JouleSort: A Balanced Energy-Efficiency Benchmark

Suzanne Rivoire
Stanford University

Mehul A. Shah
HP Labs

Parthasarathy Ranganathan
HP Labs

Christos Kozyrakis
Stanford University

ABSTRACT

The energy efficiency of computer systems is an important concern in a variety of contexts. In data centers, reducing energy use improves operating cost, scalability, reliability, and other factors. For mobile devices, energy consumption directly affects functionality and usability. We propose and motivate *JouleSort*, an external sort benchmark, for evaluating the energy efficiency of a wide range of computer systems from clusters to handhelds. We list the criteria, challenges, and pitfalls from our experience in creating a fair energy-efficiency benchmark. Using a commercial sort, we demonstrate a JouleSort system that is over 3.5x as energy-efficient as last year's estimated winner. This system is quite different from those currently used in data centers. It consists of a commodity mobile CPU and 13 laptop drives, connected by server-style I/O interfaces.

Categories and Subject Descriptors

H.2.4 [**Information Systems**]: Database Management—*Systems*

General Terms

Design, Experimentation, Measurement, Performance

Keywords

Benchmark, Energy-Efficiency, Power, Servers, Sort

1. INTRODUCTION

In contexts ranging from large-scale data centers to mobile devices, energy use in computer systems is an important concern.

In data center environments, energy efficiency affects a number of factors. First, power and cooling costs are significant components of operational and up-front costs. Today, a typical data center with 1000 racks, consuming 10MW total power, costs $7M to power and $4-$8M to cool per year, with $2-$4M of up-front costs for cooling equipment [28]. These costs vary depending upon the installation, but they are growing rapidly and have the potential eventually to outstrip the cost of hardware [2]. Second, energy use has implications for density, reliability, and scalability. As data centers house more servers and consume more energy, removing heat from the data center becomes increasingly difficult [27]. Since the reliability of servers and disks decreases with increased temperature, the power consumption of servers and other components limits the achievable density, which in turn limits its scalability. Third, energy use in data centers is starting to prompt environmental concerns of pollution and excessive load placed on local utilities [28]. Energy-related concerns are severe enough that companies like Google are starting to build data centers close to electric plants in cold-weather climates [24]. All these concerns have led to improvements in cooling infrastructure and in server power consumption [28].

For mobile devices, battery capacity and energy use directly affect usability. Battery capacity determines how long devices last, constrains form factors, and limits functionality. Since battery capacity is limited and improving slowly, device architects have concentrated on extracting greater energy efficiency from the underlying components, such as the processor, the display, and the wireless subsystems in isolation [20, 29, 31].

To drive energy-efficiency improvements, we need benchmarks to assess their effectiveness. Unfortunately, there has been no focus on a complete benchmark, including a workload, metric, and guidelines, to gauge the efficacy of energy optimizations from a whole-system perspective. Some efforts are under way to establish benchmarks for energy efficiency in data centers [33, 35] but are incomplete. Other work has emphasized metrics such as the energy-delay product or performance per Watt to capture energy efficiency for processors [13, 21, 27] and servers [34] without fixing a workload. Moreover, while past emphasis on processor energy efficiency has led to improvements in overall power consumption, there has been little focus on the I/O subsystem, which plays a significant role in total system power for many important workloads and systems.

In this paper, we propose *JouleSort* as a holistic benchmark to drive the design of energy-efficient systems. JouleSort uses the same workload as the other external sort benchmarks [1, 17, 25], but its metric incorporates total energy, which is a combination of power consumption and performance. The benchmark can be summarized as follows:

- Sort a fixed number of randomly permuted 100-byte records with 10-byte keys.

SIGMOD'07, June 12–14, 2007, Beijing, China.

- The sort must start with input in a file on non-volatile store and finish with output in a file on non-volatile store.
- There are three scale categories for JouleSort: 10^8 ($\sim$ 10GB), 10^9 ($\sim$ 100GB), and 10^{10} ($\sim$ 1TB) records
- The winner in each category is the system with the minimum total energy use.

We choose sort as the workload for the same basic reason that the Terabyte Sort, MinuteSort, PennySort, and Performance-price Sort benchmarks do [16, 17, 25]: it is simple to state and balances system component use. Sort stresses all core components of a system: memory, CPU, and I/O. Sort also exercises the OS and filesystem. Sort is a portable workload; it is applicable to a variety of systems from mobile devices to large server configurations. Another natural reason for choosing sort is that it represents sequential I/O tasks in data management workloads.

JouleSort is an I/O-centric benchmark that measures the energy efficiency of systems at peak use. Like previous sort benchmarks, one of its goals is to gauge the end-to-end effectiveness of improvements in system components. To do so, JouleSort allows us to compare the energy efficiencies of a variety of disparate system configurations. Because of the simplicity and portability of sort, previous sort benchmarks have been technology trend bellwethers, for example, foreshadowing the transition from supercomputers to clusters. Similarly, an important purpose of JouleSort is to chart past trends and gain insight into future trends in energy efficiency.

Beyond the benchmark definition, our main contributions are twofold. First, we motivate and describe pitfalls surrounding the creation of a fair energy-efficiency benchmark. We justify our fairest formulation, which includes three scale factors that correspond naturally to the dominant classes of systems found today: mobile, desktop, and server. Although we support both Daytona (commercially supported) and Indy ("no-holds-barred") categories for each scale, we concentrate on Daytona systems in this paper. Second, we present the winning 100GB JouleSort system that is over 3.5x more efficient ($\sim$ 11300 SortedRecs/Joule for 100GB) than last year's estimated winner ($\sim$ 3200 SortedRecs/Joule for 55GB). This system shows that a focus on energy efficiency leads to a unique configuration that is hard to find pre-assembled. Our winner balances a low-power, mobile processor with numerous laptop disks connected via server-class PCI-e I/O cards and uses a commercial sort, NSort [26].

The rest of the paper is organized as follows. In Section 2, we estimate the energy efficiency of past sort benchmark winners, which suggests that existing sort benchmarks cannot serve as surrogates for an energy-efficiency benchmark. Section 3 details the criteria and challenges in designing JouleSort and lists issues and guidelines for proper energy measurement. In Section 4, we *measure* the energy consumption of unbalanced and balanced systems to motivate our choices in designing our winning system. The balanced system shows that the I/O subsystem is a significant part of total power.

Section 5 provides an in-depth study of our 100GB JouleSort system using NSort [26]. In particular, we show that the most energy-efficient, cost-effective, and best-performing configuration for this system is when the sort is CPU-bound.

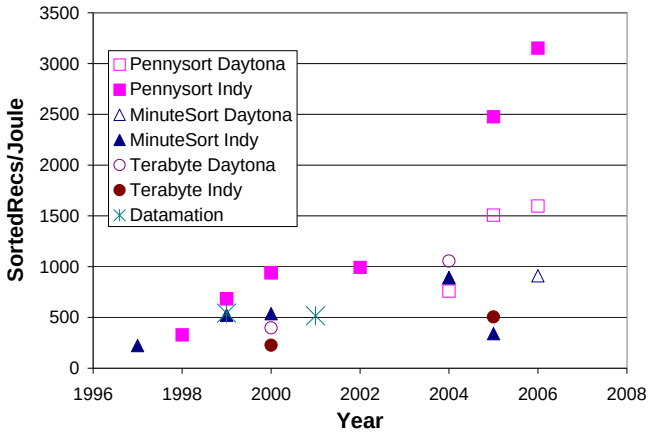

Figure 1: Estimated energy-efficiency of previous winners of sort benchmarks.

We also find that both the choice of filesystem and in-memory sorting algorithm affect energy efficiency. Section 6 discusses the related work, and Section 7 presents limitations and future directions.

2. HISTORICAL TRENDS

In this section, we seek to understand if any of the existing sort benchmarks can serve as a surrogate for an energy-efficiency benchmark. To do so, we first estimate the SortedRecs/Joule ratio, a measure of energy efficiency, of the past decade's sort benchmark winners. This analysis reveals that the energy efficiency of systems designed for pure performance (i.e. MinuteSort, Terabyte Sort, and Datamation winners) has improved slowly. Moreover, systems designed for price-performance (i.e. PennySort winners) are comparatively more energy-efficient, and their energy efficiency is growing rapidly. However, since our 100GB JouleSort system's energy efficiency is well beyond what growth rates would predict for this year's PennySort winner, we conclude that existing sort benchmarks do not inherently provide an incentive to optimize for energy efficiency, supporting the need for JouleSort.

2.1 Methodology

Figure 1 shows the estimated SortedRecs/Joule metric for the past sort benchmark winners since 1997. We compute these metrics from the published performance records and our own estimates of power consumption since energy use was not reported. We obtain the performance records and hardware configuration information from the Sort Benchmark website and the winners' posted reports [16].

We estimate total energy during system use with a straightforward approach from the power-management community. Since CPU, memory, and disk are usually the main power-consuming system components, we use individual estimates of these to compute total power. For memory and disks, we use the HP Enterprise Configurator [19] power calculator to yield a fixed power of 13W per disk and 4W per DIMM. Some of the sort benchmark reports only mention total memory capacity and not the number of DIMMs; in those cases, we assume a DIMM size appropriate to the era of the report. The maximum power specs for CPUs, usually

quoted as thermal design power (TDP), are much higher than the peak numbers seen in common use; thus, we derate these power ratings by a 0.7 factor. Although a bit conservative, this approach allows reasonable approximations for a variety of systems. When uncertain, we assume the newest possible generation of the reported processor as of the sort benchmark record because a given CPU's power consumption improves with shrinking feature sizes. Finally, to account for power supplies inefficiencies, which can vary widely [3, 5], and other components, we scale total system power derived from component-level estimates by 1.2 for single-node systems. We use a higher factor, 1.6, for clusters to account for additional components, such as networking, management hardware, and redundant power supplies.

Our power estimates are intended to illuminate coarse historical trends and are accurate enough to support the high-level conclusions in this section. We experimentally validated this approach against some server and desktop-class systems, and its accuracy was between 2% and 25%.

2.2 Analysis

Although previous sort benchmark winners were not configured with power consumption in mind, they roughly reflect the power characteristics of desktop and higher-end systems in their day. Thus, from the data in Figure 1, we can infer qualitative information about the relative improvements in performance, price-performance, and energy efficiency in the last decade. Figure 1 compares the energy efficiency of previous sort winners using the SortedRecs/Joule ratio and supports the following observations.

Systems optimized for price-performance, i.e. PennySort winners, clearly are more energy-efficient than the other sort benchmark winners, which were optimized for pure performance. There are two reasons for this effect. First, the price-performance metric motivates system designers to use fewer components, and thus less power. Second, it provides incentive to use cheaper, commodity components which, for a given performance point, traditionally have used less energy than expensive, high-performance components.

The energy efficiency of cost-conscious systems has improved faster than that of performance-optimized systems, which have hardly improved. Others have also observed a flat energy-efficiency trend for cluster hardware [2]. Much of the growth in the PennySort curve is from the last two Indy winners, which have made large leaps in energy efficiency. In 2005, algorithmic improvements and a minimal hardware configuration played a role in this improvement, but most importantly, CPU design trends had finally swung toward energy efficiency. The processor used in the 2005 PennySort winner has 6x the clock frequency of its immediate predecessor, while only consuming 2x the power. Overall, the 2005 sort had 3x better performance than the previous data point, while using 2x the power. The 2006 PennySort winner, GPUTeraSort, increased energy efficiency by introducing a new system component, the graphics processing unit (GPU), and utilizing it very effectively. The chosen GPU is inexpensive and comparable in power consumption (57W) to the CPU (80W), but it provides better streaming memory bandwidth than the CPU.

This latest winner, in particular, shows the danger of relying on energy benchmarks that focus only on specific hardware like CPU or disks, rather than end-to-end efficiency. Such specific benchmarks would only drive and track improvements of existing technologies and may fail to anticipate the use of potentially disruptive technologies.

Benchmark	SRecs/sec	SRecs/$	SRecs/J
PennySort	50%/yr.	57%/yr.	24%/yr.
Minute, Terabyte, and Datamation	37%/yr.	n/a	12%/yr.

Table 1: This table shows the estimated yearly growth in pure performance, price-performance, and energy efficiency of past winners.

Since price-performance winners are more energy-efficient, we next examine whether the most cost-effective sort implies the best achievable energy-efficient sort. To do so, we first estimate the growth rate of sort winners along multiple dimensions. Table 1 shows the growth rate of past sort benchmark winners along three dimensions: performance (SortedRecs/sec), price-performance (SortedRecs/$), and energy efficiency (SortedRecs/Joule). We separate the growth rates into two categories based on the benchmark's optimization goal: price- or pure performance, since the goal drives the system design. For each category, we calculate the growth rate as follows. We choose the best system (according to the metric) in each year and fit the result with an exponential. Table 1 shows that PennySort systems are improving almost at the pace of Moore's Law along the performance and price-performance dimensions. The pure performance systems, however, are improving much more slowly, as noted elsewhere [16].

More importantly, our analysis shows much slower estimated growth in energy efficiency than in the other two metrics for both benchmark categories. Given last year's estimated PennySort winner provides $\sim$ 3200 SRecs/J, our current JouleSort winner at $\sim$ 11300 SRecs/J is nearly 3x the expected value of $\sim$ 4000 SRecs/J for this year. This result suggests that we need a benchmark focused on energy efficiency to promote development of the most energy-efficient sorting systems and allow for disruptive technologies in energy efficiency irrespective of cost.

3. BENCHMARK DESIGN

In this section, we detail the criteria and challenges in designing an energy-efficiency benchmark. We describe some of the pitfalls of our initial specifications and how the benchmark has evolved. We also specify rules of the benchmark with respect to both workload and energy measurement.

3.1 Criteria

Although past studies have proposed energy-efficiency metrics [13, 21, 34, 27] or power measurement techniques [9], none provide a complete benchmark: a workload, a metric of comparison, and rules for running the workload and measuring energy consumption. Moreover, these studies traditionally have focused on comparing existing systems rather than providing insight into future technology trends. We set out to design an energy-oriented benchmark that addresses these drawbacks with the criteria below in mind. While achieving all these criteria simultaneously is hard, we strive to encompass them as much as possible.

Energy-efficiency: The benchmark should measure a system's "bang for the buck," where bang is work done and the cost reflects some measure of power use, e.g. average

power, peak power, total energy, and energy-delay. To drive practical improvements in power consumption, cost should reflect both a system's performance and power use. A system that uses almost no power but takes forever to complete a task is not practical, so average and peak power are poor choices. Thus, there are two reasonable cost alternatives: energy, a product of execution time and power, or energy-delay, a product of execution time and energy. The former weighs performance and power equally while the latter, popular in CPU-centric benchmarks, places more emphasis on performance [13]. Since there are other sort benchmarks that emphasize performance, we chose energy as the cost.

Peak-use: A benchmark can consider system energy in three important modes: idle, peak-use, or a realistic combination of the two. Although minimizing idle-mode power is useful, evaluating this mode is straightforward. Real-world workloads are often a combination, but designing a broad benchmark that addresses a number of scenarios is difficult to impossible. Hence, we chose to focus our benchmark on an important, but simpler case: energy efficiency during peak use. Energy efficiency at peak is the opposite extreme from idle and gives an upper bound on work that can be done for a given energy. This operating point influences design and provisioning constraints for data centers as well as mobile devices. In addition, for some applications, e.g. scientific computing, near-peak use can be the norm.

Holistic and Balanced: A single component cannot accurately reflect the overall performance and power characteristics of a system. Therefore, the workload should exercise all core components and stress them roughly equally. The benchmark metrics should incorporate energy used by all core components.

Inclusive and Portable: We want to assess the energy efficiencies of a wide variety of systems: PDAs, laptops, desktops, servers, clusters, etc. Thus, the benchmark should include as many architectures as possible and be as unbiased as possible. It should allow innovations in hardware and software technology. Moreover, the workload should be implementable and meaningful across these platforms.

History-proof: In order to track improvements over generations of systems and identify future profitable directions, we want the benchmark specification to remain meaningful and comparable as technology evolves.

Representative and Simple: The benchmark should be representative of an important class of workloads on the systems tested. It should also be easy to set up, execute, and administer.

3.2 Workload

We begin with external sort, as specified in the previous sort benchmarks [16], as the workload because it covers most of our criteria. The task is to sort a file containing randomly permuted 100-byte records with 10-byte keys. The input file must be read from, and the output file written to, a non-volatile store, and all intermediate files must be deleted. The output file must be newly created; it cannot overwrite the input file.

This workload is *representative* because most platforms, from large to small, must manage an ever-increasing supply of data [23]. To do so, they all perform some type of I/O-centric tasks critical for their use. For example, large-scale websites run parallel analyzes over voluminous log data across thousands of machines [7]. Laptops and servers contain various kinds of filesystems and databases. Cell phones, PDAs, and cameras store, retrieve, and process multimedia data from flash memory.

With previous sort implementations on clusters, supercomputers, SMPs, and PCs [16] as evidence, we believe sort is *portable* and *inclusive*. It stresses I/O, memory, and the CPU, making it *holistic* and *balanced*. Moreover, the fastest sorts tend to run most components at near-peak utilization, so sort is not an idle-state benchmark. Finally, this workload is relatively *history-proof*. While the parameters have changed over time, the essential sorting task has been the same since the original DatamationSort benchmark [1] was proposed in 1985.

3.3 Metric

After choosing the workload, the next challenge is choosing the metric by which to evaluate and compare different systems. There are many ways to define a single metric that takes both power and performance into account. We list some alternatives that we rejected, describe why they are inappropriate, and choose the one most consistent with the criteria presented in Section 3.1.

3.3.1 Fixed energy budget

The most intuitive extension of MinuteSort and PennySort is to fix a budget for energy consumption, and then compare the number of records sorted by different systems while staying within that energy budget. This approach has two drawbacks. First, the power consumption of current platforms varies by several orders of magnitude: less than 1W for handhelds to over 1000W for servers, and much more for clusters or supercomputers. If the fixed energy budget is too small, larger configurations can only sort for a fraction of a second; if the energy budget is more appropriate to larger configurations, smaller configurations would run out of external storage. To be fair and inclusive, we would need multiple budgets and categories for different classes of systems.

Second and more importantly from a practical benchmarking perspective, finding the number of records to fit into an energy budget is a non-trivial task due to unavoidable measurement error. There are inaccuracies in synchronizing readings from a power meter to the actual runs and from the power meter itself (+/- 1.5% for the one we used). Since energy is the product of power and time, it is susceptible to variation in both quantities, so this choice is not *simple*.

3.3.2 Fixed time budget

Similar to the Minute- and Performance-Price sort, we can fix a time budget, e.g. one minute, within which the goal is to sort as many records as possible. The winners for the Minute and Performance-Price sorts are those with the minimum time and maximum SortedRecs/$, respectively. Similarly, our first proposal for JouleSort specified measuring energy and used SortedRecs/Joule as the ratio to maximize.

There are two problems with this approach, which are illustrated by Figure 2. This figure shows the SRecs/J ratio for varying input sizes (N) with our winning JouleSort system. We see that the ratio varies considerably with N. There are two distinct regions: $\leq 1.5 \times 10^7$ records which

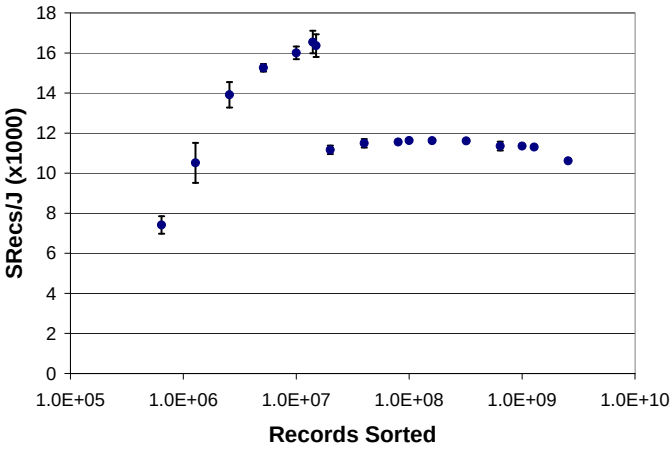

Figure 2: This figure shows the best measured energy efficiency of our 100GB winning system at varying input sizes.

corresponds to 1-pass sorts, and $> 1.5 \times 10^7$ records which corresponds to 2-pass sorts. To get the best performance for 2-pass sorts, we stripe the input and output across 6 disks using LVM2 and use 7 disks for temporary runs. For 1-pass sorts, we stripe the input and output across 10 disks. (see Section 5 for more system details). With a fixed-time budget approach, the goals of our benchmark can be undermined in the following ways for both one and two-pass sorts.

Sort progress incentive: First, in any time-budget approach there is no way to enforce continual progress. Systems will continue sorting only if the marginal cost of sorting an additional record is lower than the cost of sleeping for the remaining time. This tradeoff becomes problematic when an additional record moves the sort from 1-pass to 2-pass. In the 1-pass region of Figure 2, the sort is I/O limited, so it does not run twice as fast as a 2-pass sort. It goes fast enough, however, to provide about 40% better efficiency than 2-pass sorts. If the system was designed to have a sufficiently low sleep-state power (< 7W), then with a minute budget, the best approach would be to sort 1.5×10^7 records, which takes 10 sec, and sleep for the remaining 50 sec, resulting in a best 11800 SRecs/J. Thus, for some systems, a fixed time budget defaults into assessing efficiency when no work is done, violating our criteria.

Sort complexity: Second, even in the 2-pass region, total energy is a complex function of many performance factors that vary with N: total I/O, memory accesses, comparisons, CPU utilization, and effective parallelism. Figure 2 shows that once the sort becomes CPU-bound ($> 8 \times 10^7$ records), the SRecs/J ratio trends slowly downward because total energy increases superlinearly with N. The ratio for the largest sort is 9% lower than the peak. This decrease is, in part, because sorting work grows as $O(Nlg(N))$ due to comparisons, and the O-notation hides constants and lower-order overheads. This effect implies that the metric is *biased* toward systems that sort fewer records in the allotted time. That is, even if two fully-utilized systems A and B have same true energy efficiency, and A can sort twice as many records as B in a minute, the SortedRecs/Joule ratio will favor B. (Note: since this effect is small, our relative comparisons and conclusions in Section 2 remain valid.)

3.3.3 Our choice: fixed input size

The final option that we considered and settled upon was to fix the number of records sorted, as in the Terabyte Sort benchmark [16], and use total energy as the metric to minimize. For the same fairness issues as in the fixed-energy case, we decided to have three scales for the input size: 10^8, 10^9, and 10^{10} records, (similar to TPC-H) and declare winners in each category. (For consistency, henceforth, we use MB, GB, and TB for 10^6, 10^9 and 10^{12} bytes, respectively). For a fixed input size, minimum energy and maximum SortedRecs/Joule are equivalent metrics. In this paper, we prefer the latter because, like an automobile's mileage rating, it highlights energy efficiency more clearly.

This approach has advantages and drawbacks, but offers the best compromise given our criteria. These scales cover a large spectrum and naturally divide the systems into classes we expect: laptops, desktops, and servers. Moreover, since energy is a product of power and time, a fixed work approach is the simplest formulation that provides an incentive to optimize power-consumption and performance. Both are important concerns for current computer systems.

One disadvantage is that as technologies improve, scales must be added at the higher end and may need to be deprecated at the lower end. For example, if the performance of JouleSort winners improves at the rate of Moore's Law (1.6x/year), a system which sorts a 10GB in 100 sec. today would only take 10 sec. in 5 years. Once all relevant systems require only a few seconds for a scale, that scale becomes obsolete. Since even the best performing sorts are not improving with Moore's Law, we expect these scales to be relevant for at least 5 years. Finally, because comparison across scales is misleading, our approach is not fully history-proof.

Categories: As with the other sort benchmarks, we propose two categories for JouleSort: Daytona, for commercially supported sorts, and Indy, for "no-holds-barred" implementations. Since Daytona sorts are commercially supported, the hardware components must be off-the-shelf and unmodified, and run a commercially supported OS. As with the other sort benchmarks, we expect entrants to report the cost of the system.

3.4 Measuring Energy

There are a number of issues surrounding the proper accounting of energy-use. Specific proposals in the power-management community for measuring energy are being debated [33] and are still untested "in-the-large". Once these are agreed upon, we plan to adopt the relevant portions for this benchmark. As a start, we propose guidelines for three areas: the boundaries of the system to be measured, environmental constraints, and energy measurement.

System boundaries: Our aim is to account for all energy consumed to power the *physical system executing the sort*. All power is measured from the wall and includes any conversion losses from power supplies for both AC and DC systems. Power-supplies are a critical component in delivering power and, in the past, have been notoriously inefficient [3, 5]. Some DC systems, especially mobile devices, can run from batteries, and those batteries must eventually be recharged, which also incurs conversion loss. While the loss from recharging may be different from the loss from

System	CPU	Memory	Disk(s)	OS, FS
S1: DL360G3	Intel Xeon 2.8 GHz	2GB DDR	2xSCSI,15000rpm,36GB	Linux, XFS
S2: Blade	Transmeta Efficeon TM8000 1 GHz	256MB SDRAM	1xIDE,5400rpm,36GB	Windows 2000, NTFS
S3: NC6400	Intel Core 2 Duo T7200, 2GHz	3GB DDR2	1xSATA,7200rpm,60GB	Windows XP, NTFS

Table 2: The unbalanced systems measured in exploring energy-efficiency tradeoffs for sort.

the adapter that powers a device directly, for simplicity, we allow measurements that include only adapters.

All hardware components used to sort the input records from start to finish, idle or otherwise, must be included in the energy measurement. If some component is unused but cannot be powered-down or physically separated from adjacent participating components, then its power-use must be included. If there is any potential energy stored within the system, e.g. in batteries, the net change in potential energy must be no greater than zero Joules with 95% confidence, or it must be included within the energy measurement.

Environment: The energy costs of cooling are important, and cooling systems are variegated and operate at many levels. In a typical data center, there are air conditioners, blowers and recirculators to direct and move air among aisles, and heat sinks and fans to distribute and extract heat away from system components. Given recent trends in energy density, future systems may even have liquid cooling [28]. It is difficult to incorporate, anticipate, and enforce rules for all such costs in a system-level benchmark. For simplicity, we only include a part of this cost: one that is easily measurable and associated with the system being measured. We specify that a temperature between $20 - 25^o$ C should be maintained at the system's inlets, or within 1 foot of the system if no inlet exists. Energy used by devices physically attached to the sorting hardware that remove heat to maintain this temperature, e.g. fans, must be included.

Energy Use: Total energy is the product of average power over the sort's execution and wall-clock time. As with the other sort benchmarks, wall-clock time is measured using an external software timer. The easiest method to measure power for most systems will be to insert a digital power meter between the system and the wall. We intend to leverage the "minimum power-meter requirements" from the SPEC-Power draft [33]. In particular, the meter must report real power instead of apparent power since real power reflects the true energy consumed and charged for by utilities [22]. While we do not penalize for poor power factors, a power factor measured anytime during the sort run should be reported. Finally, since energy measurements are often noisy, a minimum of three consecutive energy readings must be reported. These will be averaged and the system with mean energy lower than all others (including previous years) with 95% confidence will be declared the winner.

3.5 Summary

In summary, the JouleSort benchmark is as follows:

- Sort a fixed number of randomly permuted 100-byte records with 10-byte keys.
- The sort must start with input in a file on non-volatile store and finish with output in a file on non-volatile store.
- There are three scale categories for JouleSort: 10^8 (10GB), 10^9 (100GB), and 10^{10} (1TB) records.
- The total true energy consumed by the entire physical system executing the sort, while maintaining an ambient temperature between 20-25^{o}C, should be reported.
- The winner in each category is the system with the maximum SortedRecs/Joule (i.e. minimum energy).

JouleSort is a reasonable choice among many possible options for an energy-oriented benchmark. It is an I/O-centric, system-level, energy-efficiency benchmark that incorporates performance, power, and some cooling costs. It is balanced, portable, representative, and simple. We can use it to compare different existing systems, to evaluate the energy-efficiency balance of components within a given system, and to evaluate different algorithms that use these components. These features allow us to chart past trends in energy efficiency, and hopefully will help predict future trends.

4. A LOOK AT DIFFERENT SYSTEMS

In this section, we measure the energy and performance of a sort workload on both unbalanced and balanced sorting systems. We analyze a variety of systems, from laptops to servers, that were readily available in our lab. For the unbalanced systems, the goal of these experiments is not to painstakingly tune these configurations. Rather, we present results to explore the system hardware space with respect to power-consumption and energy efficiency for sort. After looking at unbalanced systems, we present a balanced file-server that is our default 1TB winner. We use insights from these experiments to justify the approach for constructing our 100GB JouleSort winner (see Section 5).

4.1 Unbalanced Systems

Configurations: Table 2 shows the details of the "unbalanced" systems we evaluated, spanning a reasonable spectrum of power consumption in servers and personal computers. We include a server (S1), an older, low-power blade (S2), and an a modern laptop (S3). We chose the laptop because it is designed for whole-system energy-conservation, and S1 and S2 for comparison. We turned off the laptop display for these experiments. For S2, we only used 1 blade in an enclosure that holds 20, and, as per our rules, report the power of the entire system.

Sort Workload: We use Ordinal Technology's commercial NSort software which was the 2006 TeraByte sort Daytona winner. It uses asynchronous I/O to overlap reading, writing, and sorting operations. It performs both one and two-pass sorts. We tuned NSort's parameters to get the best performing sort for each platform. Unless otherwise stated, we use the radix in-memory sort option.

	Recs x10^7	Power(W)	Time (s)	SRecs/J	CPU util
S1	5	139.3 ±0.1	299.4 ±2.5	1206 ±10	25%
S1	10	138.5 ±0.1	596.9 ±0.6	1203 ±1	26%
S2	5	90.0 ±1.0	1847 ±52	300 ±10	11%
S3	5	21.0 ±1.0	727.5 ±28	3270 ±120	1%
S3	10	21.7 ±1.0	1323 ±48	3479 ±131	1%

Table 3: Energy efficiency of unbalanced systems.

Power Measurement: To measure the full-system AC power consumption, we used a digital power meter interposed between the system and the wall outlet. We sampled this power at a rate of once per second. The meter used was Brand Electronics Model 20-1850CI which reports true power with ±1.5% accuracy. In this paper, we always report the average power over several trials and the standard deviation in the average power.

4.1.1 Results

The JouleSort results for our unbalanced systems are shown in Table 3. Since disk space on these systems was limited, we chose to run the benchmark at 10GB and a smaller 5GB dataset to allow fair comparison. We see that S1 (the server) is the fastest, but S3 (the laptop) is most energy-efficient. System S1 uses over 6.6x more power than S3, but only provides 2.2x better performance. Although S1's disks can provide more sequential bandwidth, S1 was limited by its SmartArray 5I I/O controller to 33 MB/s in each pass. System S2 (the blade) is not as bad as the results show because blade enclosures are most efficient only when fully populated. The enclosure's power without any blades was 66W. When we subtract this from the S2's total power, we get an upper bound of 1121 ±144 SRecs/J for S2. For all these systems, the standard deviation of total power during sort was at most 10%. The power factor (PF) for S1, S2, and S3 were 1.0, 0.92, and 0.55 respectively.

The CPUs for all three systems were highly underutilized. In particular, S3 attains an energy-efficiency similar to that of last year's estimated winner, GPUTeraSort, by barely using its cores. Since the CPU is usually the highest power component, these results suggest that building a system with more I/O to complement the available processing capacity should provide better energy efficiencies.

4.2 Balanced Server

In this section, we present a balanced system that usually functions as a fileserver in our lab. Table 4 shows the components used during the sort and coarse breakdowns of total system power. The main system is an HP Proliant DL360 G5 that includes a motherboard, CPU, low-power laptop disk, and a high-throughput SAS I/O controller. For the storage, we use two disk trays, one that holds the input and output files and the other which holds the temp disks. Each tray has 6 disks and can hold a maximum of 12. The disk trays and main system all have dual power-supplies, but for these experiments, we powered them through one each. For all our experiments, the system has 64-bit Ubuntu Linux 2.6.17-10 and the XFS filesystem installed.

Table 4 shows that for a server of this kind, the disks and their enclosures consume roughly the same power as the rest of the system. When a tray is fully populated with 12 disks, the idle power is 145 W and with 6 disks the idle power is 101 W. There clearly are inefficiencies when the tray is under-utilized. To estimate the power of the 2GB DIMMs, we added two 1GB DIMMs and measured the system power with and without the 2GB DIMMs. We found that the 2GB DIMMs use 7.5W both during sort and at idle.

Comp	Model	Idle Power	Sort Power
CPU	Intel Xeon 5130 2GHz	65 W (TDP)	
Memory	2x2GB PC2-5300	7.5±0.5W (each)	
OS disk	Fujitsu, SATA, 5400rpm, 60GB MHV2060BS	n/a	
I/O Ctrl	LSI Logic SAS HBA 3801E	n/a	
Mother-board	HP Proliant DL360G5	n/a	
All of above		168±1W	181±1 W
Input / output tray	HP MSA60 6 x Seagate Barracuda ES, SATA, 7200rpm, 500GB	101±1W	111±1W
Temp tray	HP MSA60 (same as above)	101±1W	113±1W

Table 4: A balanced fileserver.

For this system, we found the most energy-efficient configuration by experimenting with a 10GB dataset. By varying the number of disks used, we found that, even with the inefficiencies, the best performing 10GB setup uses 12 disks split across two trays. This effect happens because the I/O controller offers better bandwidth when data is shipped across its two channels. A 10GB sort provides 313±1MB/s on average for each phase across the trays while only 212±1MB/s when the all disks are within a tray. The average power of the system with only one tray is 347±1W and with two trays is 406±1W. As a result, with two trays the system attains a best 3863±19 SRecs/J instead of 3038±22 SRecs/J with one tray.

The 2-tray, 12-disk setup is also when the sort becomes CPU-bound. When we reduce the system to 10 disks, the I/O performance and CPU utilization drop, and when we increase the system to 14 disks, the performance and utilization remain the same. In both cases, total energy is higher than the 12-disk point, so this balanced, CPU-bound configuration is also the most energy-efficient.

Table 6 shows the performance and energy characteristics of the 12-disk setup for 1TB sorts. This system takes nearly 3x more power than S1, but provides over 8x the throughput. This system's SRecs/J ratio beats the laptop and last year's estimated winner, even with a larger 1TB input. Experiments similar to those for the 10GB dataset show that this setup provides just enough I/O to keep the two cores fully utilized on both passes and uses the minimum energy for the 1TB scale. Thus, at all scales, the most energy-efficient and best-performing configuration for this system is when sort is CPU-bound and balanced.

Comp	Model	Price ($)	Power
CPU	Intel Core 2 Duo T7600	639.99	34W (TDP)
Motherboard	Asus N4L-VM DH	108.99	n/a
Case/PSU	APEVIA X-Navigator ATXA9N-BK/500	94.99	n/a
8-disk ctrl	HighPoint Rocket RAID 2320	249.99	9.5W
4-disk ctrl	HighPoint Rocket RAID 2300	119.99	2.0W
Memory (2)	Kingston 1GB DDR2 667	63.99	1.9W (spec)
Disk (13)	Hitachi TravelStar 5K160 5400 rpm, 160 GB	119.99	A:1.8W I:0.85W (spec)
Adapters		130.25	

Table 5: Winning 100GB system.

4.3 Summary

In conclusion, from experimenting with these systems we learned (1) CPU is wasted in unbalanced systems (2) the most energy-efficient server configuration is when the system is CPU-bound (3) an unbalanced laptop is almost as energy-efficient as a balanced server. Moreover, current laptop drives use 5x (2 vs. 10 W) less power than our server's SATA drives while offering around 0.5x (40 vs. 80 MB/s) the bandwidth. These observations suggest a reasonable approach for building the most energy-efficient 100GB sorting system is to use mobile-class CPUs and disks and connect them via a high-speed I/O interconnect.

5. 100GB JOULESORT WINNER

In this section, we first describe our winning JouleSort configuration and report its performance. We then study this system through experiments that elucidate power and performance characteristics of this system.

5.1 Winning Configuration

Given limited time and budget, our goal was to convincingly overtake the previous estimated winner rather than to try numerous combinations and construct an absolute optimal system. As as result, we decided to build a Daytona system and solely use NSort as the software. Our design strategy for an energy-efficient sort was to build a balanced sorting system out of low-power components. After estimating the sorting efficiency of potential systems among a limited combination of modern, low-power, x86 processors and laptop disks, we assembled the configuration in Table 5.

This system uses a modern, low-power CPU with 5 frequency states, and a TDP of 34W for the highest state. We use a motherboard that supports both a mobile CPU and multiple disk controllers to keep the cores busy. Few such boards exist because they target a niche market; this one includes two PCI-e slots: one 1-channel and one 16-channel. To fill those slots, we use controllers that hold 4 and 8 SATA drives, respectively. Finally, our configuration uses low-power, laptop drives which support the SATA interface. They offer an average 11 ms seek time, and their measured sequential bandwidth through XFS is around 45 MB/s. Hitachi's specs list an average 1.8W for read and write and 0.85W for active idle. We use two DIMMs whose specs report 1.9W for each. Finally, the case comes with a 500W power supply.

Our optimal configuration uses 13 disks because the PCI-e cards hold 12-disks maximum and the I/O performance of the motherboard controller with more than 1 disk is poor. The input and output files are striped across a 6-disk array configured via LVM2, and the remaining 7 disks are independent for the temporary runs. For all experiments, we use Linux kernel 2.6.18 and the XFS filesystem unless otherwise stated. In the idle state at the lowest CPU frequency, we measured 59.0 $\pm$1.3 W for this system.

Table 6 shows the performance of the system, which attains 11300 SRecs/J when averaged over 3 consecutive runs. The pure-performance statistics are reported by NSort. We configure it to use radix sort as its in-memory sort algorithm and use transfer sizes of 4MB for the input-output array and 2MB for the temporary storage. Our system is 24% faster than GPUTeraSort and consumes an estimated 3x less power. The power use during sort is 69% more than idle. In the output pass, the CPU is underutilized (see Table 6; max 200% for 2 cores), and the bandwidth is lower than in the input pass because the output pass requires random I/Os. We pin the CPU to 1660 MHz, which Section 5.3 shows is the most energy-efficient frequency for the sort.

5.2 Varying System Size

In these experiments, we vary the system size (disks and controllers) and observe our system's pure performance, cost efficiency, and energy efficiency. We investigate these metrics using a 5GB dataset. For the first two metrics, we set the CPU to its highest frequency, and report the metrics for the most cost-effective and best performing configurations at each step. We start with 2 disks attached to the cheaper 4-disk controller, and at each step use the minimum-cost hardware to support an additional disk. Thus, we switch to the 8-disk controller for configurations with 5-8 disks, and use both controllers combined for 9-12 disks. Finally, we add a disk directly to the motherboard for the 13-disk configuration.

Figure 3 shows the performance (records/sec) and cost efficiency with increasing system size. The 13-disk configuration is both the best performing and most cost-efficient point. Each additional disk on average increases system cost by about 7% and improves performance by 14% on average. These marginal changes vary; they are larger for small system size and smaller for larger system sizes. The 5-disk point drops in cost efficiency because it includes the expensive 8-disk controller without a commensurate performance increase. Although the motherboard and controllers limit the system to 13 disks, we speculate that additional disks would not help since the first pass of the sort is CPU-bound.

Next, we look at how energy efficiency varies with with system size. At each step, we add the minimum-energy hardware to support the added disk and report the most energy-efficient setup. We set the CPU frequency to 1660MHz at all points to get the best energy efficiency (see Section 5.3). For convenience, we had one extra OS disk on the motherboard from which we boot and which was unused in the sort for all but the last point. The power measurements include this disk, but this power is negligible at idle ($<$ 1W).

System	Recs	**SRecs/J**	Energy (kJ)	Power (W)	Time (sec)	BW (in,out,total) (MB/s)	CPU util. (200 max)	PF
Table 5	10^8	**11628 ±41**	8.6 ±0.03	99.3 ±0.2	86.6 ±0.4	(248±3, 222±1, 115±1)	139±1%	
(low-power)	10^9	**11354 ±29**	88.1 ±0.23	100.0 ±0.1	880.8 ±1.5	(238±0.1, 219±0.4, 114±0.2)	154±0%	0.65
Table 4 (server)	10^{10}	**3425 ±40**	2920 ±0.34	406±1	7196 ±67	(274±0.4, 282±5, 139±1)	179±2%	0.96

Table 6: Performance of winning JouleSort systems.

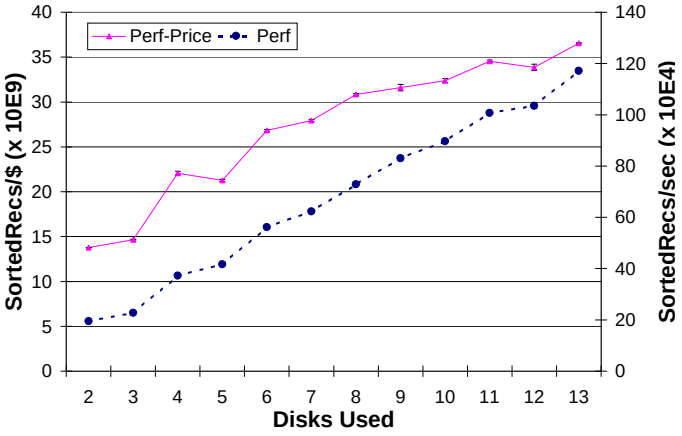

Figure 3: Shows how performance-price and performance varies with system size.

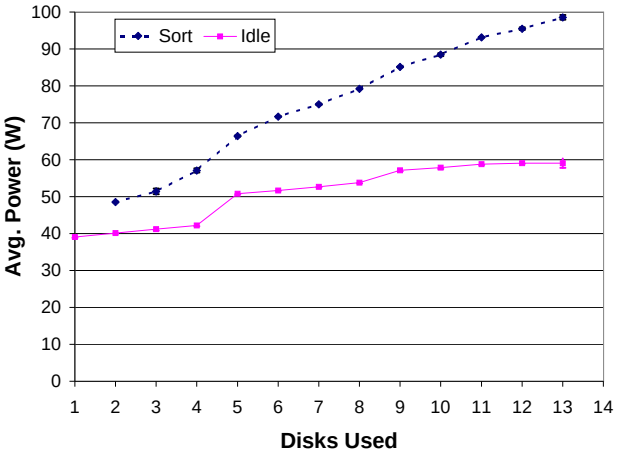

Figure 4: Shows how power varies with system size.

Figure 4 shows idle power at the lowest frequency state versus average power during sort at 1660 Mhz for the same system configurations. With the system at idle and only the motherboard disk installed, our measurements show that the 8-disk controller uses 9.5W and the 4-disk one uses 2.0W. Thus, for points between 2-4 disks, we use only the 4-disk controller, between 5-8 we use the only 8-disk controller, and for 9 or more, we use both. Figure 4 shows jumps at these transitions. The idle line indicates adding a disk increases power by 0.95±0.1W. During sorting, adding a disk increases total power on average 4.3±1.0W at sizes fewer than 8-disks and 3.4W±1.0W on average for more than 9-disks. These increases reflect end-to-end utilization of the CPU, disk, controllers, etc.

Figure 5 shows the energy-efficiency with increasing number of disks used in the sort. The curve is similar to the price-performance curve in Figure 3. The average increase

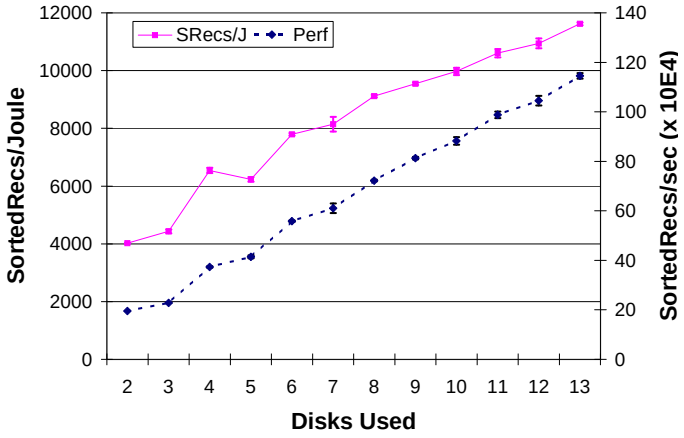

Figure 5: Shows how energy efficiency varies with system size.

in energy at each step is 6% while the average increase in performance is about 14%. The 5-disk point again is a local minimum because it incurs the power of the larger controller without enough disks to take advantage of it. The sort is CPU-bound in the most energy-efficient configuration.

There are two main points to take away from these experiments. First, the similar shapes of these curves reflect that the base dollar and energy costs of the system are high compared to the marginal dollar and energy cost of disks. If we used server-class disks that are similar in cost but consume 5x the power of mobile disks, we would see different cost and energy efficiency curves. Second, for the components we chose, the best performing, most cost-efficient, and most energy-efficient configurations are identical modulo the CPU frequency. Moreover, in this best configuration, the system is balanced with just enough I/O bandwith to keep the CPU fully utilized for the first pass.

5.3 Software Matters

Next, we vary the filesystem and in-memory sort algorithm to see how they affect energy efficiency. The winning 100GB configuration uses the XFS filesystem and a radix sort. Figure 6 examines the effect of changing the filesystem to ReiserFS and the sort algorithm to merge sort at different CPU frequencies for a 10GB dataset.

As expected, power consumption steadily increases with frequency in all cases. The power consumptions of XFS with radix sort and merge sort are similar at all frequencies. ReiserFS, however, consumes less power and also is less energy-efficient. All three configurations show improved energy efficiency from 996 MHz to 1660 MHz, and then level off or decrease. This result indicates that the sorts are CPU-bound at the lower frequencies. ReiserFS shows a 26% improvement in performance between the lowest and highest

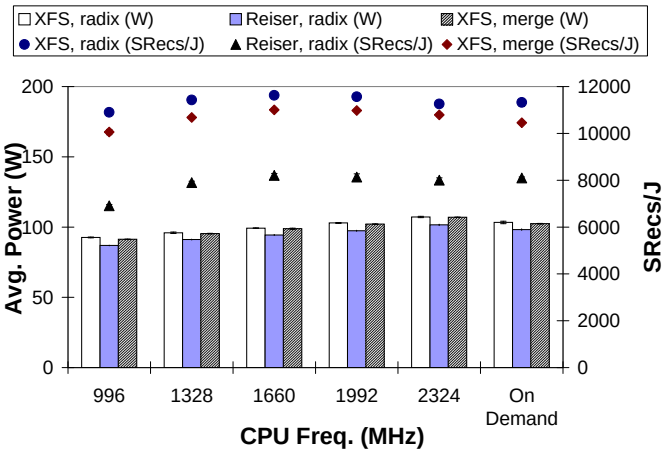

Figure 6: Shows how average power and energy efficiency vary with CPU frequency for a 10GB sort.

frequencies, while XFS radix improves only 16% and XFS merge improves only by 20%.

ReiserFS has worse energy efficiency mainly because it provides less sequential bandwidth, and thus worse performance, than XFS. Although we tuned each configuration, this result may be an artifact of our setup and not an inherent flaw of ReiserFS. Similarly, the merge sort also gives worse energy efficiency than radix entirely because its performance is worse.

The graph also shows the power and energy efficiency of the Linux on-demand CPU frequency scaling policy, which is within 10% of the lowest execution time and 15% of the lowest power for all three configurations. For ReiserFS, the on-demand policy offers the same efficiency as the best configuration. In summary, these experiments show that the algorithms and underlying software used for sort affect energy efficiency mainly through performance.

5.4 Approximate CPU vs. I/O Breakdown

We performed some micro-benchmarks exercising the I/O subsystem (disks plus controllers) and the CPU subsystem (CPU plus memory) separately to determine how much each contributes to the increase in power during sort. The system at the 12-disk point consumes 33.5W more during sort than when the system is at idle with 1660 MHz CPU frequency. This increase is nearly 35% of the total power during sort. Our benchmarks suggest that the I/O subsystem consumes a much greater fraction of this power increase than the CPU subsystem.

We first performed a test to help approximate the contribution from the I/O subsystem. In this test, we copied data from a 6-disk array to another 6-disk array, which had the same average disk bandwidth as the 12-disk sort. We found that the system power increased by 30.5W. During this test, the CPU utilization was 66% (out of 200% max for 2 cores). Next, we performed an experiment to approximate the contribution from the CPU subsystem. We put a small input file on a ram-disk and repeatedly sorted it. This test pegged the CPU to 200% utilization and the power increase was 15.7W.

Using the above values and assuming that CPU subsystem power increases linearly with CPU utilization, we estimate its contribution during the 12-disk sort as follows. During this sort, CPU utilization was an average 118% (59% per core), so we assign $0.59 \times 15.7 = 9.2$W to the CPU subsystem. Similarly, we discount the copying test for its CPU utilization and estimate that the I/O subsystem uses $30.5 - 0.33 \times 15.7 = 25.3$W. These estimates combine to 34.5W and almost match (3% error) the 33.5W measured increase due to the 12-disk sort. Thus, our tests imply that about 75% of the power increase during sort is from the I/O subsystem and 25% from the CPU subsystem. We found similar proportions at smaller system sizes.

5.5 Vary DIMMs and Power Supply

Since NSort uses only a fraction of the available memory for these experiments, we ran experiments with only 1 DIMM. Power use and execution time were statistically indistinguishable from the 2 DIMM case during sort. Power use is also within measurement error at idle.

We replaced the 500W power supply with a 145W one and found that the power-consumption during sort and idle increased by 2W. This suggests that at 68% load or less, efficiencies of the two power-supplies are similar.

Note, the power factors for the laptops and desktop systems in this paper, including our 100GB winner, are well below 1.0. Low power factors are problematic in data centers because power delivery mechanisms need to be overprovisioned to carry additional current for loads with low power factors [22]. Utilities often charge extra for this provisioning. Similar to server-class systems, power supplies will need to provide power-factor correction for systems like our 100GB winner to become a reality in data centers.

5.6 Summary

We describe the Daytona 100GB JouleSort system that is over 3.5x as energy efficient as last year's PennySort winner, the GPUTeraSort. For this system, we show the most energy-efficient sorting configuration is when the sort is CPU-bound and balanced. This configuration is also the best performing and most cost-efficient. It will be interesting to see how long this relationship holds. We see that filesystem and in-memory sort choice mainly affect energy efficiency through performance rather than power for this system.

In this paper, we focused on building a balanced system with low-power off-the-shelf components targeted for the 100GB scale. Unfortunately, because of hardware limitations and market availability, we could not easily scale this system to the 1TB category. In the future, we expect systems in other classes to win the 10GB and 1TB categories, but for completeness we report in Table 6 the best configurations we encountered for those categories.

6. RELATED WORK

Our related work falls into three categories. We first discuss the history of sort benchmarks and large-scale sorting techniques. Next, we cover the previous work on metrics for evaluating energy efficiency. Finally, we briefly discuss work on techniques for reducing energy consumption in systems.

6.1 Sort Benchmarks and Techniques

The original Datamation sort benchmark was a pure performance benchmark that measured the time to sort a million records [1]. In 1994, the developers of AlphaSort [25] recognized that the benchmark was losing its relevance, because startup and shutdown would eventually dominate the

time to sort such a small number of records. They therefore proposed two variants, MinuteSort and PennySort, hoping they would remain relevant as technology improved at the pace of Moore's Law. Recognizing that PennySort was biased against large configurations by allowing too small a time budget, researchers then proposed the performance/price sort, which is tied to the MinuteSort [17] time budget. The time budget approach undermines the goals of JouleSort.

Since the original Datamation sort benchmark, there have been many different implementations of external sort on a variety of platforms from desktops to supercomputers. The Sort Benchmark website [16], maintained by Jim Gray, lists the winners and briefly surveys past trends.

6.2 Energy Benchmarks

Several different metrics have been proposed for evaluating the energy efficiency of computer systems. In 1996, Gonzalez *et al.* [13] proposed the energy-delay product as the metric of energy-efficient microprocessor design. Alternatively, the metric of performance per Watt is also widely used to evaluate processors' energy efficiency [21]. This metric emphasizes performance less than the energy-delay product, which is equivalent to performance squared per Watt.

Energy-efficiency metrics tailored to data centers have also been proposed. Sun's Space, Watts, and Performance metric (SWaP) [34] considers the rack space taken up by a hardware configuration along with its power and performance, in an effort to promote data center compaction. Metrics based on exergy [27], which is the energy converted into less efficient forms such as heat, take into account every aspect of the data center from processors to the cooling infrastructure. However, these metrics are not applicable to the entire range of systems we want to evaluate with JouleSort.

Comparatively little work has been done on workloads for energy-efficiency benchmarks. In the embedded domain, the EEMBC EnergyBench benchmarks [9] provide a physical infrastructure to evaluate a single processor's energy efficiency on any of EEMBC's existing mobile benchmark suites. In the enterprise domain, the SPEC Power and Performance Committee [33] is currently developing an energy benchmark suite for servers, and the United States Environmental Protection Agency's EnergyStar program is developing a way to rate the energy efficiency of servers and data centers [35].

6.3 Energy Efficiency

There is a large body of prior work on energy efficiency. For example, at the component and system levels, many studies have been devoted to algorithms for dynamically exploiting different power states in processors [4, 14, 36], memory [10], and disks [8] in order to promote energy efficiency. In clusters and data centers, research has focused on energy-efficient workload distribution and power budgeting (e.g. [6, 11, 30, 32]). Other studies have focused at the application level, including energy-aware user interfaces [31] and fidelity-aware energy management [12].

7. CONCLUSIONS

In this section, we summarize the limitations of JouleSort, speculate on future energy-efficient systems, and wrap up.

7.1 Limitations

JouleSort does not address all possible energy-related concerns. Since JouleSort focuses on data management tasks, it misses some important energy-relevant components for multimedia applications. JouleSort omits displays, which are an important component of total power for mobile devices. GPUs also consume significant power and are ubiquitous in desktop systems. Although we can use GPUs to sort, our benchmark does not require their use. As a result, it loses relevance for applications where these components are essential.

There are other energy-related concerns in data centers beyond system power that were difficult to incorporate. At a high level, cooling requires (1) lowering ambient temperature and (2) extracting heat away from systems. JouleSort accounts only for part of the second. Delivering power to systems incurs losses at the rack and data-center level which are ignored in JouleSort. Moreover, many systems are used as an ensemble [32] in data centers, with sophisticated scheduling techniques to trade performance for lower energy among systems rather than at the component level. As a system-level benchmark, JouleSort may not identify the benefits of such methods.

7.2 "Greener" Systems

We speculate on two emerging technologies that may improve the energy efficiency of systems. For the 10GB scale, flash memory appears to be a promising storage technology driven by the mobile device market [18]. Per byte, it is about 4x cheaper than DRAM and provides sequential read and write bandwidth close to that of disks. More importantly, random read I/Os with flash are 100x faster than disk, and flash consumes 10x less power than disks. The random-reads allow interesting modifications to traditional 2-pass sorting algorithms. To date, the largest cards at reasonable cost are 8GB. We anticipate a system such as a laptop or low-power embedded device that can leverage multiple flash devices as the next 10GB winner.

For the larger scales, an intriguing option is a hybrid system using a low-power CPU, laptop disks, and a GPU. The GPUTeraSort has shown that GPUs can provide much better in-memory sorting bandwidth than CPUs [15]. Using a motherboard that supports more I/O controllers and a GPU, we could scale our system to use more disks. An interesting question is whether these performance benefits might be offset by the recent trend in GPUs to consume more power.

7.3 Closing

This paper proposes JouleSort, a simple, balanced energy-efficiency benchmark. We present a complete benchmark: a workload, metric, and guidelines, and justify our choices. We also present a 100GB winner that is over 3.5x as efficient as last year's estimated winner. Today, this system is hard to find pre-assembled. It consists of a commodity mobile-class CPU and 13 laptop disks connected through server-class PCI-e I/O cards.

The details of JouleSort already have undergone significant changes since its inception. Since JouleSort has not yet been tried "in the wild", we fully expect further revisions and fine-tuning to keep it fair and relevant. Nevertheless, we look forward to its use in guiding energy-efficiency optimizations in future systems.

Acknowledgments

We dedicate this work to Jim Gray. He inspired us with his enthusiasm, support, and insightful comments regarding

benchmark design. We thank Jacob Leverich, Eric Anderson, Chris Nyberg, Hernan Laffitte, and Dimitris Economou for their help in setting up, tuning, and optimizing our JouleSort configurations. We thank Naga Govindaraju, Chris Reummler, and the anonymous reviewers for their feedback that drastically improved the quality of this work. We thank John Sontag and Alistair Veitch for equipment and support. We thank all those at the Berkeley RAD Lab retreat whose suggestions helped us refine the current version of the benchmark.

8. REFERENCES

[1] Anonymous et al. A measure of transaction processing performance. In *Datamation*, pages 112–118, Apr. 1985.

[2] L. Barroso. The price of performance. *ACM Queue*, 3(7), Sept. 2005.

[3] P. Bose. Keynote address: Power-efficient microarchitectural choices at the early definition stage. In *PACS*, 2003.

[4] D. Brooks and M. Martonosi. Dynamic thermal management for high-performance microprocessors. In *HPCA*, 2001.

[5] C. Calwell and T. Reeder. Power supplies: A hidden opportunity for energy savings. Online, May 2002. http://www.ecosconsulting.com/resources_publications.html#PowerSupply.

[6] J. Chase, D. Anderson, et al. Managing energy and server resources in hosting centers. In *SOSP*, 2001.

[7] J. Dean and S. Ghemawat. MapReduce: Simplified data processing on large clusters. In *OSDI*, Dec. 2004.

[8] F. Douglis, P. Krishnan, and B. Marsh. Thwarting the power-hungry disk. In *USENIX*, 1994.

[9] Embedded Microprocessor Benchmark Consortium (EEMBC). EnergyBench benchmark software. Online. http://www.eembc.org/benchmark/power_sl.asp.

[10] X. Fan, C. Ellis, and A. Lebeck. Memory controller policies for DRAM power management. In *Low-Power Systems and Design (ISLPED)*, 2001.

[11] W. Felter, K. Rajamani, et al. A performance-conserving approach for reducing peak power consumption in server systems. In *International Conference on Supercomputing*, 2005.

[12] J. Flinn and M. Satyanarayanan. Energy-aware adaptation for mobile applications. In *SOSP*, 1999.

[13] R. Gonzalez and M. Horowitz. Energy dissipation in general purpose microprocessors. *IEEE Journal of Solid-State Circuits*, 31(9):1277–1284, 1996.

[14] K. Govil, E. Chen, and H. Wasserman. Comparing algorithms for dynamic speed-setting of a low-power CPU. In *MobiCom*, 1995.

[15] N. K. Govindaraju, J. Gray, R. Kumar, and D. Manocha. GPUTeraSort: High performance graphics coprocessor sorting for large database management. In *SIGMOD*, June 2006.

[16] J. Gray. Sort benchmark home page, Oct. 2006. http://research.microsoft.com/barc/SortBenchmark.

[17] J. Gray, J. Coates, and C. Nyberg. Performance / Price Sort and PennySort. Technical Report MS-TR-98-45, Microsoft, Aug. 1998.

[18] J. Gray and B. Fitzgerald. Flash disk opportunity for server-applications. Online, 2007. http://www.microsoft.com/ gray.

[19] HP enterprise configurator power calculators, Oct. 2006. http://h30099.www3.hp.com/configurator/powercalcs.asp.

[20] C. Jones, K. Sivalingam, et al. A survey of energy-efficient network protocols for wireless networks. *Wireless Networks*, 7(4):354–358, July 2001.

[21] J. Laudon. Performance/Watt: The new server focus. *SIGARCH Computer Architecture News*, 33(4):5–13, Nov. 2005.

[22] M. R. Lindeburg. *Mechanical Engineering Reference Manual*. Professional Publications, Tenth edition, 1997.

[23] P. Lyman and H. R. Varian. How much information? Online, 2003. http://www2.sims.berkeley.edu/research/projects/how-much-info-2003/.

[24] J. Markoff and S. Hansell. Hiding in plain sight, Google seeks an expansion of power. *New York Times*. June 14, 2006.

[25] C. Nyberg, T. Barclay, Z. Cvetanovic, J. Gray, and D. Lomet. Alphasort: A cache-sensitive parallel external sort. *VLDB Journal*, 4(4):603–627, 1995.

[26] C. Nyberg and C. Koester. Ordinal Technology - NSort Home Page. Online, 2007. http://www.ordinal.com/.

[27] C. D. Patel. A vision of energy aware computing from chips to data centers. In *Micro-Mechanical Engineering (ISMME)*, Dec. 2003.

[28] C. D. Patel and P. Ranganathan. Enterprise power and cooling. ASPLOS Tutorial, Oct. 2006.

[29] P. Pillai and K. G. Shin. Real-time dynamic voltage scaling for low-power embedded operating systems. In *SOSP*, pages 89–102, 2001.

[30] E. Pinheiro, R. Bianchini, et al. Load balancing and unbalancing for power and performance in cluster-based systems. In *Workshop on Compilers and Operating Systems for Low Power (COLP)*, 2001.

[31] P. Ranganathan, E. Geelhoed, et al. Energy-aware user interfaces and energy-adaptive displays. *IEEE Computer*, 39(3):31–38, March 2006.

[32] P. Ranganathan, P. Leech, et al. Ensemble-level power management for dense blade servers. In *ISCA*, 2006.

[33] Standard Performance Evaluation Corporation (SPEC). SPEC power and performance committee. Online. http://www.spec.org/specpower/.

[34] Sun Microsystems. SWaP (space, watts and performance) metric. Online. http://www.sun.com/servers/coolthreads/swap/.

[35] United States Environmental Protection Agency (EPA). Enterprise server and data center efficiency initiatives. Online. http://www.energystar.gov/index.cfm?c=products.pr_servers_datacenters.

[36] A. Weissel and F. Bellosa. Process cruise control: event-driven clock scaling for dynamic power management. In *Compilers, Architecture, and Synthesis for Embedded Systems (CASES)*, 2002.