Year 2005 *Article* 93

A POWER AND PERFORMANCE MEASUREMENT FRAMEWORK FOR SERVER- CLASS STORAGE

Mathew Oldham
Florida State University (Arts & Sciences - Computer Science)

A POWER AND PERFORMANCE MEASUREMENT FRAMEWORK FOR SERVER- CLASS STORAGE

Abstract

Recently researchers have studied various approaches on how to reduce energy consumption of server-class disk storage devices. Although many studies have claimed success, details of their experimental setups are either little documented or not designed to measure power consumption or performance of their systems simultaneously. This thesis describes a new framework that produces accurate results using realistic workloads by measuring both the power and performance of the disks at the same time while maintaining the privacy of the users. We have used this framework to test the power and performance characteristics of three different RAID personalities.

THE FLORIDA STATE UNIVERSITY
COLLEGE OF ARTS AND SCIENCES

A POWER AND PERFORMANCE MEASUREMENT FRAMEWORK

FOR SERVER- CLASS STORAGE

By

MATHEW OLDHAM

A Thesis submitted to the
Department of Computer Science
in partial fulfillment of the requirements for Honors in the Major

Degree Awarded:
Spring Semester, 2005

The members of the Committee approve the thesis of Mathew Oldham defended on 5 Apr 2005.

Dr. An-I Wang
Professor Directing Thesis

Dr. Ted Baker
Committee Member

Dr. Nancy Greenbaum
Outside Committee Member

Table of Contents

List of Figures

List of Tables

Acknowledgements

I would like to thank Carl Owenby for his engineering skills in creating the cables used to monitor the power consumption of the drives. I would also like to thank Dr. Geoffrey Kuenning for his insight on measuring the power consumption of the disks and associated problems that were occurring, along with providing sanity checks throughout the framework creation process. I'd like to thank Charles Weddle and Sean Toh, part of the OS Research Team. Finally, I would like to thank Nicolas Wallen for providing the external power supply and the members of my committee, Dr. Nancy Greenbaum and Dr. Ted Baker. This work was funded by NSF grant number CNS-0410896.

ABSTRACT OF THE THESIS

A Power and Performance Measurement Framework for Server-Class Storage

by

Mathew Oldham

Florida State University, Spring 2005

Dr. An-I Wang

Dr. Ted Baker

Dr. Nancy Greenbaum

Recently researchers have studied various approaches on how to reduce energy consumption of server-class disk storage devices. Although many studies have claimed success, details of their experimental setups are either little documented or not designed to measure power consumption or performance of their systems simultaneously. This thesis describes a new framework that produces accurate results using realistic workloads by measuring both the power and performance of the disks at the same time while maintaining the privacy of the users. We have used this framework to test the power and performance characteristics of three different RAID personalities.

Keywords: RAID, Hard Drive, Power

1.0 Introduction

The average cost of computers has declined over the years. Average computers that used to cost ~$2,000 now cost only half as much due to cheaper manufacturing techniques and the economy of scale. Affordable computers, along with wide-spread network access, have brought computing to the masses. This in turn has lead to the explosion of the Internet.

As the Internet has grown, so too have the number ofservers needed to store the massive amounts of information. These servers must have the ability to respond very quickly to the requests of the clients accessing them. Serving thousands of requestsevery second requires multiple fast processors, large amounts of memory, fast hard drives that can store the necessary amount of information, and one or more network connections capable of delivering the information back to the user.

As the performance of these servers have increased, so too has the amount of power consumed. For example, in March 2003, the Google search engine contained over 15,000 servers located at 4 different installations [2]. With such a dense deployment of resources in a confined space, massive amounts of energy are required for not only the servers, but also for cooling on top of the regular operating costs of a building. Over the lifetime of a server, the energy cost outweighs the operational cost of the server. One report estimates that the energy costs of a typical server center can reach up to 60% of the operational costs and can use between 10 and 20 megawatts of power per hour. The power needed to run one server center can power 10,000 to 20,000 homes with every light turned on [1].

Recently researchers have studied various approaches on how to save energy consumption of server-class disk storage devices. Hard drives alone account for up to 24% of the total power consumed by a web server and up to 77% of power consumed by a proxy server [2]. Although many studies have claimed success, their experimental setups are either little documented or not designed to measure power consumption and performance of their systems simultaneously. It is very hard to measure such benefits as current servers are built for maximum performance, not for power measurement.

In this thesis I present a new framework for measuring the performance and power consumption of server-class storage systems. The framework is easy to setup and run, and provides repeatable workloads based on real traffic patterns. The workloads themselves are realistic, repeatable, and privacy-aware and match the hardware and software environments that are used within the framework.

2.0 Background

This section introduces the terminology associated with hard drives, web servers, and the way hard drives are used in web servers. Those who are familiar with such topics may skip to Section 3.

2.1 Hard Drives

The performance and power consumed relate directly to the fact a hard drive is a mechanical device with moving parts. Figure 1 describes the internal mechanical structure of a typical hard disk. A typical hard drive consists of several cylindrical disks called platters that rotate around a spindle. Each platter contains magnetically encoded data. To read and write data onto the platters, a disk arm moves back and forth over the platters. Attached to the end of the disk arm is a read/write head that translates the data sent from the operating system to the drive.

When a user either reads or writes a file to disk, the operating system sends a disk request to the hard drive containing the information. The disk platters must be spinning at a specific speed to read or write the data. Should the platters be spinning below this speed, additional power is required to ‘spin up’ the disk. After the platters are spinning at the appropriate speed, the disk arm moves over the platters into the proper position and performs the read/write operation. After the operation is complete, the disk arm moves away from the platter and the disk returns to an idle state.

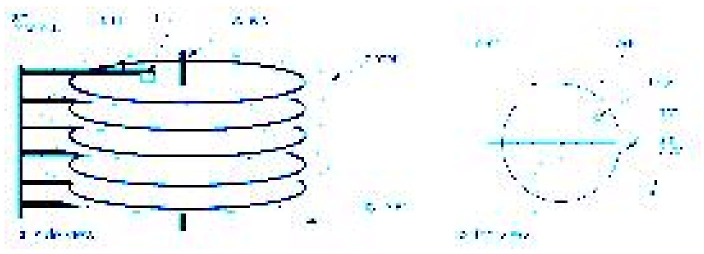


Figure 1: Internal mechanical structure of a hard disk.

2.2 Web Servers

Since different application servers have different workload characteristics, we limited the focus of our studyto web servers. Web servers allow users (clients) to access information from anywhere in the world. A typical web server consists of one or more processors, one or more fast network connections, large amounts of memory, and several hard drives to store the web content. The server itself runs a web server program, which receives requests from the client, finds the requested file, and returns the file to the user over a network.

Web servers can serve both static and dynamic content. Static content includes files such as images, web pages, and text files that are served directly to the user. Dynamic content is generated on demand. Dynamic content includes such things as stock tickers, news tickers, and other content that is generated by the server and then served on demand to the client.

Web servers consume a considerable amount of power. This is mainly due to the need for continual service to all users whether the web server is used or not. A web server contains several hard drives, which can increase the performance and throughput of the server through parallelism, but also increase power consumption. The drives are connected directly to a logic board, called a backplane, which provides power and distributes the disk requests to the appropriate drive. The hard drives work together in a redundant array of inexpensive disks (RAID) to provide performance and reliability through data redundancy.

2.3 Redundant Arrays of Inexpensive Disks (RAID)

A RAID device distributes data across several disks, which provides greater performance by allowing more than one drive to be accessed at a time. This *data striping* can either be uniform meaning that each disk contains the same amount of information, or skewed meaning that some disks contain more information than other disks. Data striping provides a greater throughput for large files where seek time is not the dominant time factor. For small files, where seek time and rotational delay dominate, RAID is only as fast as the slowest drive.

RAID provides greater reliability by spreading redundant information across several disks. This allows for one or more disks to fail without crashing the system or losing data. When the failed disk is replaced with a new one, the remaining disks can rebuild the RAID device and regenerate the data previously stored on the failed drive. Different RAID personalities provide tradeoffs between performance and reliability. For example, RAID-0 contains no redundancy but can provide excellent bandwidth but will not be able to recover from a single disk failure. RAID-5 provides redundancy striped across all disks which allows for a disk to be recovered should a disk failure occur.

3.0 Performance Evaluation Techniques

There are three types of performance evaluation techniques including analytical modeling, simulations, and empirical measurements. Analytical modeling requires the construction of a model. The model can be constructed quickly and can provide predictors of anticipated results, but such results are often based on many simplifying assumptions that reflect reality in limited ways.

Simulations are a good alternative, if the hardware is not available to conduct actual measurements. Simulations can be easily configured to explore a large parameter space. However, simulations need to be validated, and the validation process can be tedious. Simulations may take a long time to run if the physical effects being simulated are at the time scale of microseconds. For example, if a disk is powered in pulses. It is also hard to simulate the amount of energy required to spin up a disk, which can throw off results.

The most realistic form of performance evaluation in terms of validity is empirical measurement. This type of performance evaluation can be labor intensive and costly to obtain the needed equipment to implement and measure the test system. Many times new methods must be invented to take the proper measurements, which involve in-depth knowledge of the test system and what is happening within the system. Other times it may be difficult to perform the measurements due to lack of compatibility between software measurement devices and the system.

3.1 Workload Types

There are two major types of workloads. Workloads can be either synthetic or realistic. Synthetic workloads are mostly designed to measure the individual components of a system at peak performance. They often provide their own file set based off previous statistics. However, such workloads do not reflect real world situations such as human interaction.

Realistic workloads are based off workloads from a real system. Realistic workloads are more accurate in representing user behavior and intentions. However, one workload may not be representative of all workloads. For web servers, there is no such thing as an average or ideal workload. For every web server, there is a difference in the types of web sites and content of the web sites. Each trace may also be biased based on the software and hardware implemented running on the server.

3.2 Popular Web-Server Benchmarks

Several popular benchmarks exist and are used to measure the performance of web servers. Chart 1 contains an overview of various popular benchmarks along with what our benchmark aims to achieve. All the benchmarks are client-server benchmarks where a client makes a request to the server and the server responds. Some benchmarks have the ability to run traces based on web server access log files while others are synthetic and have a predetermined file set. The synthetic benchmarks tend to measure the peak performance of the web server while the more realistic benchmarks that use web access logs to drive the benchmark provide more of an average performance from a realistic workload. However, none of the previous benchmarking frameworks measure the power consumption of the hard drives within the server [6, 8, 9, 10].

	Real-Time	Daily/Weekly Cycles	Trace-Based	Concurrent Requests	Human Interaction	Power	Peak Performance	Average Performance	Matching HW Characteristics	Dynamic Content
WebStone			X	X			X			
SPECWeb 99				X			X			X
HTTPerf			X	X			X			
Self Scaling Web Benchmark			X	X				X	X	X
Trace Replay	X	X	X	X	X		X	X		X
WebClient (Ours)	X	X	X	X	X	X	X	X	X	?

Table 1

4.0 Design

The design of our new measurement framework takes into account several aspects of the entire system from the hardware the system runs to the software running on the hardware. We also take into account the workload used to drive the benchmark, how the file system is created to maintain privacy, and how the benchmark itself is designed to generate loads efficiently and report the appropriate information from power measurement to performance metrics.

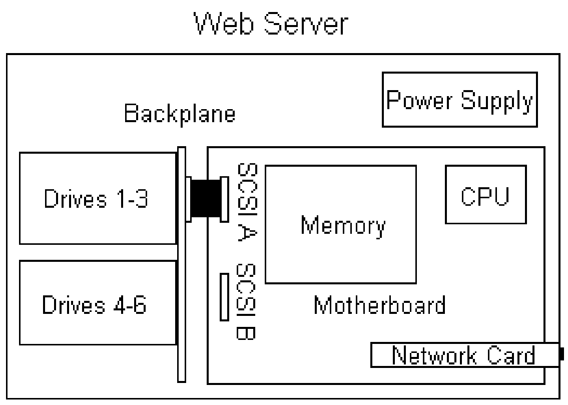


Figure 2: Internals of a Web Server

4.1 Hardware Design

Commercial servers are not designed to be research friendly for measuring the power consumption of the hard drives. Figure 2 shows the inside of a typical web server. Each hard drive is connected to a logic board called a backplane. The connection between the backplane powers the disks along with providing data transfer capabilities since server-class disks do not have a separate power connection. There is no way to measure the power of the individual drives while they are connected to the backplane. Measuring the power of the entire backplane yields the total power consumed by all drives, but also showsthe power consumed by the backplane itself , which can skew results.

The solution implemented in our design is to bypass the backplane as shown in Figure 3. The drives themselves are removed from the backplane. A separate cable having the same interface as the backplane connects the disks to the server. The cable interface is also the same as the drive interface. Since the drives have a different number of pins in their connection, an adapter is provided to connect the drives to the cable. The adapter contains a standard power connector that allows us to connect an external power supply to power the drives. This provides us with the ability to measure the power consumption of the drives since the adapter itself takes no power. We also maintain the same performance with the drives connected to the cable as with the drives connected to the backplane since the same interface is used between each.

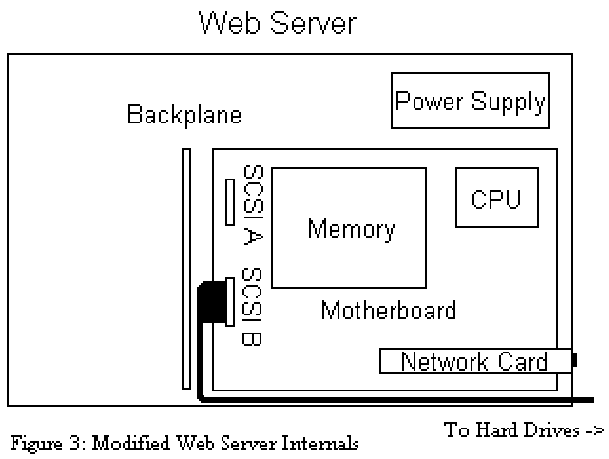


Figure 3: Modified Web Server Internals

4.2 Workload Considerations

The next decision was to find an appropriate workload that matched our hardware and software. The workload chosen came from an academic web server. Academic workloads are not the same as government or commercial workloads. Commercial servers may contain more dynamic content while academic and government servers may contain more static content. Ideally, our framework can adapt to different workloads.

Our chosen workload also has several nice properties for studying power savings. Web accesses exhibit a periodic nature where more requests are served during the midday hours than at night, and more requests during the middle of the week than on weekends [5]. This property is essential for many power-saving techniques to work well. This periodicity of requests also allows us to identify whether a web server isseverely over-provisioned or under provisioned.

Finally, the workload must be timely enough to match the hardware settings. A web trace that is too old may not fully exercise modern hardware. A modern trace on older hardware can also overwhelm the older hardware. Great care must also be taken when using modern workloads with modern hardware as the workload may still not exercise the hardware to its full potential.

4.3 File System Creation

Most studies make no mention of recreating the original file system used for their workload. They often reconstruct a file system based only on the files referenced during the logged period. Since many of the researchers used an older web trace to provide the workload for their testing, the possibility of them having access to the original file system remains extremely small. Without the original file system, errors could skew results in terms of both power and performance.

Taking into account only those web files that are publicly accessible still would not capture the entire file system. Additional overhead occurs with larger file systems in terms of file lookup, updating metadata, etc. Some files may be password protected. To gather the entire file system, we must contact every user of the web server and get prior approval. This could involve hundreds if not thousands of people and man hours.

Most researchers do not have access to the actual file system the trace comes from due to many factors (e.g. privacy). Sensitive data may be contained within user files (e.g. final exam questions). Even file names could encode information as many users name a file after the contents within the file.

Our solution was to disregard the actual information contained within the file and encrypt the names of the files. A script runs through the entire file system and gathers the name, size, and type of file along with the time stamps of the files creation, modification, and last access. We then encrypted the names of the files using the SHA-1 algorithm [7].

Even if a webpage contains a picture of other resources, they are still served as separate requests within the log file so the file is stil served appropriately.

4.4 WebClient

The WebClient program is the heart of the benchmark. Web server programs keep accurate logs of who has requested what resource from the server. The WebClient program replays these web server log files by splitting theminto two separate files. The first file is the master log file. This file contains the time followed by the number of requests issued at that time. The second file contains a sequential list of requests from the web server. For example if at time three, ten requests were processed by the original web server which is the information read from the master log file, the next ten requests from the request log file would be issued by the web client.

The WebClient program uses the master log to determine the number of requests to read from the request log file at a specific time. By the end of the trace, all requests from the request log file had been issued in the appropriate order and at the appropriate relative time intervals. To access our file system with encrypted file names, the file names appearing in thelog file were also encrypted using the SHA-1 algorithm [7].

The WebClient must efficiently represent many users accessing the web server. Each WebClient request is handled as a separate connection to the web server, with an associated thread, or single execution stream. A thread pool, or collection of threads, is maintained and the individual threads within the pool are kept alive for reuse. This reduces the overhead of creating new threads.

Another purpose of the WebClient is to report detailed performance numbers of the web server. Each client thread reports statistics about the individual request. These metrics can be found in Section 4.6. This is implemented through various performance monitors. The performance monitoring contained within the WebClient must not impact the retrieval of requests which could skew the results of the test.

Finally, the WebClient offers some additional options to be used when running tests. The first option is to limit the number of connections that can be opened at any time between the client and server. The server already has a maximum number of connections that it can handle at any given time and there is no need to overload the server, since users would not use servers with such high latencies. Also, we are testing the power consumption and performance of hard drives, not the number of connections the server can handle at one time.

4.5 Power Setup

Measuring the power consumption of any electrical device is based off Ohms Laws shown in Figure 5. These two laws state that if you can monitor the voltage drops across any device within a circuit, and if you have access to a resistor, then you can determine the amount of current that is used by the device and the power consumed. To do this one must create a closed circuit and insert a resistor in series at the beginning of the circuit as shown in Figure 4. A multimeter measures the voltage drop across the resistor by measuring the voltage at the beginning of the resistor and the voltage at the end of the resistor and subtracting the two: allowing you to calculate the amount of current used by the resistor, which is the same amount of current used by the device. By measuring the voltage drop across the device, one can then use Ohms Laws and the current previously calculated from the voltage drop of the resistor to determine the amount of power consumed at that time.

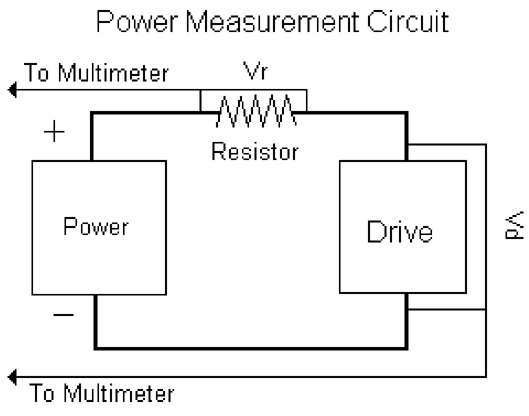


Figure 4: Power Circuit

$$I\ (Current) = V_r\ (Voltage\ Drop\ of\ \mathrm{Re}\,sistor) / R\,(\mathrm{Re}\,sistor)$$
$$P\,(Power) = V_d\ (Voltage\ Drop\ of\ Drive) * I\ (Current)$$
Figure 5: Ohms Laws

For measuring the power consumption of a hard drive, an independent power supply is provided that connects to an adapter for the server-class disk. Since the adapter consumes no power itself, all power from the power supply is directly translated to the disk. We then inserted a resistor in series between the power supply and the adapter by cutting the power cable of both the 12V and 5V lines. By using a multimeter with several channels, which allows the multimeter to take more than one reading at a time, we can measure the voltage drop across the resistor to calculate the current and the voltage drop across the drive. From Ohms laws we then calculate the amount of power consumed.

Several considerations must be taken into account when taking power measurements for a disk drive. Disks that have requests sent to them constantly are busier than those disks that are have requests sent to them on occasions. Because of this, power measurement replies on the timing of request intervals. Therefore, overloading the system to test the peak performance will not demonstrate any power benefits. Similarly, a system that is serving a very low volumeof requests or remains idle can turn off all disks to save power, which is not representative of a typical server. Thus, when measuring the power consumption of a server, the trace must show load fluctuations that are representative of the usage patterns of a server. The drives of a server that has higher activity in the afternoon should also exhibit higher power consumption while the drives during lower activity exhibit lower power consumption.

The accuracy of power measurements must also be taken into account. Multimeters with higher sampling rates can take more readings over a given time period and produce more accurate results than multimeters with lower sampling rates. However, such multimeters may not be affordable for research purposes. Additionally, these multimeters must have the ability to send the results to a separate data logging device, minimizing the interference on the performance or sampling rate of the multimeter.

The multimeter must also have the ability to monitor more than one channel at once. Each drive contains a 5V line for circuitry and a 12V line for the spindle, arm, and head movement that must be monitored. However, monitoring more channels requires greater overhead, so the multimeter must maintain a sufficient sampling rate for each channel when taking the power measurements. The framework must also be able to measure the system wide power consumption.

4.6 Metrics

A benchmark produces information that can be used to compare design alternatives. This process produces several metrics for the analysis of the power consumption of the drives and performance of the server. The framework allows us to monitor each drive individually, which produces a per drive power consumption. Combining the individual drives power consumption produces the aggregate power consumption of the drives. Finally, combining the aggregate power consumption of the drives with the power consumption of the server allows us to calculate the total power consumption of the server at any given time.

The performance information obtained through the WebClient program are called end-to-end client-server metrics. End-to-end metrics are more reflective of user experience compared to server-only metrics. The performance metrics encompass both the entire server performance and per request performance. We measure the elapsed time to receive the first response from the server (latency) and the elapsed time between sending the first byte of the request to receiving the last byte of the response for every request. We also record the total number of connections completed per second, the total number of open connections between the client and server at a given time (concurrent connections), and the number of bytes sent by the server during a given time period (throughput). The performance goal for servers is to maintain a high number of connections per second and throughput while maintaining a low average latency and file completion time.

5.0 Implementation

To test the framework we used the Florida State Computer Science Web Server, websrv2, as the basis for our implementation. We tried to match the original web server as much as possible taking into account differences that must be handled to maintain privacy and the software used for the development of the entire project. We obtained the file system snapshotalong with the web log trace from websrv2 also.

5.1 Hardware Setup

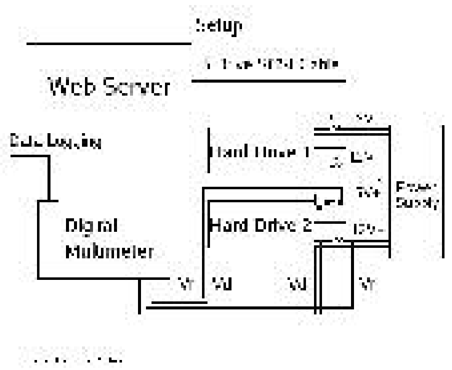
We obtained two computers from Dell that have thespecifications listed in table 2. The web server is a Dell PowerEdge 2600 and the client is a Dell PowerEdge 700.

We connected the client directly to the web server via a CAT-6 crossover cable. This allowed us to minimize any noise that might have occurred when both computers are connected to the Computer Science network. The CAT-6 crossover cable also allowed us to take full advantage of the gigabit Ethernet cards in both the client and server reducing the chance of a network bottleneck.

	Web Server	Client
Processor	Intel Xeon 2.8 Ghz	Intel Pentium 4 2.8 Ghz
Memory	2 Gigabytes	1 Gigabytes
Network	Gigabit Ethernet	Gigabit Ethernet
Drives	36.7 GB 15k RPM SCSI Ultra 320	160 GB 7200 RPM SATA

Table 2

We obtained a digital multimeter that is capable of sampling at a reasonable rate and is able to log the data to a compute for processing. The multimeter is connected directly to the client computer. The multimeter takes several samples per channel per second but only logs the average to the computer approximately once every second. The number of samples that can be logged in a given time period is limited by the interface between the multimeter and the client. The total number of samples that can be logged is limited by the software used for gathering the data.



Each hard drive was removed from the web server and connected to an adapter that provides an interface that connects the 80 pin SCSI Ultra 320 hard drives to the 68 pin Ultra 320 SCSI cable as shown in Figure 6. The SCSI cable is connected directly to the motherboard which allows the SCSI cable to maintain the same performance as if the drives were connected directly to the backplane

To measure the power of the hard drives we inserted a low 0.1 Ohmresistor in series with between the power supply and the adapter for each 12V+ and 5V+ line of the hard drives we were measuring. We attached one set of probes to each side of the resistor and another set of probes to both the positive and negative lines to the drive. This allowed the multimeter to measure the voltage drop across the resistor and voltage drop of the drive. From the voltage drop of the resistor we calculated the current of the drive. Although we have the ability to constantly measure the voltage drop across the drive, the voltage drop across the drive remains constant from test to test because of Ohms laws.

5.2 Software

The Computer Science web server uses a combination of the Linux Operating System along with the Apache Web Server to serve requests to users. We have decided to use Linux Kernel version 2.6.5 as the development environment since both websrv2 and the development of our Power Aware RAID use a version of the 2.6.X family. We are also using Apache version 2.0.52 as this is the same version as websrv2. The Apache

Web Server uses the same configuration file as websrv2 with slight modifications to reflect the SHA-1 file system. Since most web users use Microsoft Windows, the client is using Microsoft Windows XP with service pack 2. The entire WebClient program is designed using Java version 1.5.

We used software RAID for all RAID personalities tested as there is no hardware to implement all of them. Software RAID is implemented through the modified Linux raid-tools. We ran the Linux ext2 file system on the RAID device as this provided good file system performance for a web server.

5.3 File System

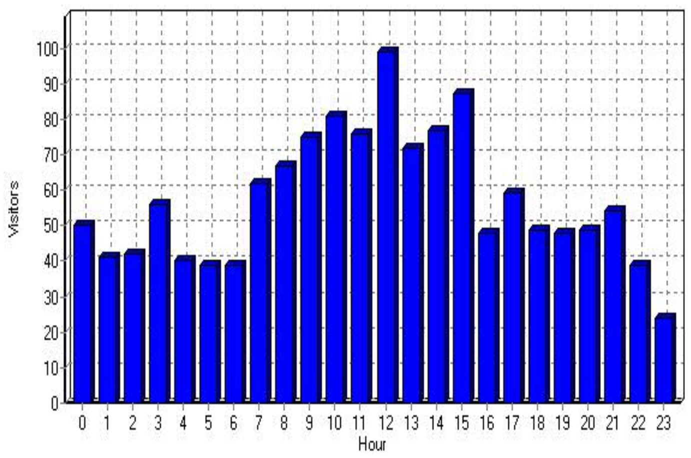
The file system was obtained from websrv2 in late November. The file system contains approximately 47GB of data with close to 38GB of total user data that can be served by the web server. The file system contains approximately 44,000 directories, 509,000 regular files, and approximately 9,000 linked files. The file system represents all files that can be accessed from websrv2 by any student, professor, or faculty member that has a Computer Science account.

5.4 Web Access Logs

The web access log file was obtained from websrv2 in late November. The access log contains websrv2 traffic from August 2005 to November 2005. All dates containing possible server attacks were removed from the possibilities of dates used for replay as the server attacks provide useless traffic that do not affect the disk in terms of requests sent to the disk.

We looked at all dates and chose September 23rd as the date for replay for our trace. It exhibited a higher number of requests than other days and as figure 7 shows, it contains the periodic affect necessary to exhibit the possible power savings of the different RAID personalities. The trace consisted of approximately 450MB worth of data transmitted in 17,000 requests over a 24 hour period. We also contained the last six hours of September 22nd in the log file to provide a warm up period for the web server cache and contained the first two hours of September 24th to provide a cool down period and allow the last requests of September 23rd to finish. The total length of the trace consisted of approximately 30 hours of web server activity.

Figure 7



We noticed when running the trace in real time that the disks were under utilized. Because of this we decided to speed the trace up by a factor of 8 to show the peaks and valleys of the usage patterns contained within the trace. Speeding up a trace too much can remove the human interaction between requests. As mentioned before, the trace should exhibit periods of high usage and low usage and the drives should reflect that. We also limited the number of concurrent connections to the web server to 1022 which is the maximum number of connections that can be handled at one time by our web server. This number was determined by using the httperf test tool.

6.0 Case Study and Validation

We used the framework to test the power consumption and performance of a new Power Aware RAID (PARAID) system developed at Florida State University. We compared RAID-0, to Least Recently Used (LRU) RAID-0 to PARAID. RAID-0 provides uniform striping of data across all disks with no redundancy. LRU is RAID-0 where each disk spins down after being idle for a certain time threshold. PARAID uses skewed striping and different disk configurations to provide the same performance as RAID-0 but reduce the power consumption of the web server.

6.1 RAID Personalities

RAID-0 distributes data across all disks and provides no data redundancy for the RAID device. Without redundancy, any single disk failure will result in loss of data. Since in its current implementation PARAID performs no redundancy, RAID-0 is the closest implementation and is used as the baseline for all tests conducted. LRU RAID-0 is a modification of RAID-0 and is based on previous studies, which spin-down individual drives based on a certain time threshold if the drive has not received any requests within the time threshold. The time threshold must be long enough such that spinning down the disks provides a power savings. Disks that are spun down immediately after a request may spin back up immediately. This causes the disk to consume more power as spinning up a disk consumes more power than keeping the disk in an idle state.

Power Aware RAID (PARAID) uses a skewed striping pattern such that certain file portions are replicated, so that a variable number of powered-on drives can deliver the same content at different speeds. The number of active disks is based on the current disk configuration. For our measurements we altered between two PARAID configurations, where the first configuration contained all four disks and the second configuration contained only two disks. Depending on drive utilization of the drive, a monitoring tool would switch the disk configuration to either all four disks or to only two disks.

6.2 Power Consumption

Figure 8 shows the power consumption comparisons between all three RAID personalities. Note that the power consumption of the drives reflects the usage patterns of the web server as shown in Figure 7. The power consumption of the drives is calculated at 15 second intervals. Greater disk drive power consumption occurs during what would be considered the midday hours of the trace while where more traffic is occurring than during the early morning and mid evening hours.

From this graph you can also determine when a four-disk configuration was used and when a two-disk configuration was used. One would assume that when the load on the server increases, drive utilization also increases, which would switch the disk configuration to four disks. This occurs around the midday hours. When the load of the server decreases, drive utilization also decreases, which then switches the disk configuration back to only two disks.

Using RAID-0 as a baseline suggests that LRU will save power on average of about three percent. However, power savings as much as 46% were obtained when running LRU, which may be attributed to caching. The average power savings for LRU was approximately 5%. PARAID actually saves between 15% to 40% power consumption when compared to RAID-0. On average, PARAID saves approximately 23% of the power consumed by RAID-0. At worse, PARAID has the same power consumption of RAID-0.

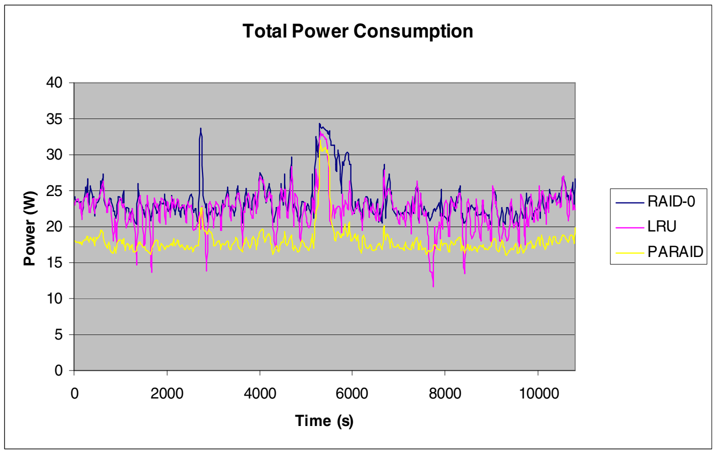


Figure 8

The power consumption by the drives is within the five percent margin as described by the hard drive manufacturer. The power consumption of the drives were tested during both the idle and active states. The spin-up of the drives also reaches the five percent margin as described by the manufacturer with the additional power consumed by the hard drive.

6.3 Performance

Preliminary numbers show that RAID-0 successfully completed approximately 54% of the requests it issued. Successful requests are defined as requests that were not cancelled due to either server errors such as files not found or server timeouts. A successful request is also a request that is not stopped by the client. This compares favorably to the 56% of requests that were served successfully during the analysis of the actual log file. The difference in requests can be attributed to the removal of dynamic content and the use of encryption. PARAID also served the same number of requests successfully as RAID-0 while LRU served 52% of the files successfully. LRU contained a more server timeouts, approximately 4%, due to the spin-up of the drives.

Figure 9 shows the cumulative distribution function (CDF) of the latency of the requests while figure 10 shows the CDF of the total file completion times The graph is read as the percentage of files (the vertical axis) completed within a certain time period (the horizontal axis). A high performance web server will serve a high percentage of files within a small period of time. This means that a web server has both a low latency and a low total file completion time.

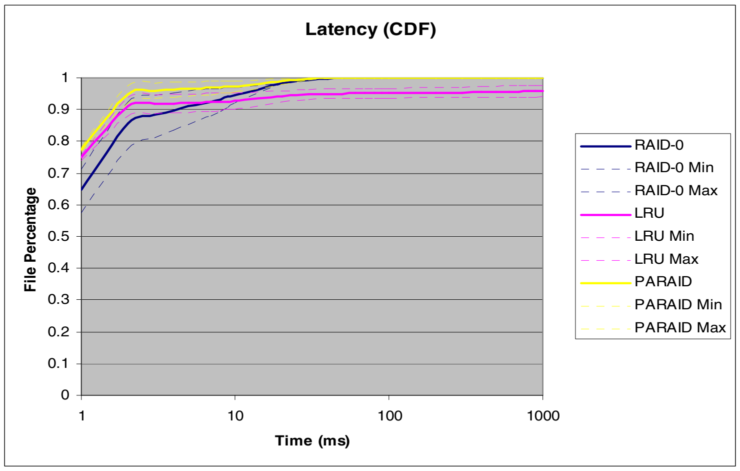


Figure 9

Preliminary numbers show that the latency of LRU is higher than that of RAID-0. For LRU the maximum latency for a successful request was over 150 times greater than that of RAID-0. Approximately 10% of the successfully completed files for LRU had a greater latency than the maximum latency for RAID-0. This is because the drives must spin-up before they can serve a request if the drives are not already active. For larger files that are striped across all four disks, this process takes approximately 20 seconds which is greater than the web server timeout. Preliminary numbers for PARAID shows that it performs slightly better than RAID-0. This is may be due to the fact that the latency and file completion time are as fast as the slowest drive. If only two drives are running rather than the four drives, statistically the two drives would have a faster response time than the four drives.

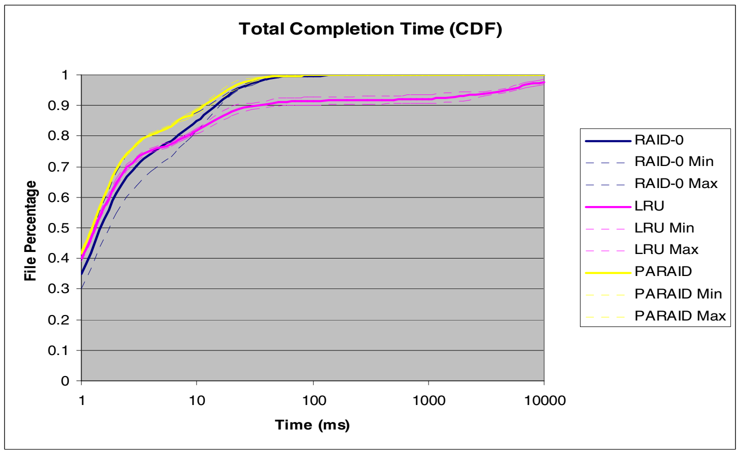


Figure 10

Preliminary numbers also show that LRU is the worst performer for the total file completion time when compared to RAID-0, while PARAID performed slightly better. The performance of LRU can once again be contributed to the spin-up of the disks as discussed above. When compared to RAID-0, the maximum file completion time was over 50 times greater than the maximum file completion time of RAID-0. Approximately 15% of the successfully completed files for LRU had a higher total file completion time than the maximum file completion time of RAID-0. PARAID once again performs slightly better than RAID-0 and may be attributed to the facts above.

For both the file completion time and the latency, LRU and PARAID initially perform much better than RAID-0. However, as the amount of time increases, RAID-0 catches up to the latency and completion time of PARAID while LRU then starts to suffer do to the spin-up of the drives from an idle state. The experiments run suggest that a 90% confidence exists with the number of tests already run. We expect 90% of the tests to fall within the minimum and maximum values on the graphs in both Figure 9 and Figure 10.

7.0 Future Work

Although this project is in a stable state, there are several improvements that can be made to the software and further tests that should be conducted to test the different types of workloads available. There are still many unanswered questions that could affect web server performance that this framework can be used to test.

7.1 Workloads

As mentioned before the workload used is from an academic web server. We wish to find a workload that is better suited to the web server hardware in our possession, so that we do not have to speedup the trace. We would like to test a workload that contains a higher load that we can run in real-time without the speedup and still see the peaks and valleys associated with the usage patterns of the users. We would also like to test the framework with different workloads from other academic servers, commercial servers, and government servers. These workloads all have different trends and usage patterns than the workload from websrv2.

We also need to determine how dynamic content affects the power consumption and performance of the drives. Since dynamic content is processed on the server side and then sent to the client side, additional power could be consumed by the hard drives with workloads that have a high amount of dynamic content. Our current implementation does not have a way to study dynamic data due to the encrypted file names and the randomness of the data contained within the file. Along with dynamic content we must test traces that contain writes. When writing files to disk, updates must occur with metadata, which may make disks consume more power.

7.2 File System Performance

Various file system performances can also occur. We must first identify the effect of using encrypted file names. With the current implementation, the encrypted file names are exactly forty characters long. However, those files that exist between both our web server and websrv2 are random in length. File names that are longer may take more operating system resources and cause longer lookup times and significant performance overhead. We would ultimately like to find a file system that does not require SHA-1 encryption. This would allow us to test the effect of encryption on the file system.

Also, we would like to streamline the file recreation process. Currently it takes approximately six hours to recreate all the directories, files, and links. This time element is due to the possibility that anyspecific file may depend on whether a directory or other file has already been created. Additional time is required to create the randomness of the files with a random number generator and table lookup takes time.

7.3 WebClient Improvements

Performing real-time performance analysis would benefit the WebClient program and the user since post-processing takes quite some time. These improvements mainly include total server performance metrics such as total server throughput. This is a tricky task as requests may overlap and last through multiple tracking intervals.

Additional WebClient improvements relate directly to the performance of the WebClient itself. The initial design of the WebClient used Java due to the ease of implementation. However, rewriting the program in a more performance-efficient programming language such as C may increase the overall performance. A second feature is to support stay-alive connections. For those requests that are waiting to be issued, should a request have the same IP address as the previous request, then the same connection could be used to avoid reopening a connection to the server.

The design of the WebClient has given it the ability to be extensible and test other network server architectures including proxy servers, e-mail servers, and file servers. Through the interface options with Java, a generic ServerClient can be created that implements a specific version to test a specific type of server. Many of the metrics between all servers are similar so the same performance metrics would be appropriate but other metrics could be added to fit the needs of any user

7.4 Automation

The biggest improvements can be made are in the area of automation of data gathering and processing. Currently it takes approximately 30 minutes to an hour to process just one test run of the WebClient. By integrating additional performance calculations to the WebClient itself, this process can be reduced as long as the associated overheads do not affect the overall performance of the WebClient. For information that cannot be processed within the WebClient, automated scripts can be written as information reported by both the WebClient and the power logging utilities are in formats that are easy to parse.

Finally, creating one package to gather and encrypt the file system would be beneficial. Developing another package to recreate the file system, as well as split and encrypt the web log files would also be beneficial. By creating these separate packages, a network administrator would only have to run the first package to gather the information for a researcher while the researcher would only have to run the second package to recreate everything.

8.0 Conclusion

We have created a framework that is easy to implement and run to gather information on the power and performance of server class disks. The framework provides straightforward replay of workloads based on real traffic patterns for both the energy consumption and performance of server class disks. The workloads are realistic, repeatable, and privacy-aware and match the hardware and software environments that are used within the framework. Although this framework was used to test the power and performance of a web server, the framework can be easily extended to test other network servers.

9.0 References

1. Abreu, Elinor. "Down on the Server Farm – Industry Trend or Event." *The Industry Standard.* 19 Feb. 2001 [online] http://www.findarticles.com/p/articles/mi_m0HWW/is_7_4/ai_70740553

2. Carrera et al. "Conserving Disk Energy in Network Servers." Technical Report DCS-TR-511. Department of Computer Science, Rutgers University. Nov. 2002

3. McLamb, Eric. "Fossils vs. Renewals: Energy's Future Today." Ecology Communications Incorporated. 2003 [online] http://www.ecology.com/feature-stories/nuclear-energy-challenge/

4. Elnozahy et al. "Energy Conservation Policies for Web Servers." Low-Power Computer Research Center, IBM Research. Mar 2003.

5. Arlitt et al. "Web Server Workload Characterization the Search for Invariants." Department of Computer Science, University of Saskatchewan. 15 Mar 1996.

6. Seltzer et al. "A Self-Scaling and Self-Configuring Benchmark for Web Servers." Department of Computer Science, Harvard University. 17 Mar 1997

7. RFC-3174 – US Secure Hash Algorithm 1 (SHA1). Sep 2001. [online] http://www.faqs.org/rfcs/rfc3174.html

8. WebStone Benchmark [online] http://www.mindcraft.com/webstone/

9. SPECWeb 99 Benchmark [online] http://www.spec.org/osg/web99/

10. HTTPerf [online] http://www.hpl.hp.com/personal/David_Mosberger/httperf.html