2013

Using Forecasting to Predict Long-term Resource Utilization for Web Services

Daniel Wayne Yoas
Nova Southeastern University, dyoas@pct.edu

Using Forecasting to Predict Long-term Resource Utilization for Web Services

by

Daniel W. Yoas

A dissertation submitted in partial fulfillment of the requirements
for the degree of Doctor of Philosophy
in
Computer Information Systems

Graduate School of Computer and Information Sciences
Nova Southeastern University

2013

An Abstract of a Dissertation Submitted to Nova Southeastern University
in Partial Fulfillment of the Requirements for the Degree of Doctor of Philosophy

Using Forecasting to Predict Long-term Resource Utilization for Web Services

by

Daniel W. Yoas
January 2013

Researchers have spent years understanding resource utilization to improve scheduling, load balancing, and system management through short-term prediction of resource utilization. Early research focused primarily on single operating systems; later, interest shifted to distributed systems and, finally, into web services. In each case researchers sought to more effectively use available resources. Since schedulers are required to manage the execution of multiple programs every second, short-term prediction has focused on time frames ranging from fractions of a second to several minutes.

The recent increase in the number of research studies about web services has occurred because of the explosive growth and reliance on these services by most businesses. As demand has moved from static to dynamic content, the load on machine resources has grown exponentially, periodically resulting in temporary loss of service. To address these short-term denial-of-service issues, researchers have tried short-term prediction to manage scheduling of service requests.

What researchers have not considered is that the same methods used for single step short-term prediction can also be used for long-term prediction if a coarse granularity of samples is used. Instead of using one or more samples per second, a coarser aggregate of minutes or hours more accurately emulates the long-term patterns. This research has shown that simple moving averages and exponential moving averages as a prediction technique can be used to more accurately predict hourly, daily, and weekly seasonal patterns of resource utilization for web servers.

Additionally, this research provides a foundation where using a resource prediction within a confidence interval range could be more useful to an administrator or system software than a single prediction point. When the focus shifts to a range, a set of probabilities can establish normal function within that system. For distributed systems, it will provide the ability to notify other systems when resource utilization is no longer normal before that system is unable to issue a notice of overloading. For web systems it can be used to provide a warning, permitting the instantiation of a second system to begin load balancing during unscheduled heavy loads. In both cases, the availability of the system can be improved by predicting a resource utilization level and the confidence interval within which that resource use has historically fallen.

Acknowledgements

First and foremost, I thank God for giving me the patience and knowledge to complete this effort. A special thanks to Dr. Simco for his patience and guiding comments throughout my two-year effort. Without his help, it would have been a lesser experience. I also thank my committee, Drs. Mitropoulos and Mukherjee, for their assistance in completing my dissertation and defense.

I would also like to thank my colleagues at the Pennsylvania College of Technology for their continued positive comments and support over the last ten years during the acquisition of my Masters and Ph.D., and my wife, Helen, who, for the last two years of my research, put up with hours of work, study, and requests for editing assistance.

Thank you for making these experiences a good learning process.

May the words of my mouth and the meditations of my heart be acceptable in Your sight, O LORD, my rock and my redeemer.

Psalms 19:14

Table of Contents

List of Tables

Tables

List of Figures

Figures

List of Equations

Equations

Chapter 1

Introduction

Introduction

Short-term resource utilization remains an active area of research for scheduling web services. One reason for this is that an overloaded system will delay or deny services for users or automated requests (Andreolini & Casolari, 2006; Sharifian, Motamedi, & Akbari, 2010). Researchers have sought to understand the connections between resources and applications by studying resource balancing, scheduling, and future needs (Chapman, Musolesi, Emmerich, & Mascolo, 2007; Dinda, 2002) so that more applications can be serviced in a more effective manner. That exploration has also used simulation and load emulation (Andreolini, Casolari, & Colajanni, 2008; P. Barford & Crovella, 1998; Schroeder & Harchol-Balter, 2006) to better explain how systems react under specific conditions.

A more effective means of scheduling and admission of web requests during times of system overload is needed to provide efficient use of resources, which will help businesses to increase revenue (Al-Ghamdi, Chester, & Jarvis, 2010; Hoffmann, Trivedi, & Malek, 2006). Software resource performance (Balsamo, Marzolla, & Mirandola, 2006; Chen, Liu, Gorton, & Liu, 2005; Marzolla & Mirandola, 2007) has also been studied to demonstrate how software affects those resources. A better understanding of software-driven resource utilization can lead to more accurately matching available system resources to software needs.

In each of the cases listed above, researchers sought to understand how resources were utilized. The researchers also directly referenced the use of prediction at some level to improve the processes they were exploring. Resource prediction is not a new area of study, but continues to be active because of the promise it holds to improve utilization in so many areas of computing.

Problem Statement

Resource utilization prediction remains an active area of research due to the difficulty in accurate forecasting. The need for forecasting also remains high since an accurate understanding of resource utilization can improve scheduling, performance optimization, load balancing, and recognition of overload conditions (Sharifian et al., 2010). Most of the research into resource forecasting has been conducted using simulation or limited real-time measurements, but these methods have provided only limited results.

Dynamic load balancing of web content is one research area where system short-term predictability is being studied (Sharifian et al., 2010). The Internet continues to grow at an astounding rate, and web services must now provide dynamic content in order to satisfy expanding user demands. The ability to predict resource utilization is a critical part of improving system availability, since overloading causes increased response time or dropped service. Sharifian et al. (2010) addressed dynamic load balance by establishing an algorithm that would examine incoming web traffic and determine how to distribute it among the various servers within the system.

The system proposed by Sharifian et al. (2010) – approximation-based load-balancing (ALB) – considers the CPU to be the resource primarily susceptible to bottlenecks in their systems. By classifying incoming requests based on estimated CPU time from similar requests, the system can queue like requests together. The system then calculates load based on queue size and estimated utilization time for the CPU and maintains balance by admitting requests to appropriate servers within the cluster. To prevent overloading, the system routinely recalculates the remaining CPU capacity at regular time intervals. The remaining unit capacity, along with CPU time for released jobs, is fed back into the system calculations to increase the accuracy of job times in the appropriate class.

Standard web traffic, dynamic web traffic, and secure traffic all require different levels of CPU time to service and are the primary class styles used within the system (Sharifian et al., 2010). Each of the three types of traffic is broken into further classes, since requests within the same area can also have different levels of CPU requirements. Sharifian et al. (2010) used weighted round-robin (WRR) and content-aware policy load-balancing algorithms as benchmarks to determine the effectiveness of their ALB system, with collection points from 5,000 to 60,000 clients. Testing (Sharifian et al., 2010) showed that under load, WRR performed well through 27,000 clients, but then efficiency fell off. Context Aware Policy (CAP) used multiclass round robin and request classification to define this third-generation load balancing algorithm. CAP showed a much better performance than WRR, providing good throughput for up to 36,000 clients, which then slowly declined. ALB was able to maintain the high throughput rate through 50,000 and remained level through 60,000 clients. Response times grew quickly for WRR load-balancing and broke ten seconds at 32,000 clients, while CAP could handle about 55,000 clients before breaking the ten-second barrier. In comparison, the ALB system never took more than two seconds to return a client request.

Resource utilization prediction models have also been explored by Andreolini and Casolari (2006). Systems require load balancing, overload management, admission control, and job dispatching. Andreolini and Casolari (2006) evaluated resource behavior for web services including CPU utilization, network throughput, open sockets, and memory load. Simulation was used to generate resource utilization patterns in step, staircase, and alternating formations. The three scenarios were evaluated for better understanding of CPU and disk utilization.

The TPC-W benchmark ("TPC-W transactional Web e-commerce benchmark (Retired 04/28/2005)," 2004) was used as a foundation for Andreolini and Casolari's (2006) study. They used sample rates of one- and five-second intervals for a ten-minute period for the study. The collected samples were processed using simple moving averages (SMA), exponential moving averages (EMA), and cubic spline (CS) analysis to provide resource utilization prediction. SMA, EMA, and CS look back at a number of samples to generate a prediction for the next cycle. These statistical methods were selected because of the averaging effect, allowing the drastic changes seen in web data to be normalized; each evaluation method was then run against the resource samples and evaluated for accuracy (Andreolini & Casolari, 2006).

The sampling sizes feeding the series analysis changed the results of the prediction, so Andreolini and Casolari (2006) chose to generate results for ten and thirty seconds into the future. The ten-second sample time frame was selected from a series of times greater than zero, up to thirty seconds since it exhibited a high level of accuracy for predictions under thirty seconds. The accuracy of the prediction was also driven by the number of previous time frames used for weighting the prediction. In the case of the ten-second intervals, a prediction window of five time frames produced the highest level of accuracy; for the thirty-second time frame, fifteen intervals were used. In both cases, smaller intervals weighed the recent past too heavily, decreasing the accuracy, while longer intervals weighed the more distant past too heavily, causing the predictions to adjust to changes too slowly.

Once it was determined that the prediction tracks aligned with the simulated pattern, Andreolini and Casolari (2006) calculated the amount of error in each prediction, by taking the actual reading from the simulator and comparing it to the prediction. The authors determined that the EMA prediction was the most accurate of the three, with an error rate of 0.065 at ten seconds in the future compared to CS's error rate of 0.16 under the same time interval. The authors noted that EMA could suffer from high delays during times of overloading in web services.

Forecasting for single web servers is another active area of research. Most organizations work with a web server until the load exceeds the unit's resources. Once the web server is no longer able to guarantee availability, administrators move toward either clustered or distributed web services, but this then wastes resources, since most excess loading is transitory. A better understanding of how and when resources are required will permit the administrator to use a single web server and its resources more effectively. One recent study used an Apache web server to better understand response time and free memory (Hoffmann et al., 2006). The research was broken into several phases to provide the highest reliability in predicting response time and free memory. In the first phase Hoffmann et al. (2006) pulled readings from logs monitoring 102 variables, which were pulled every five minutes for one hundred and sixty-five hours.

The initial variable group was composed of common variables that are traceable on an Apache server. A review of the collected information was completed to determine the variables that most accurately correlated to the server's response time and free physical memory. These predictive variables were then identified using three benchmarks (forward selection, backward elimination, and probabilistic wrapper), reducing the number of variables followed even further. Hoffman et al. (2006) concentrated on the variables that would be used for both a short-term prediction (five minutes) and a long-term prediction (two days). The root-means-square-error was calculated for each benchmark to find the calculation with the smallest error. Hoffmann et al. (2006) determined that the probabilistic wrapper benchmark had the smallest error at 0.025; forward selection was 0.047; and backward elimination was 0.336 for short-term prediction, while long-term prediction errors were 0.010, 0.011, and 0.716, respectively.

Probabilistic wrapper had the smallest error for both long-term and short-term predictions, so it was used for the model building phase of Hoffmann et al.'s (2006) research. Four models were used (multivariate linear regression, support vector machines, redial basis functions, and universal basis functions) to process the data for both short-term and long-term predictions. The authors (Hoffmann et al., 2006) used root-mean-squared-errors again to determine which of the four models provided the most accurate prediction. The support vector machines model was the most promising during validation, with a short-term error rate of 0.012 and a long-term error rate of 0.007. However, the authors (Hoffmann et al., 2006) indicated that, once the final testing was completed, the single vector machines model was outperformed by universal basis functions for both short and long-term errors. Hoffmann et al. (2006) stated that a radial basis functions model outperformed a support vector machines model for long-term prediction error, but table 5 (Hoffmann et al., 2006) indicates that the support vector machines model outperformed the radial basis functions model. This change in error rate between the five-minute prediction in table 5 and the twenty-four hour predictions of table 6 suggested that a single prediction method might not have been appropriate for forecasting different time frames of the same resource.

Hoffmann et al. (2006) concluded that variable selection is very important to the process of predicting resource utilization and that using all variables as suggested by backward elimination will greatly slow the process. Three variables (number of new processes, blocks written to disk, and the number of Ethernet packets transmitted) were used for predicting free memory and response times for the author's Apache server. Since these variables proved to have predictable qualities for web services, they are good candidates for long-term prediction as well.

Prediction is also needed in network traffic management. These predictions should neither starve nor waste bandwidth and such predictions should not incur a high cost of execution or communications (Krithikaivasan, Zeng, Deka, & Medhi, 2007). Using a basis of well-known cyclical network utilization, Krithikaivasan et al. (2007) proposed a seasonal AutoRegressive Conditional Heteroskedasticity (ARCH) based model. They collected data every five minutes and averaged them for a fifteen-minute interval. Data was collected over fifteen weekdays, while no data was collected on weekends or holidays. This collection occurred in three different months. The first ten days of the data was used to generate forecasting for the last five.

Krithikaivasan et al. (2007) found that days one, six, and eight exhibited higher usage than the other days in the study and considered these anomalies important to understand but detrimental to the foundation of the study. The outliers are not removed from the data but transformed using natural logarithmic transformation, an effective way to stabilize data. This data is then processed under minimum mean square error (MMSE) to predict a single step ahead in prediction of the network traffic level. Krithikaivasan et al. (2007) determined that the resulting error level was negligible in this study.

The final stage of the study used the information gathered to provide bandwidth provisioning. Krithikaivasan et al. (2007) used a modified allocation method called Stabilized Bandwidth Provisioning (SBP). The modification uses a Hold Down Timer (HDT) that permits extra bandwidth as needed but throttles message exchange as bandwidth is slowly reduced. The process keeps the bandwidth steps from heavy oscillation. The authors also set a maximum bandwidth of 60% to prevent starvation of other traffic. There were only two times in the study data where the 60% bandwidth limit was reached, with the remainder of the times properly provisioning bandwidth for the required service over the five days of forecasting.

Five- to ten-minute prediction methodologies have been reasonably successful at determining resource utilization (Andreolini & Casolari, 2006; Sharifian et al., 2010), while similar results in longer patterns of resource utilization and prediction remain elusive. Attempts to accurately predict resource utilization using multiple steps of short-term readings (Andreolini & Casolari, 2006; Sharifian et al., 2010) were less successful than using larger time-frames and a single step. Extending a prediction over several steps found that each step after the first in the time-series resulted in decreased accuracy of the prediction; researchers, therefore, have focused on time-series analysis that uses only a single step prediction. System degradation and overloading of web services is a constant issue for service owners and needs to be addressed to guarantee availability (Hoffmann et al., 2006). Web services continue to be a key area of interest due to the rate of growth and importance to industry (Sharifian et al., 2010).

Short-term prediction has shown that resource utilization can be reasonably determined on the scale of minutes or less; this is adequate for tasks such as scheduling, but, unfortunately, research has not been undertaken into long-term prediction. The predictability of system resource utilization deteriorates rapidly when multiple step short-term prediction is used (Andreolini & Casolari, 2006; Istin, Visan, Pop, & Cristea, 2010). Shifting from short-term prediction to long-term prediction will provide for improvements in load balancing, job dispatching, job distribution, and overload prevention (Andreolini & Casolari, 2006) by permitting a system to anticipate resource needs further into the future. The improvements provided by long-term forecasting will facilitate the ability for administrators to avoid denial-of-service issues.

Algorithms have used real-time sampling (Dinda, 2006) to solve resource limitation issues and only addressed problems as they were detected or based on short-term prediction. Being able to predict resource utilization through short-term prediction was helpful in allowing scheduling changes (Schroeder & Harchol-Balter, 2006), load balancing (Sharifian et al., 2010), and resource trending (Andreolini & Casolari, 2006). While current prediction methods deteriorate quickly (Andreolini & Casolari, 2006; Istin et al., 2010), researchers have determined that further research into longer term prediction (greater than fifteen minutes) is needed to better understand utilization and management system components (Krithikaivasan et al., 2007).

Better long-term prediction models for network traffic flow can be used to provide a higher quality of service (QoS) to the users (Krithikaivasan et al., 2007) by predicting the type of traffic expected on a network. Traffic volume prediction could also be used to expand or contract the communications channel prior to oversaturation. Another benefit of long-term prediction is in scheduling grid resources (Ramachandran, Lutfiyya, & Perry, 2010). Short-term prediction has improved usage of the volunteer grid, helping to guarantee availability of CPU and storage to the distributed system. The difficulty occurs when these systems are taken off the grid due to local usage. Long-term prediction would provide additional details about when systems are likely to go offline and provide for transition of the service to another system prior to predicted loss of service.

Overloading of web services has also been improved through short-term prediction (Sharifian et al., 2010), by shifting incoming requests based on the level of previous service for similar requests. Long-term prediction would provide a foundation for load balancing of systems by permitting fewer machines to run during low load periods and bringing on additional machines during high loads. Load levels are also important to servers: Andreolini and Casolari (2006) used short-term load prediction to limit incoming traffic to a web server. Longer load predictions could identify periods when high load is expected, permitting a mirrored service to be started prior to the expected heavy load.

Short-term prediction has been used successfully to improve resource utilization for items which occur frequently or need to be addressed quickly. In most cases researchers have focused on a time frame of minutes, seconds, or fractions of a second, but computers must also contend with time frames of a much coarser granularity. Each day, systems see a heavier load as workers log into the services, a lighter load as staff goes to lunch, and a very light load overnight when most workers don't need the services. These longer patterns exhibited over minutes, hours, days, weeks, and longer, have not yet been explored by researchers, nor have the ramifications of how administrators can more effectively use the resources available in their servers during these differing time periods.

Dissertation Goal

The goal of this research was to provide evidence that web resource utilization patterns that occur hourly, daily, and weekly are predictable with a reasonable accuracy using existing techniques of prediction. In the past, short-term prediction has focused on forecasting in time frames less than fifteen minutes (Andreolini, Casolari, & Colajanni, 2006). Krithikaivasan et al. (2007) focused on bandwidth provisioning with fifteen days of five-minute collections, not including holidays and weekends, to predict bandwidth utilization for twenty-four hours. The work of Istin, Visan, Pop, & Cristea (2010) identified reliable statistical methods for short-term patterns and future predictability of resources for scheduling. Those methods included the evaluation of average percentage of errors for SMA, Weighted Moving Average (WMA), EMA, random prediction, and three additional methods that use neural networks. The results of the analysis completed by Istin et al. (2010) identified the patterns for CPU utilization, network traffic, memory utilization, and disk I/O availability in a distributed system, along with the average percentage of errors when predicting a single step ahead.

This research has built upon Krithikaivasan et al.'s (2007) methods of collecting and evaluating resource data by using the coarser granularity of data collection at one sample every ten seconds instead of one or more samples per second (Andreolini & Casolari, 2006; Dinda, 1999, 2006; Istin et al., 2010). The same set of resources used by Istin et al. (2010) – CPU utilization, network traffic, memory utilization, and disk I/O availability – was used to determine the effectiveness of long-term prediction. A previous study used Optimistic Network Performance Index, and a Robust Network Performance Index was used to create a high and low binding range for the prediction (Abusina et al., 2005). This research will use a similar technique for each web resource by identifying the 80% confidence interval for each single step prediction, during the hourly, daily, and weekly time frames of this study. Abusina et al. (2005) also chose to use time slices over each hour of their study. This study will divide the data into minute, fifteen-minute, and hourly time slices for predictions over an hour, day, and week, respectively.

Relevance and Significance

Simulation is an accepted method of testing a theory while reducing the number of variables within the study. Andreolini and Casolari (2006) used simulation of various workload patterns to assist in understanding the effectiveness of various job servicing algorithms. Their simulation of resource utilization in step, staircase, and alternating formations permitted better run-time scheduling of job services for web services. Barford and Crovella (1998) developed Scalable URL Reference Generator (SURGE) as a simulator that would represent common workload characteristics of a web service. The

SURGE simulator was compared to SPECweb96 ("SPECweb2009 (Retired 01/12/2012)," 2011) to determine the accuracy and ability to stress various web resources, including CPU utilization, active connections, and memory usage. SURGE also added a key metric that identified how long a page view would normally take before another page was requested. Most simulators, including SPECweb96, focus on pushing as much information at the web service as possible, to help identify the maximum load. Long-term prediction of resource utilization is focused on common utilization and SURGE's ability to more accurately represent incoming traffic will allow the simulation to more accurately represent common utilization.

The existence of hourly, daily, and weekly patterns in computing resources is well known (Krithikaivasan et al., 2007), but has not been extensively explored. The utilization levels of several resources (CPU utilization, Disk I/O, free memory, and network traffic), were examined as indicators to provide increased availability of a system (Schroeder & Harchol-Balter, 2006). By determining service levels of equipment, administrators can estimate resource needs, performance trending, trouble shooting, or event reconstruction using information found in logs (Ghemawat, Gobioff, & Leung, 2003). When resources are undersized, the server is likely to experience denial of service issues (Andreolini et al., 2006; Schroeder & Harchol-Balter, 2006). When a denial of service occurs, users will abandon a web site after a short period of time. Users will also abandon sites if response times become too long, causing a company to lose revenue (Al-Ghamdi et al., 2010). When a system's resources are oversized, the company wastes money by purchasing resources that won't be used. By forecasting resource utilization needs, an administrator can properly size resources for demand (Al-Ghamdi et al., 2010; Schroeder & Harchol-Balter, 2006).

Previous research focused on near-term issues driven by computing time frames, frequently using a small granularity of time (less than a minute) to drive the analysis and prediction. Research by Rood and Lewis (2010) focused on distributed system scheduling using increments from five minutes to twenty-five hours; the results provided predictions over fifteen days. Rood and Lewis' (2010) work focused on the level of resource demand five minutes into the future by considering what had come immediately before without considering the longer cyclical patterns. In one study on network traffic predictions (Qiao, Skicewicz, & Dinda, 2004), the time frame was based on the end-to-end transfer times to provide a more accurate prediction of file transfer times. The prediction was used by the file services to balance demand and determine the routing of new requests.

Most of the research into resource predictability only provides forecasting for short periods (Istin et al., 2010); those predictions appear to behave randomly over longer time periods because the recent past has little correlation with the immediate future. Researchers have focused on the dynamic behavior of systems resources (Dinda, 2002; Rood & Lewis, 2008), often looking for trending over predictable usage, but have rarely considered the large patterns that are known to exist in resource utilization (Krithikaivasan et al., 2007).

Lampson (1983) indicates that, no matter how well a system manages its resources, once two-thirds of that resource is being utilized, bad things begin to happen. Administrators currently use logs to identify when resources are being over-utilized, solving the issue through evaluation or increasing size of the affected resource. Lampson

(1983) also argues that avoiding a disaster is an appropriate response and that manual evaluation often takes place after a disaster has occurred. Systems need to move toward dynamic allocation (Al-Ghamdi et al., 2010; Andreolini et al., 2008) to address the rapidly-changing user needs. The development of long-term resource forecasting will facilitate dynamic configuration since current methods are still too error-prone (Al-Ghamdi et al., 2010).

Barriers and Issues

Resource prediction has been accepted as an important part of better system management (Andreolini & Casolari, 2006; Chen et al., 2005) and an important step in dynamic resource allocation (Al-Ghamdi et al., 2010). Attaining that goal has been reasonably successful in the process of understanding resource prediction for short-term prediction (Hoffmann et al., 2006; Lu, Wang, & Koutsoukos, 2005). At this time short-term prediction has been favored over long-term prediction because of the inaccuracy of extending the short-term theories (Istin et al., 2010). The patterns in long-term resource utilization have been accepted as existing (Abusina et al., 2005; Krithikaivasan et al., 2007), but research has not been furthered by this acceptance.

The behavior of computer systems appears to be random (Istin et al., 2010). The execution of code, service requests, interrupts, outside communications, and human interaction create an environment that looks chaotic when considered in the time frame normally associated with a computer’s system clock. This randomness has been seen in attempts to develop fair scheduling for both operating systems and distributed systems (Dinda, 1999; Silberschatz, Galvin, & Gagne, 2003, pp. 153-156). In most cases, as a system approaches overload, the scheduling algorithm becomes unfair and the advantage gained by a specific scheduling algorithm no longer matters (Wierman & Harchol-Balter, 2003). Randomness was also seen in Rood and Lewis's work (2008) when they used recent history to predict system availability in the future.

The apparent randomness in short-term resource prediction runs counter to the suggested existence of the patterns within network traffic by Krithikaivasan et al. (2007). Since little work has been done to understand these patterns (Krithikaivasan et al., 2007) within the computing environment, appropriate methods to evaluate the data remain unknown. Krithikaivasan et al. (2007) provides some insight into the long-term patterns using ARCH. ARCH is based on time series analysis for resources and was applied to network traffic to improve QoS. Krithikaivasan et al. (2007) then modified the resource data to remove anomalies within the system to facilitate their study. Using time series analysis with seasonal consideration, long-term resource utilization prediction is feasible.

Forecasting research has shown a wide range of how much historical data is needed to determine the next step. In Krithikaivasan et al. (2007), ten days of resource history was used to generate the prediction, while Rood and Lewis (2008) appeared to use several days and hours. One research project (Andreolini & Casolari, 2006) used only ten minutes of history to try to determine how heavy resource utilization would be one minute and five minutes in the future. Another forecasting attempt used 165 hours of data collected in five-minute increments (Hoffmann et al., 2006) to determine estimated response times on an Apache server as resources became exhausted. This research has provided insight into how much history is necessary to predict resource utilization accurately.

Researchers have used a variety of time-frames to collect the raw data for short-term forecasting, but research into long-term prediction must also seriously consider the time-frames used. A collection rate at the speed of the computer would starve other systems and prove to be useless, but collection rates of thousands of samples per second are feasible without interfering with the system's functionality (Dinda, 2002; Wolski, Spring, & Hayes, 1999). For short-term prediction, this high rate of collection is necessary to provide schedulers the data needed to make choices based on current load levels. Long-term prediction is more interested at looking at the overall long-term patterns of a system (Krithikaivasan et al., 2007). An appropriate granularity for the collection is an issue since short time frames may not add clarity to the result, while too few samples lead to inaccuracy (Wolski, 2003).

The collection mechanism introduces a second issue about accuracy: the sample may not accurately reflect the overall resource utilization levels. Resources inherently move dramatically between levels of use very quickly (Dinda, 1999), so individual samples may not be representative of the true resource utilization level. Hoffmann, et al. (2006) saw this issue when data points exhibited utilization spikes which were then removed by filtering prior to the study's analysis. Many researchers have chosen to use sample averaging to help increase the accuracy and remove heavy fluctuation of the resource utilization levels (Dinda, 2002; Krithikaivasan et al., 2007; Rood & Lewis, 2008; Wolski, 2003).

Research Questions

Two questions have been answered by means of this research, during two phases. The first phase used SURGE (P. R. Barford, 2001) to exercise a web server with specific pattern usage to determine if the simulated pattern can be predicted. The second phase used trace data obtained from a live web server to evaluate the utilization pattern for long-term predictions. The results of both experiments were evaluated to answer two questions.

The first question to be addressed was whether systems exhibit predictable long-term resource patterns. Usage patterns are accepted as existing on an hourly level over a 24-hour period (Krithikaivasan et al., 2007), but research has primarily focused on short-term needs. This work has helped to understand the larger long-term patterns in web services (Schroeder & Harchol-Balter, 2006), since overloading of web services can lead to lost revenue for businesses (Al-Ghamdi et al., 2010). This research evaluated common web resources including percentage CPU utilization, Disk I/O time, free memory, and network traffic, using seasonality to identify the pattern of resource usage over the time frames of hours, days, and weeks. The answer to this question is important to administrators so that they can properly size servers and identify resource upgrades before issues arise. When administrators are able to identify resource utilization levels, peak usage, and periods of low demand, they can balance system requirements and maintain system availability (Hoffmann et al., 2006).

The second question this research addressed was identifying the confidence interval of the prediction for each resource. Administrators should be able to trust the resource prediction, or else actions based on those numbers will be no more effective than those based on general observation. To provide a level of trust in the prediction, the study must define a confidence interval for the prediction (Traeger, Zadok, Joukov, & Wright, 2008). The research found a balance between the confidence interval and the level of precision needed for an informed decision to be made by the administrator. A high confidence interval can result in an extreme range of values around the prediction, making the prediction nearly useless since nearly all of the results fall into that range. On the other hand, if the range of resource utilization is too narrowly defined in order to allow the administrator to be aware when the prediction is outside that range, trust in that prediction becomes very low, due to the large number of results outside the range. The balance point permits the smallest possible range of utilization values with the highest reasonable confidence, allowing the administrator to make quick decisions when the actual resource utilization falls outside the prediction range.

Summary

Short term research has provided advancements in job scheduling (Dinda, 2006), distributed system management (Rood & Lewis, 2010), and web services (Sharifian et al., 2010). Researchers have used a variety of statistical methods to provide short-term predictability for system resources. To make these predictions, researchers have focused on collecting multiple data points every second to estimate utilization in the next few seconds, up to a few minutes in the future (Dinda, 2002). These attempts have remained focused on the short-term results, since longer-term predictions required researchers to use multiple iterations, and with each iteration a greater level of error in the prediction was generated (Andreolini & Casolari, 2006; Istin et al., 2010; Sharifian et al., 2010).

Krithikaivasan, et al. (2007) used the techniques previously established for short-term prediction and extended those methods to provide predictions up to fifteen minutes in the future, by building aggregate values for resource utilization over five minute periods which were then used as data points to generate that prediction. Krithikaivasan, et al. (2007) also focused on single steps instead of multiple steps into the future, since previous research had determined multi-step prediction to be a poor methodology. By extending the techniques used by Krithikaivasan, et al. (2007), this research examined whether long-term predictions over hours, days, and weeks are also possible.

Chapter 2

Review of the Literature

Introduction

Short-term resource prediction has been studied frequently and researchers have found that using short-term data for long-term predictions provides poor results. By using the techniques developed for short-term prediction with long-term prediction data points, this difficult problem can be addressed. Hoffman et al. (2006) reviewed a variety of resource prediction studies and assembled a best practices guide for establishing predictive services for the Apache web server. This study identified the CPU, network, disk I/O, and free memory as critical resource for web services. The steps provided by Hoffmann et al. (2006) were used as a framework for this study and are as follows:

First: gather data that will be used to define the patterns of the resource and are used to generate the forecasts. Web services continue to grow at an astounding rate and organizations are continuing to find new and innovative ways of providing content to consumers. Since web servers are common and critical to an organization's functions, this study targeted the resources required to maintain availability of web services. Further justification is included in *Forecasting for Web Services*. Because each system is different, it was important to identify resources that could provide an accurate forecast of the system. Resources considered to be critical are available memory, network bandwidth, available CPU cycles, and disk access (Istin et al., 2010). Short-term studies frequently target one or more of these resources, but these resources are just as critical to long-term prediction. Additional details are provided in *Resource Identification in*

Forecasting. The data was collected at appropriate intervals in order to be useful. It is possible to collect hundreds of samples per second; this is useful for a very fine-grained prediction when multiple decisions are needed every second. This type of sampling is important to scheduling but is not appropriate for long-term predictions. Long-term prediction needs a very coarse-grained view of the computer system; samples that are taken every minute, five minutes, or every hour are more appropriate for this type of study. Data collection will be discussed in *Data Collection for Forecasting.* Long-term prediction is also faced with the issue that a random sample may not appropriately represent the system resource utilization for a specific period of time. For example, a new job start is likely to spike CPU utilization, while jobs entering a waiting cycle simultaneously would cause CPU utilization to look as if it were idle. Researchers have used data aggregation to address issues where coarse-grained measurements are required; this will be discussed in *Data Aggregation in Forecasting*.

Second: a statistical model should be used that will evaluate the collected data and provide a pattern of events for each resource. Each set of data needs to be processed and evaluated; each resource exhibits different patterns based on the web service being performed. Statistics provide a variety of methods to evaluate sequential readings which were used to predict future behavior. Time-series analysis is a proven method of evaluating this data, including the ability to address patterns that are considered "seasonal". Those seasonal patterns of resource utilization have been seen in network traffic and were apparent in the hourly, daily, and weekly utilization of computer resources. The analysis of the data is discussed further in *Data Analysis in Forecasting.*

Third: those patterned events are then used to forecast future events for each resource. The data analysis provides a predicted future value of the series. The time-series prediction can be compared to the actual resource utilization and used to generate a confidence level, which gives administrators more information about their system resource utilization than they had previously. Additional discussion occurs in *Prediction Accuracy in Forecasting,* providing information about how various time-series were used for long-term prediction.

Fourth: in order for forecasting to be useful, the forecast must be sufficiently accurate that data fed back into the system improves the process. The statistical method called time-series analysis is used to analyze patterns when a series of measurements are related and expected to behave “seasonally”. Simple time-series methods use a fixed number of information points to provide a prediction, while more advanced techniques weight each data point based on the proximity to the prediction point. Feedback can be used to fine tune a time-series prediction system after proof is provided that the series of data provides trending or “seasonal” patterns. The methods previously used by short-term predictability research will be discussed in *Data Analysis in Forecasting* and *Prediction Accuracy in Forecasting*. This research used time-series analysis to begin to identify whether resource utilization in web services shows long-term trends that are predictable and helped take that first step in understanding the accuracy of long-term prediction.

Now that some of the patterns are better understood, it is possible to generate a feedback process which will increase the accuracy of future predictions. This feedback process can also be used to identify specific patterned events that have a higher rate of accuracy than the prediction and remove that portion of the prediction to focus on what is left. For example, if the morning pattern determines that 98% of the staff logs into the system between 7:50 and 8:10 am each work day, a future study on the resource needs of the Kerberos system could identify what resource utilization is required for staff logging in during that time frame. The Kerberos predictability would have one level of accuracy while the remainder of resource utilization would have another. Feedback from these two portions could then be used to increase the overall accuracy of the prediction. This research focused on showing that patterns exist and are predictable with some accuracy, while the feedback of information to improve that accuracy is left to future research.

Forecasting for Web Services

Business use of web services continues to grow exponentially, as businesses rely more heavily on these services, but when a denial-of-service occurs it causes a loss of revenue (Al-Ghamdi et al., 2010; Hoffmann et al., 2006). To address this demand, administrators can use clustered or cloud services. Each of these solves the issue of denial-of-service by placing enough resources at the customer's disposal that the failure or overuse of one system doesn't prevent the requested information from being delivered. The use of cloud services also protects against network failures or congestion. These solutions help guarantee availability but waste resources, since many systems remain underutilized. Researchers have explored short-term resource utilization to help improve the effective use of these wasted web service resources (Hoffmann et al., 2006).

Web systems must be adaptable by adjusting scheduling, balancing loads, and controlling overloads (Andreolini et al., 2008), in order to guarantee the availability of services. Short-term forecasting provides predictions to increase the effectiveness of load balancing, throttling, and scheduling, using information about the state of the system many times per second. Load balancing permits multiple units to share the load but doesn't prevent one of the units from becoming overloaded, since the length of service is rarely known. Throttling the incoming traffic has also been used by web services to prevent system failure at the cost of denying services to some. Scheduling determines how jobs will be serviced but has no predictive function on job length or resource needs. Web services consistently become overloaded; scheduling, throttling, and load-balancing provide some stability, but each strategy has a cost. To help prevent web service overloading, predictions need to extend beyond several minutes (Andreolini et al., 2008; Schroeder & Harchol-Balter, 2006).

Resource Identification in Forecasting

Resource overloading leads to impaired system availability and studies have identified appropriate sets of resources to monitor, if improvement in the system is to be gained. A wide variety of resources is available for monitoring computing systems. Those resources can then be split into two groups: fixed and variable (Istin et al., 2010). Examples of fixed resources include the machine's name, the speed of the processor, and the maximum bandwidth of the network. Resources that are fixed in value can be removed from consideration since they don't change or affect a system's availability. Of the remaining resources, the majority never experience binding, so, while an understanding of their utilization may be informative, predictions from these resources will not increase availability. Examples of resources that are unlikely to bind under web services would include the serial or printer ports, screen updating, and keyboard input. Just as with the resources that don't change, there is no point in measuring a resource that doesn't threaten the availability of the computer. It is important to identify those resources at risk of binding, along with identifying the pattern of their utilization: this provides advanced warning when a web system is likely to experience availability issues.

Resource utilization forecasting provides valuable details about system functionality and understanding when a resource is likely to become exhausted, permitting the operating system to take preventive action and allow increased short-term availability (Hoffmann et al., 2006). When a dynamically allocated resource becomes overloaded, the system should wait for the current load to pass before it introduces additional work, or all of the jobs will receive a smaller portion of that resource, extending the time the overload lasts; however, if the resource has a fixed allocation level, then new jobs must wait or be denied services. An example of a resource that is dynamic is the CPU, while a fixed resource is the amount of hard drive space. Memory is able to act as either a dynamic or a fixed resource since the operating system may use only the available memory, or it can offload portions to the hard drive and then reuse the same memory space, retrieving the original information when needed. All three types of resources are subject to overloading and need to be considered for long-term prediction, since each resource type is likely to display different utilization patterns. Only one resource needs to become overloaded to generate a denial-of-service on the computer affected.

This study focused on four system resources – network bandwidth, disk bandwidth, CPU utilization, and free memory – since they have been shown to have high utilization rates and are likely to become a bottleneck within a system (Istin et al., 2010). Focusing on a few key resources which most commonly become exhausted provides the most effective method of prediction. Tracking other resources for this study is not likely to indicate when a system is likely to experience a loss of availability due to a resource exhaustion, because increasing the availability of a resource that remains underutilized won't improve system performance (Harchol-Balter, Schroeder, Bansal, & Agrawal, 2003).

Previous studies (Andreolini et al., 2008; Dinda, 2006; Harchol-Balter et al., 2003; Hoffmann et al., 2006) used specific resources for their short-term predictions which permitted them to increase system availability. Two resources, CPU and network bandwidth, were used by Harchol-Balter et al. (2003) to manage queuing of jobs, since those resources frequently caused bottlenecks within web systems. This study used CPU and network bandwidth. CPU utilization is one of the most commonly overloaded resources. While it is dynamic in nature, the more jobs the CPU handles, the smaller the slice of time available to any one program. The CPU is the heart of the machine and understanding the long-term pattern of usage provides insights into how much processing power the web services needs to handle the current load pattern. Network bandwidth is another important resource, since overloading of this resource generates lost data, connection retries, and data retransmission. Each attempt to solve the problem of overloading compounds the issue, lengthening the amount of time until the overload can be cleared and services return to normal. Understanding traffic load patterns and generating long-term predictions of overloading will permit software to begin to offload server traffic to another server or identify other methods of mitigation prior to the event.

A larger set of variables was examined by Andreolini et al. (2008) in their study. These variables included CPU utilization, disk and network throughput, open sockets and files, process load, and percentage of utilized memory. The disk traffic was a measure of how much information the service needs to gather before formulating a response. For web services that have static pages, this resource utilization was likely to be low, since pages can be loaded to memory and additional requests serviced from that location. But web services now provide dynamic resources, which are frequently backed by a database that requires a variety of queries to acquire the data requested, rather than static resources. This higher level of demand from the disk subsystem indicates that the pattern of utilization for this resource is another important resource that should be investigated for long-term patterns in overloading. Memory was the last resource to be studied in this research. As requests were made of the server, additional jobs serviced those requests. Dynamic web page services use large amounts of memory to manage database requests, temporary data storage during response generation, and session management. Memory utilization patterns differed from the other three resources in the study, since this resource changes inversely to demands. Each running service reduces the amount of free memory.

Data Collection for Forecasting

The use of recorded data provides the advantage of using real data from a system, while processing that information on another system. The collection of data should have a granularity that is appropriate for the study (Traeger et al., 2008). For example, early studies that were focusing on scheduling and managing jobs needed to take data samples many times per second. Dinda's (1999) early work used load averages for thirty-five machines and gathered hundreds of samples per second to make appropriate predictions for managing processes. It would be impractical if the samples were taken every five minutes, since the computer has a constant need to manage the various processes.

Unlike short-term prediction sampling, long-term prediction uses a granularity of minutes, hours, days, or even weeks. Sampling for long-term prediction at hundreds of times per second would lead to tens of thousands of data points that would not increase the accuracy of the prediction. A sample rate of ten to sixty seconds provides an appropriate granularity for long-term prediction needs. Additionally, as long as enough data is collected and it represents an accurate workload, that data can be reused for multiple tests (Traeger et al., 2008). An appropriate sample rate was determined for the hourly predictions and the same sample data was used for the daily and weekly predictions in this study.

Krithikaivasan, et al. (2007) collected network traffic rate data at a granularity of five minutes and then used that information for their research. The longer time frame between collection points was selected because of the granularity chosen in the study. Since the object was to manage network bandwidth, a time frame of fifteen minutes was chosen. It was determined that adjusting the bandwidth for QoS more frequently than fifteen minutes could result in the thrashing of bandwidth throttling, while longer time frames could result in insufficient or excessive bandwidth. The selection of the five-minute sampling rate provided enough data to generate the network bandwidth demands for each adjustment every fifteen minutes.

Data Aggregation for forecasting

One issue that occurs with the coarse granularity of collection is that random samples may not truly represent the utilization level of the resource. This happens because resources fluctuate as usage changes: a resource may be at 80% utilization during one time period and at 10% in immediately surrounding periods. If the sample was pulled from the heavy time period, the data point would shift the prediction to a higher level of utilization than the resource was actually experiencing. This skewing of the results can be reduced by collecting samples at a smaller granularity than the prediction and averaging the results over that time frame. Krithikaivasan, et al. (2007) addressed this issue by averaging the three five-minute sample points to create a granularity of fifteen-minute increments which were used to predict network traffic demands.

Istin et al. (2010) used the trace replay to gather data and generate averages for system resource utilization over one, five, and fifteen-minute intervals; however, the study (Istin et al., 2010) didn't mention the sampling rate used for the generation of the averages. CPU utilization, I/O wait times, and memory utilization were a few of the resources that contributed data readings for analysis. Istin et al. (2010) chose averaging with a coarser granularity of sampling to reduce calculation overhead, so that the evaluation process didn't become overwhelmed with extraneous information.

Krithikaivasan, et al. (2007) collected packet level measurements each minute during a fifteen-day period. Those measurements were gathered from the routers, which aggregated the data into five-minute averages. The study then aggregated three five minute readings into a single fifteen-minute sample point. Just as the QoS demand drove the granularity of Krithikaivasan, et al's (2007) study, the prediction accuracy for hours, days, and weeks drove the granularity of the collection for this study.

Data Analysis in Forecasting

Short-term forecasting of resource utilization is frequently addressed by using various time-series analysis methods. Long-term forecasting of resource utilization has not been conducted, so researchers have not yet determined the appropriate conditions for the different time-series calculations. The prediction of a machine's state in a distributed system (Istin et al., 2010) compared a SMA, WMA, and EMA, with WMA having the smallest error in prediction, compared to the actual reading. The study also used several neural network algorithms for prediction, but they performed only slightly better than the three moving average algorithms.

Time-series algorithms are a common mechanism to analyze data that can be correlated in time. The collection of that information is as important as the analysis. Dinda's (2006) toolkit included mechanisms to gather data from selected resources and to analyze it using selected time-series analysis algorithms. Auto-regressive Moving Average (ARMA), Auto-regressive Integrated Moving Average (ARIMA), Auto-regressive Fractionally Integrated Moving Average (ARFIMA), and several nonlinear models were made available so that the users of the toolkit could get output according to the needs of the study or administrative task. Dinda (2006) evaluated the algorithms based on the necessary completion time, instead of the accuracy of the prediction, since the primary focus of the study was the creation of a toolkit that could run in a live system without interfering with normal business. As research continues on long-term resource utilization, it is important to know that real-time data collection and evaluation will be possible so that systems can constantly monitor resource utilization.

This research developed a proof of concept for one way to manage the information and predictions in real-time. This proof of concept used foundational time-series analysis methods. Naïve prediction, SMA, and EMA provided a mechanism in which the resource predictions were evaluated from a simulation data stream and a live data stream to determine the effectiveness of predicting long-term resource utilization.

More complex methods of time-series analysis have been used to improve short-term predictions and it is reasonable to expect future research into long-term utilization prediction to experience similar improvement.

Prediction Accuracy in Forecasting

Accuracy in forecasting is a necessity since predictions need to be trusted by those who intend to use the information to improve the system. One way to provide accuracy is to determine a confidence level instead of using standard deviation (Traeger et al., 2008). Identification of the range in which a resource is likely to fall at a given time provides a metric for administrators to use for making choices about load balancing, upgrading monitored resources, combining virtual machines with different resource requirements, or generating a notification of abnormal utilization for further investigation – just to name a few possibilities. Additional data analysis will help shrink the high and low utilization prediction of the monitored resources, tightening the confidence interval. When a prediction is too inaccurate, reverting a system to random selection provides better results (Schroeder & Harchol-Balter, 2006), so a confidence interval of 80% was selected since a level of 95% or 99.7% provided similar results when using Naïve prediction.

With the accuracy of the confidence level and long-term prediction shown to be possible, the results can be fed back into the prediction algorithm to improve the accuracy of the system in future research. As the accuracy improves, issues like active denial of service attacks, random events, or flash mobs will be more easily identified before the system becomes overwhelmed, causing the web system to experience a denial-of-service.

Summary

Effective studies in resource prediction use the same common framework (Hoffmann et al., 2006) to conduct the research. The studies reviewed by Hoffmann, et al. (2006) determined that collected data must come from a source that is directly related to the service of interest, and that those samples should be representative of the resource being targeted. Previous research efforts used time-series analysis to generate the prediction and also focused on the accuracy of the prediction by determining the confidence interval of that prediction (Hoffmann et al., 2006). Finally, researchers seek to use the results of a comparison between the prediction and the actual results to provide feedback information for future predictions.

Early research into resource prediction focused on system scheduling and resource utilization (Dinda, 1999). The prediction techniques were also used to extend prediction to distributed systems and resource availability (Rood & Lewis, 2008). Research has also determined that data could be collected without adversely affecting the target systems (Dinda, 2006). With the increased capability of systems to handle the prediction process and businesses’ need to guarantee the availability of web-enabled services, researchers have turned to various prediction techniques to improve service processing and availability (Andreolini et al., 2008).

Chapter 3

Methodology

Introduction

Researchers recognize that forecasting may improve scheduling (Schroeder & Harchol-Balter, 2006), load balancing (Istin et al., 2010; Sharifian et al., 2010), and resource utilization trending (Andreolini & Casolari, 2006; Hoffmann et al., 2006). They have also recognized that resource prediction is a difficult task and, while short-term predictions are reasonably accurate, when the same short-term data is used over multiple time periods, the resulting long-term predictions become much less accurate (Andreolini & Casolari, 2006; Istin et al., 2010). By focusing on near-term behavior of resources, the larger patterns have been lost. This occurs because the short-term data can only provide the current trend in resource utilization. When this is applied to multiple steps, ignoring the natural patterns of resource utilization, the prediction is less accurate.

This research demonstrates that long-term resource utilization patterns of web servers are a more accurate predictor of future resource utilization than random prediction. Resource overloading is the most common cause of general service delay or denial (P. R. Barford, 2001, p. 37). Since service availability is lost when resources become overloaded (Hoffmann et al., 2006), a more accurate understanding of future resource utilization improves the availability of those servers. Long-term resource predictability helps identify periods of system overloading and trends that lead to overloading, allowing administrators to take preemptive actions to prevent future overloading of resources. Overloaded resources are also one of the major causes of slow response or dropped responses specific to web services (Andreolini & Casolari, 2006; Schroeder & Harchol-Balter, 2006), so better methods of predicting future resource utilization permit longer periods of availability for web services.

When web services are not available, businesses suffer from a loss of sales if they rely on the availability of their web services for income (Al-Ghamdi et al., 2010). To address the possibility of overloading resources, administrators today use clustered and cloud services to guarantee maximum availability. A clustered server solution addresses periodic maximum resource needs, but it also leaves many of the system resources idle or underutilized the rest of the time. Long-term resource prediction allows administrators to better understand the pattern of resource requirements for their services. Resource prediction facilitates dynamic capacity planning, allowing an administrator to allocate additional resources when and where those resources are needed to facilitate availability of the services. Additionally, better understanding of resource utilization patterns permits services having opposing resource demands to share resources without overloading a server's resources. In each of these instances, system availability is improved.

A number of other actions may occur once a system's patterns of utilizing resources are better understood: applications may be developed to switch scheduling algorithms to better address the load (Harchol-Balter et al., 2003); distributed systems could use such information to act proactively instead of reactively to normal events that exhaust resources normally causing a system to enter into graceful degradation (Andreolini et al., 2006); system resource prediction could be used to enhance cloud computing's dynamic allocation of resources (Islam, Keung, Lee, & Liu, 2012); an additional level in quality of service and service level agreements could be created

(Krithikaivasan et al., 2007); or cloud technologies could be extended to preemptively move virtual machines to other hardware with additional appropriate resources. System resource prediction could also be used to recognize that a utilization pattern is broken, allowing services like intrusion detection systems to take a closer look at the unexpected changes. In each of the previous cases, research into using long-term resource prediction has not occurred, since the extension of short-term prediction was considered too inaccurate to be of use and researchers have been focused on other methodologies.

In this study, four resources (six metrics) were monitored to determine how predictable long term patterns using a coarser granularity of samples. These resources were free memory, network traffic volume, CPU utilization levels, and disk utilization volumes. Network traffic flow was represented by two metrics that measured the volume of traffic being sent to the web server and the amount of information transmitted back to the requester. Disk access was also comprised of two metrics that tracked the volume of data read and written to the hard drive. Hoffman et al (2006) were able to identify the resources that were most commonly the cause of system degradation for Apache web services after reviewing a large variety of resources the system was able to track. The previously mentioned resources were selected because they were identified as most likely to overload (Hoffmann et al., 2006) from either flash events or increased demand on the web service.

Resource demand generated by modern dynamic web services is highly variable. Even when web services provided only static web pages, the demand for services was growing rapidly, and continues to do so, also driving research into resource forecasting to better understand how service performance can be measured (P. Barford & Crovella, 1998). The extreme volatility of web services often leads to periods of overloading of resources (Schroeder & Harchol-Balter, 2006). As businesses continue to expand e-commerce as a mechanism for customer transactions, their reliance on web services has increased over the years, and can cause situations where customers are denied access and revenues are lost (Al-Ghamdi et al., 2010). For the preceding reasons, this research used a web-based system to gather the data required to show that long-term prediction of web service resource utilization is possible.

The remaining sections of this chapter will provide the following information: current resource prediction methodology is explored in C*urrent Research*; reasoning behind the selection of the six metrics used in this study is discussed in *Resource Selection*; since researchers use a wide variety of time series algorithms to analyze resource utilization, the *Time Frame and Evaluation Method Selection* reviews those methods most appropriate to this research; the best practices previously established in short-term studies as they apply to this work are presented in *Framework for the Research*; and, finally, the physical resources required to complete the study are identified in *Resource Requirements.*

Current Research

Previous research (Dinda, 1999; Rood & Lewis, 2008) focused on the behavior of resources to provide scheduling choices needed within the next few seconds. Sampling hundreds of times per second allows the information needed by the operating system algorithms to address issues like scheduling, load balancing, and network traffic management. But long-term patterns in resource utilization also exist and those patterns are observable. It is well-known that network patterns cycle over a twenty-four hour period (Krithikaivasan et al., 2007). Despite the acceptance of cyclical patterns that occur each day, formal research continues to explore these patterns using fine-grained sampling and short-term prediction applications. Krithikaivasan et al.'s (2007) research used the coarsest granularity found so far in research. This research extended the aggregation methodology used by Krithikaivasan et al (2007) by collecting resource utilization levels every ten seconds and aggregating that data into minute, fifteen minute, and hourly values for long-term evaluation.

This research project, by predicting future resource needs, may help advance resource availability through two active areas of research: distributed systems and web services. Research in the distributed systems area includes work by Istin et al. (2010), who focused on the prediction of resource availability. This area of research was concerned with identifying the availability of distributed web services and their ability to provide service. The moving averages were slightly less accurate than the neural networks used by Istin et al. (2010), but were much faster than the neural networks and had average percentage errors only marginally higher than the neural network algorithms. The predictions were based on samples acquired at one, five, and fifteen minute intervals. Samples were pulled for CPU and I/O statistics and provided predictions for a single step ahead (Istin et al., 2010). This study used a ten-second sampling rate and aggregated the data into coarser-grained time frames. Each time frame was then used for a single step predictions, since every additional step beyond the first amplifies the error prediction rate (Istin et al., 2010).

In addition to distributed systems research, web service is the other area of active research most relevant to this project. Andreolini, et al. (2008) focused on this area because of the volatility of activity in web services and the need to be able to predict system resource overloading. Andreolini, et al.'s (2008) work to understand loading levels is used to help prevent denial of service issues and requires the ability for efficient real-time resource evaluation. Andreolini, et al (2008) found that moving averages met this requirement for short-term prediction. Samples were taken from CPU utilization and disk throughput at one-second and five-second intervals, and were drawn from the system as it was exercised in four scenarios, including a step-up, staircase up then down, alternating between a high and low level, and a realistic scenario. After the data was collected, the study (Andreolini et al., 2008) evaluated the resulting moving averages using between thirty and one hundred twenty data points to create the prediction. In addition to the basic moving averages, Andreolini, et al (2008) evaluated the more complex moving average models of CS, Quartile-Weighted Median (QWM), AR, and ARIMA.

Resource Selection

Web services have grown quickly over the years, dramatically increasing the volume of traffic, the number of users, and the amount of data transmitted. This growing demand continues to strain existing servers (Sharifian et al., 2010). Businesses also expect Internet services to be available on demand with as few delays or outages as possible. To help compensate for this rapid growth, researchers have actively worked on resource predictability to improve availability and better manage web services. Previous research has focused on short-term predictions and has avoided longer term cyclical patterns of resource utilization because attempts to stretch the short-term predictions into multiple steps have resulted in inaccurate predictions (Andreolini & Casolari, 2006; Istin et al., 2010).

Andreolini and Casolari's (2006) study focused on the collection of CPU utilization, disk throughput, network throughput, open sockets, open files, and free memory. These readings were taken in five-second increments and used TPC-W ("TPC-W transactional Web e-commerce benchmark (Retired 04/28/2005)," 2004) as a simulator to generate the workload. Andreolini and Casolari's (2006) selection of resources helped support run-time decisions and also identified the need for an understanding of resource load conditions in order to better predict future needs.

Hoffman, et al (2006) focused on the available resources used by Apache Webserver and evaluated the effectiveness of each resource's ability to predict response times of the server. Samples were taken and then aggregated into five-minute and twenty-four hour data points. Instead of averaging all data points into the aggregation, the authors used the median value to represent the most common utilization level for the aggregate period. Hoffman, et al. (2006) selected free memory and server response time as the two variables that were most predictive of the time taken for the server to respond to a client request.

Schroeder and Harchol-Balter (2006) studied the effect of scheduling on web services experiencing a denial-of-service due to overloading. In order to better understand loading levels, they monitored CPU utilization, network traffic, disk I/O, and memory utilization. Schroeder and Harchol-Balter (2006) considered two types of overloading for their research; the first is a persistent overload caused by a constant demand on a system resource, while the second is an intermittent overload caused by a temporary bottleneck.

The research identified that, under either persistent or intermittent overload, the shortest-remaining-process-time (SPRT) scheduling algorithm provided an improvement over other scheduling algorithms, despite its bias toward processes using fewer resources. The SPRT algorithm managed service requests without starving those requests that placed a heavier demand on the resources.

Istin, et al. (2010) focused on understanding the state of a machine within a distributed system by the use of various statistics, including CPU utilization, free memory, swap memory, running and blocked processes, and I/O statistics including network and disk traffic. The resource utilization levels were then used in one-, five-, and fifteen-minute load averages and input into a neural network to predict the probability of a web service being available within the distributed system. Istin, et al (2010) reinforced the understanding that short-term prediction cannot be effectively extended to multiple steps, since the error rate becomes too high for the prediction to be reliable.

For this study, CPU utilization, free memory, sent and received network traffic, and disk reads and writes were used as the resource set for predictability, since they are the most frequently identified as overloaded resources on a web server (Andreolini et al., 2006; Hoffmann et al., 2006; Schroeder & Harchol-Balter, 2006). Additionally, these resources are easily measured through operating system logs and provide an effective indicator of the health of the web services. (Andreolini & Casolari, 2006; Istin et al., 2010). The web logging permitted samples to be taken every ten seconds; which were then aggregated using the mean value into one- and fifteen-minute groupings.

Time Frame and Evaluation Method Selection

Previous research focused on time frames of under fifteen minutes (Andreolini & Casolari, 2006; Dinda, 1999; Rood & Lewis, 2008) or on a specific subset of a server's needs such as scheduling (Chapman et al., 2007; Harchol-Balter et al., 2003; Sharifian et al., 2010), network traffic (Krithikaivasan et al., 2007), or resource availability (Rood & Lewis, 2008; Wolski et al., 1999). This study used the collection of data in ten-second increments, and then averaged that data into larger blocks for analysis, as Krithikaivasan, et al. (2007) had done. Unlike previous studies, the samples were taken over multiple seconds and not multiple times per second (Dinda, 2006). The larger time frame was selected because long-term prediction using finer data to predict multiple steps ahead is considered inaccurate (Istin et al., 2010). The use of the longer time frame and the aggregation of the data to predict a single step ahead has been successful for short-term predictions of over five minutes into the future (Krithikaivasan et al., 2007).

Andreolini, et al. (2008) evaluated SMA and EMA with several other more aggressive time-series analysis techniques in a highly volatile web service environment. Their research (Andreolini et al., 2008) found that the EMA method provided the most reasonable prediction of future load values for their web server. The newer statistical methods became too erratic in such a volatile environment and took much longer to generate the prediction than EMA. For this reason, Andreolini et al. (2008) concluded that the moving averages were appropriate at this time for predicting resource utilization, since they provided a reasonable prediction and could be used within a real-time system. Al-Ghamdi, et al. (2010) also reviewed a variety of prediction methods and also found that SMA and EMA produced similar or superior predictions to more complex prediction functions.

Istin, et al. (2010) studied a variety of prediction methods, including SMA, WMA, EMA, random predictions, and neural network techniques. In each case they (Istin et al., 2010) used the average percentage error to measure the accuracy of the single step prediction. Random prediction was less accurate than moving averages and moving averages were only slightly less accurate than neural network predictions. The study also reviewed the average error rate for multiple steps, but Istin, et al. (2010) found that with each step beyond the first, the prediction's error rate became amplified. For these reasons, Istin, et al. (2010) concluded that moving averages would provide reasonable predictions when used for a single-step prediction. This study used Naïve, SMA, and EMA prediction methods for a single step ahead.

Framework for the Research

Overview

The research was broken into two components, with each having three steps. The first component used simulation to provide a proof of concept. The second component involved the expansion of time-series analysis from short-term to long-term resource utilization prediction using trace data; the same techniques previously used were extended for the coarser granularity of the samples. Once the process and methods are better understood, they can be used for real time testing for future research. Each component also followed the same three-step process to gather, predict, and evaluate the accuracy of the selected methods.

The *Data Collection Process* discusses the resources that were monitored, which metrics were collected, and how the collection worked under the simulation and live servers. The *Data Collection: Simulation Based* section provides the justification for the use of the selected simulation and the validity of that portion of the study. The *Data Collection: Trace Based* section describes the purpose and use of live web services for this research. *Data Analysis,* a description of how the metrics were prepared for use in the prediction, also discusses how the accuracy of these predictions was measured. The final section, *Resource Predictability,* presents how the analysis of several different time frames was used to identify patterns of resource utilization for extended periods of time.

Data Collection: Process

The first step for both components involved the collection of resource utilization while the systems are under load. The logs collected data for resource utilization of CPU Utilization, network I/O, disk I/O time, and free memory (see Figure 1). The resource utilization levels were gathered using the Windows Log Manager program (Logman) to collect the data for the study. Data was collected over an eight week period for the simulator and over six months for the live systems.

The Logman program is a command line utility that creates a counter to collect performance metrics from the resources available on the Windows OS. Performance monitor (Perfmon.exe) is the equivalent graphical interface that can be used to collect real-time data. By default, the output from Logman is in a format that can be reviewed by Perfmon, but Logman was configured to record the data as a comma separated version (CSV) file. The CSV file was used as input to create and evaluate the predictions for both the simulation and real-world servers.

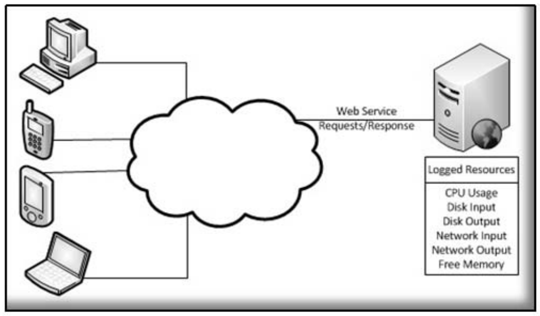

Figure 1: Server Resource Logging

The specific names of the Windows resources used in the experiment were: “\Processor(_Total)\% Processor Time” to represent the CPU utilization of the system; “\Network Interface(*)\Bytes Received/sec” and “\Network Interface(*)\Bytes Sent/sec” to represent the network I/O; “\LogicalDisk(_Total)\% Disk Read Time” and “\LogicalDisk(_Total)\% Disk Write Time” to represent the system disk I/O time; and finally, “\Memory\Available KBytes” to represent the free memory. These system counters make up the six metrics that were collected for the study. The Logman command was configured to record each of these metrics every ten seconds. The data was then stored in a raw text format using the command in Figure 2.

```
logman create counter yoas_log -f csv -v mmddhhmm -cnf 24:00:00 -c
"\Processor(_Total)\%% Processor Time" "\Memory\Available KBytes"
"\Network Interface(*)\Bytes Received/sec" "\Network
Interface(*)\Bytes Sent/sec" "\LogicalDisk(_Total)\%% Disk Read
Time" "\LogicalDisk(_Total)\%% Disk Write Time" -si 00:00:10 -o
"C:\perflogs\yoas_log"
```

Figure 2: Logman Command

The Logman configuration created a counter object called yoas_log and used the csv output (-f). The file containing the raw data was named “yoas_log” and appended a date time stamp to guarantee uniqueness (-v) during the collection process. Each file holds the data collected every ten seconds (-si) over a twenty-four hour period (-cnf) from the resources previously mentioned (-c). This configuration generated output similar to the content in Figure 3.

Data was collected at the ten-second increments and then aggregated into longer time periods as appropriate for the time frame being evaluated. Predictions based on an hourly cycle used an aggregated mean for each minute, while daily and weekly cycles aggregated the data into fifteen- and sixty-minute increments respectively. The trace-based data was collected over eight weeks for the simulation and six months for the real-time data. Once the startup process for each of the moving averages was fulfilled, the single step prediction was compared to the actual aggregate in that time slot.

```
"(PDH-CSV 4.0) (Eastern Standard Time)(300)","\\WIN-W5CXW4H8QSK\Processor(_Total)\% Processor
Time","\\WIN-W5CXW4H8QSK\Memory\Available Bytes","\\WIN-W5CXW4H8QSK\Network Interface(Marvell
Yukon 88E8050 PCI-E ASF Gigabit Ethernet Controller)\Bytes Received/sec","\\WIN-W5CXW4H8QSK
\Network Interface(Marvell Yukon 88E8050 PCI-E ASF Gigabit Ethernet Controller)\Bytes
Sent/sec","\\WIN-W5CXW4H8QSK\LogicalDisk(_Total)\% Disk Read Time","\\WIN-W5CXW4H8QSK\LogicalDisk
(_Total)\% Disk Write Time"
"01/11/2012 13:00:01.593","0.15625000000000222","1675198464","0","0","3.5659999999999998","0.01"
"01/11/2012 13:00:06.609","0","1675337728","0","0","0","0.091713395638629291"
"01/11/2012 13:00:11.593","0","1675337728","0","0","0","0.13843260188087775"
"01/11/2012 13:00:16.593","0","1675419648","0","0","0","0.10200000000000001"
"01/11/2012 13:00:21.593","0","1675501568","0","0","0","0.012"
"01/11/2012 13:00:26.593","0","1675137024","0","0","0","0.01"
"01/11/2012 13:00:31.593","0","1675214848","0","0","0","0.0080000000000000002"
"01/11/2012 13:00:36.593","0","1675173888","0","0","0","0.013999999999999999"
"01/11/2012 13:00:41.593","0","1675321344","0","0","0","0.01"
"01/11/2012 13:00:46.593","0","1675460608","0","0","0","0.01"
"01/11/2012 13:00:51.593","0","1675440128","0","0","0","0.0080000000000000002"
"01/11/2012 13:00:56.593","0","1675436032","0","0","0","0.01"
"01/11/2012 13:01:01.593","0","1675501568","0","0","0","0.0080000000000000002"
"01/11/2012 13:01:06.593","0","1675563008","0","0","0","0.012"
"01/11/2012 13:01:11.593","0.15625000000000222","1675603968","0","0","0","0.0060000000000000001"
```

Figure 3: Logman Output

For the simulation portion of the research, a single Windows IIS web server was used along with twenty-two clients on a closed network. The web server was set up using Windows 64-bit 2008 R2 Server operating system, loaded onto a Gateway E4300 (Intel Pentium 4 dual processor running at 3.4 GHz). The server was also equipped with a Marvell Yukon Gigabit Ethernet interface and a Western Digital 1600JD ATA hard drive. Both DNS and DHCP services were provided by the server for the client machines attached to a closed network for this experiment. The server and clients were connected to a Cisco Catalyst 2950 48-port switch so they could access web content provided by IIS 7.0 web services.

Each client machine was a Dell Precision T3500 with 64 bit openSUSE 11.2 installed on a portable Toshiba HDDR500E04X USB hard drive. The clients were Intel Xeon 8-core CPU running at 2.8 GHz, a Broadcom NetXtreme gigabit Ethernet card, and eight gigabytes of RAM. The installation of openSUSE used the text-based interface to maximize the amount of RAM available to the simulator being used for the experiment.

The twenty-two machines running SURGE were able to generate a sufficient traffic flow to generate a fixed pattern each hour (see Figure 4). Patterns are expected to arise in hourly, daily, and even longer sessional use of computing equipment (Krithikaivasan et al., 2007). To emulate these patterns, the scripts fired SURGE client-request sequence every fifteen minutes. The first request starts eight processes with 75 threads each at the top of the hour, the second request starts six processes of 75 threads each at quarter after, the third request starts four processes of 75 threads each at half past, and the final request starts two processes of 75 threads each at quarter of the next hour. The batch file was then set to loop for the duration of the experiment. SURGE generates random requests for files stored on the server, followed by a pause representative of the delay between requests (P. R. Barford, 2001, p. 20).

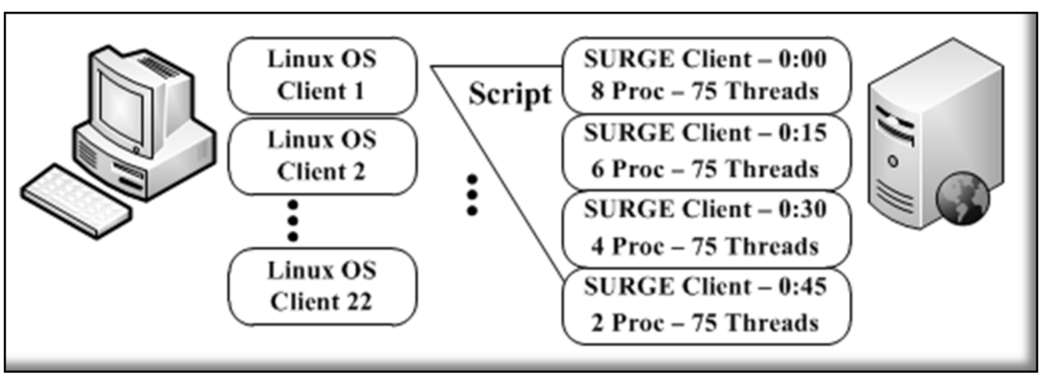

Figure 4: Client Configuration

In addition to the simulation network, data was collected from a pair of clustered web servers used by the Pennsylvania College of Technology (PCT). These servers provide information to the public, as well as to students and employees of the college. The resource utilization of the web servers used the same Logman configuration as the simulator, collecting resource utilization levels every ten seconds. The information was collected over a six month period prior to evaluation.

The logs of the simulation and live web systems were stored on the servers during the process of sampling (see Figure 3 for details of the data format). Logman was configured to start a new log file every twenty-four hours which was then processed for use in the study. The headers were stripped from all of the log files. The files were then combined into a single file and a single header was placed at the beginning of the data. SAS was then used to aggregate the ten-second data points into minute, fifteen-minute, and hour data points for the hourly, daily, and weekly forecasts respectively.

Data Collection: Simulation Based

Research frequently uses simulation as a way of providing evidence that a new method of evaluation is valid. Simulation provides a test bed for the method that is similar to a real world environment but allows the researcher to control variables that could otherwise obfuscate the results. The simulation also demonstrates the validity of the evaluation method selected for the study. This research followed this proven path of using a simulated environment in its first step. SURGE is a simulator based on user web demands that emulates user activity and pauses (P. Barford & Crovella, 1998) intended to exercise a web service. This research used SURGE to generate web traffic, while collecting the resource utilization of the CPU, memory, disk activity, and network activity on the target server. The collected data was then evaluated for long-term predictability and the accuracy of the selected moving averages used for prediction.

Barford (2001, pp. 18-19) developed the SURGE simulator to exhibit self-similar properties using analytically-based Web reference streams to accurately represent real world utilization and server behavior. Barford (2001) substantiated the accuracy of SURGE by comparing the results against SPECweb96, which was based on trace results, while SURGE is based on analytical workloads. SURGE was designed to emulate the behavior of user requests for web services and is configurable so that it can be manipulated for a variety of research purposes (P. Barford & Crovella, 1998).

Barford and Crovella (1998) created an analytic workload generator called SURGE that permits researchers to explore a variety of web service resource loads. SURGE was developed as an alternative to existing simulations that were intended to stress the resources of the target web server instead of emulating typical loads. The research to develop SURGE compared similar levels of web traffic with SPECweb96. It was determined that SURGE provided a higher level of self-similar traffic, exercised the services at a higher level, and more accurately emulated a fixed population of users than did SPECweb96 (P. R. Barford, 2001, pp. 18-19).

SURGE provides a mechanism to simulate real world traffic in a controlled environment. Using SURGE for this experiment provided the opportunity to examine resource utilization under load without having to deal with the flash traffic that can overload a live system. The SURGE-generated web requests created a clean pattern of resource loading at predictable levels and developed patterns for evaluation to determine the effectiveness of the long-term prediction methods. The results of the simulated traffic are detailed in Chapter 4.

This research extended existing short-term methods of predicting resource utilization for web services through simulation. Others (Andreolini & Casolari, 2006) have used simulation to mimic the variability seen in resources to determine the requirements for prediction of resource utilization to be effective. The use of simulation remains a viable method of testing research solutions in a controlled environment that appropriately simulates the real world characteristics of a system (P. Barford & Crovella, 1998). The Rice University Bidding System (RUBiS) simulator was used by Sharifian, et al. (2010) to test their ALB scheduling algorithm. After they (Sharifian et al., 2010) were able to show how the algorithm worked, they compared the effectiveness of scheduling to TPC-W ("TPC-W transactional Web e-commerce benchmark (Retired 04/28/2005)," 2004).

Simulation was selected as part of this research, since most prediction research has focused on prediction times under 15 minutes (Andreolini et al., 2008; P. Barford & Crovella, 1998) and the use of simulation or benchmarks is a common practice for providing proof of concept (Traeger et al., 2008). Live environments can introduce anomalous information that is not easily understood at the time of research, and therefore that anomaly must be adjusted (Hoffmann et al., 2006) or thrown out, for the research objective to be valid. SURGE provides the ability to manage the user-equivalent traffic so that usage patterns can be built into the request process. The pattern of usage built from SURGE for this experiment exercised the web service at a variety of levels and resource patterns over fixed periods of time. The resource patterns are not expected to emulate larger demand patterns, such as monthly, yearly, or seasonally, but should provide the foundation necessary to show that user equivalent requests will generate predictable long-term patterns in resource utilization.

Data Collection: Trace Based

While simulation was used to present a clean known environment, there are times when the ability of a simulation to mimic real-world characteristics fails (Hoffmann et al., 2006). When a simulation has been previously used, or if a simulation is not available for the specific characteristics the researcher is interested in, traces from live resources provide a foundation for the effort. One difficulty with using trace data lies in determining that previously-collected data contains all information required by the study. When data is missing, researchers must turn to collecting their own data.

Chapman, et al. (2007) were able to use data collected from the Condor cluster, which had 940 nodes and 1100 students, and recorded data over several weeks. The data was used to determine job length to help improve job scheduling within the cluster. Researchers have also successfully used trace data from the 1998 World Cup web site (Arlitt & Jin, 1999) to identify how well a system could handle increasing levels of traffic (Schroeder & Harchol-Balter, 2006) without suffering a loss of service. In each case, existing trace data contained the information necessary to complete the study, advancing the process of short-term prediction.

Bandwidth provisioning offered a good opportunity for using trace-based data. While various datasets are available for network traffic, Krithikaivasan, et al. (2007) was interested in using the daily activity of the network traffic at the routers within the network. Krithikaivasan, et al. (2007) used their local network to gather that trace data, since they could establish logging of specific activity on their routers. Live data was also collected by Hoffmann, et al. (2006), since their study involved reviewing a large number of resources that previous studies had not yet considered for collection. Additionally, Hoffmann, et al. (2006) didn't have access to previous data-traces that could be applied equally for evaluation of their predictions. In each of these cases, existing trace data didn't exist or didn't provide the correct information to complete the study, so the researchers established logs to collect data for later evaluation. The prediction methods demonstrated in the simulated environment were then used with the trace data.

Data analysis

The second part of the research aggregated the ten-second data point (*DP*) readings from each of the resources, CPU, network I/O, disk I/O time, and free memory (see Figure 3), into an appropriate aggregate *A* value used by the moving average functions (see Equation 1). In Equation 1, *n* is the number of data points used for the aggregated mean and *t* represents the specific samples at a given time. Resource data was collected at ten-second intervals and a given sample will be denoted as DP_t. The previous data point is identified as DP_{t-1} while the next data point is DP_{t+1}.

$$A = \frac{1}{n}\sum_{i=1}^{n} DP_{t-i}$$

Equation 1: Aggregate Function

The analysis of the aggregated data for each of the selected cycles was fed into three functions which created the predictions from the collected data. The first function was a Naïve prediction. Naïve forecasting uses the last recorded value to predict the next value (Equation 2). This method is considered an equivalent to guessing future value.

Naïve prediction is used as a baseline to compare the other forecasting functions. When a function is statistically worse than Naïve forecasting, it is considered to be inappropriate for forecasting. Conversely, if a function performs better than Naïve prediction it is considered better than guessing the next prediction value.

$$\boxed{\text{Naïve} = A_{t-1}}$$

Equation 2: Naïve Forecasting

The second function used in the study was SMA (see Equation 3). This method uses *n* samples over a period of time and adds them together, then divides the result by the number of samples to determine predicted average of the next step in time. This study used 13 steps for *n*. The drawback to using SMA is that it provides a trailing result that may be slow to catch up to the current value unless those values have been level for a while. One major advantage to this analysis method is that flash changes, very high or low values, are muted and will not create violent shifts in the prediction.

$$\boxed{SMA_t = \frac{1}{n}\sum_{i=1}^{n} A_{t-i}}$$

Equation 3: Simple Moving Average

The EMA is the most complex method of evaluating utilization levels that will be used in this study (see Equation 4). α is the weight used to split the prediction between the last reading and all of the previous readings. When α approaches 1, the EMA prediction mimics the most recent sample but is tempered by previous samples. When α approaches 0, the most recent sample tempers the prediction of all previous values. As a result, EMA can follow recent trends more quickly than a SMA prediction. The EMA process also requires initialization, which uses a prediction value based on a SMA using a small number of samples. The study used 13 samples to initialize the prediction process.

$$EMA_{t+1} = \propto * A_t + (1- \propto) * EMA_{t-1}$$

Equation 4: Exponential Moving Average

To find the prediction for the next reading, the current reading is multiplied by α while the last EMA value is multiplied by (1-α), and the two results are then added together. The result is then used as the last EMA value in the next iteration of the process. While more complex statistical time-series analysis methods have been used in other research studies, those methods provided only marginal improvement over the methods chosen for this study. Additionally, the simpler methods supplied the results more quickly than did more complex time-series equations (Al-Ghamdi et al., 2010; Andreolini & Casolari, 2006).

The effectiveness of the weight used for EMA determines the accuracy of a prediction, and a variety of formulas are used in statistics to evaluate that prediction error. To increase the accuracy of a weighted prediction, the system that is being measured should be tested using various weight combinations. A training process is used with a variety of weight settings, comparing the prediction to the actual value determining the error of the prediction. That error set is then evaluated to determine which weight yields the most accurate prediction of the future values (Makridakis, Wheelwright, & Hyndman, 1997, pp. 17-35) based on the lowest results from the error formula.

A variety of error calculations exist to help determine the accuracy of a prediction. Most of these calculations generate a value that can only be compared to other values calculated with the same formula. The other issue with many of the error values is created since the scale of the error is very difficult to understand. This study used the Mean Average Percentage Error (MAPE) (Equation 5), including the determination of the most effective weight for EMA. MAPE was selected because it provides the error value as a percent of the true value compared to the prediction. This error rate is then averaged, since it is being expressed over multiple values in the forecasting process. MAPE was found to provide a more accurate evaluation of the effectiveness of the SMA and EMA predictions (Al-Ghamdi et al., 2010) over other error calculations.

$$\boxed{MAPE = \frac{1}{n}\sum_{i=1}^{n}\left|\left(\frac{A_t - F_t}{A_t}\right) * 100\right|}$$

Equation 5: Mean Average Percentage Error

Another purpose of this study is to identify the accuracy of a prediction. To do this, the confidence interval will be calculated for each prediction using a moving standard deviation. The standard deviation will then be multiplied by the confidence level and added to the prediction to create the upper band, and subtracted from the prediction to create the lower band (see Equation 6). The use of upper and lower ranges of possible prediction values has been done by Krithikaivasan, et al. (2007) to visually identify excessive prediction errors and to review the accuracy of their predictions. For their study (Krithikaivasan et al., 2007), a confidence level of 90% was used to set the two binding curves; it was found that the results remained inside those curves most of the time.

$$\boxed{CI = \bar{x} \pm 1.282 \frac{\sigma}{\sqrt{n}}}$$

Equation 6: Confidence Interval

This study used the forecasted values, actual utilization level, and the upper and lower confidence intervals to evaluate the accuracy of the prediction. The confidence interval provides a basic mechanism of anomaly detection (Chandola, Banerjee, & Kumar, 2009). A confidence interval of 80% was selected to provide the range of normal prediction. While a higher level of 95.45% or 99.73% could have been selected for the standard deviation to generate the confidence interval, these higher values capture most of the possible samples. However, this experiment has focused on the predictability of a resource utilization level and the appropriate confidence interval to determine when a machine is no longer acting normally is left for future work.

Resource predictability

There are known patterns contained within network utilization over a 24-hour period (Krithikaivasan et al., 2007); their research indicates that those patterns may also exist at the hourly level. Krithikaivasan, et al. (2007) sampled data over ten days but eliminated the weekends from the collection process due to expectations of lower than normal network utilization. The evaluation points of this study were based on these assumptions (Krithikaivasan et al., 2007). Each time frame was sliced based on the seasonality and compared to the comparable cycle over the length of the data. For example, the hourly evaluation looked at each of the minutes, while the daily evaluation looked at each corresponding 15-minute block, and the weekly evaluation looked at each hour.

The simulation ran for six weeks and the data needed to evaluate the hourly and daily seasonality was extracted from the data set. The trace data from the live system was collected over a six month period and had data extracted to evaluate the hourly, daily, and weekly seasonal patterns. Each data set was sorted to align the data seasonally to prepare it for generating the forecast values. For each of the time frames, next step predictions were generated for CPU usage, free memory, disk input time, disk output time, network input, and network output over a range of time.

Finally, a variety of graphs and charts were created to evaluate the effectiveness of Naïve, MA, and EMA forecasting. Results were graphed to show the prediction, actual reading, and the 80% confidence interval above and below the prediction. Box and whisker charts for the resource, MAPE, and predictions were created to show the pattern of resource utilization and error rates of each cycle over an hour, day, or week. Distribution analysis was used to identify the distribution percentages of the resource and MAPE values, and linear regression analysis graphs were produced for each resource cycle to show any trend and the 80% confidence interval for that trend.

The hourly prediction time frame has sixty predictions over an hour – one for each minute. To create the predictions, the study aggregated the ten-second readings from each selected minute into a single value (see Figure 5). These values were then used as input to the Naïve, SMA, and EMA forecasting algorithms. In addition to the forecast, the confidence intervals and MAPE for each step were also calculated. The process was repeated for the second minute, third minute, and so on until all sixty minutes within the hour have predictions over the selected week.

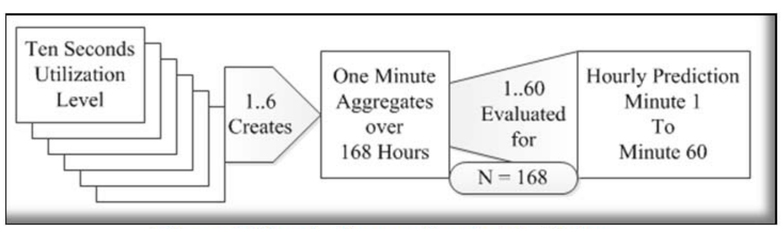

Figure 5: Hourly Aggregation for Prediction

The daily prediction time frame generated ninety-six predictions per day: one for every fifteen minutes. To create the prediction, the study aggregated ninety ten-second aggregations into fifteen-minute blocks from the hourly process. These aggregates were evaluated over twenty-eight days using the Naïve, SMA, and EMA equations (see Figure 6). The twenty-eight values over a four-week period were used to generate a prediction, the upper and lower values for the confidence interval, and the MAPE for each of the fifteen-minute increments per day. Daily analysis was conducted for the simulated data and the trace data pulled from the web servers.

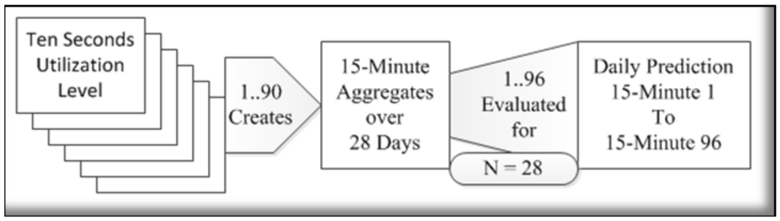

Figure 6: Daily Aggregation for Prediction

The weekly prediction time frame used three-hundred and sixty data points aggregated into an hourly mean which was used to evaluate the weekly cycles (see Figure 7). The hourly usage levels were evaluated based on each hour during the week for twenty weeks. The twenty-four data points for each hour were used with the three calculations (Naïve, SMA, and EMA) to show that the moving averages are more accurate than the Naïve forecasting method. This evaluation was completed using only the trace data from the live web servers, since at least five months of data collection was required for the evaluation to be completed.

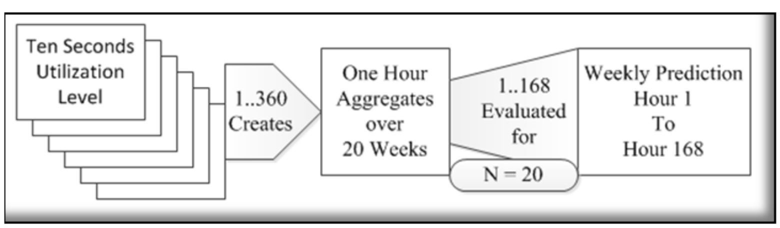

Figure 7: Weekly Aggregation for Prediction

Resource Accuracy and Confidence

This research was conducted to show that resource utilization has a degree of predictability in a long-term time frame of hours, days, and weeks. This was accomplished by comparing the prediction accuracy of Naïve forecasting to the SMA and EMA methods of forecasting. The accuracy was measured using MAPE, which expresses accuracy as a percentage of difference between the prediction value and the actual resource utilization level. The MAPE value from the Naïve prediction was compared to the MAPE values for the SMA and EMA forecasts for the minute, hour, and day prediction series. The primary experiment sliced the resource utilization levels and compared the MAPE for Naïve, SMA, and EMA. The MAPE values for each forecast were also compared when the predictions were evaluated sequentially through time.

This research also addressed the issue of confidence in the prediction. The use of a confidence interval is derived from the previous resource utilization levels and based on a standard deviation of 1.282. This represents an estimate that 80% of all readings will fall between the mean ±1.282 standard deviations. Within a time series, the mean and the upper and lower values are recalculated at each step. This information was then graphed along with the actual reading from the resource to provide a visual representation of the prediction confidence. The use of the 80% confidence interval was also reflected in the linear regression analysis.

Summary

Simulation is often used to extend research into new areas and helps to determine if the research methods are valid for the new use. This research used a closed network and simulation software to determine the effectiveness of moving average forecasting over Naïve forecasting. Short-term resource prediction methodologies were extended for use on long-term resource predictions. A variety of simulators have been used for short-term prediction, but most focused on stressing the resources in the web server instead of simulating common use. SURGE was developed specifically to emulate web traffic by simulating a number of clients that establish contact with a web server and the patterns of request and wait seen on other systems (P. R. Barford, 2001). Since SURGE was specifically designed to emulate common web request patterns, it was chosen for the simulation part of this experiment.

Researchers also use trace data to provide an environment where an experiment can be repeated and retested. The trace data also provides information where part of the data may be used for testing the experiment; the results can then be compared to the remaining data. For this study, the trace data from two clustered web servers was used as input to compare moving average forecasting to Naïve forecasting from a live environment.

In both cases, simulation and real world trace data, the validity of the predictions was determined through the evaluation of the error between the prediction and the actual resource utilization. This error measurement allowed for the selection of appropriate weight for the EMA and the comparison of the various forecasting methods used in this experiment. The results of the forecasts were graphed along with the actual utilization levels to provide a visual reference of the accuracy of the prediction.

The study also graphically mapped the confidence interval using linear regression for each step of prediction. The 80% confidence intervals provide two bands above and below the prediction to indicate where 80% of the previous utilization levels fell. Along with the actual utilization level, the graph is able to quickly determine when a resource is no longer behaving normally based on its past behavior.

Chapter 4

Results

Introduction

Resource prediction has been used to improve scheduling for operating systems (Dinda, 2002), distributed systems (Dinda, 2006), network traffic control (Abusina et al., 2005; Krithikaivasan et al., 2007), and web services (Harchol-Balter et al., 2003). In each case, short-term prediction techniques were used to improve computing service. This research used the short-term techniques previously studied to determine the effectiveness of long-term forecasting for resource utilization forecasting on web services.

This study used two sets of trace data to examine the predictability of six resource utilization levels. The first set of trace data was generated using the SURGE simulator (P. R. Barford, 2001) and collected on a Windows 2008 web server. The second set of trace data was collected from a pair of live clustered web servers located at the Pennsylvania College of Technology (Penn College). Each data set was used to create and evaluate predictions using Naïve, SMA, and EMA forecasting. The forecasting for both data sets examined predictions based on minute-by-minute blocks over an hour, and fifteen-minute blocks over a day. One final forecast set was derived from the live data, looking at an hour-by-hour block over a week. The results of this study are contained in the following sections of this chapter.

The report will be broken into two additional sections: *Simulation Results* and *Live System Results*; each section will then be broken into subsections. The *Simulation Results* will provide information about the minute-by-minute predictions in the *Each*

Minute over an Hour subsection and fifteen-minute predictions in the *Each Quarter-Hour over a Day* subsection. The *Live System Results* will be discussed in three subsections: *Each Minute over an Hour*, *Every Quarter-Hour over a Day*, and hourly predictions in the *Each Hour over a Week*.

Simulation Results

Twenty-two machines were set to run the SURGE client using four fifteen-minute sessions of seventy-five threads at decreasing process levels. Each hour, the system would reset to the highest level and repeat the sequence. The threads used HTTP 2.1 GET requests to pull files from the web server. The web server also provided DNS and DHCP services during the experiment and all of the equipment ran on a closed network. Logman was used to collect the data from six server resources: CPU, free memory, sent and received bytes through the network card, and disk read/write times. Each metric was sampled every ten seconds and logged for later analysis.

One side effect of the simulator was that it generated enough traffic against the Web Server that IIS began to record additional logging information, when the request queue periodically became full. This extra logging managed to fill the 150MB hard drive of the server midway through the testing period, resulting in a shutdown of Logman. Once the extra IIS log data was removed from the drive, the process returned to normal. This gap in data affected only the daily analysis, since the hourly analysis could be contained within the data collected at the start of the experiment. The missing data did not create any known issues for the daily analysis, since the analysis focused on fifteen-minute time slices and the simulator was generating the same level of page requests over a one-hour period.

The minute over an hour analysis aggregated and averaged six log entries for each metric into a single value, while the quarter-hour over a day aggregated and averaged ninety log entries. By using a predictable set of requests on a closed network, a known repeated level of demands could be placed on the web server, resulting in repeated patterns over each hour and day that were then analyzed for predictability. The results of the analysis for the simulation follow.

Each Minute over an Hour

Data from the logs generated by Logman between May 24, 2012 and May 31, 2012, were aggregated into one-minute data points, and then used to generate Naïve, SMA, and EMA forecasts. MAPE was used to determine how much error existed between the prediction and the actual resource level. The utilization rates generated by the CPU were relatively small, frequently falling below 35% under the heaviest simulator load and below 15% at the lowest simulator load level.

Within the four levels of simulation load, the aggregate exhibited a tight range around the forecast. The heaviest level of demand remained within a 5% range of the mean at an 80% confidence interval. At the second level the range was within 4%, while the third level of the simulation found the confidence interval pulled within 3%. At the lowest level, the interval remained under 1%.

A sample of the third level CPU utilization can be seen in Minute 34 (Figure 8). This minute was found to have an MAPE rate from 2.5% for the SMA and 3.2% for the Naïve MAPE. It also exists as a sample that closely represents an error rate at one standard deviation below the mean MAPE for all three forecasting methods (See Table 1). Table 1 also shows that both the average error rate and the deviation of the error rate were smaller for the SMA and EMA methods over the Naïve forecasting method. The blanks found in Table 1 exist because the value at one STD below the mean was less than the minimum MAPE for that resource.

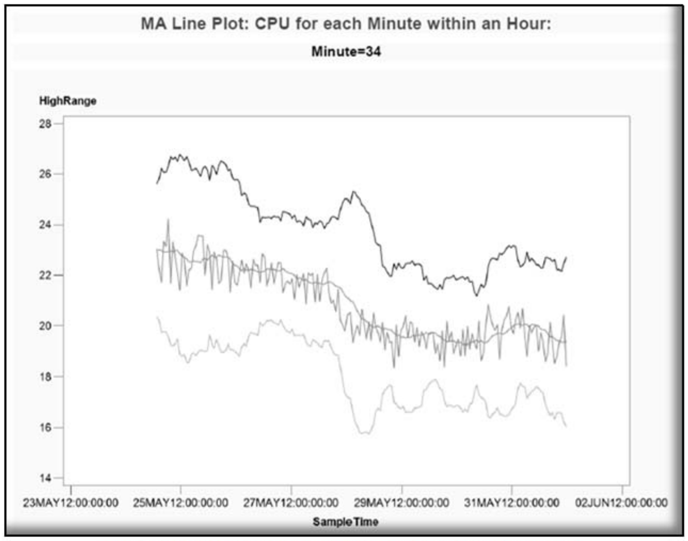

Figure 8: SMA for Hourly Cycle 34 of the CPU

The results for available memory and network traffic also show a fairly tight range of prediction values with a small standard deviation and low MAPE. This tight pattern can be seen in a box with whiskers chart (Figure 9) of the resulting quartiles for each minute within an hour. Figure 9 shows a memory utilization pattern of rising steps, since the heaviest load at the beginning of the hour requires the most memory for IIS to manage the page requests and the least amount of free memory at the end of the hour when the number of page requests is the smallest.

The box plot (Figure 9) also shows that a higher demand on the server generates a larger range of values and results in larger first and fourth quartiles. As demand shrinks, so does the variability of demand on memory. The largest range of values occurs during Minute 18, when free memory ranged from around 1.1 million to around 1.3 million bytes. With two gigabytes of RAM and a range of 0.2 million, this represents a shift in utilization of around 2% between the low and high values during that minute from hour-to-hour over the life of the sample. Table 1 shows that the network traffic also has a tightly clustered utilization range and a relatively small error rate.

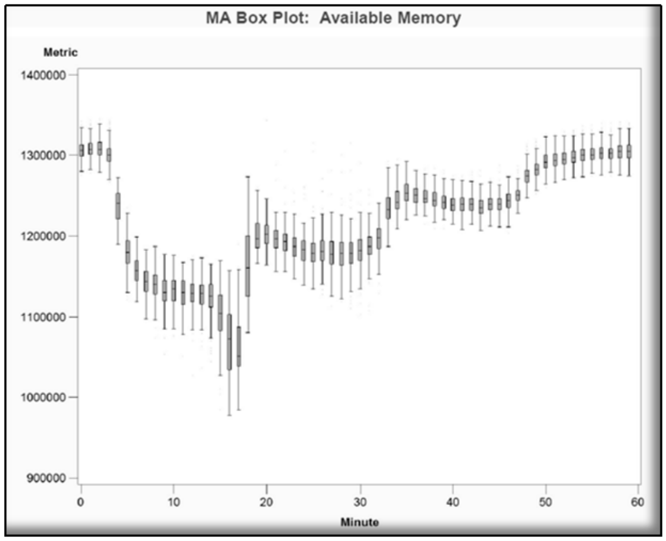

Figure 9: Box plot of Simulation Memory Utilization for each Minute

The resource readings from the disk percent utilization time exhibited a different set of characteristics than the other four data sets. Both disk read and write times appeared to be very heavily skewed. As a result, the error rates remained below 50% through the first standard deviation above the mean and then soared above 500% for the last few minutes with the highest error rates (Figure 10, Table 1).

EMA Disk Write Time MAPE Statistics

The MEANS Procedure

Variable	Mean	Std Dev	Minimum	Maximum	N
MIN_of_MAPE	0.0744677	0.0930336	0.000470878	0.5179266	60
AVG_of_MAPE	20.2417619	71.3887096	8.1978233	563.3447696	60
MAX_of_MAPE	898.2884137	6380.12	27.0950946	49492.24	60

Figure 10: MAPE Error Rate for EMA Hourly Disk Write Times

The disk read and writes appeared to exhibit this large error rate because of the low amount of time required to complete normal tasks. Then, periodically, even under the simulation, the disk reads and writes became overwhelmed with an exceptional event that was much greater than the average, which created the heavily skewed results. These outliers were also seen on the live server and deserve additional consideration in the future, since the methods used for this study did not show a high rate of predictability.

Simulation Web Server Forecasting Statistics			Results by Minute within an Hour										Summary	
			Minimum Mean/STD		~-STD Mean/STD		~Mean Mean/STD		~+STD Mean/STD		Maximum Mean/STD		Mean	STD
CPU	Naïve	MAPE	2.71		3.18		7.50		12.22		22.69		7.67	4.50
		Utilization	14.26	0.37	20.77	1.41	20.46	1.41	31.89	3.49	22.54	4.56		
		Cycle	58		34		45		12		16			
	MA	MAPE	2.03		2.53		5.75		9.03		16.53		5.83	3.29
		Utilization	14.28	0.37	20.77	1.41	33.74	4.83	31.65	3.37	22.54	4.56		
		Cycle	56		34		4		9		16			
	EMA	MAPE	2.47		2.87		6.73		11.23		20.86		7.01	4.14
		Utilization	14.26	0.37	20.77	1.41	20.46	1.41	31.89	3.49	22.54	4.56		
		Cycle	58		34		45		12		16			
Available Memory (Utilization in Millions)	Naïve	MAPE	0.26		0.30		1.19		2.04		4.99		1.19	0.88
		Utilization	1.30	0.01	1.31	0.01	1.20	0.02	1.12	0.02	1.07	0.04		
		Cycle	56		2		21		14		16			
	MA	MAPE	0.43				0.99		1.51		3.51		0.98	0.59
		Utilization	1.31	0.01			1.20	0.02	1.18	0.03	1.07	0.04		
		Cycle	2				20		29		16			
	EMA	MAPE	0.27		0.28		0.98		1.68		3.96		0.98	0.70
		Utilization	1.30	0.01	1.30	0.01	1.20	0.02	1.12	0.02	1.07	0.04		
		Cycle	56		57		21		14		16			
Network Bytes Sent (Utilization in Millions)	Naïve	MAPE	0.94		1.53		2.44		3.39		4.47		2.46	0.95
		Utilization	23.22	0.27	21.50	2.48	21.23	0.44	21.18	0.60	21.86	1.08		
		Cycle	17		18		37		8		29			
	MA	MAPE	0.78		1.24		1.97		2.88		5.04		2.04	0.86
		Utilization	21.24	0.25	22.76	0.36	15.92	0.67	20.46	0.81	16.19	4.69		
		Cycle	56		45		49		21		48			
	EMA	MAPE	0.83		1.25		2.04		2.79		3.72		2.05	0.77
		Utilization	23.22	0.27	20.63	0.32	20.72	0.49	21.42	0.69	21.85	1.09		
		Cycle	17		53		36		13		29			
Network Received (Utilization 100K)	Naïve	MAPE	0.72		1.79		6.81		12.36		22.14		7.23	5.11
		Utilization	5.61	0.08	4.87	0.13	11.72	0.79	12.94	1.39	8.81	1.76		
		Cycle	0		49		20		13		16			
	MA	MAPE	0.57		1.44		5.61		9.43		15.83		5.58	3.70
		Utilization	5.61	0.08	4.87	0.13	13.28	1.80	12.72	1.44	8.81	1.76		
		Cycle	0		49		4		14		16			
	EMA	MAPE	0.61		1.49		5.87		9.99		17.86		5.96	4.18
		Utilization	5.61	0.08	4.87	0.13	6.47	0.75	11.15	1.27	8.81	1.76		
		Cycle	0		49		18		30		16			
Disk Write (Utilization in Hundreds)	Naïve	MAPE	10.17				22.64		39.93		564.15		22.50	71.23
		Utilization	1.60	0.15			0.80	0.50	1.26	0.40	9.85	110.3		
		Cycle	35				48		17		29			
	MA	MAPE	7.45				19.24		33.39		545.49		19.01	69.22
		Utilization	1.55	0.16			0.80	0.50	1.26	0.40	9.85	110.3		
		Cycle	7				48		17		29			
	EMA	MAPE	8.20				18.29		33.77		563.34		20.24	71.39
		Utilization	1.55	0.16			0.80	0.50	1.26	0.40	9.85	110.3		
		Cycle	7				48		17		29			
Disk Read (Utilization in Thousands)	Naïve	MAPE	5.60				22.40		36.99		590.35		23.44	74.62
		Utilization	9.89	0.59			9.48	1.67	6.85	2.15	71.56	813.0		
		Cycle	57				16		17		29			
	MA	MAPE	4.41				15.67		29.94		586.34		20.11	74.46
		Utilization	9.89	0.59			4.21	2.39	6.85	2.15	71.56	813.0		
		Cycle	57				48		17		29			
	EMA	MAPE	4.83				17.77		30.97		586.83		20.92	74.43
		Utilization	9.82	0.55			9.48	1.67	6.85	2.15	71.56	813.0		
		Cycle	54				16		17		29			

Table 1: Simulation System Distribution of Each Minute Within an Hour

Every Quarter-Hour over a Day

The patterns exhibited in the fifteen-minute aggregates were also found to have a predictable pattern in most cases. This part of the research aggregated ninety of the ten-second raw data measurements into ninety-six seasonal data points. The distribution analysis exhibited by the received network transmissions (see Figure 11) was similar to other distributions in the research. Many of the distributions in both the minute and fifteen-minute blocks for the simulation frequently clustered 80% of the resource utilization level results into well-defined ranges. Figure 11 indicates that the number of bytes received over the network, in the first fifteen-minute cycle of a twenty-four hour period, was consistently between 960,000 and 1,117,000. Figure 11's distribution analysis also indicates that 90% of the aggregates over the twenty-eight days occurred within that range. Because the first cycle in the season has such a high percentage of hits within the defined range, it also exhibits the smallest MAPE rate (0.07%) (see Table 2) of this part of the study.

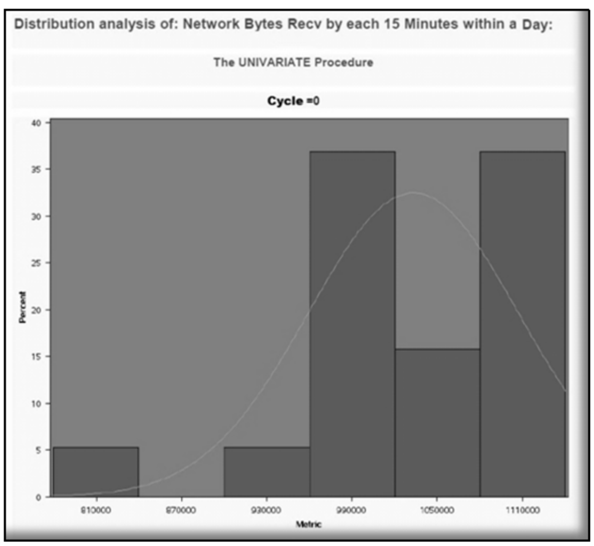

Figure 11: Distribution for Daily Cycle 0 of Received Bytes

The simulation data in the fifteen-minute cycles had only one set of error rates over 33%, which came from the declining disk write times over the analysis period and early spikes from the server OS. Disk write times ranged between 40 seconds and 58,090 seconds over the life of the experiment. Even with that high value, Cycle 64 (Figure 12) remains bounded within the assigned confidence interval.

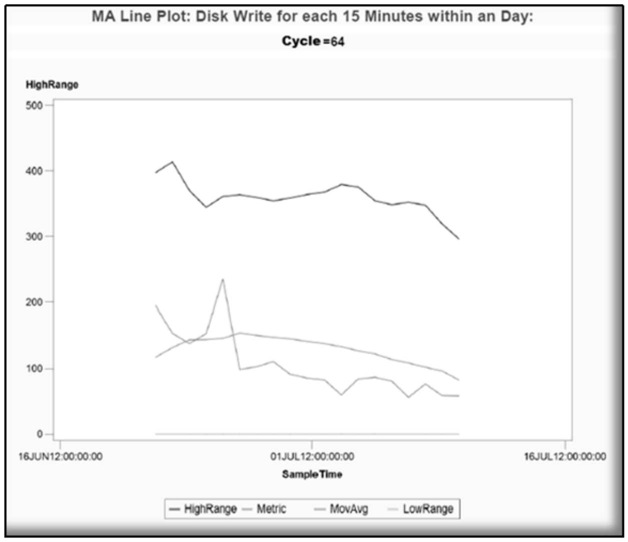

Figure 12: Disk Writes that Exhibit a Large Error Rate

The stair-step generated by the SURGE simulator started with the highest level of requests during the first fifteen minutes of the hour and stepped down to the lowest level in the last fifteen minutes of the hour, repeated for the term of the experiment. The box plot of the CPU utilization over the day shows the pattern (see Figure 13) generated by the simulator against the web services. In this case, the first set consistently pulled between 22 and 30 percent of the CPU while the last set settled between 14 and 16 percent.

These tight patterns are the norm throughout the study, but exceptions do exist. In Figure 14, the number of bytes sent from IIS in response to the simulator requests was confined to specific ranges for each fifteen minute cycle over the day, but the graph also shows that there were outliers appearing down through 16 million. These outliers are likely to always exist, leaving an open question: do they occur at predictable intervals?

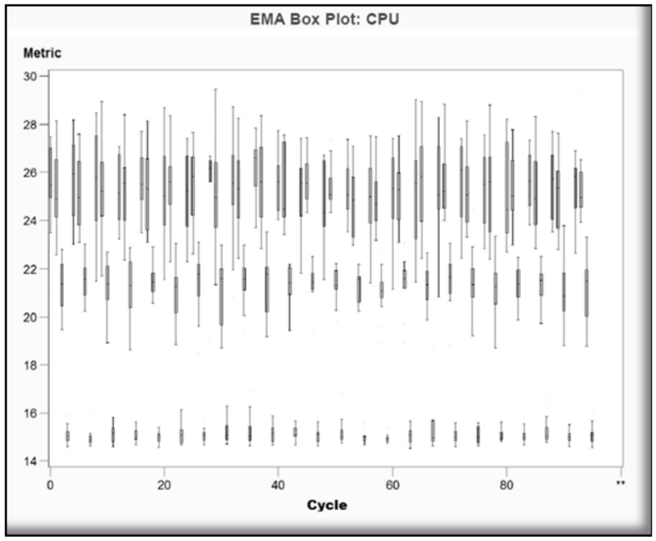

Figure 13: CPU Box Plot for Cycles over a Day

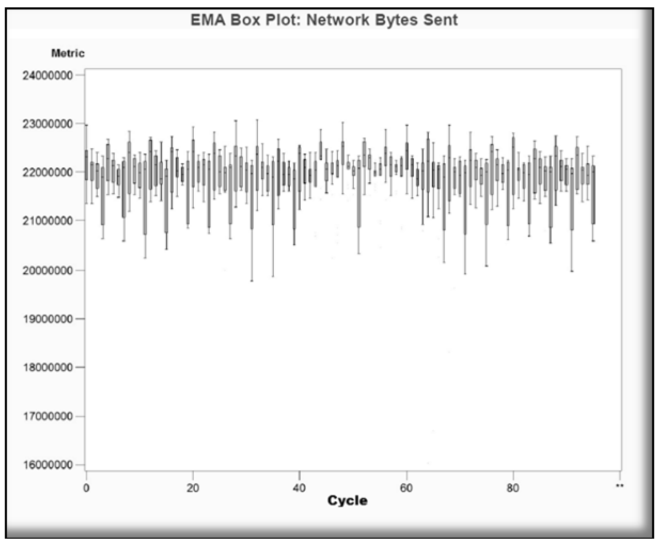

Figure 14: Net Send for Each Fifteen Minutes

The fifteen-minute cycles, like the one-minute cycles, show a level of predictability when a known input from a simulator makes web requests. The fifteen-minute forecasting shows similar or improved accuracy for SMA and EMA over Naïve prediction for CPU and network traffic utilization. Free memory and disk utilization were less accurate than the Naïve prediction for a daily seasonality.

Simulation Web Server Forecasting Statistics			Results by 15 Minutes within a Day										Summary	
			Minimum Mean/STD		~-STD Mean/STD		~Mean Mean/STD		~+STD Mean/STD		Maximum Mean/STD		Mean	STD
CPU	Naïve	MAPE	1.07		2.44		3.98		5.54		7.93		3.97	1.56
		Utilization	15.12	0.31	15.20	0.39	21.40	1.00	26.10	2.20	15.69	1.54		
		Cycle	19		11		2		28		67			
	MA	MAPE	1.87		3.15		4.72		6.39		8.08		4.76	1.60
		Utilization	15.03	0.26	15.28	0.56	21.28	1.12	25.40	1.63	25.66	2.37		
		Cycle	59		35		22		1		64			
	EMA	MAPE	1.02		2.25		3.37		5.23		8.03		3.73	1.49
		Utilization	15.12	0.31	21.51	0.65	25.63	1.08	25.82	1.98	15.69	1.54		
		Cycle	19		54		49		20		67			
Available Memory (Utilization in Millions)	Naïve	MAPE	2.65		2.98		3.42		3.88		4.69		3.43	0.44
		Utilization	1.05	0.12	0.97	0.12	0.97	0.12	0.97	0.12	0.94	0.11		
		Cycle	79		76		20		32		65			
	MA	MAPE	12.02		12.44		12.91		13.33		14.09		12.89	0.46
		Utilization	1.05	0.12	1.01	0.11	1.00	0.11	0.94	0.11	0.93	0.11		
		Cycle	71		78		50		77		53			
	EMA	MAPE	4.53		4.96		5.36		5.77		6.44		5.36	0.41
		Utilization	1.04	0.11	1.00	0.11	0.96	0.11	0.97	0.12	0.94	0.12		
		Cycle	51		30		44		12		41			
Network Bytes Sent (Utilization in Millions)	Naïve	MAPE	0.51		0.67		1.28		1.83		4.90		1.27	0.65
		Utilization	21.83	0.30	21.84	0.33	21.89	0.44	21.78	0.54	21.64	1.52		
		Cycle	54		50		17		46		64			
	MA	MAPE	1.10		1.37		2.13		2.87		4.75		2.12	0.75
		Utilization	21.83	0.30	21.92	0.34	21.92	0.62	21.45	0.77	21.64	1.52		
		Cycle	54		49		84		83		64			
	EMA	MAPE	0.63				1.29		2.00		5.36		1.29	0.74
		Utilization	21.79	0.33			21.48	0.77	21.73	0.59	21.64	1.52		
		Cycle	18				23		69		64			
Network Received (Utilization 100K)	Naïve	MAPE	0.72		2.71		4.45		6.06		22.14		4.32	1.61
		Utilization	10.33	0.74	8.75	0.43	8.67	0.57	8.71	0.40	10.34	0.57		
		Cycle	0		34		38		42		16			
	MA	MAPE	0.57		4.02		5.18		6.75		15.83		5.24	1.48
		Utilization	10.33	0.74	5.82	0.19	10.28	0.69	5.83	0.24	10.34	0.57		
		Cycle	0		19		33		7		16			
	EMA	MAPE	0.61		2.44		3.95		5.59		15.83		4.11	1.45
		Utilization	10.33	0.74	5.85	0.24	5.82	0.19	5.83	0.19	10.34	0.57		
		Cycle	0		3		19		47		16			
Disk Write (Utilization in Hundreds)	Naïve	MAPE	9.40		11.04		14.68		18.43		26.69		14.70	3.62
		Utilization	1.06	0.36	1.10	0.37	0.98	0.36	0.98	0.33	0.90	0.43		
		Cycle	13		41		16		36		67			
	MA	MAPE	34.69				43.02		55.60		159.49		42.96	12.74
		Utilization	1.17	0.38			1.10	0.42	0.90	0.43	1.14	0.49		
		Cycle	69				21		67		65			
	EMA	MAPE	13.55		15.82		19.29		22.68		33.41		19.29	3.47
		Utilization	1.07	0.35	1.11	0.40	1.10	0.44	0.80	0.32	0.9	0.43		
		Cycle	49		6		85		35		67			
Disk Read (Utilization in Thousands)	Naïve	MAPE	6.97		7.79		9.05		10.33		15.62		9.04	1.24
		Utilization	8.18	1.42	7.86	1.42	9.18	1.43	8.31	1.55	8.14	1.8		
		Cycle	13		4		71		41		64			
	MA	MAPE	14.10				19.76		30.99		170.64		19.76	15.66
		Utilization	8.55	1.32			8.69	1.66	966.0	1922	8.25	1.6		
		Cycle	69				47		68		65			
	EMA	MAPE	9.24		12.04		13.97		15.88		24.95		13.97	1.97
		Utilization	7.54	1.27	8.23	1.45	8.69	1.49	8.17	1.64	966.0	1922		
		Cycle	48		6		51		38		68			

Table 2: Simulation System Distribution of Each Fifteen-Minutes Within a Day

Live System Results

Two clustered public web servers had the Logman utility configured and started in late January 2012; they logged the six resources of interest every ten seconds through the end of July. During that time, each server accumulated over 1.7 million log entries for analysis in this research. The live trace data from the first cluster machine was analyzed for each of the live data prediction series. From that dataset, the hourly analysis used data collected between February 26th and March 5th; the daily analysis used data collected between May 16th and July 15th; and the weekly analysis used the entire data set from the first cluster server which ranged from January 20th until July 26th, 2012.

Six metrics were chosen for evaluation in this study and the numbers were run for each, but the numbers for the MAPE of the disk read metric ranged from a minimum of 70.9% error to a mean error of 2403.3% and a maximum error of 140,946.5%. These values required further investigation due to the extremely large range. A review of the raw data found that many of the ten-second samples for disk read times were zero on the live web server. The disk read times of zero were then aggregated into very low values or zero, which exaggerated the error rate for this forecast. Further investigation into the time frame where the large forecasting errors occurred found that the web server was processing scheduled system backups. For the purposes of this study, the disk read times for the live system have been excluded from these results and an analysis of disk read times will be left for future consideration.

Each Minute over an Hour

Log entries from the live trace data were aggregated into one minute units before being used for forecasting. A single week of data was used, since that number of data points permitted a good view of the prediction, the actual reading, and the two bands that define the confidence interval of the prediction (see Figure 15). The mean MAPE for the EMA prediction set over the hour was 132.36%, which happened in the 26th cycle. While the error seems to be large, it should be noted that the CPU had a mean utilization of 2.38% and a median utilization of 1.29% with a standard deviation of 5.27%, yielding an 80% confidence interval between 0% and 9.14%.

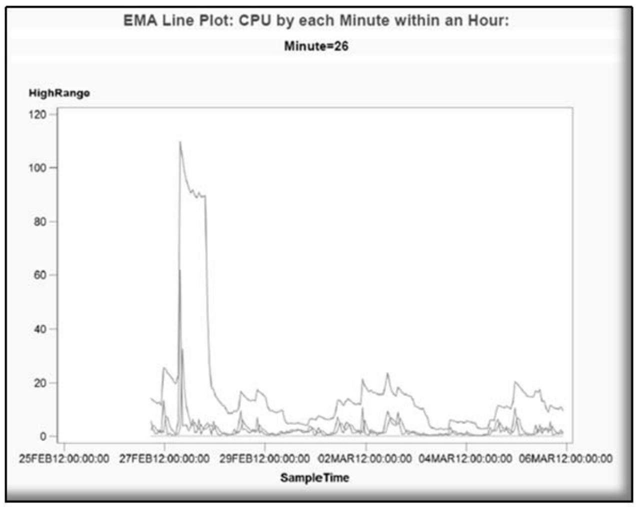

Figure 15: CPU Utilization for Cycle Twenty-Six for the Hourly Forecast

The large MAPE is seen throughout the live analysis due to the low CPU utilization levels, and the linear regression for Cycle 26 shows the early spike outside of the confidence interval along with six other data points (see Figure 16). Linear regression is normally used to show trend and correlation between two variables. This study is more interested in identifying the line that is the product of the least-squared error rate and the 80% confidence bands. The linear regression provides another example of the resource being frequently bound within a utilization range. The tight clustering of CPU can also be seen in the box and whisker graph (see Figure 17) of the hourly cycles. The height of the box and whisker graph accommodates the periodic outliers that occur within each minute, as can be easily seen in Figure 15 and Figure 16 at the beginning of each graph.

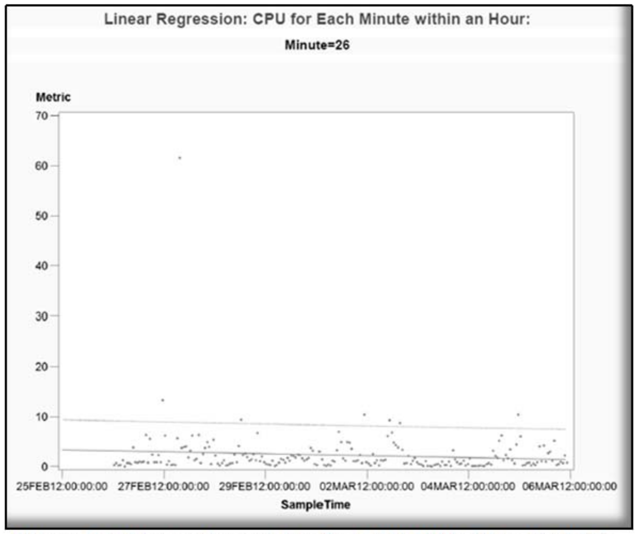

Figure 16: Linear Regression for CPU Utilization Cycle 26

In contrast, free memory for the EMA evaluation had a MAPE of 0.56% which occurred during Minute 24 of the hour (see Figure 18). Both the mean and the median of free memory during cycle 24 was 14.4 million with a standard deviation of 127,000. The 80% confidence interval for this cycle of free memory ranged between 14.24 and 14.56 million. The memory box and whisker graph (Figure 19) shows stabilization just below 14.4 million with outliers between 13.8 and 15 million.

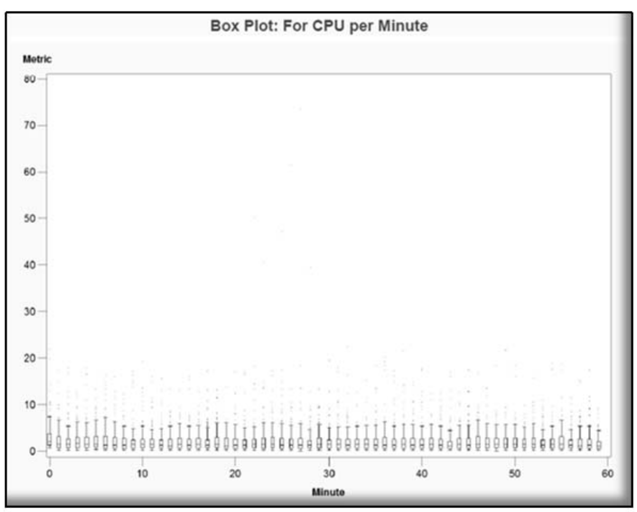

Figure 17: CPU Box with Whiskers for Each Minute over an Hour

A range of results can be seen in Table 3, reflecting the MAPE results for the minimum, mean, maximum, and ±STD values. Below each MAPE value is the minute that most closely represents that error rate, along with the metric's mean and STD. Blank values in the table indicate that the ±STD falls either below the minimum or above the maximum error rate. The error rate for the CPU and disk writes had a slightly smaller STD for both the SMA and EMA forecasting methods over the Naïve prediction method. The memory resource found that all three prediction methods had the same STD, while both network traffic utilization levels had a slightly smaller STD using EMA and a slightly larger STD using SMA over the Naïve forecast.

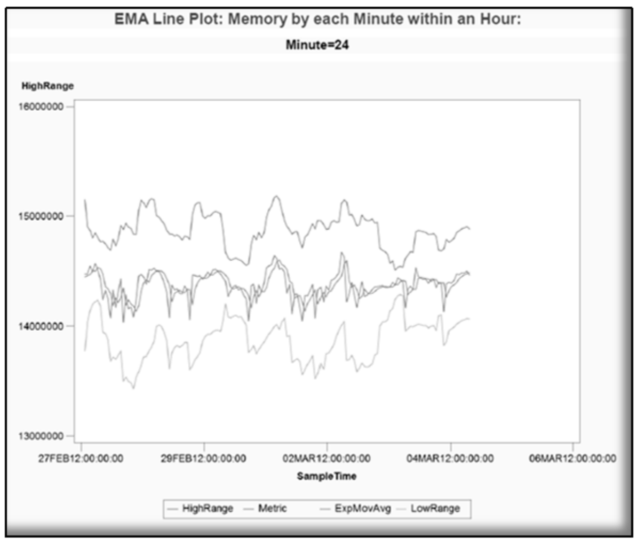

Figure 18: EMA prediction for Hourly Cycle 24 of Free Memory

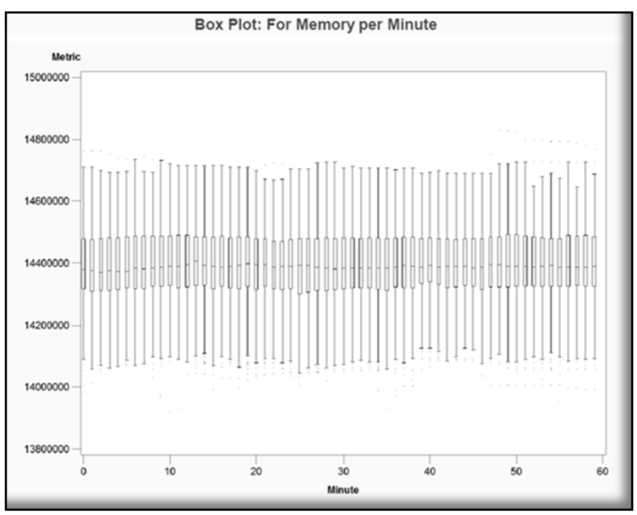

Figure 19: Memory Box with Whiskers for Each Minute over an Hour

Live Web Server Forecasting Statistics			Results by Minute within an Hour										Summary	
			Minimum Mean/STD		~-STD Mean/STD		~Mean Mean/STD		~+STD Mean/STD		Maximum Mean/STD		Mean	STD
CPU	Naïve	MAPE	97.74		113.99		138.27		161.10		206.92		138.19	23.89
		Utilization	3.28	3.41	2.10	2.71	2.12	2.30	2.48	3.52	2.04	2.40		
		Cycle	0		32		8		36		58			
	MA	MAPE	98.75		152.15		176.34		201.12		225.78		176.45	24.28
		Utilization	3.28	3.41	2.02	2.31	2.26	2.38	2.30	3.10	2.31	2.80		
		Cycle	0		30		46		38		55			
	EMA	MAPE	100.00		120.39		142.90		161.74		197.17		141.40	21.86
		Utilization	3.28	3.41	2.16	2.46	2.20	2.63	2.27	2.86	2.16	2.27		
		Cycle	0		45		10		35		1			
Available Memory (Utilization in Millions)	Naïve	MAPE	0.59		0.61		0.66		0.71		0.75		0.66	0.05
		Utilization	14.41	0.15	14.40	0.15	14.41	0.16	14.40	0.15	14.41	0.16		
		Cycle	41		22		47		3		55			
	MA	MAPE	0.71		0.74		0.77		0.79		0.83		0.77	0.03
		Utilization	14.41	0.15	14.41	0.15	14.40	0.15	14.41	0.16	14.41	0.16		
		Cycle	39		38		46		51		53			
	EMA	MAPE	0.54		0.57		0.61		0.65		0.70		0.61	0.04
		Utilization	14.41	0.15	14.41	0.15	14.41	0.15	14.40	1.49	14.41	0.16		
		Cycle	41		38		9		2		55			
Network Send (Utilization in Thousands)	Naïve	MAPE	145.47		159.75		268.35		366.53		649.13		262.04	103.71
		Utilization	56.48	68.27	54.94	49.42	48.32	46.3	56.27	80.31	63.66	82.3		
		Cycle	29		54		9		15		17			
	MA	MAPE	264.32		327.37		423.79		513.50		667.85		419.25	89.96
		Utilization	59.86	104.7	67.47	112.5	52.28	61.0	56.27	80.31	74.17	297.2		
		Cycle	32		24		20		15		2			
	EMA	MAPE	175.20		185.01		276.58		376.70		751.39		279.13	95.13
		Utilization	51.83	50.59	56.13	95.57	67.82	192.6	56.27	80.31	68.45	219.6		
		Cycle	51		42		22		15		41			
Network Received (Utilization in Thousand)	Naïve	MAPE	314.29		431.85		565.64		700.08		857.03		560.84	129.43
		Utilization	121.9	258.9	129.0	289.9	148.6	348.5	174.0	376.5	136.0	339.1		
		Cycle	42		28		10		55		50			
	MA	MAPE	516.39		623.78		759.17		893.88		1155.09		760.84	131.07
		Utilization	95.9	194.2	166.9	514.0	134.6	272.5	151.6	331.5	174.0	376.5		
		Cycle	53		49		47		48		55			
	EMA	MAPE	349.71		480.54		582.40		676.87		830.31		581.13	106.36
		Utilization	135.4	267.7	188.2	350.0	166.9	514.0	132.0	302.9	158.6	280.7		
		Cycle	59		2		49		13		38			
Disk Write	Naïve	MAPE	31.85		31.85		58.31		79.08		165.58		57.75	25.04
		Utilization	0.38	0.17	0.38	0.17	0.52	0.86	0.61	0.52	0.80	2.93		
		Cycle	32		32		22		59		2			
	MA	MAPE	25.50		26.99		47.18		67.46		140.83		48.74	21.86
		Utilization	0.38	0.17	0.38	0.21	0.48	0.51	0.59	1.36	0.80	2.93		
		Cycle	32		57		52		20		2			
	EMA	MAPE	27.90		30.12		52.73		74.87		150.28		51.64	22.62
		Utilization	0.38	0.17	0.42	0.21	0.90	1.24	0.59	1.36	0.80	2.93		
		Cycle	32		55		5		20		2			
Disk Read	Naïve	MAPE	70.92				24937.64				36267.90		24937.64	-24937.64
		Utilization	0.14	1.70			0.17	2.11			0.11	1.29		
		Cycle	40				39				52			
	MA	MAPE	79.94				5149.82		14647.00		140946.48		5149.82	9497.18
		Utilization	0.14	1.70			0.12	1.43	0.15	1.79	0.19	2.30		
		Cycle	40				24		37		18			
	EMA	MAPE	77.50				2761.07		7592.28		21258.21		2761.07	4831.21
		Utilization	0.46	3.40			0.14	1.78	0.17	2.14	0.17	2.11		
		Cycle	2				17		19		39			

Table 3: Live System Distribution of Each Minute Within an Hour

Because of the large number of zero readings pulled from the disk read times, the STD was large enough to create a deviation curve that was heavily skewed to the right.

This resulted in a negative result for the -STD position, and since MAPE is measured as an absolute value and is always positive, it doesn't make sense to have a negative value; therefore, the boxes were left blank. The results also indicated that it isn't reasonable to use either the SMA or EMA prediction when most of the readings are zero. While the Naïve forecast had a 70% MAPE, the EMA forecast rose to 4897% and the SMA forecast was at 18,265% for one standard deviation.

The minute by minute cycle shows some improvement in forecasting of EMA for the available memory and disk writes over Naïve forecasting and similar results for network sent and received bytes. When SMA is compared to the Naïve forecasting, it showed an improvement only for the disk write times, which was also had a smaller average MAPE than EMA. For all of the other metrics, the SMA was substantially higher than the Naïve forecasts. Due to the large number of zeros for the disk read, a different methodology of measurement will be needed to determine predictability.

Every Quarter-Hour over a Day

The daily analysis was done over twenty-eight days and divided each day into fifteen-minute cycles. The web server used for the evaluation saw level utilization during Cycle 49 (See Figure 20). This time period started each day at 12:15 p.m. and frequently generated less than 100,000 bytes of incoming traffic to the web server. Only three of the days had their average traffic above the upper confidence interval. One day saw a level of around 750,000 bytes of incoming traffic. The mean for this cycle was 74,482 bytes with a STD of 107,260. The distribution of this cycle shows that about 95% of the traffic averages below 180,000 bytes (see Figure 21).

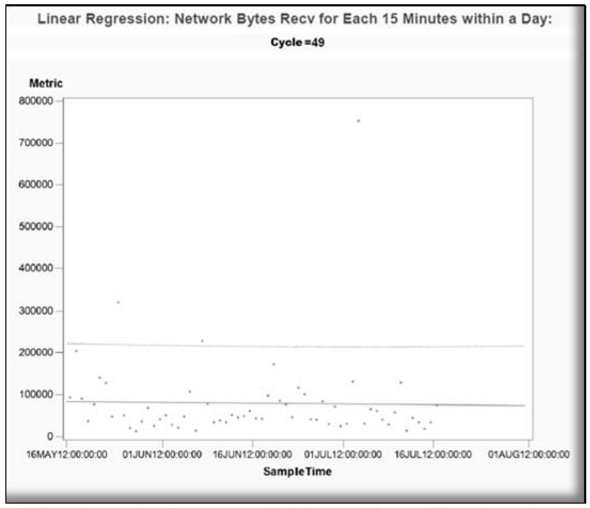

Figure 20: Linear Regression for Daily Cycle 49 of Received Data

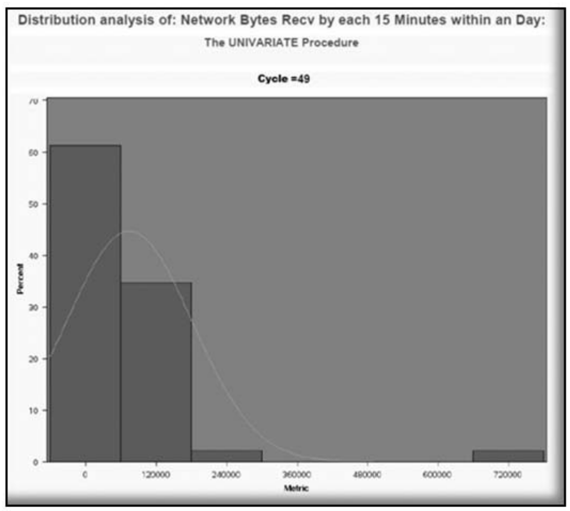

Figure 21: Distribution for Daily Cycle 49 of Received Data

While Cycle 49 occurs just after noon, Cycle 14 represents the time slot of 3:30 a.m. and shows a very low pattern of incoming data (see Figure 22). Most of the traffic was generating an average traffic level of 27,446, with nearly 80% of all traffic below 40,000 bytes. Like Cycle 49, this time frame saw only a few average samples outside the 80% confidence level. Eight points could be examined as outliers, with two of those samples hovering around an average of 120,000 bytes received over the fifteen-minute period.

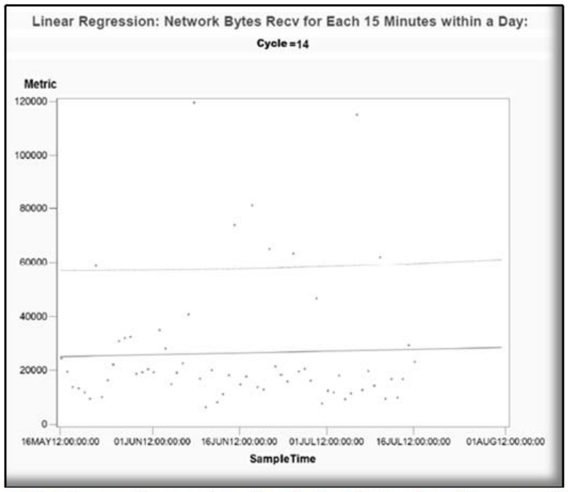

Figure 22: Linear Regression for Daily Cycle 14 of Received Data

The daily forecasting showed that both the SMA and EMA had a lower average MAPE for every resource, excluding the disk read times, over the use of Naïve forecasting. In each case, the STD of SMA and EMA were also smaller than the Naïve forecasting for the CPU, free memory, network traffic, and disk write times. Additionally, both the minimum and maximum errors for the SMA and EMA forecasting are lower than the Naïve forecast method. Unlike the hourly predictions, which showed some improvement and favored the EMA forecasting method, the daily patterns were most accurately predicted by the SMA forecasting method.

In some cases, there was a substantial improvement in the prediction accuracy. The SMA for the CPU had an error rate (62%) that was a 24% improvement over the Naïve error rate (77%). The network bytes sent MAPE was improved by 28%, reducing the MAPE for the Naïve forecast from 269% to 210% for the SMA MAPE. The disk write time MAPE also saw a 22% improvement from 28% for the Naïve MAPE to 23% for the SMA MAPE. The free memory and bytes sent error rates also saw an improvement over the Naïve forecasting error, but both were less than 5%.

Live Web Server Forecasting Statistics			Results by 15 Minutes within a Day										Summary	
			Minimum Mean/STD		~-STD Mean/STD		~Mean Mean/STD		~+STD Mean/STD		Maximum Mean/STD		Mean	STD
CPU	Naïve	MAPE	16.13				77.21		146.56		601.24		77.33	80.39
		Utilization	3.12	0.52			1.68	1.27	1.67	1.98	2.91	4.28		
		Cycle	4				86		95		93			
	MA	MAPE	11.52		26.54		62.59		95.78		266.29		62.43	37.94
		Utilization	3.12	0.52	1.32	0.39	2.26	2.39	1.84	2.02	2.91	4.28		
		Cycle	4		74		57		2		93			
	EMA	MAPE	14.13				68.48		130.28		446.02		68.19	59.99
		Utilization	3.12	0.52			2.34	2.00	1.85	2.53	2.91	4.28		
		Cycle	4				55		1		93			
Available Memory (Utilization in Millions)	Naïve	MAPE	0.85		1.01		1.25		1.49		1.76		1.25	0.25
		Utilization	14.75	0.16	14.56	0.19	14.53	0.18	14.46	0.21	14.47	0.25		
		Cycle	20		90		78		66		48			
	MA	MAPE	0.62		0.85		1.02		1.20		1.32		1.02	0.18
		Utilization	14.75	0.16	14.67	0.16	14.56	0.19	14.43	0.21	14.45	0.22		
		Cycle	21		9		90		64		67			
	EMA	MAPE	0.70		0.85		1.05		1.25		1.51		1.05	0.20
		Utilization	14.75	0.16	14.70	0.18	14.48	0.20	14.44	0.20	14.47	0.25		
		Cycle	21		23		72		60		48			
Network Send (Utilization in Thousands)	Naïve	MAPE	39.63				244.39		919.23		4575.50		269.18	748.48
		Utilization	47.32	21.99			35.15	16.2	54.85	86.89	45.73	51.5		
		Cycle	66				79		86		93			
	MA	MAPE	30.79				214.01		510.26		1445.31		209.81	314.00
		Utilization	42.31	23.41			61.05	117.3	31.07	13.0	55.48	75.4		
		Cycle	70				31		91		85			
	EMA	MAPE	35.79				235.32		748.36		2914.87		245.06	498.24
		Utilization	42.31	23.41			101.6	152.7	55.48	75.4	45.73	51.5		
		Cycle	70				32		85		93			
Network Received (Utilization in Thousand)	Naïve	MAPE	49.96				162.33		291.15		1170.72		162.85	153.88
		Utilization	44.42	20.45			63.21	127.1	87.96	127.1	73.29	115.1		
		Cycle	74				81		86		94			
	MA	MAPE	51.54		61.61		152.21		252.71		522.94		156.20	95.42
		Utilization	26.17	13.45	50.19	29.85	43.31	76.16	48.9	54.3	63.21	127.1		
		Cycle	6		71		2		82		81			
	EMA	MAPE	48.07				152.66		260.79		744.90		151.27	109.01
		Utilization	44.42	20.45			32.99	18.68	44.14	127.0	73.29	115.1		
		Cycle	74				0		20		94			
Disk Write	Naïve	MAPE	7.51		10.17		27.95		43.79		150.44		27.77	16.62
		Utilization	0.39	0.04	0.42	0.05	0.54	0.31	0.44	0.24	1.28	3.76		
		Cycle	79		77		24		92		12			
	MA	MAPE	9.39				23.45		38.57		132.53		23.43	14.96
		Utilization	0.39	0.04			0.47	0.17	0.51	0.46	1.28	3.76		
		Cycle	79				30		26		12			
	EMA	MAPE	8.04				25.21		41.01		170.13		25.10	18.63
		Utilization	0.39	0.04			0.543	0.160	0.440	0.240	1.275	3.762		
		Cycle	79				3		92		12			
Disk Read	Naïve	MAPE	279.19				15446.13		85852.46		395526.59		15829.27	59041.16
		Utilization	0.002	0.005			0.332	2.256	1.625	5.450	1.280	4.053		
		Cycle	80				5		1		0			
	MA	MAPE	298.70				36366.57		106198.67		591271.14		38703.57	104679.3
		Utilization	0.049	0.147			0.06	0.42	0.62	2.52	1.36	4.57		
		Cycle	28				75		64		3			
	EMA	MAPE	230.70				32276.70		114882.87		552890.44		30947.44	92669.35
		Utilization	0.001	0.003			0.444	1.833	1.625	5.450	1.357	4.572		
		Cycle	85				68		1		3			

Table 4: Live System Distribution of Each Fifteen-Minutes Within an Day

Each Hour over a Week

The length of the simulation was just over four weeks, which didn't provide enough data for a weekly analysis. Even with the live data, the amount of time needed to gather enough records to use the moving averages totaled six months. During that period, approximately 1,681,000 records were collected and aggregated into approximately 28,000 data points for analysis. Each data point represented the average utilization of the resource over a one hour period. A total of 168 cycles per week was then used to predict future utilization levels. Naïve, SMA, and EMA forecasting were used to determine how well these predictions fit within the resulting confidence interval.

The hourly cycles showed improved predictability for available memory and disk write time for both the SMA and EMA forecasting over the Naïve method. The MAPE for the EMA prediction of the CPU utilization also showed marginal improvement over the Naïve error rate, while the SMA error rate for the CPU prediction was 13% worse. SMA was also 39% less accurate for the network bytes sent and nearly 54% worse for the network bytes received.

The mean error rate of the Naïve prediction for the network bytes received was 96.29% during the 82nd hour of the week. The linear regression chart (see Figure 23) and Naïve forecast chart (see Figure 24) show the results of the prediction for the aggregated data. The Naïve hour that represented the mean occurred on Tuesdays at 10:00 a.m. and had an averaged 162,700 bytes sent with a STD of 128,400 (see Figure 25). The SMA had the mean MAPE during the 129th hour of the week. The linear regression chart for the SMA analysis (see Figure 26) and forecast chart (see Figure 27) also indicate a tight pattern of aggregate data. The SMA hour occurred on a Thursday at 9:00 a.m. and had a lower average of 108,400 bytes sent with a STD of 99,100 (see Figure 28).

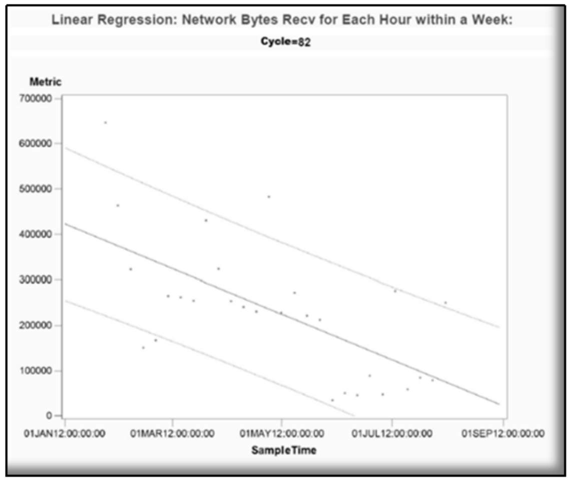

Figure 23: Linear Regression for Weekly Cycle 82 of Received Data

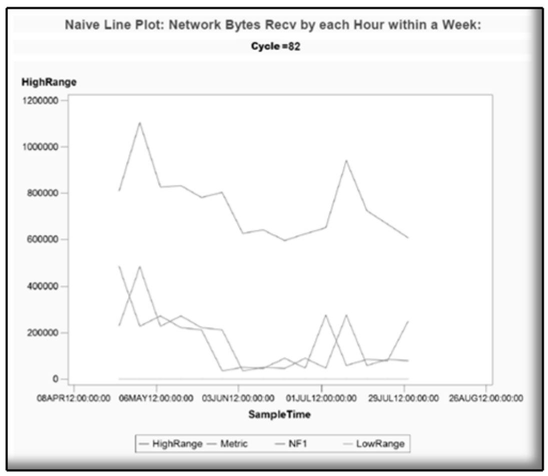

Figure 24: Naïve Forecast for Weekly Cycle 82 of Received Data

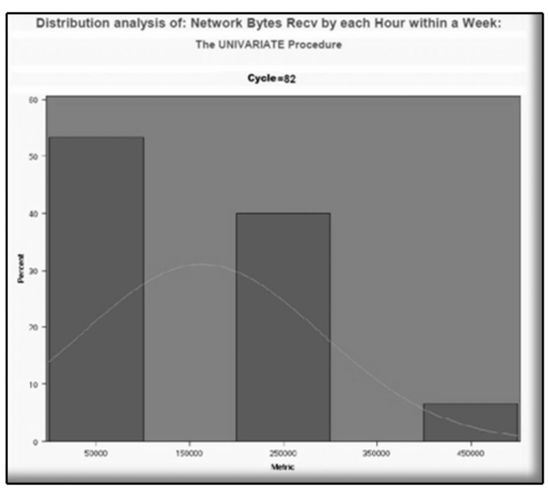

Figure 25: Distribution for Weekly Cycle 82 of Received Data

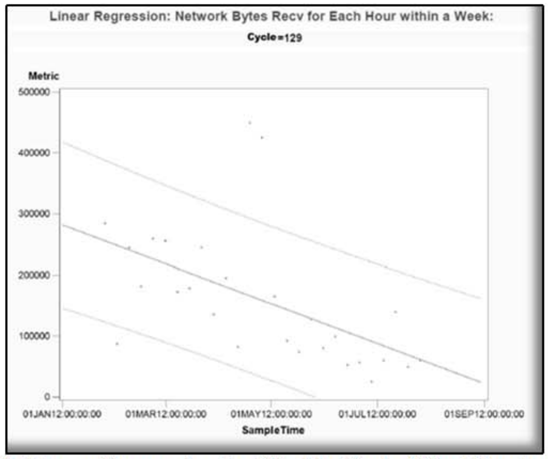

Figure 26: Linear Regression for Weekly Cycle 129 of Received Data

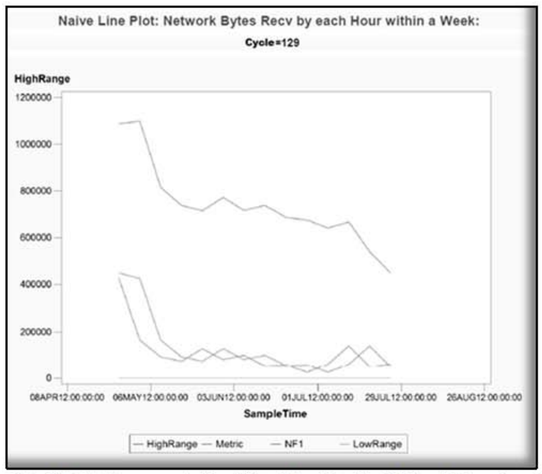

Figure 27: SMA Forecast for Weekly Cycle 129 of Received Data

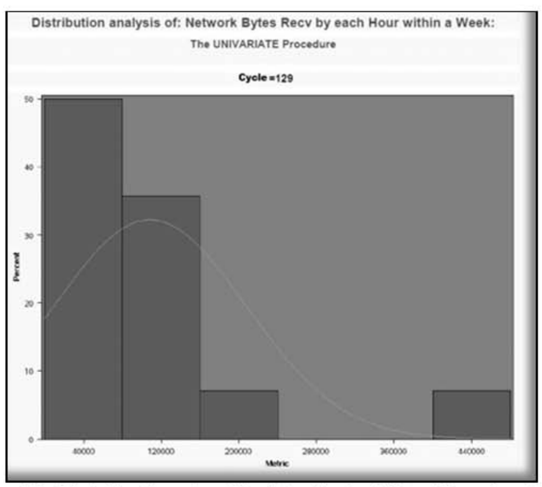

Figure 28: Distribution for Weekly Cycle 129 of Received Data

The error rate for the SMA was larger than the Naïve forecast for network bytes sent. However, SMA has a well-defined range of values with nearly 85% of the aggregates falling between 4,000 and 16,000. In both cases the aggregates show a strong trend of declining traffic which is most notable in the middle of May: a drop most likely caused by the departure of the student body from the school, although it exists in a seasonal pattern of yearly events which is too large for this study to analyze.

The box and whisker analysis of bytes sent from the live server for a weekly season is represented in Figure 29. The graph shows seven rises in outgoing traffic and consistent valleys between each peak. The highest utilization occurs each day from around 8:00 a.m. until about 11:00 p.m., peaking around 3:00 p.m. each day. The valley appears to be at its lowest values around 4:00 a.m. This cycle appears to repeat itself for each day over the week.

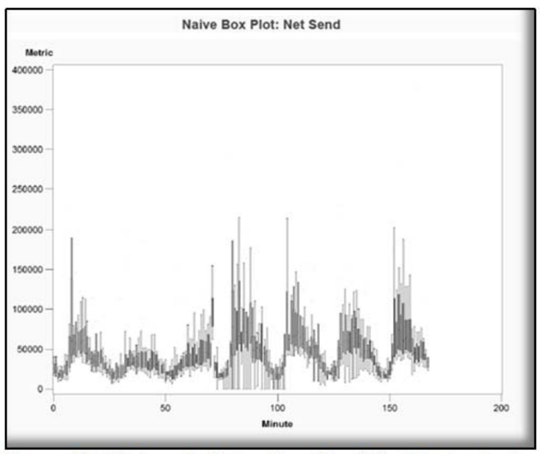

Figure 29: Network Bytes Sent for Weekly Analysis

Summary

Both the SMA and EMA forecasting were found to be better predictors of resource utilization than Naïve forecasting for certain time periods. The best forecasting results were seen for live daily prediction of resource utilization. The worst results were seen from attempts to analyze the disk read times, most likely due to the large number of times during which that resource had no activity during the initial logging phase. Finally, mixed results were seen from the live system when using the hourly or weekly forecasting.

Live Web Server Forecasting Statistics			Results by Hour within an Week										Summary	
			Minimum Mean/STD		~-STD Mean/STD		~Mean Mean/STD		~+STD Mean/STD		Maximum Mean/STD		Mean	STD
CPU	Naïve	MAPE	15.47				53.71		95.37		358.68		54.31	40.84
		Utilization	1.69	0.44			1.97	0.88	2.02	2.06	2.00	1.30		
		Cycle	1				68		63		92			
	MA	MAPE	12.84		21.34		60.71		102.07		317.96		60.82	41.28
		Utilization	1.41	0.19	1.41	0.33	2.98	1.46	2.22	2.36	2.00	1.30		
		Cycle	38		97		84		78		92			
	EMA	MAPE	12.67				53.18		101.56		362.83		53.45	46.91
		Utilization	1.41	0.19			1.95	1.90	2.29	2.25	1.72	2.10		
		Cycle	38				167		116		95			
Available Memory (Utilization in Millions)	Naïve	MAPE	0.57		0.82		1.30		1.81		2.74		1.31	0.50
		Utilization	14.67	0.10	14.58	0.18	14.58	0.22	14.53	0.21	14.76	0.28		
		Cycle	1		42		17		131		124			
	MA	MAPE	0.56		0.84		1.12		1.40		1.81		1.12	0.28
		Utilization	14.68	0.08	14.67	0.14	14.36	0.18	14.59	0.22	14.53	0.30		
		Cycle	76		30		110		59		128			
	EMA	MAPE	0.60		0.82		1.12		1.43		2.04		1.12	0.31
		Utilization	14.68	0.08	14.59	0.18	14.50	0.22	14.36	0.24	14.62	0.33		
		Cycle	76		63		134		83		97			
Network Send (Utilization in Thousands)	Naïve	MAPE	13.87				116.63		149.66		6108.55		118.90	534.37
		Utilization	35.19	7.11			42.36	44.77	64.74	63.20	49.58	34.38		
		Cycle	35				32		7		93			
	MA	MAPE	29.17				171.55		203.11		5479.21		165.90	619.59
		Utilization	35.19	7.11			17.02	6.60	46.12	26.10	49.58	34.38		
		Cycle	35				75		91		93			
	EMA	MAPE	13.97				148.24		180.12		6891.76		148.67	689.37
		Utilization	40.21	8.46			64.74	63.20	46.12	26.10	49.58	34.38		
		Cycle	166				7		91		93			
Network Received (Utilization in Thousand)	Naïve	MAPE	27.97				96.29		122.57		1184.62		96.59	121.81
		Utilization	43.06	15.82			162.7	128.4	82.06	99.50	98.59	94.75		
		Cycle	48				82		18		93			
	MA	MAPE	36.41				148.87		207.18		1613.97		149.32	203.58
		Utilization	35.65	43.07			108.4	99.11	109.2	71.38	89.36	65.96		
		Cycle	50				129		130		92			
	EMA	MAPE	29.78				118.35		203.34		2126.07		115.64	207.92
		Utilization	131.0	105.0			104.2	84.97	84.57	55.75	85.68	119.0		
		Cycle	155				68		91		95			
Disk Write	Naïve	MAPE	3.99				23.85		29.09		367.01		23.87	29.36
		Utilization	0.39	0.02			0.48	30.00	0.60	0.18	2.15	3.94		
		Cycle	61				102		121		123			
	MA	MAPE	3.80				17.31		20.38		250.60		17.26	20.40
		Utilization	0.40	0.02			0.51	0.11	0.48	30.00	2.15	3.94		
		Cycle	52				3		102		123			
	EMA	MAPE	3.46				19.21		29.43		372.12		19.54	29.38
		Utilization	0.39	0.03			0.47	0.11	0.68	0.22	2.15	3.94		
		Cycle	64				99		97		123			
Disk Read	Naïve	MAPE	56.93				2322.90		7340.16		71428.16		2323.30	7228.57
		Utilization	0.001	0.001			0.01	0.02	0.17	0.50	7.29	8.73		
		Cycle	42				9		51		88			
	MA	MAPE	54.94				4121.49		17360.36		104619.05		4447.82	12561.63
		Utilization	0.002	0.002			0.17	0.34	0.16	0.40	7.29	8.73		
		Cycle	37				129		96		88			
	EMA	MAPE	67.15				3175.66		15254.53		144831.50		3082.74	13109.74
		Utilization	0.001	0.001			0.10	0.25	0.08	0.18	7.29	8.73		
		Cycle	42				99		86		88			

Table 5: Live System Distribution of Each Hour Within a Week

Chapter 5

Conclusions, Implications, Recommendations, and Summary

Conclusions

Basic moving average forecasting is a viable method of prediction of web service resources including CPU, free memory, network bytes sent and received, and disk write times. In some cases, test results for moving averages were as good as those for Naïve forecasting. Only a few tests – disk write time for the simulator daily forecasting and network bytes sent/received for the live weekly forecast – found both moving average forecasts to be worse than Naïve forecasting by more than 10%.

Linear regression, box and whisker, and line graphs provide visual indicators that each of the analyzed resources has a small range of seasonal data and falls within an 80% confidence level. Each seasonal test (hourly, daily, and weekly) had some success at predicting the resource utilization patterns. The most successful set of predictions was the daily analysis, which showed that both the SMA and EMA were better predictors than Naïve forecasting on the live web server. Even when SMA or EMA performed similarly to the Naïve forecast, linear regression showed that the resource range can be considered small with an 80% confidence interval when the range is compared to the full range of all data collected for that resource.

For example, the CPU utilization on the live web server averaged around 3% with a 80% confidence interval of ±3%, effectively providing a range of 0% to 6% for CPU utilization. The live system had free memory averaging around 14,400,000 with an 80%

confidence interval of ±325,000. Because free memory is very stable and the values are large, the MAPE was frequently below 1%.

The network traffic statistics showed that received bytes were around 140,000 and ±400,000 for the 80% confidence interval, and sent bytes around 60,000 with the 80% confidence interval of ±150,000. This provides an average range of 0 to 540,000 for network bytes received and 0 to 210,000 for network bytes sent. The Internet connection for the live web servers has a bandwidth of 32,768,000 bytes from a 250Mb Internet connection. The web server received bytes range represents about 1.6% of the maximum allowable traffic, while the bytes sent represents about 0.6% of the maximum allowable traffic.

In each case, the range of the resource utilization created by the 80% confidence interval is small compared to the maximum possible range. The CPU exhibited a 6% range; the free memory range represented less than 0.1% of the total available memory; the network bytes received range represented about 4.1% of the total available bandwidth; and the network bytes sent range was about 1.6% of the available bandwidth. Each resource examined in this research found the range of utilization was relatively small compared to the total availability of that resource.

Businesses rely on the availability of their Web servers (Al-Ghamdi et al., 2010; Hoffmann et al., 2006), and administrators use baselines and logging to understand the normal state of their systems. This research provides a foundation to begin to understand that administrators can monitor critical web server resources and create a variety of seasonal baselines, using moving averages to identify the common behavior of those resources over hours, days, and weeks. This information may then be used to show when a chosen resource moves outside that baseline and begins to behave abnormally.

A denial-of-service is currently recognized when the web service is no longer available (Schroeder & Harchol-Balter, 2006). This should be considered a failed state until service can be restored. With an understanding of long-term resource prediction, administrators have an additional tool to use in preventing the denial-of-service, and this tool will permit administrators to recognize when a resource is no longer behaving normally so they can begin to investigate the issue before the availability of the web server is compromised. Just as Krithikaivasan et al. (2007) used forecasting to help balance the network bandwidth to manage QoS and Abusina et al. (Abusina et al., 2005) used bands to determine the range within which the prediction was likely to fall, this research of web system resources can help to predict resource utilization, reducing the loss of service by understanding the difference between normal and abnormal utilization.

The simulator (P. R. Barford, 2001) used for the first part of the study was based on human interaction with web systems. Abusina et al. (2005) also recognized the influence of human factors in network traffic patterns. This study reinforces that conclusion in the patterns seen from the analysis of the live web server. The patterns can also be seen in the weekly chart of network transmissions from the web server (see Figure 29) as it responds to the user's requests rising during the day and falling at night.

Implications

Web server configurations are diverse and demands on that equipment have an even greater diversity (Andreolini et al., 2006). The variety of configurations and demands on each web server will require an administrator to determine if there are any additional resources, beyond those discussed in this research, that need to be monitored (Hoffmann et al., 2006). Once the set of resources is selected, the baseline can be created. Each server will require its own baseline of predictions to be effective in maintaining the availability of the machine throughout forecasting and to address the unique demands of that unit (Abusina et al., 2005). Those differences come from different operating systems, functionality, or installed hardware.

One purpose of this study was to identify the accuracy of SMA and EMA against Naïve forecasting and to identify that predictions can support web system availability. Once an analysis is completed, the administrator will have to select a moving average function to monitor and predict a reasonable next step for the resource based on the desired results. While those results fall within the accepted confidence interval, the administrator will have a reasonable assurance that the system is behaving normally and will continue to remain available to its users and services.

This research shows that long-term resource utilization has a degree of predictability. Through the use of SMA or EMA and confidence intervals, a range can be found to predict normal behavior in seasonal patterns that occur each hour, day, and week. Tests demonstrated that even when the error rate of the other prediction methods are higher than the Naïve prediction error rate, a defined range can be found that is substantially smaller than the available resource's range. A less accurate prediction with a reasonable confidence interval can be used as long as the range is appropriate for the resource and the intended prediction.

One implication of using long-term prediction to monitor resource utilization is the ability to detect when that resource is no longer acting normally, so that an administrator can evaluate whether that abnormal behavior will jeopardize the availability of the system. An administrator who finds that a change in resource utilization will lead to a loss of availability may then take action to preempt the event or minimize it.

Previous work in forecasting examined short-term prediction and indicated that long-term prediction was not reasonable since multiple steps were required to achieve that goal (Andreolini & Casolari, 2006; Istin et al., 2010). This research looked at the problem similarly to business and applied long-term research approaches to this problem. By using aggregate information that combined the data points for prediction instead of attempting to forecast multiple steps ahead or account for every fluctuation, the accuracy of the forecast was greatly increased (Papagiannaki, Taft, Zhang, & Diot, 2003), permitting long-term forecasts of hours, days, and weeks to be obtained. This provides a variety of opportunities for future research focusing on the overall behavior of the resource, instead of focusing on the resource utilization at one point in time.

A wide variety of other services experience patterns of usage due to human activity (Arlitt & Jin, 1999; Huang, 2008; Rood & Lewis, 2008), and this research also had an underlying driver of human activity, in continuing to support the hypothesis that human activity on computers is also predictable. Applications have also been shown to exhibit predictable run times (Chen et al., 2005; Marzolla & Mirandola, 2007; Smith, Foster, & Taylor, 2004) and resource usage patterns. These applications are executed based on user demand, so if human activity helps determine resource demand, then the applications will also have a degree of impact on utilization as well. A few researchers (Mitzenmacher, 2000; Rood & Lewis, 2010) have also looked at human and application interactions' effect on system availability of volunteers within a distributed system, suggesting that future research using long-term techniques may also apply to systems of resources as a group as well as individually.

Recommendations for Future Research

A variety of research possibilities exist from this point. This research identified that an hourly seasonal view of the web server provided forecasting results with the smallest error rate. The use of the confidence interval to define a range for the result, instead of expecting a precise value, provides a more appropriate expectation for understanding normal use of a resource (Abusina et al., 2005; Papagiannaki et al., 2003). While this study extended previous work in network utilization (Krithikaivasan et al., 2007) to other resources used by web services, future work should be able to extend this work to other servers and services including network servers, virtual servers, and distributed systems.

Previous research has recognized the use of various seasonality for prediction, since resource loads frequently exhibit different characteristics in different time frames (Andreolini & Casolari, 2006) and not all of them lead to accurate prediction of future utilization. This research extends the use of seasonality from predicting several minutes ahead to examining predictions for hourly, daily, and weekly seasonal cycles. Other seasonal time frames will also need to be investigated to determine the most appropriate seasonal view each server class researched. Different seasonal results have been used in short-term prediction for cloud computing (Dinda, 1999), network services (Krithikaivasan et al., 2007), virtual systems (Park & Humphrey, 2008), and application utilization (Chen et al., 2005; Smith et al., 2004). These research areas built foundations in short term prediction which may now be extended into long-term resource prediction.

Once a normal level of resource utilization is understood, research may begin to focus on the outliers within the prediction system (Papagiannaki et al., 2003). Currently the volatility and outliers are an issue to stable prediction (Abusina et al., 2005; Andreolini et al., 2008; Krithikaivasan et al., 2007; Rood & Lewis, 2010). Future research may use the volatility and outliers as a unique pattern of interest. Researchers could then explore the probability of those patterns and begin to provide valuable information about predicting the probability of the next event.

Prediction has also been used as a foundation to address the probability of a systems state and resource allocation. Krithikaivasan et al. (2007) used a probability-hop algorithm to determine when additional bandwidth was to be allocated, while Rood & Lewis (2008) used probabilities based on prediction to determine scheduling for a distributed system. Forecasting of resource utilization can lead to probability analysis. By determining the probability that a resource will remain within the forecast confidence interval, a level of normal may be understood, and the probabilities of abnormal may also be better defined. From there, a three position finite state machine as a guide for normal, abnormal, and failing systems. Research into a system's state (Andreolini et al., 2006) to improve distributed system availability and increase the time available for graceful degradation was conducted using single step prediction. With the knowledge of this study, the system state could now be reflected by normal resource utilization within the designated confidence interval.

Moving averages do not use feedback loops to update or adjust the prediction process to help correct estimation errors. Many short-term studies have used methods like ARMA, which incorporates a feedback loop to improve the prediction accuracy

(Sharifian et al., 2010). When changes are made to the system based on long-term forecasting, adjustments to future predictions will be needed or the administrator will be forced to wait through several seasons to regenerate the baseline for accurate forecasting (Papagiannaki et al., 2003). One example of such a use would be the combination of two virtual machines, after forecasting determines that most of the resources used heavily on one system are lightly used on the other. As soon as the two virtual machines are placed on the same hardware, the forecasting will change. Future research will need to find appropriate prediction methods to implement that feedback loop.

Summary

System resource prediction has been used by operating systems for the management of the program scheduler. Researchers (Dinda, 2002) focused on gathering data from system resources to predict the utilization level in the near future, so that the operating system could balance system resources to maintain system availability as much as possible. Research found that the use of data collected in fractions of a second did well at predicting the next step but were not able to easily predict the resource utilization several steps out (Istin et al., 2010). Results from those attempts found that the error rates increased with the number of steps and the predictions were less effective than using the last resource utilization level for the prediction.

Research has also focused on web services (Schroeder & Harchol-Balter, 2006), because businesses are much more dependent on those systems than they were ten years ago. Many businesses now use web services as a primary method of customer contact and sales, so the availability of those systems is considered to be critical. Researchers took short-term prediction results and extended them for scheduling requests on web services.

But this research also found that if the scheduling was extended several steps into the future, the error rate was no longer reasonable.

Research has used prediction to help manage QoS by rebalancing network bandwidth for traffic every fifteen minutes. Krithikaivasan et al.'s (2007) work was able to look at aggregates of five minutes in length and make adjustments every fifteen minutes. By extending the work to fifteen minutes, Krithikaivasan et al. (2007) showed that an aggregate coarser than fractional seconds is possible. Network utilization levels were aggregated into ninety-minute cycles to reduce the fluctuation seen at smaller time frames (Papagiannaki et al., 2003) to provide a range for the prediction instead of a single prediction value, since a value showing normal utilization is more important than a precise prediction at a single point in time (Abusina et al., 2005).

This research used Naïve, SMA, and EMA forecasting to examine the effectiveness of providing long-term forecasting. By generating aggregates with a coarser granularity, the predictability of a resource's utilization range was examined based on the seasonality of hours, days, and weeks. CPU utilization, free memory, network bytes sent, network bytes received, disk write time, and disk read time were selected as the resources that were evaluated. Disk read time, however, needed to be dropped from the analysis due to the large number of zero values in the data points and aggregate values.

By comparing the average MAPE values for each of the predictions, the research could determine the most accurate method of prediction between Naïve, SMA, or EMA. Most of the resources within each season had the smallest error rate using the SMA or EMA methods of forecasting. These results existed for both the simulation and the live data collected for the study.

For one set of tests, the daily seasonal time frame for the live data, every resource saw improvement in the prediction error rate for both the SMA and EMA forecast methods. For CPU, network bytes sent, and disk write time, a substantial improvement was seen by the SMA over the Naïve prediction method. SMA for those three resources improved the prediction error rate by more than 20%.

Previous research established the use of aggregation and utilization ranges (Abusina et al., 2005; Papagiannaki et al., 2003) for network traffic. The results of this research extended the use of aggregates and range evaluation to general computing resources as each resource utilization level was logged; then the data was aggregated into a timeframe appropriate for the seasonality of the prediction required. Once the aggregates were available, the information was processed using SMA or EMA to generate the prediction and confidence interval appropriate to the administrator's needs. The real data aggregate was then compared to the prediction and confidence interval to determine if the resource was still acting normally, and the next forecast was then generated for the next cycle.

References

Abusina, Z. U. M., Zabir, S. M. S., Ashir, A., Chakraborty, D., Suganuma, T., & Shiratori, N. (2005). An engineering approach to dynamic prediction of network performance from application logs. *Int. J. Netw. Manag., 15*(3), 151-162. Retrieved from doi:10.1002/nem.554

Al-Ghamdi, M., Chester, A. P., & Jarvis, S. A. (2010). Predictive and Dynamic Resource Allocation for Enterprise Applications. *10th IEEE International Conference on Computer and Information Technology (CIT 2010), 0*, 2776-2783. Retrieved from http://doi.ieeecomputersociety.org/10.1109/CIT.2010.463 doi:10.1109/CIT.2010.463

Andreolini, M., & Casolari, S. (2006). Load prediction models in web-based systems. *Proceedings of the 1st international conference on Performance evaluation methodolgies and tools*, 27. Retrieved from doi:10.1145/1190095.1190129

Andreolini, M., Casolari, S., & Colajanni, M. (2006). A Distributed Architecture for Gracefully Degradable Web-Based Services. *Fifth IEEE International Symposium on Network Computing and Applications (NCA'06), 0*, 235-238. Retrieved from http://doi.ieeecomputersociety.org/10.1109/NCA.2006.1

Andreolini, M., Casolari, S., & Colajanni, M. (2008). Models and framework for supporting runtime decisions in Web-based systems. *ACM Trans. Web, 2*(3), 1-43. Retrieved from doi:10.1145/1377488.1377491

Arlitt, M., & Jin, H. (1999). Workload Characterization of the 1998 World Cup Web Site *HPL-1999-35 (R.1)* (pp. 90): HP Laboratories Palo Alto.

Balsamo, S., Marzolla, M., & Mirandola, R. (2006). Efficient Performance Models in Component-Based Software Engineering. *Proceedings of the 32nd EUROMICRO Conference on Software Engineering and Advanced Applications*, 64-71. Retrieved from doi:10.1109/euromicro.2006.34

Barford, P., & Crovella, M. (1998). Generating representative Web workloads for network and server performance evaluation. *SIGMETRICS Perform. Eval. Rev., 26*(1), 151-160. Retrieved from doi:10.1145/277858.277897

Barford, P. R. (2001). *Modeling, Measurement and Performance of World Wide Web Transactions.* (Doctor of Philosophy Ph.D.), Boston University, Boston. Retrieved from http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.22.6112&rank=1 Available from Penn State University SiteSeer database.

Chandola, V., Banerjee, A., & Kumar, V. (2009). Anomaly detection: A survey. *ACM Comput. Surv., 41*(3), 1-58. Retrieved from doi:10.1145/1541880.1541882

Chapman, C., Musolesi, M., Emmerich, W., & Mascolo, C. (2007). Predictive Resource Scheduling in Computational Grids. *0*, 116-116. Retrieved from http://doi.ieeecomputersociety.org/10.1109/IPDPS.2007.370306 doi:10.1109/IPDPS.2007.370306

Chen, S., Liu, Y., Gorton, I., & Liu, A. (2005). Performance prediction of component-based applications. *J. Syst. Softw., 74*(1), 35-43. Retrieved from doi:10.1016/j.jss.2003.05.005

Dinda, P. A. (1999). The statistical properties of host load (Extended Version). *Sci. Program., 7*(3-4), 211-229. Retrieved from

Dinda, P. A. (2002). A Prediction-Based Real-Time Scheduling Advisor. *International Parallel and Distributed Processing Symposium (IPDPS'02), 1*, 0010b-0010b. Retrieved from http://doi.ieeecomputersociety.org/10.1109/IPDPS.2002.1015480 doi:10.1109/IPDPS.2002.1015480

Dinda, P. A. (2006). Design, Implementation, and Performance of an Extensible Toolkit for Resource Prediction in Distributed Systems. *IEEE Transactions on Parallel and Distributed Systems, 17*, 160-173. Retrieved from doi:10.1109/TPDS.2006.24 doi:doi:10.1109/TPDS.2006.24

Ghemawat, S., Gobioff, H., & Leung, S.-T. (2003). The Google file system. *Proceedings of the nineteenth ACM symposium on Operating systems principles*, 29-43. Retrieved from doi:http://doi.acm.org/10.1145/945445.945450

Harchol-Balter, M., Schroeder, B., Bansal, N., & Agrawal, M. (2003). Size-based scheduling to improve web performance. *ACM Trans. Comput. Syst., 21*(2), 207-233. Retrieved from http://doi.acm.org/10.1145/762483.762486 doi:10.1145/762483.762486

Hoffmann, G. A., Trivedi, K. S., & Malek, M. (2006). A Best Practice Guide to Resources Forecasting for the Apache Webserver. *12th Pacific Rim International Symposium on Dependable Computing (PRDC'O6), 0*, 183-193. Retrieved from http://doi.ieeecomputersociety.org/10.1109/PRDC.2006.5 doi:10.1109/PRDC.2006.5

Huang, J. (2008). Short-Term Traffic Flow Forecasting Based on Wavelet Network Model Combined with PSO. *International Conference on Intelligent Computation Technology and Automation, 1*, 249-253. Retrieved from http://doi.ieeecomputersociety.org/10.1109/ICICTA.2008.74

Islam, S., Keung, J., Lee, K., & Liu, A. (2012). Empirical prediction models for adaptive resource provisioning in the cloud. *Future Generation Computer Systems, 28*(1), 155-162. Retrieved from http://www.sciencedirect.com/science/article/pii/S0167739X11001129 doi:10.1016/j.future.2011.05.027

Istin, M., Visan, A., Pop, F., & Cristea, V. (2010). Decomposition Based Algorithm for State Prediction in Large Scale Distributed Systems. *Ninth International Symposium on Parallel and Distributed Computing, 0*, 17-24. Retrieved from http://doi.ieeecomputersociety.org/10.1109/ISPDC.2010.13 doi:10.1109/ISPDC.2010.13

Krithikaivasan, B., Zeng, Y., Deka, K., & Medhi, D. (2007). ARCH-based traffic forecasting and dynamic bandwidth provisioning for periodically measured nonstationary traffic. *IEEE/ACM Trans. Netw., 15*(3), 683-696. Retrieved from doi:10.1109/tnet.2007.893217

Lampson, B. W. (1983). Hints for computer system design. *Proceedings of the ninth ACM symposium on Operating systems principles*, 33-48. Retrieved from doi:10.1145/800217.806614

Lu, C., Wang, L., & Koutsoukos, X. (2005). Feedback Utilization Control in Distributed Real-Time Systems with End-to-End Tasks. *IEEE Transactions on Parallel and Distributed Systems, 16*, 550-561. Retrieved from http://doi.ieeecomputersociety.org/10.1109/TPDS.2005.73

Makridakis, S., Wheelwright, S. C., & Hyndman, R. J. (1997). *Forecasting: Methods and Applications* (3rd ed.). New York, NY: John Wiley & Sons.

Marzolla, M., & Mirandola, R. (2007). Performance prediction of web service workflows. *Proceedings of the Quality of software architectures 3rd international conference on Software architectures, components, and applications*, 127-144. Retrieved from

Mitzenmacher, M. (2000). How Useful Is Old Information? *IEEE Trans. Parallel Distrib. Syst., 11*(1), 6-20. Retrieved from doi:10.1109/71.824633

Papagiannaki, K., Taft, N., Zhang, Z.-L., & Diot, C. (2003). Long-term forecasting of Internet backbone traffic: observations and initial models. *INFOCOM 2003. Twenty-Second Annual Joint Conference of the IEEE Computer and Communications. IEEE Societies, 2*, 1178-1188 vol.1172. Retrieved from doi:10.1109/infcom.2003.1208954

Park, S.-M., & Humphrey, M. (2008). Feedback-controlled resource sharing for predictable eScience. *2008 ACM/IEEE conference on Supercomputing, 0*, 1-11. Retrieved from http://doi.ieeecomputersociety.org/10.1145/1413370.1413384

Qiao, Y., Skicewicz, J., & Dinda, P. (2004). An Empirical Study of the Multiscale Predictability of Network Traffic. *0*, 66-76. Retrieved from http://doi.ieeecomputersociety.org/10.1109/HPDC.2004.3

Ramachandran, K., Lutfiyya, H., & Perry, M. (2010). Decentralized Resource Availability Prediction for a Desktop Grid. *2010 10th IEEE/ACM International Conference on Cluster, Cloud and Grid Computing, 0*, 643-648. Retrieved from http://doi.ieeecomputersociety.org/10.1109/CCGRID.2010.54 doi:10.1109/CCGRID.2010.54

Rood, B., & Lewis, M. J. (2008). Resource Availability Prediction for Improved Grid Scheduling. *Fourth IEEE International Conference on eScience, 0*, 711-718. Retrieved from http://doi.ieeecomputersociety.org/10.1109/eScience.2008.66

Rood, B., & Lewis, M. J. (2010). Availability Prediction Based Replication Strategies for Grid Environments. *10th IEEE/ACM International Conference on Cluster, Cloud and Grid Computing, 0*, 25-33. Retrieved from http://doi.ieeecomputersociety.org/10.1109/CCGRID.2010.121 doi:10.1109/CCGRID.2010.121

Schroeder, B., & Harchol-Balter, M. (2006). Web servers under overload: How scheduling can help. *ACM Trans. Internet Technol., 6*(1), 20-52. Retrieved from doi:http://doi.acm.org/10.1145/1125274.1125276

Sharifian, S., Motamedi, S. A., & Akbari, M. K. (2010). An approximation-based load-balancing algorithm with admission control for cluster web servers with dynamic workloads. *J. Supercomput., 53*(3), 440-463. Retrieved from doi:10.1007/s11227-009-0303-8

Silberschatz, A., Galvin, P. B., & Gagne, G. (2003). *Operating Systems Concepts, sixth ed.*: John Wiley & Sons, Inc;.

Smith, W., Foster, I., & Taylor, V. (2004). Predicting application run times with historical information. *Journal of Parallel and Distributed Computing, 64*(9), 1007-1016. Retrieved from http://www.sciencedirect.com/science/article/B6WKJ-4D8F6S2-2/2/a10f088331e9937ff7f09f92b91454a3 doi:DOI: 10.1016/j.jpdc.2004.06.008

SPECweb2009 (Retired 01/12/2012). (2011). Retrieved October 27,2012, from http://www.spec.org/web2009/

TPC-W transactional Web e-commerce benchmark (Retired 04/28/2005). (2004). Retrieved October 27, 2012, from http://www.tpc.org/tpcw/

Traeger, A., Zadok, E., Joukov, N., & Wright, C. P. (2008). A nine year study of file system and storage benchmarking. *Trans. Storage, 4*(2), 1-56. Retrieved from doi:10.1145/1367829.1367831

Wierman, A., & Harchol-Balter, M. (2003). Classifying scheduling policies with respect to unfairness in an M/GI/1. *Proceedings of the 2003 ACM SIGMETRICS international conference on Measurement and modeling of computer systems*, 238-249. Retrieved from doi:http://doi.acm.org/10.1145/781027.781057

Wolski, R. (2003). Experiences with predicting resource performance on-line in computational grid settings. *SIGMETRICS Perform. Eval. Rev., 30*(4), 41-49. Retrieved from doi:10.1145/773056.773064

Wolski, R., Spring, N. T., & Hayes, J. (1999). The network weather service: a distributed resource performance forecasting service for metacomputing. *Future Gener. Comput. Syst., 15*(5-6), 757-768. Retrieved from doi:10.1016/s0167-739x(99)00025-4