$ to 0.40. Uniform error was chosen for a few reasons. First, its simplicity in implementation. As will be seen later, the analysis is tractable albeit non-trivial. Second, we felt that uniform error showed the least bias between under-estimation and over-estimation. Finally we felt that this was an accurate representation of a system that had been simply estimated, as there was an equal likelihood of over and underestimation and the over and under estimation both tend to be larger given larger initial values. Though this is a solid foundation we acknowledge different strategies could be taken in the future, as will be discussed in the future work section of the thesis.

Next we will discuss the methodology for SITA-E despite the uncertainty of the estimates. That is to say, how can one determine a set of x boundaries that will yield equally-loaded servers despite the fact there is inherent error in task size estimates. The intention here is to see how gracefully the system degrades when there are these uniformly distributed estimation errors, and a detailed solution on how to compensate for a known error distribution.

There are two separate methods to compensate for errors, the first is to calculate the new with-error probability distribution function of the system, that is to calculate the distribution of $\hat{S}$ (call its associated density g) and use this as the basis for all of the calculations described in the first part of the thesis. Recall (4.1.3) describes the process with which one can find a single boundary x_1 from k and successively find all future boundaries by finding each x_i using x_{i-1} with (5.3.1) here we can replace $f(x)$ with the distribution of $\hat{S}$ and use (5.3.1) after which one can use (5.3.2) successively until all the boundaries are calculated.

$$\int_{k}^{x_1} g(x) * x \ dx \ = \ \rho = \frac{E[X]}{j} \tag{5.3.1}$$

$$\int_{x_i}^{x_{i+1}} g(x) * x \ dx \ = \ \frac{\rho}{\lambda} = \frac{E[X]}{j} \tag{5.3.2}$$

With all of these boundaries set we have created a new fully defined system following the rules of SITA-E, where even given the error we guarantee that the load on each server will be equal. This is a very powerful statement, we have taken a system which has some amount of known error, and which has a very high possible amount of variance and made a guarantee that loads will still be equal. Having said that, this method comes with its own points of difficulty. Firstly the implementation is not simple: the distribution of $\hat{S}$ is difficult to calculate. There are ways to determine the distribution representing this multiplication's different outcomes and their likelihoods however it is a non-continuous piecewise combination of three disparate sections of the underlying distributions that must all be calculated separately. Secondly system limitations can quickly lead to problems when trying to use the aforementioned construction. In early tests with this methodology Matlab's symbolic library was processing for so long we simply decided to terminate the process. For this reason we came up with a second method, which came with its own issues.

The second approach involves determining the proportion of tasks that are misclassified into another interval due to the error. For example we might have considered a task that utilized a server for nine seconds originally as a task that would utilize the server for 11 seconds. This is not problematic if the bounds involved are above and below both those numbers, but it will cause issue if some bound exists such that $9 \leq x_i \leq 11$. The reason that this causes issues is because of the misclassification of tasks, we will have accounted for a task that takes nine seconds to be handled by a resource intended for greater than x_i seconds. Here we will consider where those edge cases will be causing problems and deal with that problem specifically, thus avoiding needing to construct the distribution for $\hat{S}$. We do this by taking the integral in points of interest as the probability that the specified x lands within our given region multiplied by the value of the probability density function at that location. That is to say, in general we replace (5.3.2) with:

$$(A) \quad \int_{x_i/(1+n)}^{x_i/(1-n)} ([BP(k,p,\alpha)(x)] * x * (P[U > \frac{x_i}{x}]))\ dx \quad + \tag{5.3.3}$$

$$(B) \quad \int_{x_i/(1-n)}^{x_{i+1}/(1+n)} ([BP(k,p,\alpha)(x)] * x)\ dx \quad + \tag{5.3.4}$$

$$(C) \quad \int_{x_{i+1}/(1+n)}^{x_{i+1}/(1-n)} ([BP(k,p,\alpha)(x)] * x * (P[U < \frac{x_{i+1}}{x}]))\ dx \quad = \tag{5.3.5}$$

$$\rho = \frac{E[X]}{j} \tag{5.3.6}$$

So the three cases are described as the following: (A) is the leftmost portion of the integral, where the task is included in our current server based upon the probability that it will land in the server after being multiplied by U. (C) is similar but on the right side. Part (B) are the tasks that remain within the interval, even after the error is included.

We examine a specific example's upper bound to explain the idea in a little more detail.

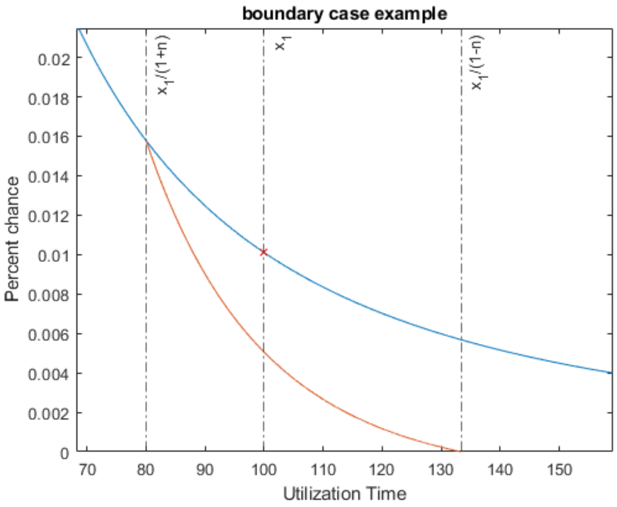

Figure 5.12: Boundary Case example

In the above example we see a case where our first bound happens to fall at $x_1 = 100$. In blue we see a typical slice of the bounded Pareto density function. Since we are concerned with the server that runs tasks whose estimates fall on the left side of x_1 we will be treating this as a "right bound" and use (5.3.5) for compensating on the right bounds. To the left of $\frac{x_1}{1+n}$ there is no chance that a task of length less than x_1 falls outside of our boundaries, so in this range we can use the unmodified bounded Pareto density. Since there is now error in our system there is a chance that we will be dealing with sizes greater than our intended bound because of estimation error. Overall this has a relatively small effect on our system mainly for two reasons. First, in a typical system these boundaries are far apart and the edge-cases affect only (relatively) small portions of the overall system. Second, as you can see in the above image there is a portion discounted on the left side of x_1 (above the red line, but below the blue line), but to countermand this there is a new section included on the right (below the red line but above zero). This means that as our system degrades in accuracy so long as we have a method to understand the amount of inaccuracy in the system our expected results do not degrade too quickly. Without these adjustments one can see how for heavily loaded systems "misplacing" a few tasks could lead to catastrophic results by making what was a stable – though highly loaded – server into an unstable server.

In summary we have a method that is simple to understand and easily implemented; it requires much less computer algebra, and as such runs in a fraction of the time. This method manages to compensate for given errors in a manner that gracefully degrades the intended outcome of savings through our original methods, which assumed precise knowledge of task sizes.

5.3.2 Trends and Numerical Findings

Here we will discuss the numerical findings for the savings in a transition from a non-energy aware set of SITA servers to a SITA-EA server set. This section will begin with an unfortunate disclaimer. We have erroneous findings wherein the system has performed better on occasion with added errors. It would make sense that the addition of errors would result in increasing the variance seen by the system as such we do not believe that this would be possible. We believe that since there are so many cascading calculations including ones with values that are quite close to each other that we are likely seeing a manifestation of the 'catastrophic cancellation' phenomena due to arithmetic underflow in Matlab. Catastrophic cancellation has been studied in great detail but is also well outside of the scope of the discussion at hand. We see this phenomena exaggerated at lower values of ρ, which makes sense as catastrophic cancellation appears to have greater impact in such scenarios.

Having said all of this we do not feel that the findings are entirely without use or merit, and as such we will discuss some trends that might be possible to be seen when error is accounted for within our system.

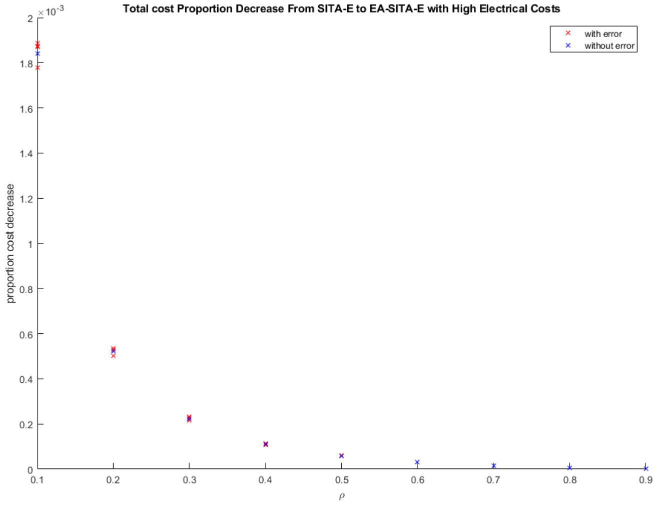

Figure 5.13: Impact of Errors as a Function of Utilization

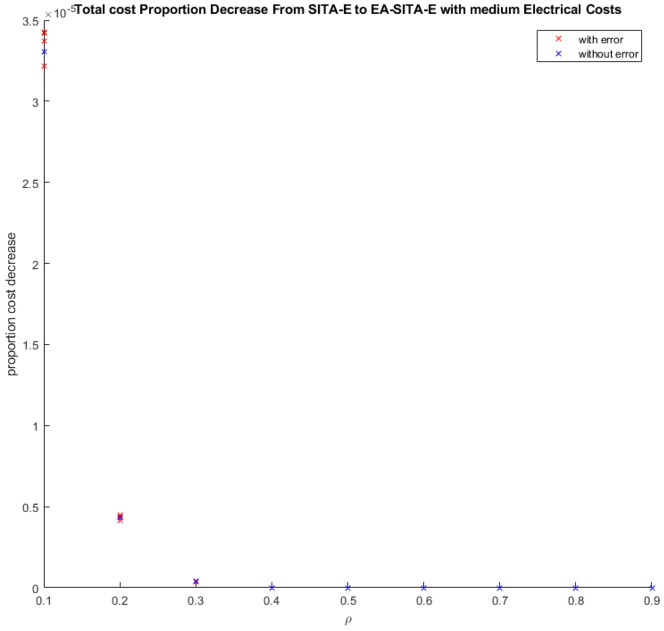

Figure 5.14: Total cost Decreases with Error, and Medium Electrical Costs

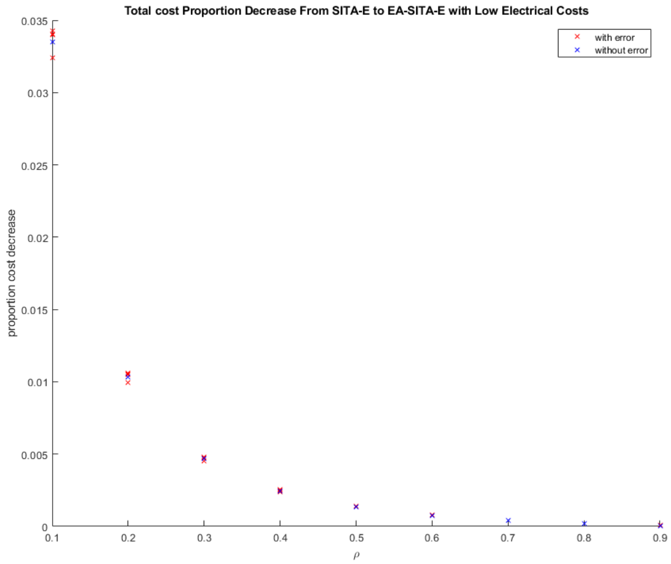

Figure 5.15: Total cost Decreases with Error, and Low Electrical Costs

The most significant finding is that the value of the error seems to be mostly ineffectual to the final product – that is savings over the original system. We see tightly grouped results for 0 error all the way up to 40 percent error, a quite high value of error considering this means a value could be over or underestimated by as much as half its original value. Overall, the portion of savings seen by the system is not largely altered, this is a promising insight because it means that if we were to actively correct for error we see in real life systems it stands to reason that at least eventually we could implement an effective SITA-E or EA-SITA-E system. Second, this shows that as the utilization seen by a server increases generally the error will affect those servers less in terms of the proportion that they will be able to save. This makes sense because we would expect the few task allocation decisions we make in a lightly loaded system to be more impactful than any one of the many decisions we make in a more heavily loaded system. This too is a promising finding because it means that the systems we tend to target more frequently with optimisations like the ones presented here are heavily loaded systems due to the fact they will have a more significant impact on our costs at the bottom line. An interesting finding is that within the 27 systems we tested, regardless of the amount of error there was never a switch in whether or not there was savings. This indicates that if previously we had seen an entirely static set of servers we would continue to see that after any amount of error, and likewise if there were servers that were not static there still will be. This means reinforces the notion that estimation errors only make a small amount of difference to the results that we see given proper mitigation techniques.

To summarize: the effects of error can be mitigated rather effectively by the above discussed methods, and they are even less significant in the systems we tend to care about the most – heavily loaded systems.

Chapter 6

Conclusion

6.1 Summary and Possible Applications

Throughout this thesis we have discussed different tools and algorithms to reduce energy costs while avoiding significant increases in mean response times. This is mainly in response to growing concerns about the environment, and growing energy costs for data centers – both of which are important.

We proposed a new assignment policy called EA-SITA-E which modifies the existing SITA-E policy to be electrically aware. This allows the inclusion of energy costs in decisions in operating a set of servers. We provided a new algorithm that is intended to re-balance the utilizations seen by servers across our system. This is done in such a way that we can still maintain the benefits of lowering variance seen by a server provided by SITA-E. We also include contributions assessing the aforementioned policies when estimation errors are present. Finally we provide an algorithmic approach to lowering the effects of error given that we understand what level and type of error we will be seeing in our system.

In our studies we found that in cases we can have better results than SITA-E in terms of mean response time and also in terms of energy costs after applying the algorithm discussed above. This is an interesting finding because it means that there is potential for a "win-win" solution. Even if in cases where mean response times and energy costs cannot simultaneously be decreased we found that in a large set of cases – or at least a large set of cases in which we consider energy costs a significant enough portion of the total costs – we can see significant energy savings. This is coupled with increased mean response times and a designer may want to determine whether the tradeoff is acceptable or not.

Of course these findings have a wide range of applications, but mainly these should be applied in areas where there is large variance in task sizes. As noted we see this commonly in computing, whether it be in small-scale examples like on a CPU of a PC, or in larger scale examples such as on web servers where some downloads or tasks might be orders of magnitude larger than others. Of course the most applicable problem for this algorithm or set of algorithms to be applied to is in the case of a set of servers as commonly seen in data centers.

6.2 Future Work

In the future we would like to see whether the results we presented hold for systems that have lower task size variance. Though most of our findings were presented in such a way that they are applicable in general, we developed our insights through the lens of the task sizes following a bounded Pareto distribution, which is well known for its high variance and as such lends itself to performing well in the conditions that we are presenting in this thesis.

In a similar fashion we would like to expand the error distributions we presented to beyond that of the uniform distribution, which is limited in terms of how well it models practical situations. Some relevant things to consider would be a distribution that does not have an error PDF with a mean value of one (a mean value of one means there is no bias towards under or over estimation). Does it impact a system more if we underestimate or overestimate job sizes, on average?

Again while the thesis does discuss the possibility, cycling costs were largely ignored in favour of more coherently examining the trade-off between energy cost and mean response time, it is also worth noting that the static servers we are attempting to favour would not suffer from cycling costs, and as such in future studies we may see the cycling costs lowered as well. In the future this could be considered more directly through a lens of how seriously cycling costs could be affected, which might be important to certain data centers depending on the specific hardware issues that they could be facing in terms of wear and tear from frequent server shutdowns.

Finally something that could be developed further is the specific reasoning behind the erroneous behaviour in the numerical analysis of the portion of the thesis discussing error in the *a priori* approximation of job sizes. This would help to make the tentative findings that we have discussed more trustworthy.

Bibliography

1. V.J. Maccio, D.G. Down, “On Optimal Policies for Energy Aware Servers,” Performance Evaluation vol. 90 , p. 36–52, 2015

2. M. Gebrehiwot S. Aalto, P. Lassila, “Optimal Energy–Aware Control Policies for FIFO Servers,” Performance Evaluation vol. 103 ,p. 41-59, 2016

3. M. Gebrehiwot, S. Aalto, P. Lassila, “Energy–Performance Trade–Off for Processor Sharing Queues with Setup Delay,” Operations Research Letters vol. 44 ,p. 101-106, 2016

4. M. Gebrehiwot, “Energy–Aware SRPT Server with Batch Arrivals: Analysis and Optimization,” Performance Evaluation vol. 115 , p. 92–107, 2017

5. Samuli Aalto, P. Lassila, “Near–Optimal Dispatching Policy for Energy–Aware Server Clusters,” Performance Evaluation vol. 115, 2019

6. M. Harchol–Balter, M.E. Crovella, C.D. Murta, “On Choosing a Task Assignment Policy for a Distributed Server System,” Lecture Notes in Computer Science vol. 1468 p. 231-242, 1998

7. J. Anselmi, J. Doncel, “Asymptotically Optimal Size-Interval Task Assignments,” IEEE Transactions on Parallel and Distributed Systems, vol. 30 (11) p. 2422-2433, 2019

8. W. Winston "Optimality of the Shortest Line Discipline," Journal of Applied Probability, vol. 14, p. 181–189, 1977.

9. R Weber "On the Optimal Assignment of Customers to Parallel Servers," Journal of Applied Probability, vol. 15, p. 406–413, 1978.

10. W. E. Leland, T.J. Ott, "Load–Balancing Heuristics and Process Behavior," SIGMETRICS '86/PERFORMANCE '86: Proceedings of the 1986 ACM SIGMETRICS joint international conference on Computer performance modelling, measurement and evaluation p. 54–69, 1986

11. M. Harchol–Balter, A.B. Downey, "Exploiting Process Lifetime Distributions for Dynamic Load Balancing", ACM SIGMETRICS Performance Evaluation Review, vol. 24, p. 13-24, 1996

12. V. Paxson, S. Floyd,"Wide Area Traffic: the Failure of Poisson Modeling," IEEE/ACM Transactions on Networking, vol. 3, (no. 3), p. 226-244, 1995

13. D. L. Peterson,D. B. Adams,"Fractal Patterns in DASD IO Traffic," The 22nd International Conference for the Resource Management & Performance Evaluation of Enterprise Computing Systems, vol. 2, 1996

14. Rachel Mailach, D.G. Down,"Scheduling Jobs with Estimation Errors for Multi–Server Systems," 29th International Teletraffic Congress (ITC 29), p. 10-18, 2017

15. B. Schroeder, M. Harchol–Balter,"Web Servers Under Overload: How Scheduling can Help," ACM Transactions on Internet Technology, vol. 6 (No. 1), p. 20–52, 2006

16. D. Lu, H. Sheng, P. Dinda,"Size-Based Scheduling Policies with Inaccurate

Scheduling Information," 12th International Symposium on Modeling, Analysis, and Simulation of Computer and Telecommunication Systems, 2004

17. P. Jelenkovic, X. Kang, and J. Tan,"Adaptive and Scalable Comparison scheduling," Proceedings of the 2007 ACM SIGMETRICS International Conference on Measurement and Modeling of Computer Systems, 2007

18. Pollaczek, F,"Über eine Aufgabe der Wahrscheinlichkeitstheorie," JOURNAL, Mathematische Zeitschrift vol. 32, p. 64–100, 1930

19. M. Harchol-Balter, M.E. Crovella, C.D. Murta,"Task Assignment in a Distributed System: Improving Performance by Unbalancing Load," Sigmetrics Performance Evaluation Review - SIGMETRICS, vol. 26, p. 268-269, 1997

20. L. Schrage, L. Miller, "The Queue M/G/1 with the Shortest Remaining Processing Time Discipline," Operations Research, vol. 14 (4), p. 670-684, 1966

21. Y. Chen., A. Das,W. Qin, A. Sivasubramaniam, Q. Wang, N. Gautam, "Managing Server Energy and Operational Costs in Hosting Centers," Association for Computing Machinery, vol. 14(4), p.303-314, 2005

22. P. Barford, A. Bestavros, A. Bradley, M Crovella,"Changes in Web client access patterns: Characteristics and caching implications," World Wide Web 2, p. 15-28, 1999