On the Energy Consumption and Performance of Systems Software

Appears in the proceedings of the 4th Israeli Experimental Systems Conference (ACM SYSTOR 2011)

Zhichao Li, Radu Grosu, Priya Sehgal, Scott A. Smolka, Scott D. Stoller, and Erez Zadok
Department of Computer Science, Stony Brook University
{zhicli,grosu,psehgal,sas,stoller,ezk}@cs.stonybrook.edu

ABSTRACT

Models of energy consumption and performance are necessary to understand and identify system behavior, prior to designing advanced controls that can balance out performance and energy use. This paper considers the energy consumption and performance of servers running a relatively simple file-compression workload. We found that standard techniques for system identification do not produce acceptable models of energy consumption and performance, due to the intricate interplay between the discrete nature of software and the continuous nature of energy and performance. This motivated us to perform a detailed empirical study of the energy consumption and performance of this system with varying compression algorithms and compression levels, file types, persistent storage media, CPU DVFS levels, and disk I/O schedulers. Our results identify and illustrate factors that complicate the system's energy consumption and performance, including nonlinearity, instability, and multi-dimensionality. Our results provide a basis for future work on modeling energy consumption and performance to support principled design of controllable energy-aware systems.

Categories and Subject Descriptors

H.3.4 [**Information Storage and Retrieval**]: Systems and Software—*Performance evaluation (efficiency and effectiveness)*; C.4 [**Performance of Systems**]: Modeling techniques; I.2.8 [**Artificial Intelligence**]: Problem Solving, Control Methods, and Search—*Control theory*

General Terms

Measurement, Performance

Keywords

Energy efficiency, System identification, Data compression

1. INTRODUCTION

The carbon footprint of the IT industry, though only 2% of the world economy, is estimated to be equal to that of the entire aviation industry [7]. Energy consumption is emerging as a critical issue in the design of computing systems [5, 14, 20, 22, 29, 33, 43]. The goals of energy-aware system design include saving energy without sacrificing performance, and supporting flexible, dynamic trade-offs between energy consumption and performance. Accurate models of energy consumption and performance provide a foundation for the design of energy-aware systems.

A large portion of the energy consumed by IT infrastructure is due to desktop machines and commercial servers [8]. Moreover, the total amount of electronic data stored world-wide is rising exponentially. By 2020, that figure is expected to reach 35 Zetta Bytes [16]; energy consumption is expected to grow just as rapidly. Thus, it is desirable to develop highly scalable solutions that are significantly better than today's solutions.

In prior work, we analyzed the energy and performance profiles of server workloads, such as Web servers, email servers, database servers, and file compression [24, 35]. We discovered large deviations for both performance and energy consumption—as much as 10 times—suggesting that there are significant opportunities to save energy and improve performance. Our past work considered those systems only as black-boxes and reported their performance and energy consumption without a deeper understanding of the exact reasons for those deviations.

Seeking a better understanding of the system internals of these workloads, we tried to identify their internal behavior, so we could build advanced controllers to better manage both energy and performance. Unfortunately, our initial attempts to identify these systems using traditional linear-systems identification techniques resulted in poor models with low prediction accuracy (under 50%).

In this paper, we shed considerable light on the complexities underlying systems-software energy consumption and performance. In particular, we present an in-depth experimental evaluation of the energy consumption and performance of a relatively simple yet familiar file-compression workload as a representative workload involving both substantial CPU usage and disk I/O. We also analyze the effects of several input parameters, including choice of compression algorithm, compression level, file type, persistent storage media (e.g., SATA, SAS, and SSD), CPU Dynamic Voltage and Frequency Scaling (DVFS) level, and disk I/O scheduler—all under the Linux operating system.

Our experimental results show that energy consumption and performance are unexpectedly complex and cannot be easily modeled using standard system-identification techniques. We identify several factors that contribute to this complexity, in terms of nonlinearity, instability, and multi-dimensionality. Our results suggest that hybrid discrete-continuous models [1, 18] may provide a suitable foundation for modeling and control of energy consumption and performance in energy-aware systems software.

The rest of the paper is organized as follows. Section 2 con-

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.
SYSTOR'11, May 30–June 1, 2011, Haifa, Israel.
Copyright 2011 ACM 978-1-4503-0773-4/11/05 ...$10.00.

siders related work. Section 3 provides the requisite background. Section 4 provides the motivation for this work. Section 5 presents our experimental setup and benchmarks. Section 6 contains our experimental results. We conclude in Section 7 and describe future work in Section 8.

2. RELATED WORK

This section places our work in the context of past work.

2.1 Energy Efficiency

Many energy-saving techniques have been developed at both the hardware and software levels. For example, virtualization allows multiple Operating Systems (OSs) to run on one server, sharing most of the resources, thereby reducing energy consumption. Moreover, there are energy-aware cache replacement algorithms [41], energy-aware task and interrupt management techniques [37], online learning-based power management [10], predictive data grouping and replication [14], and energy-aware file systems configuration pruning techniques [35]. Some modeling based approaches have been proposed by Isci, Sarikaya, and others [21, 34]. Some of our own past studies show significant energy savings possible in commodity Linux servers running common workloads such as Web, email, database, compression, etc. [24, 35]. Generally, optimal use of energy-saving techniques requires accurate models of system energy consumption with respect to appropriate parameters; the work described in this paper is a step towards the development of such models.

2.2 Energy Consumption of Data Compression

Our prior work, conducted by Kothiyal et al., evaluated energy consumption and performance of data compression on servers [24] and demonstrated that compression reduces energy consumption in some situations but not all. A careful application of compression can save energy in some cases by a factor of 10×, but a careless application of compression can easily waste energy and slow performance by 200×. In contrast to the work described in this paper, our past study did not focus on accurate modeling of energy consumption and hence did not discuss system identification or analyze the behavioral characteristics of energy consumption and performance that make accurate modeling difficult.

3. BACKGROUND

In this section, we describe background work in terms of compression algorithms (Section 3.1), I/O schedulers (Section 3.2), and power and energy consumption (Section 3.3).

3.1 Compression Algorithms

In Linux, there are three main compression utilities: `gzip`, `bzip2`, and `lzop`, each of which has compression levels ranging from level 1 to level 9. A higher level tries to achieve a better compression ratio at the expense of additional CPU cycles.

Gzip [15] is based on the DEFLATE algorithm, which is a combination of LZ77 and Huffman coding. Bzip2 uses the Burrows-Wheeler transform to convert frequently recurring character sequences into strings of identical letters and then applies a move-to-front transform and Huffman coding [6]. Lzop [30] uses the LZO library and produces files a bit larger than Gzip's but with a lower CPU use. For `lzop`, compression levels 1 to 6 are identical.

3.2 I/O Schedulers

I/O scheduling has been studied aggressively [2, 4, 19, 23, 39] especially since the speed of disk lags far behind the speed of CPU and RAM.

Normally, a disk scheduler tries to maintain a balance between fairness, performance, and latency (or real time guarantees). Fairness guarantees that every process has fair share of the access to disk on a multi-user system. Performance requires the scheduler to serve requests predictably to save both time and energy. Latency means that any request must be served within a given time limit. There are four main I/O schedulers in Linux systems: (1) CFQ (the default), which emphasizes fairness; (2) ANTICIPATORY, which emphasizes performance; (3) DEADLINE, which is designed for low latency and real time access; and (4) NOOP, which is a simple first-come-first-served scheduler.

3.3 Power and Energy Consumption

In this subsection, we introduce the power and energy consumption patterns for both CPU and disk, since our workload is both CPU-intensive and disk-intensive.

The power consumed in a processor consists of three portions: dynamic power $P_{dynamic}$, static power P_{static}, and short-circuit power [26]. For Complementary Metal Oxide Semiconductor (CMOS) chips, dynamic power refers to the energy consumption in switching transistors, while static power refers to the flowing leakage current when a transistor is off. Short-circuit power is consumed only during signal transitions and is insignificant. The dynamic power is calculated as follows:

$$P_{dynamic} = C \times V^2 \times f \tag{1}$$

where C is the capacitance per cycle, V is the supply voltage and f is the processor clock frequency.

Although dynamic power is the primary source of power dissipation in CMOS chips, static power is becoming an important issue. Static power is computed as follows:

$$P_{static} = V \times I_{th} + V_{bs} \times (I_{jn} + I_{bn}) \tag{2}$$

where I_{th} is the sub-threshold leakage current, V_{bs} is the body bias voltage, and I_{jn} and I_{bs} are the drain and source to body junction leakage current, respectively.

Processors with Dynamic Voltage and Frequency Scaling (DVFS) are capable of operating at multiple frequency and voltage levels. Dynamic power is considered to be the dominant portion of the processor's energy consumption. As seen from Equation 1, $P_{dynamic}$ depends linearly on frequency and quadratically on voltage. However, operating at a lower voltage and frequency does not necessarily result in overall energy savings, as we see later in Section 6.3. The main reason is that when running at a lower frequency, it usually takes longer to accomplish the same work, which can increase the total energy consumption.

The energy consumed by a Hard Disk Drive (HDD) follows the following equation:

$$E_{disk} = E_{spin} + E_{head} \tag{3}$$

where E_{spin} refers to the energy consumed by the spinning platter, and E_{head} refers to the energy consumption incurred by the movement of the disk head.

4. MOTIVATION

Section 4.1 gives some background on system identification. Section 4.2 describes the problems we encountered when we tried to apply system identification techniques to model the energy consumption of our workload.

4.1 System Identification

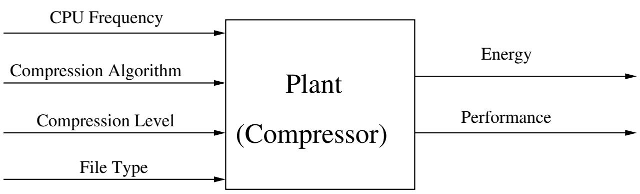


Figure 1: Plant: Compressor

System identification is the first step of control engineering that uses statistical methods to build models from observed behavior.

As shown in Figure 1, our system has four inputs: compression algorithm, compression level, file type, and CPU frequency. Our system has two outputs: energy and performance. Applying off-the-shelf technology for system identification, such as MATLAB's system identification tool-box [25] has considerable appeal, since one needs to know only the inputs and outputs. It does not require a detailed understanding of the system's behavior. By applying statistical techniques to data collected from the target system, system identification attempts to construct a mathematical model of the relationships between inputs and outputs.

A typical workflow for system identification follows these four steps: (1) Specify the model in the form of inputs and outputs, and design experiments to collect data; (2) Apply the system identification algorithm to estimate the values of the coefficients of the model; (3) Verify the accuracy of the resulting model by evaluating it against additional measured data; (4) Decide whether the model is acceptable. If the prediction accuracy is unacceptably low, one or more steps in the workflow need to be revisited.

In our experiments, we used a traditional linear state-space model of the following form:

$$x(n+1) = Ax(n) + Bu(n) + Kw(n) \tag{4a}$$
$$y(n) = Cx(n) + Du(n) + w(n) \tag{4b}$$

where $u(n)$ are the inputs, $y(n)$ are the outputs, $x(n)$ the internal states of the plant, and $w(n)$ is a white Gaussian noise representing uncontrollable inputs and output measurement errors (e.g., errors introduced by the default system daemons) at time n. The parameter $x(n+1)$ denotes the next internal states of the plant. Matrices A, B, C, D, and K denote the significance or weight that each element in the input, output, and Gaussian noise have in determining the next state and output of the system.

4.2 Problems Encountered

Our system is a simple file compressor. System inputs x can be file type (ZERO, TEXT, BINARY, or RANDOM), compression level (1 to 9), compression algorithm (GZIP, BZIP2, LZOP, or NONE for no compression), and CPU frequency/voltage (eight available choices). We considered energy consumption and performance as the outputs y.

The system inputs and outputs must be quantified in order to apply system identification. Energy is measured in Watt-hours. Performance is measured as the number of files compressed per second. The CPU frequency is measured in Hertz. However, it is difficult to choose appropriate numerical values to represent file types, compression levels, and compression algorithms.

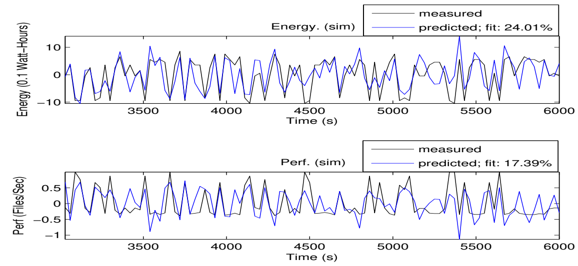


Figure 2: A typical example for poor accuracy. The two inputs are File Type and CPU Frequency. Compression algorithm is fixed to be gzip along with its default compression level. The two outputs are energy and performance which are normalized to be zero-mean.

The compression level is numerical, but the level number is actually just a label (in other words, a name); the numerical value has no direct significance other than ordering. Similarly, file types and compression algorithms are naturally identified by discrete, non-numerical labels but must be represented numerically to apply the system identification algorithm. The numbers chosen are significant, because they must be related to the next states and outputs by Equation 4 for system identification to succeed and should not impose arbitrary quantitative relationships. However, we have no a-priori way of deciding what values to use.

We tried a simple linear approach using consecutive integers (e.g., 0 for NONE, 1 for GZIP, 2 for BZIP2 and 3 for LZOP), as well as other numbers and ordering. We also tried a non-linear approach, assigning each compression algorithm a number corresponding to its compression ratio; but the compression ratio varies with file type and hence is not a fixed value associated solely with the compression algorithm.

In conclusion, labels are similar to the discrete states of a finite automaton. In our case, they represent different modes of system behavior; that is, they represent the modes of a hybrid automaton. Any attempt to give them a numerical meaning is doomed to fail.

We prepared two data sets of the same size to identify the system. One data set is used to estimate the parameters of the model using least-squares techniques; the other is used to evaluate the quality of the model fit. Accuracy is the percentage of model fit. We applied the MATLAB's system identification tool-box to learn Single-Input-Single-Output (SISO) and other system models. However, we achieved only limited accuracy, less than 50% in overall. A typical error graph appears in Figure 2.

This was clearly insufficient as a basis to design a controller. In order to better understand the causes of the problem, and to find ways of splitting the nonlinear behavior into segments that can be more accurately modeled as linear systems, we decided to study the system's energy consumption and performance in more detail.

5. METHODOLOGY

This section details our experimental setup and benchmarks.

5.1 Experimental Setup

We conducted our experiments on a Dell PowerEdge R710 server consisting of one quad-core Intel® Xeon™ Nehalem CPU with a maximum frequency of 2.395GHz with dynamic frequency and voltage scaling (DVFS) support: 7 different frequencies at a difference of 133MHz each without the Turbo Mode, and 8 different frequencies at a difference of 1MHZ for the top 2 frequencies and a difference of 133MHZ for the remaining 7 frequencies with the Turbo Mode on. The machine has 24GB RAM, out of which we used only 2GB to force I/O to take place. The server has two 146GB Seagate SAS disks with 15,000RPM rotation speed and a 16MB cache, two 250GB internal Fujitsu SATA disks with 7,200 RPM rotation speed and 16MB cache, and one 80GB Intel SSD disk model SSDSA2MH080G1C5. We ran all of our benchmarks on all of these three different kinds of disk drives. The server was running the Linux 2.6.18 kernel with the `acpi_cpufreq` module installed to enable software control of the CPU frequency.

We connected the server to a WattsUP Pro ES in-line power meter [12], which measures the energy drawn by a device plugged into the meter's receptacle. The power meter uses non-volatile memory to store measurements every second. Its resolution is 0.1 Watt-hours (1 Watt-hour = 3,600 Joules). The accuracy is $\pm 1.5\%$ of the measured value plus a constant error of ± 0.3 Watt-hours. Its resolution for power measurements is 0.1 Watts. We used the `wattsup` Linux utility to download the recorded data from the meter over a USB interface to the test machine.

We conducted 216 combinations of experiments (repeated five times each), and collected a large data set: 4,810,320 data points in total for a single run. Running one complete set of benchmarks took about 15 calendar days to complete. We ran and reran experiments many times over a period of more than a year, as we kept refining our experimentation methodology and developed automation tools. Retrieving information from this large data set and drawing figures were made simpler thanks to the automation tools we built.

To automate the measurements, we developed a tool called auto-ebench, written in Perl and Bash, that helped us benchmark the energy and power consumption under different scenarios while launching `vmstat` to record the number of block reads and block writes. We measured the total number of block reads and writes at the whole-system level; this saved us significant time and effort.

5.2 Benchmarks

The workload for each test is to compress 20 identical files with 20 threads concurrently, and write the compressed files to disk. Each file is 65MB. Several factors influence energy consumption for data compression, as we will discuss in Section 6.3. In order to fully explore these factors and their interactions, we conducted experiments for each combination. Specifically, we consider the following factors: persistent storage media (SAS disk, SATA disk, and SSD disk), I/O scheduler (anticipatory, CFQ, deadline and NOOP), compression algorithm (gzip, bzip2, and lzop) and compression level (1–9), and file type (text, binary, and random). We ran the above workload for each combination of these factors. Between each compression level, we inserted some sleeping intervals, so that each experiment for each compression level started at the same exact time. The elapsed time for compression plus the sleeping interval was the same and fixed during each compression level, in order to align the graphs for each compression level. Auto-ebench is responsible for repeatedly launching the experiments and recording the results multiple times and under multiple scenarios. Our experiments follow this pattern unless otherwise noted.

We ran all the tests five times and computed the 95% confidence intervals using the Student-t distribution. The error bars shown in our graphs are the half widths of the 95% confidence intervals. We used version 1.3.5 of `gzip`, version 1.0.3 of `bzip2`, and version v1.02rc1 of `lzop`.

The I/O scheduler can be set per device and is easy to configure. In order to set the I/O scheduler, we write the desired scheduler name to `/sys/block/$dev/queue/scheduler` and launch the experiments after that.

We ran the tests on the specified disk drive, formatted with Ext3 file system and mounted using the default options. To avoid caching effects, we unmounted the file system after each test iteration to flush the data in memory to disk. Our measurements include this flushing time.

6. EVALUATION

In this section, we provide evaluation and deep analysis for the energy consumption pattern of our file-compression workload. Sections 6.1, 6.2, and 6.3 focus on non-linearity, instability, and multi-dimensionality, respectively.

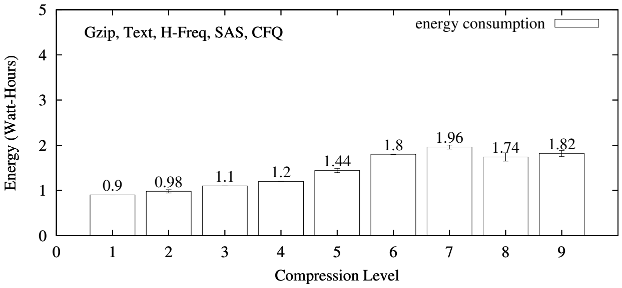
6.1 Nonlinearity

For compression algorithms, a higher compression level usually means a better compression ratio (CR). Table 1 shows the CR for all algorithms and levels.

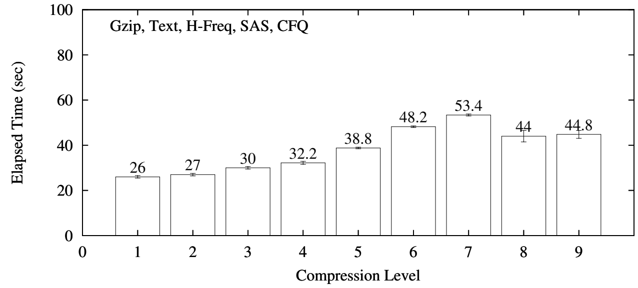
Tool	File Type		
	Text	Binary	Rand
gz-1	3.61	2.14	1.00
gz-2	3.77	2.18	1.00
gz-3	3.90	2.21	1.00
gz-4	4.18	2.26	1.00
gz-5	4.35	2.30	1.00
gz-6	4.43	2.32	1.00
gz-7	4.45	2.33	1.00
gz-8	4.46	2.33	1.00
gz-9	4.46	2.33	1.00
bz-1	4.72	2.38	0.99
bz-2	5.02	2.45	0.99
bz-3	5.18	2.53	0.99
bz-4	5.28	2.57	0.99
bz-5	5.36	2.60	0.99
bz-6	5.40	2.64	0.99
bz-7	5.44	2.65	1.00
bz-8	5.49	2.67	1.00
bz-9	5.50	2.69	1.00
lzo-(1∼6)	2.82	1.77	1.00
lzo-7	3.80	2.15	1.00
lzo-8	3.84	2.16	1.00
lzo-9	3.84	2.17	1.00

Table 1: Compression ratios achieved by various compression utilities and levels

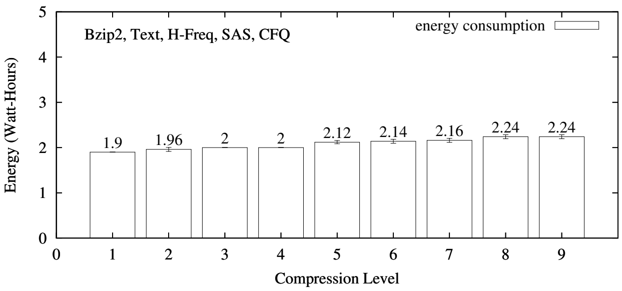
Although it is true that a higher compression level generally commits fewer blocks to disk for the same workload and hence might save energy due to reduced I/O activity, the overall energy consumption might not follow the same pattern. One possible reason is that the CPU may have to perform a lot more work in order to achieve a better CR, which takes longer time and consumes more energy. The actual energy consumed under certain workloads is in fact a trade-off between these factors. Therefore, as we can see from Figure 3, which presents measurements for `gzip`, `bzip2`, and `lzop`, the energy consumption is not a linear function of the compression level. Moreover, it is also not monotonically increasing with the compression level. For example, in Figure 3(b), energy consumption peaks at level 7, then unexpectedly drops at levels 8 and 9.



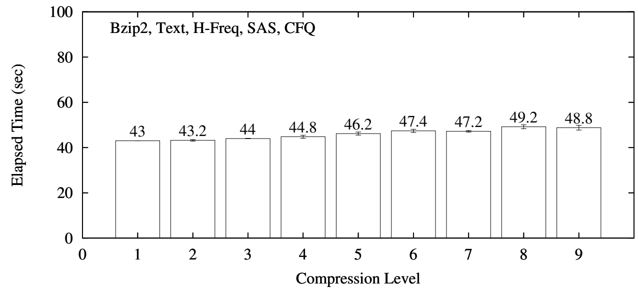
(a) **Energy** consumption in each compression level of gzip



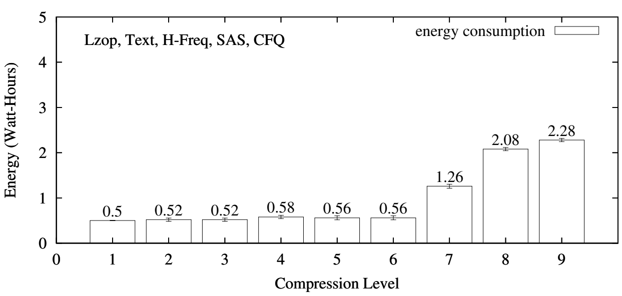
(b) **Time** taken in each compression level of gzip



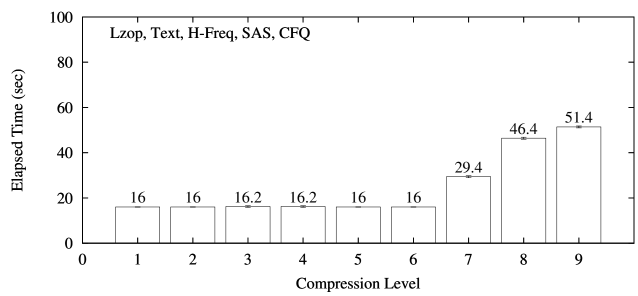
(c) **Energy** consumption in each compression level of bzip2



(d) **Time** taken in each compression level of bzip2



(e) **Energy** consumption in each compression level of lzop



(f) **Time** taken in each compression level of lzop

Figure 3: An example combined graph for illustrating nonlinearity. Experiments compressing text files using the highest CPU frequency, SAS disks, and the CFQ I/O scheduler. In 3(a), the x axis denotes the compression level and the y axis denotes Watt-hours (equals to 3,600 Joules). In 3(b), the unit for Elapsed Time is seconds. This representation is kept the same for 3(c), 3(d), 3(e), and 3(f).

Comparing the graphs in the left and right columns of Figure 3, we also observe that the energy consumption for the whole system depends heavily on the total elapsed time during the compression period [9, 32].

As we can see from Figure 3(a), in the case of `gzip`, the energy consumption goes up non-linearly and then goes down slightly as the compression level increases. Figure 3(b) shows that the elapsed time follows the same trend. In the case of `bzip2` as shown in Figure 3(c), the energy consumption is relatively stable, increasing only slightly across all 9 compression levels, which suggests that a balance between the CPU energy consumption and disk energy consumption has been achieved. The elapsed time, shown in Figure 3(d), follows the same pattern. With `lzop`, as shown in Figure 3(e), the energy consumption is the same for the first six identical compression levels and then increases monotonically but non-linearly. This reflects that for the last three compression levels, due to the longer elapsed time, the entire system (including the disk drive, even when it is just spinning, not reading or writing) is consuming more power at higher compression levels even though slightly fewer blocks are written to disk. Figure 3(f) show that the elapsed time strictly follows the same pattern.

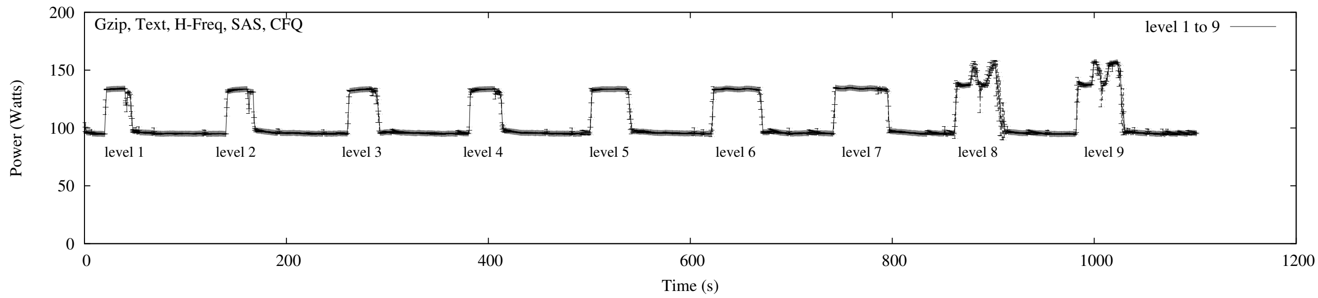
In summary, it is clear that the energy consumption and elapsed time relate non-linearly and in some cases non-monotonically with the compression level. Consequently, controlling the system's energy usage by adjusting the compression level is complex.

6.2 Instability

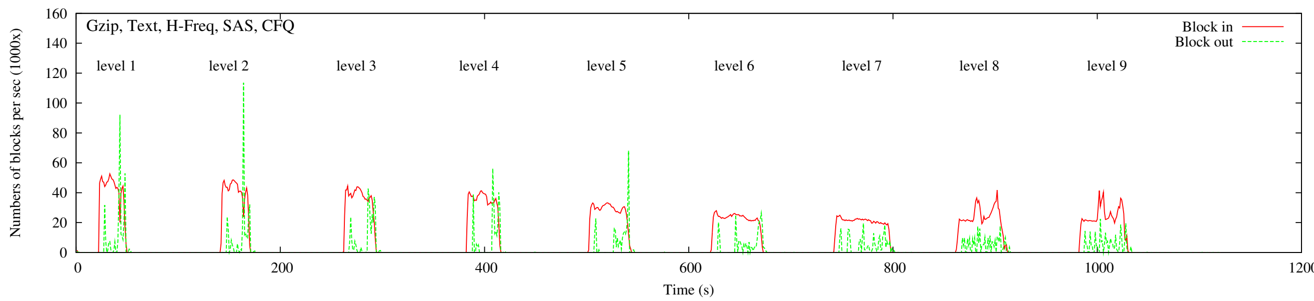
This section examines how the power consumption varies during each run. We found that in some cases, the power consumption response is unstable and fluctuates significantly, as we can see from Figures 4, 6, and 7. This should be taken into consideration when designing power-aware systems.

Our experiments revealed that the cause of those fluctuations lies in the interleavings between disk reads and writes when the CPU frequency is maintained at the same level. We discuss this in more detail below.

For `gzip` (Figure 4(a)), the power consumption response is relatively stable from level 1 to level 7. However, it becomes unstable

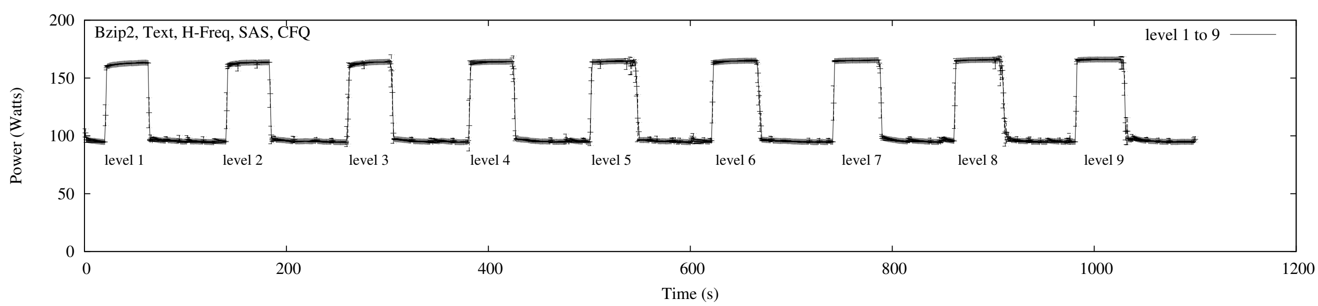


(a) Power consumption response for each level of compression of gzip

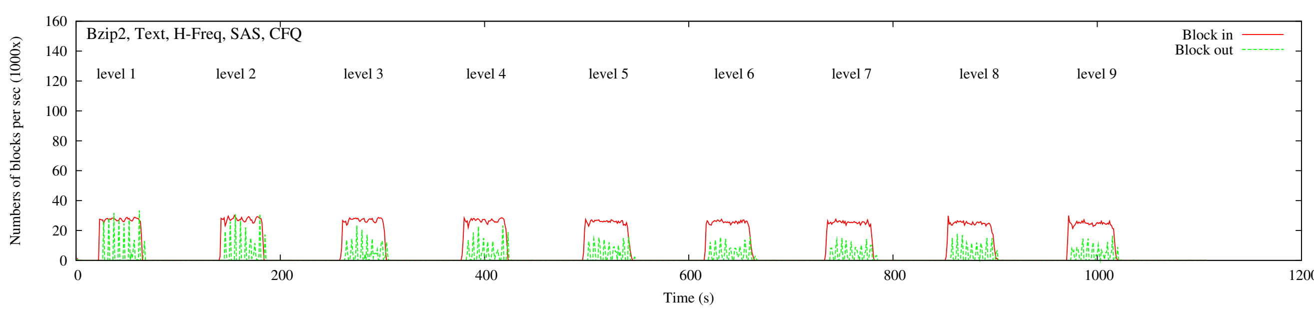


(b) Rate of block reads and writes for each level of compression of gzip

Figure 4: Relationship between the rates of block reads/writes and power consumption of gzip. The y axis is in units of thousands of reads/writes. The CPU frequency is set to the highest frequency in the above experiments. One can see that there are fluctuations in levels 8 and 9.

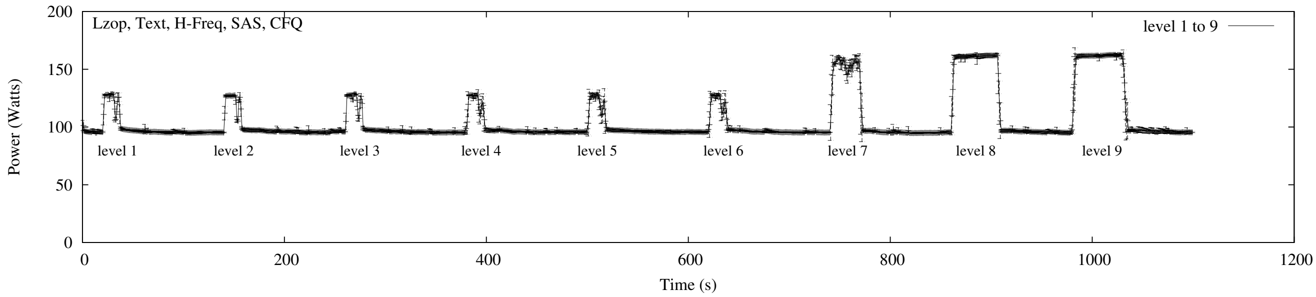


(a) Power consumption response for each level of compression of bzip2

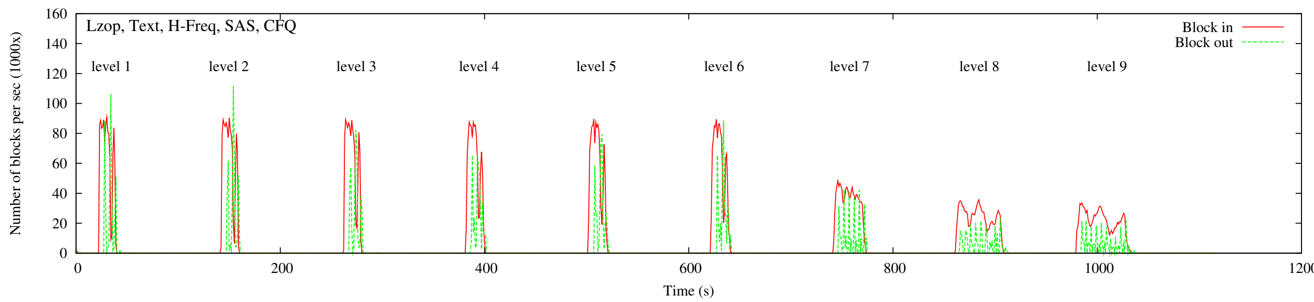


(b) Rate of block reads and writes for each level of compression of bzip2

Figure 5: Relationship between the rates of block reads/writes and power consumption of bzip2. The CPU frequency is set to the highest frequency in the above experiments. One can see that the power response is stable for each compression level.

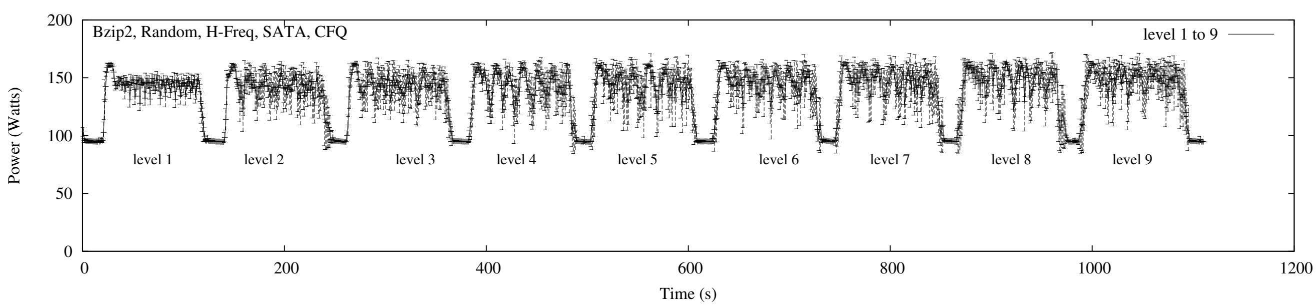


(a) Power consumption response for each level of compression of lzop

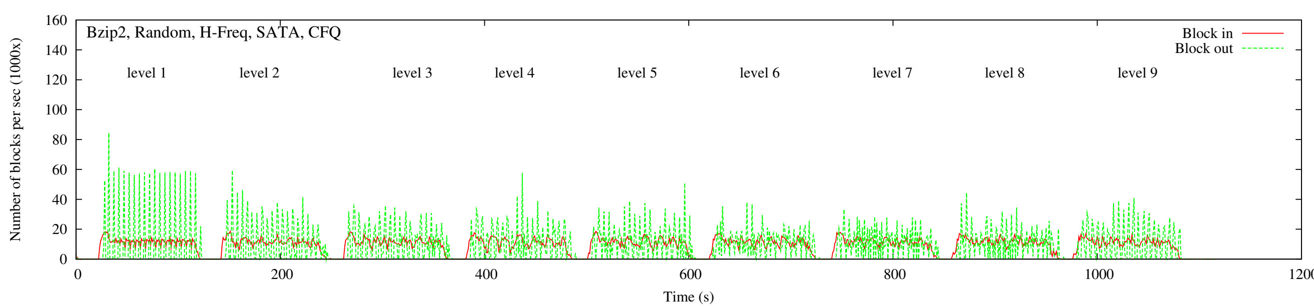


(b) Rate of block reads and writes for each level of compression of lzop

Figure 6: Relationship between the rates of block reads/writes and power consumption of lzop. The CPU frequency is set to the highest frequency in the above experiments. One can see fluctuations from levels 1 to 7.



(a) Power consumption response for each level of compression of bzip2 with SATA and random files



(b) Rate of block reads and writes for each level of compression of bzip2 with SATA and random files

Figure 7: An even more complex example. The CPU frequency is set to the highest frequency in the above experiments. One can see large fluctuations during every compression level.

in levels 8 and 9. Furthermore, Figure 4(b) reveals that the rate at which blocks are read exhibits the same pattern of stability in levels 1 to 7 and fluctuation at levels 8 and 9. Looking at Figure 4(b) in detail, especially in levels 8 and level 9, it also reveals more frequent interleavings between block reads and writes. For levels 1 through 4, there are just small fluctuations in power consumption towards the end of the compression job. A similar pattern appears in the interleavings between block reads and writes for levels 1 to 4. Moreover, the stable response in levels 5, 6, and 7 suggests equally distributed interleavings between the rates of block reads and writes. We believe that when block reads and writes are interleaved beyond a certain level, I/O scheduler algorithms (and possibly algorithms inside the disk) begin to break down and their efficiency goes down considerably as a result.

For `bzip2` (Figure 5(a)), the power consumption response is relatively stable. In Figure 5(b), we can see clearly that the rate of disk block reads is maintained at a stable level, and the rate at which disk blocks are written is equally distributed throughout the compression period. This leads the power consumption response to be stable.

For `lzop` (Figure 6(a)), the power consumption response follows a different pattern compared with the previous two scenarios. We can see from Figure 6(b) that for the first six levels, the I/O rate is much higher than in the remaining levels. However, the run is shorter in terms of elapsed time, as we can see from the width of the active intervals, and the interleavings between the rates of block reads and writes are in some degree not equally distributed across the compression level, resulting in a few fluctuations towards the end of each compression level. For levels 8 and 9, since the interleavings are equally distributed, the power response is relatively stable. The fluctuations in level 7 suggest there exists unequally distributed interleavings between the rates of disk reads and writes.

An even more complicated example appears in Figure 7(a). In this scenario, random files are being compressed and SATA is the persistent storage media. The power response is extremely unstable in each compression level. The interleavings between the rates of block reads and writes are ill regulated, as we can see from Figure 7(b). This suggests that the harder the file is to compress (e.g., high entropy), the less predictable the performance and energy consumption are. There is no simple way to model systems exhibiting such complex and diverse behavior.

We conclude that the power response exhibits instability in many cases. This contributes to the complexity of the energy usage of the system and makes controlling a serious challenge.

6.3 Multi-Dimensionality

In this section, we illustrate the dependence of the energy consumption for our system on several factors, such as CPU frequency, compression algorithm and level, file type, persistent storage media, and disk I/O scheduler.

The compression algorithm is clearly an important factor of energy consumption here, as we have already seen in Figure 3. For example, `bzip2` takes much longer time to compress than `lzop` does. Thus, `bzip2` usually takes more energy to compress than `lzop` does.

One might expect that a lower CPU frequency will result in lower energy consumption. However, as we can see from Figures 8(a) and 8(b), that is not necessarily true. With lower CPU frequency, the energy consumption is actually increased for all the compression levels. The reason is that when the CPU frequency is lower, it takes longer to finish the compression, which generally results in a higher total energy consumption. We can also see from Figure 8 that for both the highest frequency and the lowest frequency, the consumed energy increases as a function of compression level. However, there is also a possibility that when the CPU frequency is lower, the rate at which the CPU compresses data in the blocks will be closer to the rate at which the disk drive produces blocks. If this happens, it can save energy at the end, since there is no wasted energy.

The disk I/O scheduler influences the order of disk writes and hence may affect the energy consumption. Figure 9 shows the energy consumption with 4 different I/O schedulers. We can see that anticipatory and CFQ have largely the same effect, while deadline and NOOP also have similar effect to each other but different from anticipatory or CFQ. As the unit for the y axis is Watt-hours, the difference in energy consumption between anticipatory and CFQ is actually significant, especially for larger workloads.

The file type affects different compression strategies for each compression algorithm and hence plays a role in energy consumption. The left column of Figure 10 shows the energy consumption of the compression workload for different file types. We see that the workload with binary files consumes more energy than the workload with text files when other parameters are the same; this makes sense because text files have more common patterns that can be compressed (e.g., lower entropy). Also for text and binary files, more energy is consumed with compression level 9 than with other compression levels. Surprisingly, for random files, level 8 turns out to be the most energy-consuming one, instead of level 9. We conclude that the file type affects the energy consumption response in a manner that is not easy to predict, and an approach involving adaptive feedback control may thus be required.

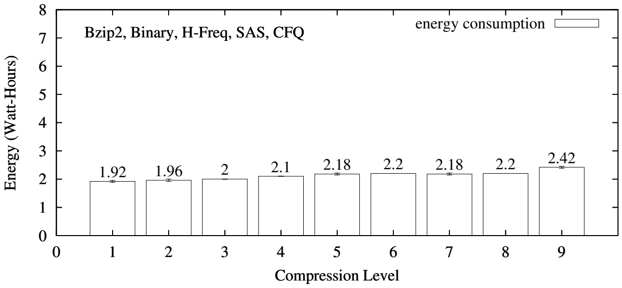
Different disk types usually have different electronics and firmware, different physical features, and different storage strategies. This should affect energy consumption. The right column of Figure 10 shows the energy consumption of the compression workload for different persistent storage media. As expected, SAS is generally faster than SATA, so the workload runs faster and consumes less energy, 2–12% less. SSD is the fastest storage media among the three, consuming the least energy, 3–5% less than SAS and 6–16% less than SATA. This is because an SSD contains no energy-consuming moving parts (cf. Equation 3) and stores data on non-volatile flash memory chips using a Flash Translation Layer (FTL) that allows the linear device to look like a traditional disk. These results also show that the workload is not completely CPU bound, even though it is CPU intensive.

In summary, we observe that the total energy consumption of computer systems follows a complicated pattern, because the energy consumption for each subsystem contributes to it. This suggests that instead of trying to develop system-level energy models purely in a bottom-up fashion, a more practical approach may be to use machine learning methods in the development of such models to guide the design of energy-aware systems.

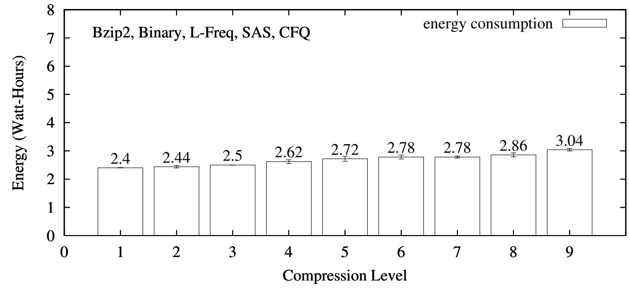
7. CONCLUSIONS

Accurate models of energy consumption and performance are vital for the design and implementation of energy-efficient systems. Our detailed experimental results show that the behavior of these quantities is far more complicated than one might expect, even for a relatively simple workload such as data compression. The complexity is reflected in nonlinearity, instability, and multi-dimensionality. These factors must be considered in the design of energy-efficient systems.

Although we have measured and analyzed the effects of several factors, there may be other important factors to consider, depending on the system, such as the workload itself, and even the server and machine-room temperatures.

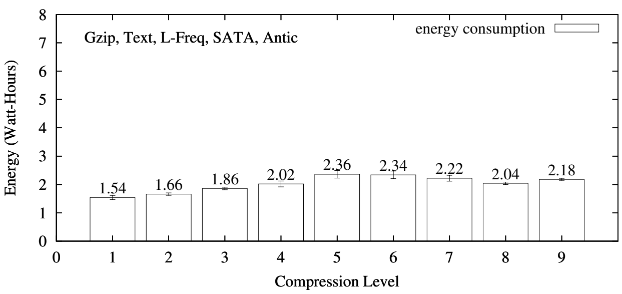


(a) **Energy** consumption in highest frequency with each compression level

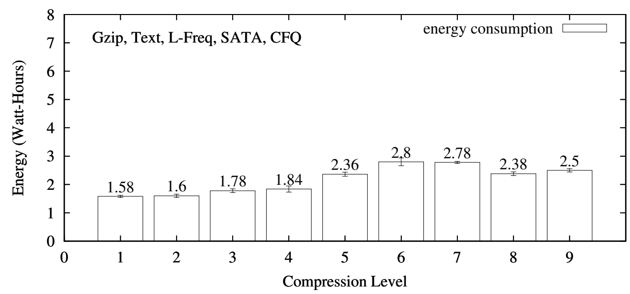


(b) **Energy** consumption in lowest frequency with each compression level

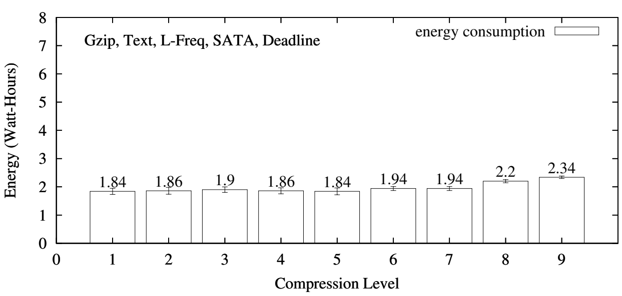
Figure 8: Energy consumption at the highest and lowest CPU frequencies



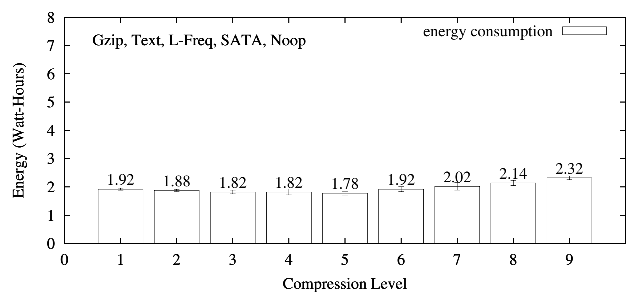
(a) **Energy** consumption with anticipatory for each compression level



(b) **Energy** consumption with CFQ for each compression level



(c) **Energy** consumption with deadline for each compression level



(d) **Energy** consumption with NOOP for each compression level

Figure 9: Energy consumption under 4 different I/O schedulers

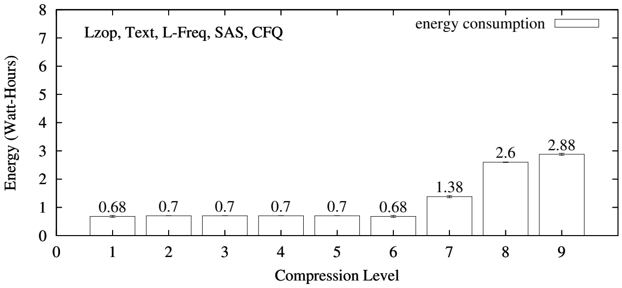
8. FUTURE WORK

Software systems often perform poorly because of poor handling of dynamic changes in workloads and resources (e.g., due to failures). The problem of dynamic changes exists in many engineering disciplines, such as aeronautic, electrical and mechanical engineering [17]. Feedback control theory [10, 27, 31, 36, 44, 45] has been proven effective in the design of some energy-efficient computer systems. We plan to apply control theory to complex software systems that can balance performance and energy consumption given user preferences.

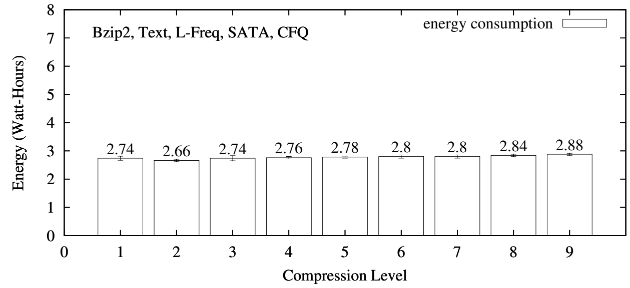
To achieve this goal, we first need to build our knowledge of the energy and performance of the complex systems. Our critical measurements and analysis of energy and performance patterns will in the end help building better energy-aware software systems, better schedulers [3, 28, 40], better operating systems and even better controllers [11]. We plan to investigate techniques to overcome the non-linearity of systems, by exploring linearization techniques such as segmenting a behavior into groups where each group can be modeled linearly, and combining the groups using a state machine. We have already begun to explore the design of controllers that can handle multiple inputs, multiple outputs, and multiple internal states (e.g., MIMO).

There are several directions for future work on measurement and modeling of energy consumption. Currently, we can only measure the energy consumption of the entire system. However, in some situations, it may be desirable to measure, model, and control energy consumption of individual components, such as the CPU, the disk drive [42], and so on. In addition, to better understand the dependence of the energy consumption of file compression on the file contents, we plan to generate and use files of the same type but with different entropy levels [13].

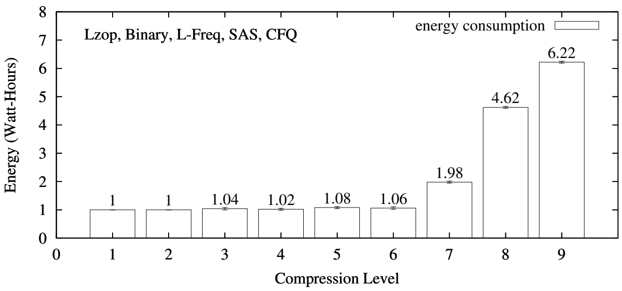
Moreover, because it is so tedious to measure the energy consumption under so many different scenarios, we plan to continue developing the auto-ebench tool-set mentioned in Section 5 to au-



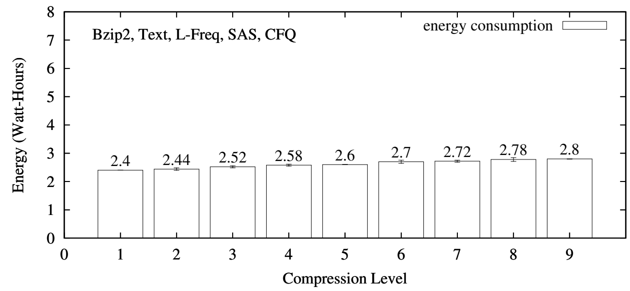
(a) **Energy** consumption of text files with each compression level



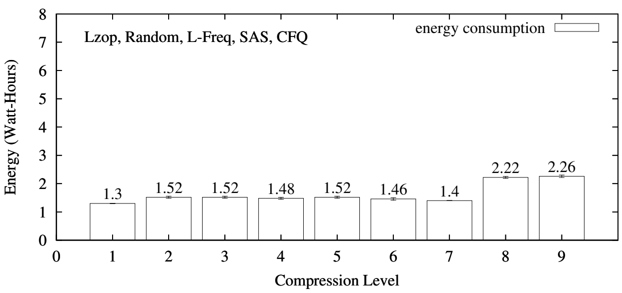
(b) **Energy** consumption of SATA



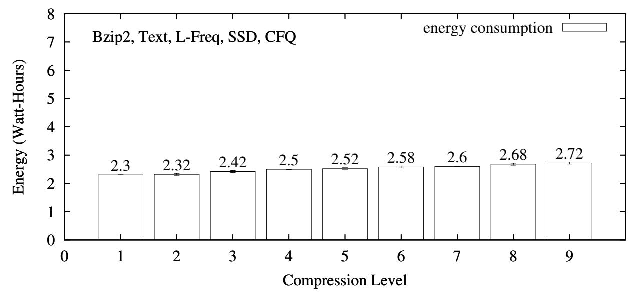
(c) **Energy** consumption of binary files with each compression level



(d) **Energy** consumption of SAS



(e) **Energy** consumption of random files with each compression level



(f) **Energy** consumption of SSD

Figure 10: Energy consumption for different File types and disk types

tomate energy benchmarking, similar to what auto-pilot [38] does to automate file system benchmarking. We plan to enhance auto-ebench to support additional features such as automatic calculation of confidence intervals, automatic detection of memory leaks, and intelligent analysis of results.

9. ACKNOWLEDGEMENTS

This work is sponsored in part by NSF awards CCF-0937854, CCF-0926190, CCF-1018459, and CNS-0831298; AFOSR grant number FA0550-09-1-0481; and an IBM Faculty Award. This work performed in part under the Stony Brook Advanced Energy Research and Technology Center (AERTC, `www.aertc.org`).

10. REFERENCES

[1] R. Alur and D. L. Dill. A theory of timed automata. *Theoretical Computer Science*, 126(2):183–235, 1994.

[2] J. Axboe. CFQ IO Scheduler, 2007. `http://mirror.linux.org.au/pub/linux.conf.au/2007/video/talks/123.ogg`.

[3] H. Aydin, R. Melhem, D. Mosse, and P. Mejia-Alvarez. Power-Aware Scheduling for Periodic Real-Time Tasks. *IEEE Transactions on Computers*, 53:584–600, 2004.

[4] N. Bansal, A. Blum, S. Chawla, and A. Meyerson. Approximation algorithms for deadline-tsp and vehicle routing with time-windows. In *Proceedings of the thirty-sixth annual ACM symposium on Theory of computing*, pages 166–174. ACM, 2004.

[5] P. Bohrer, E. N. Elnozahy, T. Keller, M. Kistler, and L. C. Mcdowell. The case for Power Management in Web Servers, 2002. `www.research.ibm.com/people/l/lefurgy/Publications/pac2002.pdf`.

[6] Bzip2 Wikipedia Documentation. `http://en.wikipedia.org/wiki/Bzip2`.

[7] J. Chang, J. Meza, P. Ranganathan, C. Bash, and A. Shah.

Green server design: Beyond operational energy to sustainability. In *Proceedings of the 2010 International Conference on Power Aware Computing and Systems*, HotPower'10, 2010.

[8] D. Colarelli and D. Grunwald. Massive Arrays of Idle Disks for Storage Archives. In *Proceedings of the 2002 ACM/IEEE conference on Supercomputing*, pages 1–11, 2002.

[9] D. C. Montgomery. *Engineering Statistics*. Wiley, 3 edition, 2004.

[10] G. Dhiman and T. Rosing. System-level Power Management using Online Learning. *IEEE Transaction on Computer-Aided Design of Integrated Circuits Systems*, 28(5):676–689, 2009.

[11] Y. Diao, N. Gandhi, J. L. Hellerstein, S. Parekh, and D. M. Tilbury. Using MIMO feedback control to enforce policies for interrelated metrics with application to the apache web server. In *Proceedings of the Network Operations and Management Symposium*, pages 219–234, 2002.

[12] Watts up? PRO ES Power Meter. `www.wattsupmeters.com/secure/products.php`.

[13] Introduction to Entropy. `http://en.wikipedia.org/wiki/Introduction_to_entropy`.

[14] D. Essary and A. Amer. Predictive Data Grouping: Defining the Bounds of Energy and Latency Reduction through Predictive Data Grouping and Replication. *ACM Transactions on Storage (TOS)*, 4(1):1–23, May 2008.

[15] J. L. Gailly. GNU Zip. `www.gnu.org/software/gzip/gzip.html`, 2000.

[16] J. Gantz and D. Reinsel. The digital universe decade - are you ready? `www.emc.com/digital_universe`, May 2010.

[17] J. L. Hellerstein, S. Singhal, and Q. Wang. Research Challenges in Control Engineering of Computing Systems. *Transactions on Network and Service Management*, 6(4), 2009.

[18] T. A. Henzinger, P.-H. Ho, and H. Wong-Toi. Hytech: The next generation. In *IEEE Real-Time Systems Symposium*, pages 56–65, 1995.

[19] M. Hofri. Disk Scheduling: FCFS vs. SSTF revisited. *Communication of the ACM*, 23(11), November 1980.

[20] H. Huang, W. Hung, and K. Shin. FS2: Dynamic Data Replication in Free Disk Space for Improving Disk Performance and Energy Consumption. In *Proceedings of the 20th ACM Symposium on Operating Systems Principles (SOSP '05)*, pages 263–276, Brighton, UK, October 2005. ACM Press.

[21] C. Isci, G. Contreras, and M. Martonosi. Live, runtime phase monitoring and prediction on real systems with application to dynamic power management. In *Proceedings of the 39th Annual IEEE/ACM International Symposium on Microarchitecture*. IEEE Computer Society, 2006.

[22] N. Joukov and J. Sipek. GreenFS: Making enterprise computers greener by protecting them better. In *Proceedings of the 3rd ACM SIGOPS/EuroSys European Conference on Computer Systems 2008 (EuroSys 2008)*, Glasgow, Scotland, April 2008. ACM.

[23] R. King. Disk Arm Movement Anticipation of Future Requests. *ACM Transactions on Computer Systems*, 8(3):214–229, 1990.

[24] R. Kothiyal, V. Tarasov, P. Sehgal, and E. Zadok. Energy and Performance Evaluation of Lossless File Data Compression on Server Systems. In *Proceedings of the 2nd Israeli Experimental Systems Conference (ACM SYSTOR '09)*, Haifa, Israel, May 2009. ACM.

[25] L. Ljung. *System Identification (2nd ed.): Theory for the User*. Prentice Hall, 1999.

[26] S. M Martin, K. Flautner, T. N. Mudge, and D. Blaauw. Combined dynamic voltage scaling and adaptive body biasing for lower power microprocessors under dynamic workloads. In *Proceedings of the 2002 IEEE/ACM International Conference on Computer-Aided Design*, pages 721–725, 2002.

[27] R. Minerick, V. W. Freeh, and P.M. Kogge. Dynamic Power Management using Feedback , 2002. `www4.ncsu.edu/~vwfreeh/colp.pdf`.

[28] R. Mishra, N. Rastogi, D. Zhu, D. Mosse, and R. Melhem. Energy Aware Scheduling for Distributed Real-Time Systems. In *International Parallel and Distributed Processing Symposium*, 2003.

[29] D. Narayanan, A. Donnelly, and A. Rowstron. Write off-loading: practical power management for enterprise storage. In *Proceedings of the 6th USENIX Conference on File and Storage Technologies (FAST 2008)*, 2008.

[30] M.F.X.J. Oberhumer. lzop data compression utility. `www.lzop.org/`.

[31] C. Pereira, V. Raghunathan, S. Gupta, R. Gupta, and M. Srivastava. A Software Architecture for Building Power Aware Real Time Operating Systems. In *Proceedings of the IEEE CAS Workshop on Wireless Communication and Networking*, 2002.

[32] R. Jain. *The Art of Computer System Performance Analysis*. Wiley, 1991.

[33] S. Gurumurthi and A. Sivasubramaniam and M. Kandemir and H. Franke. DRPM: Dynamic Speed Control for Power Management in Server Class Disks. In *Proceedings of the 30th Annual International Symposium on Computer Architecture*, pages 169–181, 2003.

[34] R. Sarikaya, C. Isci, and A. Buyuktosunoglu. Runtime workload behavior prediction using statistical metric modeling with application to dynamic power management. In *In IEEE International Symposium on Workload Characterization*, 2010.

[35] P. Sehgal, V. Tarasov, and E. Zadok. Evaluating Performance and Energy in File System Server Workloads extensions. In *Proceedings of the Eighth USENIX Conference on File and Storage Technologies (FAST '10)*, pages 253–266, San Jose, CA, February 2010. USENIX Association.

[36] A. Soria-Lopez, P. Mejia-Alvarez, and J. Cornejo. Feedback scheduling of power-aware soft real-time tasks. In *ENC '05: Proceedings of the Sixth Mexican International Conference on Computer Science*, pages 266–273, Washington, DC, USA, 2005. IEEE Computer Society.

[37] V. Srinivasan, G. R. Shenoy, S. Vaddagiri, and D. Sarma. Energy-aware task and interrupt management in linux. In *Ottawa Linux Symposium*, July 2008.

[38] C. P. Wright, N. Joukov, D. Kulkarni, Y. Miretskiy, and E. Zadok. Auto-pilot: A Platform for System Software Benchmarking. In *Proceedings of the Annual USENIX Technical Conference, FREENIX Track*, pages 175–187, Anaheim, CA, April 2005. USENIX Association.

[39] Y. Yu, D. Shin, H. Eom, and H. Yeom. NCQ vs I/O Scheduler: Preventing Unexpected Misbehaviors. *ACM Transaction on Storage*, 6(1), March 2010.

[40] W. Yuan and K. Nahrstedt. Energy-efficient CPU Scheduling for Multimedia Applications. *ACM Transactions of Computer System*, 24(3):292–331, 2006.

[41] J. Yue, Y. Zhu, Z. Cai, and L. Lin. Energy and thermal aware buffer cache replacement algorithm. In *The 26th IEEE Symposium on Massive Storage Systems and Technologies*, 2010.

[42] J. Zedlewaki, S. Sobti, N. Garg, F. Zheng, A. Krishnamurthy, and R. Wang. Modeling hard-disk power consumption. In *Second Conference on File and Storage Technologies*, 2003.

[43] Q. Zhu, F. M. David, C. F. Devaraj, Z. Li, Y. Zhou, and P. Cao. Reducing Energy Consumption of Disk Storage Using Power-Aware Cache Management. In *Proceedings of the 10th International Symposium on High-Performance Computer Architecture*, pages 118–129, 2004.

[44] Y. Zhu and F. Mueller. Feedback EDF Scheduling Exploiting Dynamic Voltage Scaling. In *RTAS '04: Proceedings of the 10th IEEE Real-Time and Embedded Technology and Applications Symposium*, pages 33 – 63, Washington, DC, USA, 2004. IEEE Computer Society.

[45] Y. Zhu and F. Mueller. DVSleak: Combining Leakage Reduction and Voltage Scaling in Feedback EDF Scheduling. In *ACM SIGPLAN Notices, Proc. of LCTES'07*, pages 31–40, 2007.