Bulletin of the Technical Committee on

Data Engineering

March 2011 Vol. 34 No. 1

IEEE Computer Society

Letters

Special Issue on Energy Aware Big Data Processing

Conference and Journal Notices

The TC on Data Engineering

Membership in the TC on Data Engineering is open to all current members of the IEEE Computer Society who are interested in database systems. The TC on Data Engineering web page is http://tab.computer.org/tcde/index.html.

The Data Engineering Bulletin

The Bulletin of the Technical Committee on Data Engineering is published quarterly and is distributed to all TC members. Its scope includes the design, implementation, modelling, theory and application of database systems and their technology.

Letters, conference information, and news should be sent to the Editor-in-Chief. Papers for each issue are solicited by and should be sent to the Associate Editor responsible for the issue.

Opinions expressed in contributions are those of the authors and do not necessarily reflect the positions of the TC on Data Engineering, the IEEE Computer Society, or the authors' organizations.

The Data Engineering Bulletin web site is at http://tab.computer.org/tcde/bull_about.html.

Letter from the Editor-in-Chief

TCDE Activities

Those of you who read my Bulletin letters know that I was recently elected as Chair of the Technical Committee on Data Engineering (TCDE). One of my first responsibilities is to form an Executive committee. My overall approach in this has been to seek a combination of new members plus experienced members, to try to ensure that all geographic regions are represented, and to maintain a strong connection with the ICDE Steering Committee. The end result is that the new Executive Committee has a mix of new and prior members, and retains ties to the Steering Committee. The people I am appointing all have been involved with the TCDE, mostly via ICDE involvement. They are Calton Pu (Vice Chair), Thomas Risse (Secretary/Treasurer), Malu Castelanos, Paul Larson, Erich Neuhold, and Kyu-Young Whang (members), Guy Lohman (Chair DEW: Self Managing Database Sys.), and Christian Jensen (SIGMOD Liason). I want to thank Calton Pu and Karl Aberer for having served on the Executive Committee. Their thoughtful and careful examination of issues sets a high standard for the new committee.

The Current Issue

How does our industry become "green". While there are many aspects to being "green", in the immortal words of Kermit the Frog, "it is hard being green". The high tech industry has, for many years now, employed some very good practices to enable resources to be recycled or, at a minimum, disposed of safely. For example, I regularly mail back to the printer manufacturer my used printer cartridges. The industry has also paid attention to energy use in terms of battery charge duration, where how many hours you can use your laptop or your cell phone without a power cord directly impacts the usability of the device. But the study of and minimization of energy use will only grow in importance. It may be "hard being green" but it will be unavoidable before long.

While it is not true that energy consumption of desktop and server machines has been ignored, it is true that this has not been a major consideration, as power from the electrical grid was on hand to provide as much as was needed. But times are changing. Our industry is such a large part of the world economy now, that how effectively our industry uses power now shows up on charts describing the entire economy.

The current issue focuses on energy management for "Big Data Processing". One of the remarkable artifacts of our age is the number and size of huge data centers, and their significant power consumption. Energy consumption is not just a "green" aspiration, it is a "nuts and bolts" economic issue. Those companies building large data centers, especially those who intend to compete in the cloud services market, now see energy cost and consumption as a major factor. Further, the electrical grid will not become "smart" without the application of data intensive analysis to the problem.

The current issue draws on work and research done by both university and industrial technologists. This is very much in the Bulletin tradition of bringing these parts of the database community together for information interchange. Brian Cooper has assembled an issue that accomplishes just that, providing a broad perspective on this area. I want to thank Brian for his fine job of handling this issue. Even now, a focus on energy is not mainstream. But this issue is a good start on raising awareness of the technical challenges in this area.

David Lomet
Microsoft Corporation

Letter from the Special Issue Editor

One thing that I have always found fascinating about computer science is how a relatively technical discipline, with arcana not understood by the majority of the world's population, can relatively quickly produce inventions that impact the lives of large portions of that population. Big data processing is an example - it was big data processing, among other things, that helped the internet grow from a technical curiousity to a daily essential part of billions of people's lives. Big data processing enables the planning and tracking of the movement of products around the globe, manages the financial records of the world's economy, and impacts the world in myriad other ways.

However, we have begun to see that big data processing has the potential to impact the world in another, less desirable way. Its insatiable desire for more CPU cycles, more disk accesses, and more network bits means that ever-increasing energy resources are needed to power and cool the massive arrays of servers that actually do the work. This appetite for energy has driven the creation of new types of datacenters, server hardware and cooling technologies. But more work is needed to control energy usage that continues to grow.

Luckily for computer science researchers, there is an opportunity to do interesting research on this problem while helping the world as a whole. In this issue, we highlight four efforts that illustrate the challenges, and the opportunities, for energy-aware big data processing.

- The real electric grid is not just a smooth, unending stream of electrons, and renewable energy sources in particular provide varying amounts of power based on the current wind or solar radiation. Krioukov et al explore how batched analytic workloads are well suited to become "supply-following" - doing more work when clean renewable energy is available, and drawing less power when the only supplies are from dirty, non-renewable sources.

- Often, computing query results "as fast as possible" is not strictly required. Lang, Kandhan and Patel suggest that query processors can pick the most energy efficient plan that is still "fast enough" to meet SLA requirements.

- Recent batched data architectures, such as Dryad or MapReduce, are explicitly designed to use whatever resources are available. In reality, however, different jobs have different needs for CPU, disk and memory. Xiong and Kansal describe how batch systems can schedule work at a fine grain to use only necessary server resources, allowing unneeded resources to be used for other jobs or not to be used at all.

- Of course, most efforts to reduce energy usage are moot if we cannot effectively measure the usage! While TPC is popular for measuring performance, energy aware versions of TPC can be used to measure consumption. Poess and Nambiar describe techniques to analyze TPC results and estimate the associated power usage.

The first three articles share a common motivation of adapting query scheduling and planning to minimize usage of dirty, wasteful energy. The fourth article demonstrates how mature, well-used benchmarking techniques can be extended to evaluate the effectiveness of these query scheduling and planning techniques. It is my hope that these works, in addition to standing in their own right, will spark discussion and further research in how to manage energy consumption even as we continue to process ever larger amounts of data.

I would like to thank the issue authors for contributing and revising their work. I would also like to thank Dave Lomet for his advice and assistance throughout this process.

Brian Cooper
Google, Inc.
Mountain View, California, USA

Integrating Renewable Energy Using Data Analytics Systems: Challenges and Opportunities

Andrew Krioukov, Christoph Goebel†, Sara Alspaugh, Yanpei Chen, David Culler, Randy Katz
Department of Electrical Engineering and Computer Science
University of California, Berkeley
International Computer Science Institute†
{krioukov,alspaugh,ychen2,culler,randy}@eecs.berkeley.edu
goebel@icsi.berkeley.edu

Abstract

The variable and intermittent nature of many renewable energy sources makes integrating them into the electric grid challenging and limits their penetration. The current grid requires expensive, large-scale energy storage and peaker plants to match such supplies to conventional loads. We present an alternative solution, in which supply-following loads adjust their power consumption to match the available renewable energy supply. We show Internet data centers running batched, data analytic workloads are well suited to be such supply-following loads. They are large energy consumers, highly instrumented, agile, and contain much scheduling slack in their workloads. We explore the problem of scheduling the workload to align with the time-varying available wind power. Using simulations driven by real life batch workloads and wind power traces, we demonstrate that simple, supply-following job schedulers yield 40-60% better renewable energy penetration than supply-oblivious schedulers.

1 Introduction

A major challenge for the future electric grid is to integrate renewable power sources such as wind and solar [26]. Such sources are variable and intermittent, unlike traditional sources that provide a controllable, steady stream of power. Integrating a substantial fraction of renewable sources into the energy mix typically requires extensive backup generation or energy storage capacity to remove the variable and intermittent nature of such sources [11]. Given technological and economic limitations in current energy storage techniques, it will be difficult to meet even the current mandates for renewable energy integration [6, 18, 26].

Some have proposed creating *supply-following electric loads* from home appliances, lighting, and electric vehicles [5, 10]. This approach would schedule or sculpt the electric load such that it is synchronized with power availability from renewable sources, e.g., charge electric vehicles only when sufficient wind or solar power is available. This dispatchable demand approach represents an advance over traditional demand response techniques, which focus only on shedding load during times of high demand. However, home appliances, lighting, and electric vehicles all directly interact with humans. Such human dependencies can limit when, how

Bulletin of the IEEE Computer Society Technical Committee on Data Engineering

much, and how quickly such loads can be re-scheduled or sculpted. Subjective aspects of human comfort and perception can make it challenging to quantify and to compare alternate systems.

Recent green computing efforts have addressed components of a solution to this problem: energy efficiency [2, 8, 13, 15, 27], power proportionality [4, 14, 17, 21, 25], and service migration to geographic areas of lower real-time electricity prices [19]. These efforts are only components because even if we have energy efficient, power proportional systems that minimize energy bills, we will still have the problem of matching variable and intermittent energy sources with so-far less variable and continuous energy demand.

However, we show natural extensions of these techniques that address the matching problem on data analytics computer clusters. These clusters exhibit several properties. First, such clusters have varying levels of utilization [4], with the serviced workload having significant scheduling slack [10]. Second, the automatic and batch processing nature of computations on these clusters partially remove human limitations on when and how much the workload can be re-scheduled or sculpted. Third, the highly engineered and networked nature of such clusters allow rapid response to control signals from renewable sources. Taken together, these properties make data analytics computer clusters a compelling building block for supply-following loads.

This paper shows how to build supply-following loads using data analytics computer clusters.

- We make the case that data analytics workloads presenting a unique opportunity to implement supply-following mechanisms to help address the problem of integrating renewable energy.
- We introduce a quantitative metric to measure the degree renewable energy integration.
- We describe a simple supply-following job scheduler, evaluate it using realistic wind power and data analytic workload traces, and attain 40-60% improvement in the level of renewable energy integration.

The rest of the paper is organized as follows. Section 2 surveys the technical landscape to explain why the techniques we present are not in use today. Section 3 formalizes the problem of integrating renewable energy and introduces a metric for quantifying the degree of integration. Section 4 describes our simulation-based methodology, and the particular wind power traces and data analytic workloads we considered. Section 5 describes our supply-following scheduling algorithms. Section 6 presents the results of our simulations, which show that our algorithm yields significant improvement in renewable energy integration. Lastly, we discuss in Section 7 the key opportunities and challenges for future research in the area.

2 Technical Landscape

The intermittent and variable nature of renewable sources of energy, such as wind and solar, pose a problem for electric grid operators, who face increasing pressure to enlarge their renewable generation capacity. The current model of electric grid operation predicts the load in advance and then schedules the supply portfolio to service the load. The baseline generation capacity comes from sources that output constant, relatively inexpensive power, such as large coal and nuclear power plants. A portfolio of smaller, rapid-response, but more expensive and intermittent peaker plants track variation in demand and bridge any transient discrepancies between predicted and actual loads. This represents a model of *load-following supplies*, in which the electric loads are oblivious to the amount or type of supply, and supplies must track the electric load. Increasing the proportion of renewable supplies severely disrupts this model because renewable sources simply cannot be scheduled on demand.

One approach is to compensate for the variance in renewable supply using energy storage or additional peaker plants. This is an expensive proposition using current technologies - the energy storage and peaker plants must meet the full peak-to-zero swing in supply, instead of just meeting the small gap between predicted and actual load. An alternate solution is to flip the relationship and schedule the loads, thus creating *supply-following loads*. In this approach, loads must be prepared to consume electricity when supply is available and not otherwise. Only some loads form appropriate building blocks for supply-following loads.

Data analytics clusters represent a good example of electricity consumers with inherent scheduling flexibility in their workload. In a data analytics or batch processing cluster, users submit jobs in a non-interactive fashion. Unlike interactive web service clusters, these clusters do not have short, strict deadlines for servicing submitted jobs. Job completion deadlines typically create slack for a scheduler to shift the workload in time and consequently adjust energy consumption to, for instance, the amount of renewable energy available, or when electricity is cheaper.

If this is the case, why aren't such techniques in common practice? Part of the answer is that green computing remains an emerging field, with existing research focused on "low-hanging-fruits". Only recently has renewable energy integration been recognized as an unsolved problem. We briefly illustrate this transition in research focus. Early efforts in green computing included the Power Utilization Efficiency (PUE) of large scale data centers. PUE is defined as the ratio of total data center consumption to that consumed by the computing equipment, with typical values of 2 or greater [9, 24], i.e., to deliver 1 unit of energy to the computers, the data center wastes 1 or more units of energy in the power distribution and cooling infrastructure [3]. This revealed huge inefficiencies in the physical designs of data centers, and intense design efforts removed this overhead and reduced PUE to 1.2-1.4, much close to the ideal value of 1.0 [20, 22].

Once PUE values became more acceptable, data center operators recognized that real measure of effectiveness is not the power ratio between servers and the power distribution/cooling facilities, but the actual work accomplished on the servers per unit energy. In fact, servers in data centers are actively doing work typically only about 25% of the time [4]. Such low utilization levels naturally follow from the gap between peak and average requests rates, amplified by overprovisioning to accommodate transient workload bursts. Consequently, data center designers identified the need for *power proportionality*, i.e., that systems should consume power proportional to the dynamically serviced load and not to the static overprovisioning [4, 14, 17, 21, 25].

Power proportionality is a prerequisite for successfully turning data analytics clusters into supply-following loads. Otherwise, the cluster consumes approximately the same amount of energy regardless of the work it is doing. Unfortunately, modern server platforms are far from power proportional despite substantial improvements in power efficiency of the microprocessor, including Dynamic Voltage/Frequency Scaling (DVFS) and the introduction of a family of sophisticated power states. Even for specially engineered platforms [4], the power consumed when completely idle is over 50% of that when fully active, and idle consumption often over 80% of peak for commodity products [7].

Recently, we demonstrated the design and implementation of power proportional clustered services constructed out of non-power proportional systems. The basic approach is fairly obvious – put idle servers to sleep and wake them up when they are needed, keeping just enough extra active capacity to cover the time to respond to changes [12]. Thus, the stage is set for creating supply-following loads from data analytic compute clusters.

3 Problem Formulation

Our high-level goal is to use increase renewable energy use by turning data analytics clusters into supply-following electric loads. We consider a specific scenario where are data centers located near sources of clean electricity seek to maximize the use of local, directly attached wind turbines (or solar panels). In addition to the local intermittent power source, we can also draw energy from traditional sources in the grid. We observe the available renewable power at a given time and respond accordingly by sculpting the data analytics workload. If our data center is truly supply-following, it would draw most of its energy from the local, directly attached renewably supply, and very little energy from the rest of the grid.

> Key idea: Measure the degree of renewable integration by the fraction of total energy that comes from the renewable source, i.e., wind energy used divided by the total energy used. Better wind integration corresponds to a higher percentage.

Alternate problem formulations include optimizing a grid supply "blend" using remote control signals from grid operators, or responding to real time energy price, with the price being a function of the renewable and conventional power blend. These formulations assume that renewable sources have already been integrated in the grid signaling/pricing structures, and complicates validating the quality of such integration. Thus, we choose the strict formulation in which the data center operators directly contribute quantifiable improvements in integrating renewable sources.

A key feature of data analytics clusters is that jobs often do not need to be executed immediately. We use the term *slack* to describe the leeway that allows computational loads to be shifted in time. Slack is the number of time units that a job can be delayed, i.e., the slack for job j with submission time b_j, deadline d_j, that executes for t_j units of time is $s_j = d_j - b_j - t_j$.

Slack allows scheduling mechanisms to align job execution with the highly variable renewable power supplies. The quality of alignment, measured by the ratio of renewable to total energy used, depends on both the slack in the data analytic workload and the variability in the available renewable power. To obtain realistic results, we used batch job workload from a natural language processing cluster at UC Berkeley (Section 4.1), and wind power traces from the National Renewable Energy Laboratory (NREL) (Section 4.2).

We make several simplifying assumptions. We assume the cluster is power proportional. Otherwise, the cluster consumes roughly the same power all the time, making it incompatible with variable and intermittent sources. Also, we consider only data analytics applications that are inelastic, i.e., they cannot adjust the amount of consumed resources at runtime. An example of inelastic application is Torque [23], and an example of elastic applications is Hadoop [1]. Further, the application is "interruptible", meaning it can stop and resume as needed. At job submission time, we know the job deadline, run time, and resource requirements. We also assume that all the data needed by the application resides on a SAN that is under separate power management – it remains an open question to effectively power manage systems that co-locate computation and storage.

Slack is a key enabler for supply-following scheduling algorithms, in conjunction with power proportionality. Unlike traditional batch schedulers that try to maximize job throughput or minimize response time, supply-following schedulers seeks to find a good tradeoff between throughput, response time, and running jobs only when renewable energy is available.

4 Methodology

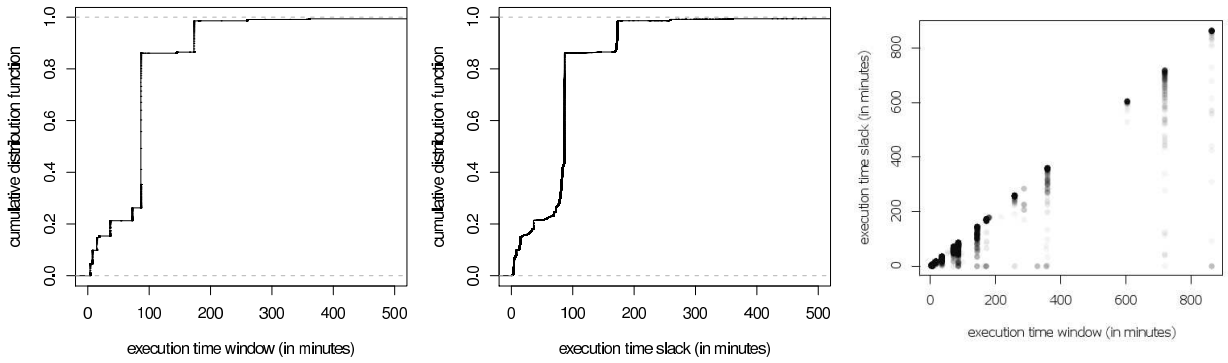
Two key components of our evaluation of supply-following scheduling are the input cluster workload and the input wind power traces. The degree of renewable integration depends on the slack in the particular cluster workload and the ability of the workload to align with particular wind traces.

4.1 Data Analytics Traces

We use batch job traces collected from a natural language processing cluster of 576 servers at UC Berkeley. Natural language processing involves CPU-intensive signal processing and model fitting computations. These jobs execute in a parallelized and distributed fashion on many processors. The completion deadlines are rarely critical. The cluster job management system is Torque [23], a widely used, open source resource manager providing control over batch jobs and distributed compute nodes. When submitting jobs to Torque, users specify the amount of processors and memory to be allocated, as well as the maximum running time. During job execution, the scheduler keeps track of the remaining running time of each job.

We collect job execution traces using Torque's `showq` command to sample the cluster state at 1 minute intervals. We collected a one month trace of 128,914 jobs and extracted job start times, end times and user-specified maximum running times. Deadlines are defined as the start time plus the maximum running time.

Figure 1(a) shows the CDF of the extracted job execution times. Figure 1(b) provides the CDF of execution



(a) CDF of execution times

(b) CDF of execution time slack

(c) Execution time window versus slack

Figure 1: Characteristics of batch job traces

time slack. The CDF shows that most of the jobs extracted from the cluster logs have a significant amount of execution time slack, generally ranging from 40 to 80 minutes of slack. Figure 1(c) shows the joint distribution of the job execution times and the execution time slack. The plot shows accumulations at certain execution time intervals (vertical lines), indicating different amounts of slack associated with jobs with the same execution times.

4.2 Wind Traces

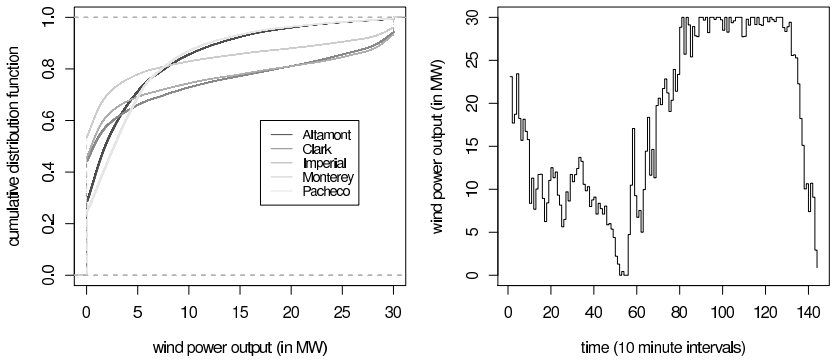
We used the wind speed and power data from the National Renewable Energy Laboratory (NREL) database [16]. This database contains time series data in 10 minute intervals from more than 30,000 measurement points in the Western Interconnection, which includes California. The measurement points in the NREL database are wind farms that hold 30 MW of installed capacity each. This capacity roughly equals 10 Vestas V-90 3MW wind turbines. For our experiments we picked one measurement point out of each major wind region in California.

Using wind output data from different regions is equivalent to considering data centers located in near different wind supplies. Our intention is to evaluate how well the the supply-following schedulers perform in range of possible locations. Figure 2(a) shows the cumulative distribution functions of wind power output at the different sites, suggesting considerable variation. Interestingly, some regions, such as Monterey, exhibit no power generation at all for large fractions of the time. Zero wind power generation results from either no wind or heavy storms causing the turbines to shut down.

Figure 2(b) shows the wind power output of a single wind farm during one day in the Altamont region. Wind power production can decline from maximum to zero output quickly, as indicated by the power drop at the right of the graph. Such steep rises and declines occur often in the traces. These fast transitions are arguably too short for re-scheduling human-facing loads such as home appliances and lighting.

4.3 Simulation Setup

The simulation takes as input the job submission times, job deadlines, required number of processors, and wind power availability over time. The simulation runs the candidate scheduler, and outputs the job execution order and power consumed over time. From these outputs, we then compute the percentage of total energy consumed that comes from wind. For the results in this paper, we run the simulation using one month of cluster jobs and wind power traces.



(a) CDFs of wind power output

(b) Altamont wind power output over time

Figure 2: Characteristics of wind traces

5 Algorithms

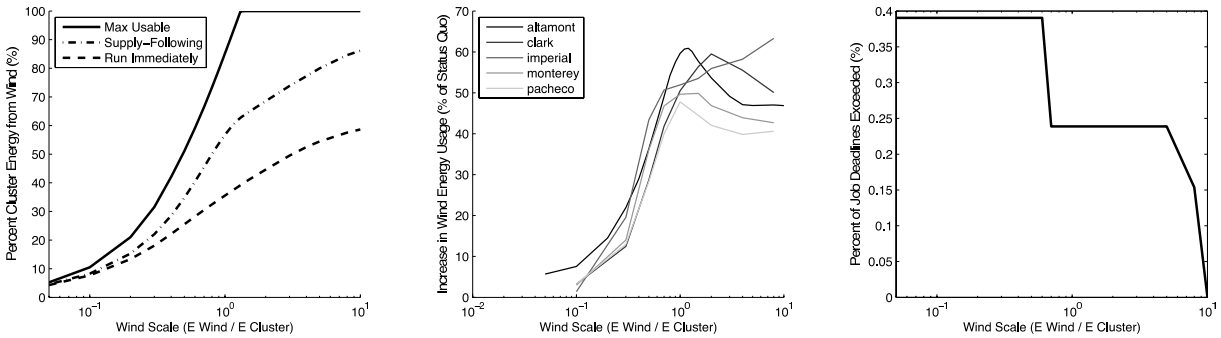
We compare two scheduling algorithms. The supply-oblivious, *run-immediately algorithm* executes jobs as soon as enough processors become available. Jobs that do not complete by their deadline are killed. The run-immediately algorithm represents the default scheduling behavior of Torque.

The *supply-following algorithm* attempts to align power consumption with the amount of wind power available, while minimizing the amount time by which jobs exceed their deadlines. It makes scheduling decisions at regular time intervals. At each interval, it schedules jobs that require immediate execution, beginning with jobs that have exceeded their deadlines the most, through jobs that will exceed their deadlines in the next time interval if left idled. If there are no such jobs that need immediate execution, the scheduler checks the wind power level. If some wind power is available, the scheduler executes the remaining jobs in order of increasing remaining slack, until either wind power or processors are fully used, or there are no more jobs on queue.

We use the heuristic of scheduling jobs in order of increasing slack, since jobs with a lot of slack can wait longer until more wind power becomes available. Thus, in the absence of accurate wind power or cluster workload predictors, this execution order increases the likelihood that we exploit all the available slack to align cluster workload and wind power availability.

One complication is that deferring jobs with slack can potentially aggravate resource bottlenecks. For example, if all jobs on the queue have slack and no wind power is available, the supply-following algorithm defers all jobs, while the run-immediately algorithm runs some of them. Thus, if periods of low wind are followed by periods of increased job submission, the slack of the delayed jobs may expire at the same time as new jobs that require immediate execution arrive. How often such situations occur depends on the particular mix of cluster workloads and wind power behavior, making it vitally important to use realistic wind traces and cluster workloads to quantify tradeoffs between renewable integration and performance metrics such as deadline violations.

Neither of these algorithms guarantees optimal job scheduling, i.e., always yield the highest possible percentage of wind energy to total energy used. Optimal job scheduling is impractical because it requires advance knowledge of cluster workload and wind availability, even though accurate, long-term workload predictors and wind forecasts remain elusive. Even if we have a workload and wind oracle, it is computationally infeasible to search for an optimal schedule out of all all possible job execution orders. Thus, the heuristic in the supply-following algorithm represents a compromise between optimality and practicality.



(a) Percentage of wind power usage (b) Percentage improvement (c) Percentage of delayed jobs

Figure 3: Evaluation of supply-following job scheduling

6 Evaluation

The scaling of the wind resource plays a crucial role in performance. Our raw wind traces vary between 0 and 30 MW, compared with our maximum cluster power consumption of 57.6 kW. A poor scaling factor would give trivial results. For example, if available wind power is orders of magnitude larger than what is needed by the cluster, under any scheduling algorithm 100% of energy used comes from wind. Conversely, if available wind power is orders of magnitude smaller, any scheduling algorithm would result in nearly 0% of energy coming from wind. We considered a range of scaling factors, such that the total available wind energy ranges from 0.1 to 10 times the total energy required by the cluster over the month long trace.

Figure 3(a) shows changes in the fraction of energy use that comes from wind for the two scheduling algorithms and a measure of the maximum usable wind energy given the fixed size of our cluster. Using the Pacheco wind trace, we scale the wind energy from 0.01 to 10 times the cluster's energy needs. The supply-following scheduler significantly out performs the run-immediately algorithm for all scale factors. The more wind available, the larger the performance gap. The supply-following algorithm undergoes a phase change around a wind scaling factor of 1 and exhibits diminishing returns for larger scale factors. This is likely due to the fact that as wind energy is scaled up, less of it can be used by a fixed size cluster – power spikes exceed the maximum cluster power.

Figure 3(b) shows the improvement of the supply-following versus the run-immediately algorithm for different wind traces. We compute improvement as:

$$\frac{\%\text{ energy from wind for supply-following algorithm} - \%\text{ energy from wind for run-immediately algorithm}}{\%\text{ energy from wind for run-immediately algorithm}}$$

We observe a range of improvements. At scaling factors of 1 and above, the supply-following scheduling yields a roughly 40-60% improvement.

> Key observation: The degree of renewable energy integration depends on renewable source variability and intermittence, as well as scheduling slack in the data analytic workload. Our supply-following scheduler attains a 40-60% improvement for realistic wind power and workload profiles.

To quantify how frequently the supply-following scheduling algorithm may cause jobs to exceed their deadlines, Figure 3(c) shows the percentage of all jobs that exceeded their deadlines, quantified at different wind scaling factors for the Pacheco wind trace. The percentage is very low and decreasing as the wind scaling factor increases. Also, job deadlines are never exceeded by more than one time interval, i.e. 10 minutes in our simulations. Compared with the 100s of minutes of execution time and slack shown in Figures 1(a) and 1(b), 10

minutes represents a very small amount. Thus, even though we can easily construct pathological wind traces and cluster workloads that lead to unacceptable deadline violation, for realistic wind traces and cluster workloads, deadline violations occur infrequently and have small impact.

7 Call to Arms

We must address the problem of integrating intermittent and variable renewable energy sources into the electric grid to have any hope of meeting legislative targets for renewable penetration. Current technologies and economic limits make it unlikely that we can construct load-following renewable supplies using large scale energy storage and peaker plants. We advocate for the alternative approach of constructing supply-following loads and we argue that server clusters are good candidates for tracking supplies. We have shown that simple, supply-aware scheduling algorithms can drastically increase the fraction of renewable energy consumed by data analytics clusters.

Future work includes exploring whether additional information regarding cluster workloads and wind traces can significantly improve the performance of the schedulers described in this paper. Ideally, we would like to construct a scheduling algorithm that is provably optimal and show how close to this bound practical schedulers can perform. Additionally, we want to extend our scheduler to support non-interruptible jobs and jobs with a minimum running time.

Looking forward, many opportunities and unanswered questions remain. We invite researchers and industry collaborators to implement the infrastructure for extensively tracing both cluster workloads and wind power profiles, and making such traces available. As we have shown in this paper, the level of renewable integration is highly dependent on workload and wind characteristics. Thus, having access to more cluster workloads is crucial.

Other open problems include supporting traditional DBMS or data-warehouse systems which would potentially require a different architecture. It remains an open and challenging problem to achieve power proportionality on systems that co-locate compute and storage on the same servers. We also want to consider tradeoffs between distributed supply-aware decisions made at each load, versus centralized decisions made by the electric grid operator. In this study we have assumed a data center with local, directly attached wind sources, independent of other loads. A more general scenario would consider a set of such loads.

We believe creating the information-centric energy infrastructure represents an interdisciplinary, society-wide enterprise. Computer scientists and engineers have much to contribute because of the exponentially growing energy foot-print of the technology industry, and our expertise in design, construction, and integration of large scale communication systems. Our paper demonstrates that another reason to contribute comes from the unique properties of electric loads caused by large scale computations. Consequently, data engineers in particular may end up leading the efforts to integrate renewable energy into the electric grid. We hope this paper serves as a first step in addressing this important challenge, and we invite our colleagues to join us in exploring the broader problem space.

Acknowledgements

The authors acknowledge the support of the Multiscale Systems Center, one of six research centers funded under the Focus Center Research Program, a Semiconductor Research Corporation program. This work was supported in part by NSF Grant #CPS-0932209 and EIA-0303575, the FCRP MuSyC Center, the German Academic Exchange Service, and Amazon, eBay, Fujitsu, Intel, Nokia, Samsung, and Vestas Corporations.

References

[1] Apache hadoop. `hadoop.apache.org`.

[2] Y. Agarwal, S. Hodges, R. Chandra, J. Scott, P. Bahl, and R. Gupta. Somniloquy: Augmenting network interfaces to reduce pc energy usage. In *NSDI'09: Proceedings of the 6th USENIX symposium on Networked systems design and implementation*, pages 365–380, Berkeley, CA, USA, 2009. USENIX Association.

[3] L. A. Barroso. The price of performance. *ACM Queue*, 3(7):48–53, 2005.

[4] L. A. Barroso and U. Hölzle. The case for energy-proportional computing. *Computer*, 40(12):33–37, 2007.

[5] A. Brooks, E. Lu, D. Reicher, C. Spirakis, and B. Weihl. Demand dispatch: Using real-time control of demand to help balance generation and load. *IEEE Power and Energy Soc.*, 8(3):20–29, 2010.

[6] California Public Utilities Commission. California Renewables Portfolio Standard. `http://www.cpuc.ca.gov/PUC/energy/Renewables/`, 2006.

[7] S. Dawson-Haggerty, A. Krioukov, and D. E. Culler. Power optimization - a reality check. Technical Report UCB/EECS-2009-140, EECS Department, University of California, Berkeley, Oct 2009.

[8] A. Greenberg, J. Hamilton, D. A. Maltz, and P. Patel. The cost of a cloud: research problems in data center networks. *SIGCOMM Comput. Commun. Rev.*, 39(1):68–73, 2009.

[9] S. V. L. Group. Data center energy forecast. `http://svlg.org/campaigns/datacenter/docs/DCEFR_report.pdf`, 2009.

[10] R. Katz, D. Culler, S. Sanders, S. Alspaugh, Y. Chen, S. Dawson-Haggerty, P. Dutta, M. He, X. Jiang, L. Keys, A. Krioukov, K. Lutz, J. Ortiz, P. Mohan, E. Reutzel, J. Taneja, J. Hsu, and S. Shankar. An information-centric energy infrastructure: The berkeley view. *Sustainable Computing: Informatics and Systems*, 2011.

[11] B. Kirby. Frequency regulation basics and trends. Technical report, Oak Ridge National Laboratory, December 2004. Published for the Department of Energy. Available via `http://www.osti.gov/bridge`.

[12] A. Krioukov, P. Mohan, S. Alspaugh, L. Keys, D. Culler, and R. H. Katz. Napsac: design and implementation of a power-proportional web cluster. In *Green Networking '10: Proceedings of the first ACM SIGCOMM workshop on Green networking*, pages 15–22, New York, NY, USA, 2010. ACM.

[13] M. Lammie, D. Thain, and P. Brenner. Scheduling Grid Workloads on Multicore Clusters to Minimize Energy and Maximize Performance. In *IEEE Grid Computing*, 2009.

[14] D. Meisner, B. T. Gold, and T. F. Wenisch. Powernap: eliminating server idle power. In *ASPLOS '09*, 2009.

[15] R. Nathuji and K. Schwan. Vpm tokens: virtual machine-aware power budgeting in datacenters. In *HPDC '08: Proceedings of the 17th international symposium on High performance distributed computing*, pages 119–128, New York, NY, USA, 2008. ACM.

[16] National Renewable Energy Laboratory. National wind technology center data. 2010.

[17] S. Nedevschi, J. Chandrashekar, J. Liu, B. Nordman, S. Ratnasamy, and N. Taft. Skilled in the art of being idle: reducing energy waste in networked systems. In *NSDI'09: Proceedings of the 6th USENIX symposium on Networked systems design and implementation*, pages 381–394, Berkeley, CA, USA, 2009. USENIX Association.

[18] Office of the Governor, California. Executive Order S-14-08, 2008.

[19] A. Qureshi, R. Weber, H. Balakrishnan, J. Guttag, and B. Maggs. Cutting the electric bill for internet-scale systems. In *SIGCOMM '09: Proceedings of the ACM SIGCOMM 2009 conference on Data communication*, pages 123–134, New York, NY, USA, 2009. ACM.

[20] N. Rasmussen. Electrical efficiency modeling of data centers. Technical Report White Paper #113, APC, 2006.

[21] J. A. Roberson, C. A. Webber, M. McWhinney, R. E. Brown, M. J. Pinckard, and J. F. Busch. After-hours power status of office equipment and energy use of miscellaneous plug-load equipment. Technical Report LBNL-53729-Revised, Lawrence Berkeley National Laboratory, Berkeley, California, May 2004.

[22] R. K. Sharma, C. E. Bash, C. D. Patel, R. J. Friedrich, and J. S. Chase. Balance of power: Dynamic thermal management for internet data centers. *IEEE Internet Computing*, 9(1):42–49, 2005.

[23] G. Staples. Torque resource manager. In *SC '06: Proceedings of the 2006 ACM/IEEE conference on Supercomputing*, page 8, New York, NY, USA, 2006. ACM.

[24] The Green Grid. The green grid data center power efficiency metrics: PUE and DCiE. *Technical Committee White Paper*, 2007.

[25] N. Tolia, Z. Wang, M. Marwah, C. Bash, P. Ranganathan, and X. Zhu. Delivering energy proportionality with non energy-proportional systems – optimizing the ensemble. In *Proceedings of the 1st Workshop on Power Aware Computing and Systems (HotPower '08)*, San Diego, CA, Dec. 2008.

[26] United States Senate, One Hundred Eleventh Congress. *First Session to Receive Testimony on a Majority Staff Draft for a Renewable Electricity Standard Proposal, Hearing before the Committee on Energy and Natural Resources*. U.S. Government Printing Office, February 2009.

[27] B. Zhai, D. Blaauw, D. Sylvester, and K. Flautner. Theoretical and practical limits of dynamic voltage scaling. In *DAC '04: Proceedings of the 41st annual Design Automation Conference*, pages 868–873, New York, NY, USA, 2004. ACM.

Rethinking Query Processing for Energy Efficiency: Slowing Down to Win the Race

Willis Lang, Ramakrishnan Kandhan, Jignesh M. Patel
Computer Sciences Department
University of Wisconsin – Madison

Abstract

The biggest change in the TPC benchmarks in over two decades is now well underway – namely the addition of an energy efficiency metric along with traditional performance metrics. This change is fueled by the growing, real, and urgent demand for energy-efficient database processing. Database query processing engines must now consider becoming energy-aware, else they risk missing many opportunities for significant energy savings. While other recent work has focused on solely optimizing for energy efficiency, we recognize that such methods are only practical if they also consider performance requirements specified in SLAs. The focus of this paper is on the design and evaluation of a general framework for query optimization that considers both performance constraints and energy consumption as first-class optimization criteria. Our method recognizes and exploits the evolution of modern computing hardware that allows hardware components to operate in different energy and performance states. Our optimization framework considers these states and uses an energy consumption model for database query operations. We have also built a model for an actual commercial DBMS. Using our model the query optimizer can pick query plans that meet traditional performance goals (e.g., specified by SLAs), but result in lower energy consumption. Our experimental evaluations show that our system-wide energy savings can be significant and point toward greater opportunities with upcoming energy-aware technologies on the horizon.

1 Introduction

Energy management has become a critical aspect in the design and operation of database management systems (DBMSs). The emergence of this new paradigm as an optimization goal is driven by the following facts: a) Servers consume tremendous amounts of energy – 61B kiloWatt-hours in 2006 alone and doubling by 2011 [3]; b) The energy component of the total cost of ownership (TCO) for servers is already high, and growing rapidly. The server energy component of the three-year TCO is expected to dwarf its initial purchase cost [9]. A big contributing factor to this trend is that processors are expected to continue doubling in the number of cores every 18 months, but the performance per Watt doubles at a slower rate of once every two years [8]; c) To make matters worse, typical servers are over provisioned to meet peak demands, and as a result, they are idle or underutilized most of the time. Barroso and Hölzle [7] have reported average server utilization in the 20–30% range; d)

Bulletin of the IEEE Computer Society Technical Committee on Data Engineering

Unfortunately, when servers are idle, or nearly idle, they tend to consume energy that is disproportional to their utilization – for example, an idle server can consume more than 50% of its peak power [7].

With these rising energy costs and energy-inefficient server deployments, it is clear that there is a need for DBMSs to consider energy efficiency as a first class operational goal. In fact, driven by requests from its customers, the Transaction Processing Performance Council (TPC) has moved in this direction, and all TPC benchmarks now have a component for reporting the energy consumed when running the benchmarks.

The challenge here is to reduce the energy consumption of a DBMS while maintaining the performance levels that are typically expected and accepted by the end users. Thus, there is huge opportunity for the database community to take on this challenge head-on and find methods to make DBMSs more energy-efficient. In this paper, we tackle the query processing component, and develop a framework for energy-efficient query processing.

To begin, one might think that perhaps doing business as usual might work for energy-aware query processing. Specifically, we already know how to optimize queries for response time/throughput metrics. So it is natural to pose the following question: Is processing (optimizing and executing) the query as fast as possible always the most energy-efficient way to operate a DBMS? As we show in this paper, the answer to this question is no. The reason for this negative answer has to do with typical server operational characteristics and the power/performance characteristics of hardware components in modern servers.

First, typical servers often run at low utilization. This means that a server has many opportunities to execute the query slower, if the additional delay is acceptable. For example, this situation may occur if the Service Level Agreement (SLA) permits the additional response time penalty. Since typical SLAs are written to meet peak or near peak demand, SLAs often have some slack in the frequent low utilization periods, which could be exploited by the DBMS. (The incentive for doing this is particularly high when the DBMS server is running as a hosted cloud service, and any cost savings that do not violate the SLAs, directly adds to the bottom line.)

Second, components on modern server motherboards offer various power/performance states that can be manipulated from software. Of these components, the CPU is currently the most amenable for transitioning between such power/performance states, but nearly every power–hungry component connected to the server motherboard (e.g., memory chips and network cards) is moving towards providing multiple power/performance states for more energy-efficient operation. For instance, recent research has shown that 'hot' power-down (parking) of physical memory DIMMs may save upwards of 40% of system energy consumption [6, 29].

This paper proposes a new way of thinking about processing and optimizing queries. In our framework, we assume that queries have some response time goal, potentially driven by an SLA. The query optimization problem now becomes: *Find and execute the most energy-efficient plan that meets the SLA*.

A crucial aspect of our work that distinguishes it from previous work [31, 32] is our focus on the slack available in performance between the optimal performance and the SLA goal, and leveraging this slack to reduce the energy consumption (and thereby overall operating cost). To the best of our knowledge this is the first paper that proposes looking at query optimization and evaluation in a DBMS from this new perspective.

To enable this framework, we propose extending existing query optimizers with an energy consumption model, in addition to the traditional response time model. With these two models, the enhanced query optimizer can now generate what we call an *Energy Response Time Profile (ERP)*. The ERP is a structure that details the energy and response time cost of executing each query plan at every possible power/performance setting under consideration. The ERP of a query can then be used to select the appropriate "energy-enhanced" strategy to execute the query.

Notice that the framework proposed in this paper is not limited to the single node database environments that were studied by [31, 32], as it can be applied to optimizers for parallel DBMSs as well. Such parallel DBMSs also have system settings that include cluster configuration as well as server configuration [20, 21]. Such considerations for parallel DBMSs is part of future work.

To show the potential and validity of this approach, in Figure 1 we show an actual ERP for a single equijoin query on two Wisconsin Benchmark relations [11]. The plot shows the system energy consumption and response

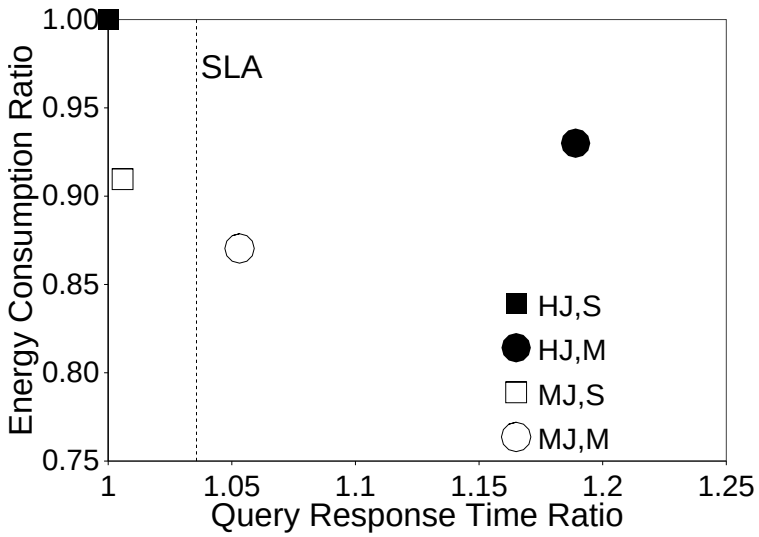


Figure 1: Energy Response Time Profile (ERP) for an equijoin query on two 50M tuple (5GB) Wisconsin Benchmark relations, on the attribute $four$, and a 0.01% selection on both relations. The energy and response time values are scaled relative to the stock settings (HJ, S) that is currently used by DBMSs.

time measurements for executing the query using two different query plans – hash join (HJ) and sort-merge join (MJ) at two different system settings (labeled 'S', 'M'), on an actual commercial DBMS. System setting 'S' corresponds to the "stock" (high power/performance) system settings. System setting 'M' is the power/performance setting that can save energy by reducing the memory capacity [6, 29]. (Details about the experimental setup and systems setting for this experiment is presented in more detail in Section 4.) The energy measurement is the actual energy that is drawn by the entire server box – i.e., we measure the energy drawn from the wall socket by the entire system. In this figure, we have plotted all the other data points proportionally relative to (HJ, S), for both the energy consumption (on the y-axis), and the response time (on the x-axis). Traditional query optimizers often incorporate response time into their primary metric so they would choose HJ and system setting S to execute the query, since this is the fastest query plan. However, by choosing the sort-merge (MJ) plan and system setting M, we can reduce the energy consumption by 13% for only a 5% increase in response time.

However, it may not always be possible to choose a lower energy setting to run the query due to some performance constraint. For example, let us say an SLA exists between the client and the server, which only allows for an additional 4% delay over the optimal response time (HJ, S). This SLA is represented in Figure 1 as a vertically dotted line. Now the most energy-efficient choice is MJ and system setting S, which reduces the energy consumption by 9% and still meets the SLA! *In this case, changing the join algorithm while holding the system settings constant reduces the energy consumption.* This ability to optimize for energy while maintaining performance constraints makes our framework particularly attractive for DBMSs.

To keep the scope of this work limited to a single paper, we focus on the following important class of queries: single join queries with selection and projections. However, our framework can be extended for more complex query processing, using the single join queries as essential building blocks. As the reader will see, our research points to many open research problems in this emerging field of energy-aware data management, only a small part of which we can address in this paper (details in Section 4.3).

This paper makes the following contributions:

- We present a new design for query optimizers that uses both energy and performance as optimization criteria while adhering to performance constraints.

- We present the notion of an Energy Response Time Profile (ERP) that can be used by our query optimization framework to explore all the combinations of system power/performance settings that the hardware provides. The ERP can then be used by the optimizer in picking an "energy-enhanced" query plan that meets any existing response time targets.
- We validate the energy model using an actual commercial DBMS and demonstrate the end-to-end benefit of our approach, which can result in significant energy savings.

The remainder of this paper is organized as follows: Section 2 presents our new optimizer framework, Section 3 describes our energy cost models. The experimental results are presented in Section 4. Related work and conclusions are described in Section 5 and 6 respectively.

2 Framework

In this section we present a general query processing framework that uses both energy and response time as the optimization criteria. The questions we tackle are: (1) How to redefine the job of the query optimizer in light of its additional responsibility to optimize for energy consumption? (2) Given this new role, how do we design a query optimizer? We discuss answers to these questions below.

2.1 New Role of the Query Optimizer

The traditional query optimizer is primarily concerned with maximizing the query performance. The optimizer's new goal now is to find query plans that have *acceptable* performance, but consume *as little energy as possible*. As shown in Figure 1, we want the query optimizer to return an energy-enhanced query plan that is to the left of the SLA-dictated performance requirement, and as low as possible along the y-axis. (We note that performance SLAs are often not rigid and violating SLAs could be compensated by other mechanisms – e.g., some financial compensation. There are potential business decisions to be made about when violating SLAs are okay, but these considerations are beyond the scope of this study.)

The main task here is to generate ERP plots like that shown in Figure 1. To generate these plots, the query optimizer needs to *quickly* and *accurately* estimate both the response time and the energy consumption for each query plan. Query optimizers today are fairly good at predicting the response time, but doing so while accounting for varying hardware configuration is a new challenge. Of course, this challenge of predicting the energy consumption of a given query plan for a specific system setting also requires that the query optimizer understands the power/performance settings that the hardware offers.

2.2 System States, Optimization, and ERP

Driven by the need for energy-efficient computing, hardware components such as CPU [19], memory [29], and NIC [12] are increasingly becoming energy-aware, and offer multiple power/performance settings that can be manipulated directly from software at runtime. Many studies have looked at CPU power/ performance control by capping the maximum CPU frequency [19, 31, 32] and so we do not focus on system states based on this type of control. We have performed our own CPU studies on a manufacturer-provided instrumented NUMA server (which none of the prior works have studied) that gave direct on-motherboard measurements of the CPU, memory, and northbridge energy consumption. For an in-memory clustered index scan on a commercial DBMS, with this server we found that dropping the maximum CPU frequency by 25% using standard Microsoft Windows Server CPU controls can save 12% in the three component aggregate consumption while being penalized 2% in response time. The reason for this result was that capping the maximum frequency increases CPU utilization and energy efficiency. (We omit these results here in the interest of space.)

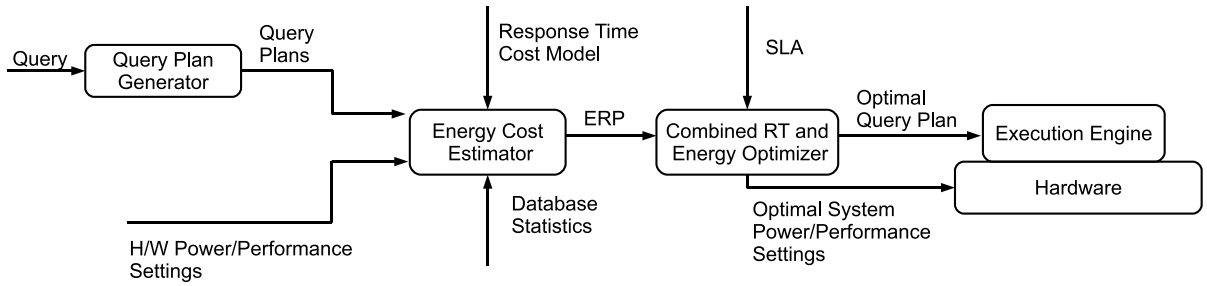


Figure 2: An overview of the framework that optimizes for both energy and response time. The energy cost estimator is described in more detail in Figure 3.

While we saw interesting wins with CPU capping, as mentioned above, this paper focuses on memory-based system configurations, since database query working sets are increasingly completely or nearly fully resident in main memory. In this setting, most of the database processing costs are outside the traditional IO subsystem, which in turn makes memory a significant source of energy consumption (up to 45% and 30% as has been discussed in [29] and [6] respectively). The specific memory mechanism we are exploiting is the ability to "park" main memory DIMMs similar to how CPU cores can currently be "parked". Major manufacturers like Intel have begun developing these mechanisms and methods for exploiting these mechanisms are been actively explored [6,29]. Our idea for using this mechanism is that if the current query does not require all of the available main memory, then some of the DIMM banks can be powered down to save energy. This new ability requires research on issues like physical DIMM-aware buffer pools and the impact this will have on join algorithm selection. (We note that we are simply scratching the surface of various research problems in this space – for example, instead of memory parking one can think of freeing up memory for use by other concurrent task, or considering both memory and CPU power performance states and expanding that scope to consider networking components in a parallel or cluster data processing system. Further discussion can be found in Section 4.3.)

The "energy-enhanced" query optimizer must be provided with a set of system operating settings, where each system operating state is a combination of different individual hardware component operating settings. The optimizer then uses an energy prediction model along with its current response time model to produce an ERP for that query, like the example shown in Figure 1.

Given a query Q with a set of possible plans $P = \{P_1, ..., P_n\}$ that is to be executed on a machine with system operating settings $H = \{H_1, ..., H_m\}$, the ERP contains one point for every plan for every system operating setting (and hence an ERP has $n \times m$ points). Note that heuristics could be developed to prune the logical plans P so that only a small subset of the n plans are explored for specific queries – e.g., plans for where the 'stock' setting does not meet the performance requirement may be assumed to not meet the requirement in all other system settings. (An interesting direction for future work is to explore this option.)

Several methods can be envisioned to predict the energy cost of a query plan. One simple method is to assume some constant power drawn by the system for any specific query such that Energy α Response Time. But, we have found in our analysis that such simple models are not very accurate. Note that an accurate energy estimation model is essential so that the optimizer does not make a wrong choice – such as choosing query plan *HJ* and system setting *M* in Figure 1, which incurs penalties in response time that lie beyond the SLA and does not save enough energy to make this choice attractive.

To enable this new query optimizing framework, we develop an accurate analytical model to estimate the energy consumption cost for evaluating a query plan. We discuss this model in Section 3.

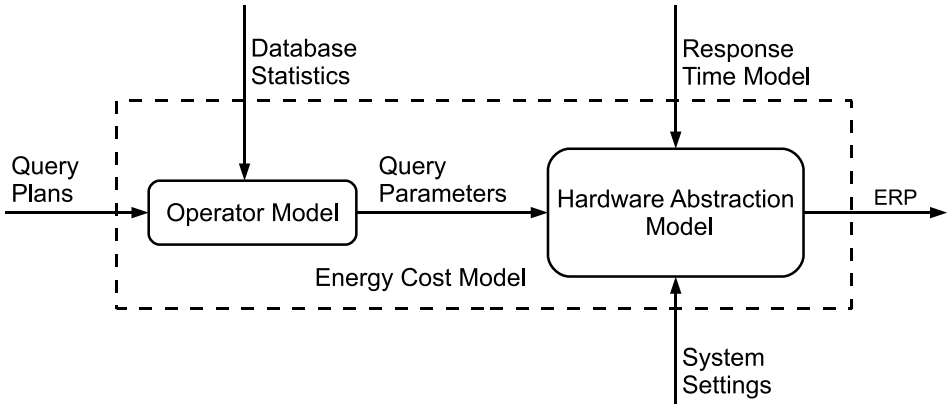


Figure 3: An overview of the energy cost model. The operator model estimates the query parameters required by the hardware abstraction model for each query plan from the database statistics available. The hardware abstraction model uses the query parameters estimated and the system parameters learnt to accurately estimate the energy cost.

2.3 Our Implementation

In this section we discuss our implementation of the framework described above. We had two implementation options – the first one was to integrate this framework into an existing query optimizer in a open-source DBMS like MySQL, and the other to implement it as an external module for a commercial DBMS. We have found that the commercial DBMS that we are using is appreciably faster than the open-source alternative – as much as a factor of 10 in many cases. Consequently, we decided to use the commercial DBMS for our implementation. This choice also forced us to think of a design that is generic and likely to be more portable across other systems, as we have to build the framework using only the high level interfaces that the DBMS provides. (Besides, if one is concerned about energy efficiency, starting with a system that is significantly faster is likely to be a better choice from the energy efficiency perspective.)

Figure 2 gives an overview of our implementation. The query is supplied as input to the query plan generator in the commercial DBMS, which is then requested to list all the promising query plans that the optimizer has identified. These query plans along with the information about available system settings is provided as input to the Energy Cost Estimator, which generates the ERP using an analytical model for predicting the energy consumption (see Section 3). The generated ERP is then used by the combined energy-enhanced query optimizer to choose the most energy-efficient plan that meets SLA constraints. Then, a command to switch to the chosen system operating state is sent to the hardware, followed by sending the optimal query plan to the execution engine.

3 Energy Cost Model

In this section, we briefly describe an analytical model that estimates the energy cost of executing a query at a particular power/ performance system setting. Our model abstracts away the energy cost of a query in terms of system parameters that can be learnt through a "training" procedure, and query parameters (CPU instructions, memory fetches, etc.) that can be estimated from available database statistics.

3.1 Model Overview

We want to develop a simple, portable, practical, and accurate method to estimate the energy cost of a query plan. Unfortunately, prior techniques used to estimate energy consumption fail to satisfy one or more of these goals. For example a circuit-level model of hardware components [1, 2, 13] can accurately predict the energy consumption, but these models also have a high computational overhead which make them impractical for query optimization. On the other hand, a higher level model that treats the entire system as a black box, though simple and portable, is not very accurate.

In our approach, we use an analytical model that offers a balance between these two extremes. In our models, the power drawn by different hardware components are abstracted into learn-able system parameters that are combined with a simple operator model. Figure 3 gives an overview of the energy cost model that we have designed.

The operator model takes as input the query plan and uses the database statistics to estimate the query parameters that is required by the hardware abstraction model to estimate the energy cost. The hardware abstraction model that we describe below required estimations of four query parameters from the operator model: the number of CPU instructions, the number of memory accesses, the number of disk read and write requests anticipated during query execution. Our operator model provided estimates for these four query parameters for three basic operations: selection, projection, and joins. (See [18] for details.) The hardware abstraction model then uses these four query parameters and the response time model (since response time is dependent on system settings) to estimate the energy cost of evaluating a query using a particular query plan at a particular power/performance system setting, essentially computing the ERP.

While we have considered various hardware abstraction models, in the interest of space, we describe the model that we found to be the most accurate. Equation 1 shows the model for average system power during a query as defined by three types of variables:

1. time (T)
2. query parameters for query Q: CPU instructions - I_Q, disk page reads - R_Q, disk page writes - W_Q, and memory page accesses - M_Q
3. train-able system parameters: CPU - C_{cpu}, disk read - C_R, disk write - C_W, memory access - C_{mm}, remaining system - C_{other}). These parameters quantify the component power properties.

The intuition behind this power model is to sum of the average CPU power ($C_{cpu} * \frac{I_Q}{T}$), read power ($C_R * \frac{R_Q}{T}$), write power ($C_W * \frac{W_Q}{T}$), memory power ($C_{mm} * \frac{M_Q}{T}$), and remaining system power (C_{other}) during the duration of the query.

$$P_{av} = C_{cpu} * \frac{I_Q}{T} + C_R * \frac{R_Q}{T} + C_W * \frac{W_Q}{T} + C_{mm} * \frac{M_Q}{T} + C_{other} \quad (1)$$

In the interest of space, we omit the details of deriving the query parameters in this paper. One can model these query parameters using the work of [24, 28] to provide accurate component power draw.

The key features of our energy cost model are:

- **Simplicity:** The models require no additional database statistics other than those used in traditional query optimizers for response time cost estimation. Also, minimal overhead is incurred by the query optimizer in calculating the most energy efficient power/performance operating setting and query plan. The computational complexity is $O(|H| * |P|)$ where H is the set of valid power/performance system settings and P is the set of query plans for the query.

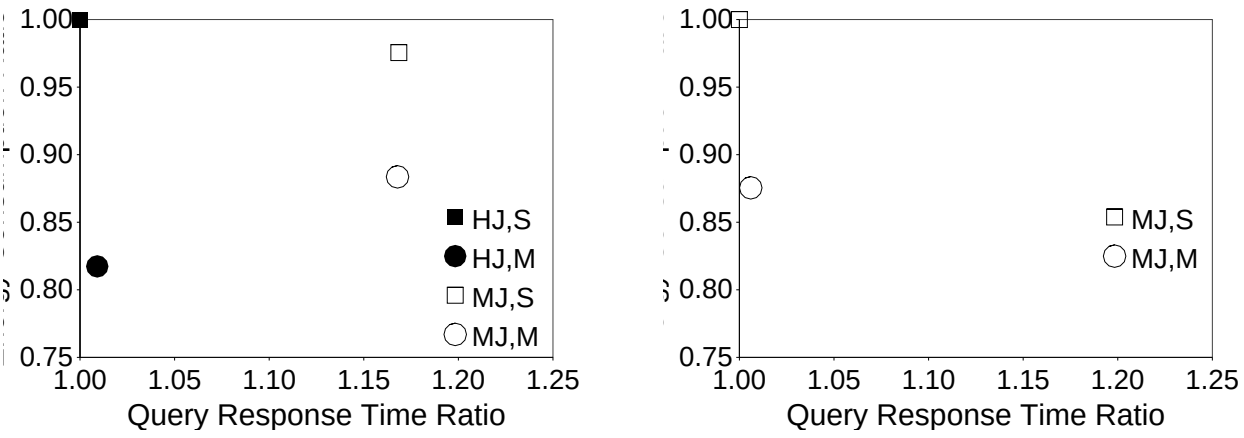


Figure 4: ERPs of two equijoin query classes: (a) Low memory requirement (b) Low memory and I/O heavy. Two join algorithms are used: hash join (HJ) and sort-merge join (MJ); along with 'stock' (S) and low memory (M).

- **Portability:** The model makes few assumptions about the underlying hardware or the database engine internals and can be ported across many DBMSs and machines. In fact, we have been able to implement this model on top of a commercial DBMS, treating it as a black box, and the model still achieves high accuracy.
- **Practical:** Detailed system simulators like DRAMSim [1] and Sim-Panalyzer [2] model hardware components at the circuit level to estimate power consumption. This process though accurate is computationally very expensive, and is hence not practical for use in a query optimizer. In our model we abstract the power drawn by different components into learn-able system parameters.
- **Accuracy:** In our tests, our models have an average error rate of around 3% and a peak error rate of 8%.

4 Evaluation and Discussion

In this section we will present results demonstrating the potential benefit of our optimization framework on a commercial DBMS using end-to-end measurements.

4.1 System Under Test

The system that we use in this paper has the following main components: ASUS P5Q3 Deluxe Wifi-AP motherboard (which has inbuilt mechanisms for component-level power measurements), Intel Core2 Duo E8500, 4x1GB Kingston DDR3 main memory, ASUS GeForce 8400GS 256M, and a 32G Intel X25-E SSD. The power supply unit (PSU) is Corsair VX450W. System power draw was measured by a Yokogawa WT210 unit (as suggested by SPEC and TPC energy benchmarks) connected to a client measuring system. Energy consumption was measured by tapping into the line that supplied power to the entire box, so our measurements and results below are real end-to-end gains. The operating system used was Microsoft Windows Server 2008, and we use a leading commercial DBMS. Note that we use an SSD-based IO system as many studies have shown that SSD-based configurations are more energy efficient [31] and also provide a better total cost of ownership because of their higher performance-per-$ [4].

Table 1: Both queries have the template (SELECT * FROM R, S WHERE <predicate>). These queries are used for the ERP plotted in Figure 4. All relations are modified (100 byte tuples) Wisconsin Benchmark relations.

Query	Predicate	Size of R, S
A	`R.unique2 < 0.1*||R|| AND R.unique1 = S.unique2`	1GB, 1GB
B	`R.unique2 = S.unique2`	5GB, 5GB

4.2 End-to-End Results: ERP Effectiveness

We now present end-to-end results using the techniques that we have proposed in this paper. We use the two system settings described in Section 1; namely, (1) **S** – the default stock settings of our system, and (2) **M** – a reduced memory setting where the memory is reduced from 4GB to 2GB.

In the interest of space, here we present results with only the two queries shown in Table 1. Both queries are join queries on two modified Wisconsin Benchmark [11] tables, where the tuple length has been reduced to 100 bytes. Query A is a 10% selectivity join on two small 1GB tables while Query B is a full join on the sorted keys of two 5GB tables.

First, let us examine the scenario when switching to a lower power/performance state has little effect on the response time. With only 2GB of memory, we expect that a query whose peak main memory requirement is less than 2GB will take approximately the same amount of time to execute, and hence will provide significant energy savings. Figure 4 (a) shows the ERP of query A in Table 1. As we can see, retaining the same hash join plan but using system setting 'M' *reduces the energy consumption by 18% but increases the query response time by only 1%.* In comparison, changing the join algorithm to sort-merge incurs significantly higher response time penalties.

Query B in Table 1, is an equijoin on two tables that are clustered on the join attribute 'unique2'. The two 5GB tables are relatively large compared to the amount of available main memory, but since the tables are already sorted on the join attribute the inputs do not need to be sorted before joining using a merge join operation. Query B is I/O-intensive, requires minimal computation, and has a low peak memory requirement. For this query, as shown in Figure 4 (b), using the sort-merge join along with reducing the memory requirement produces a win of 12% energy savings for less than 1% performance penalty. The response times for the hash join plans are far greater than sort-merge for Query A, and are therefore left out of Figure 4 (b).

4.3 Summary

Our preliminary results described above shows that significant (>10% for the queries above) energy savings can be attained by careful optimization of both query plans and system settings in a commercial DBMS based on end-to-end measurements. We now summarize key takeaways messages from our study.

Energy-Aware Query Optimization: Current DBMS query optimizers can be made energy aware using our modular optimization framework. By introducing a Hardware Abstraction Model (Figure 3) in addition to the traditional Operator Model, we are able to create Energy Response Time Profiles – ERPs (Figure 1). The Hardware Abstraction Model is a train-able model for estimating both the query response time *and* the query energy consumption. ERPs allow us to maximize the query's energy efficiency while adhering to performance SLAs.

SLA-Driven Resource Allocation: The main concept behind the ERP is that by analyzing the relationship between different query plans and different hardware power/ performance settings, we can trade decreased query performance for increased energy efficiency in the presence of slack in SLAs. In some cases, this trade-off is disproportionate, and in some cases, this trade-off can be highly favorable for energy efficiency. By plotting the SLA performance limit on the ERP, we can easily discover the most energy-efficient combination

of query execution plan and hardware system settings while still adhering to the SLA limit. Figures 4 (a,b) show preliminary results where only two plans with two system settings can provide very interesting execution options.

Exploring New Hardware Mechanisms: Traditionally, CPU has been the first target of studies in server energy management [19]. In these results, we have shown that emerging hardware mechanisms such as memory DIMM parking [6, 29] can provide significant energy savings for DBMSs. Other hardware parts such as the networking components [5, 12] are also becoming energy-aware, which will further increase the number of system settings that comprise the ERP in cluster database environments. In addition, other parameters such as cluster size [20] and cluster heterogeneity may also impact the sizd of the ERP space that must be considered.

Complex Queries and Concurrent Queries: The preliminary results presented here point to interesting opportunities for energy savings with simple queries, and it would be interesting to extend this study to more complex queries, and query workloads (e.g., concurrent queries).

5 Related Work

Interest in the area of energy management for DBMSs has begun to grow since the early database publications on this subject [14–16, 19, 27]. Much of the drive has come because of the observation that the energy costs is a big and growing component in the total cost of ownership (TCO) for large server clusters [17, 22, 25].

There are many data center-wide methods for energy management. One popular method is consolidation which forces cluster nodes to be highly utilized and thus, as energy-efficient as possible [10, 26, 30]. Other mechanisms of powering down cluster nodes have been explored in [20, 21, 23]. Cluster level methods can also benefit from the optimization framework we have presented here since our energy optimization module neatly fits in with traditional performance-based optimizers. Cluster level methods can result in improvements in the energy efficiency of data centers and can largely be used orthogonally to "local-level" methods (that work on single nodes) for reducing energy consumption.

Local-level results found in [32] also showed that opportunities for energy-based query optimization exist in PostgreSQL. However, they primarily focused on energy efficiency as the only goal, while our work targets energy efficiency along with SLA constraints. This consideration of SLAs is also a key difference between our work and another recent study [31]. Furthermore, the conclusions from that work suggested that based on traditional server configurations, no new energy-based optimization is necessary because energy efficiency directly follows performance. However, our work shows that when considering SLAs, energy optimal and performance optimal are not always the same operating points.

In summary, we believe that this is the first work to cast energy efficiency as a first-class goal that works in conjunction with performance-based SLA constraints for DBMS query processing. With the move towards cloud computing, we believe this setting is more appropriate and interesting from the perspective of hosted DBMS deployments.

6 Conclusions and Future Work

This paper presents a new framework for energy-aware query processing. The framework augments query plans produced by traditional query optimizers with an energy consumption prediction to produce an Energy Response Time Profile (ERP) for a query. These ERPs can then be used by the DBMS in various interesting ways, including finding the most energy-efficient query plan that meets certain performance constraints (dictated by SLAs). To enable the above framework, a DBMS needs an energy consumption model for queries, and we have developed a simple, portable, practical, and accurate model for an important subset of database operations and algorithms. We have used our framework to augment a commercial DBMS using actual energy measurements, and demonstrated that significant energy savings are possible in some cases.

This area of energy-aware data processing is in its inception, and there are many directions for future work. Some of these directions include extending our framework to more query operations/algorithms, considering more complex/concurrent queries, and considering breaking down "complex" operations such like hash join into smaller components (the build and the probe phase) while also switching between hardware power/performance states. Other promising directions include designing new DBMS techniques to deal with new and evolving power-related hardware mechanisms, studying the application to parallel query optimizers, and working and influencing the development of new hardware features (e.g. memory) to make it more amenable for DBMS to exploit for energy-aware query processing.

7 Acknowledgments

This research was supported in part by a grant from the Microsoft Jim Gray Systems Lab and by the National Science Foundation under grant IIS-0963993.

References

[1] DRAMSim: A Detailed Memory-System Simulation Framework. *http://www.ece.umd.edu/dramsim/*.

[2] The SimpleScalar-Arm Power Modeling Project. *http://www.eecs.umich.edu/~panalyzer/*.

[3] Report To Congress on Server and Data Center Energy Efficiency. In *U.S. EPA Technical Report*, 2007.

[4] A Comparison of SSD, ioDrives, and SAS rotational drives usi ng TPC-H Benchmark. Technical Report White Paper, HP Development Company, 2010.

[5] D. Abts, M. R. Marty, P. M. Wells, P. Klausler, and H. Liu. Energy Proportional Datacenter Networks. In *ISCA*, 2010.

[6] R. Ayoub, K. R. Indukuri, and T. S. Rosing. Energy Efficient Proactive Thermal Management in Memory Subsystem. In *ISLPED*, 2010.

[7] L. A. Barroso and U. Holzle. The Case for Energy-Proportional Computing. *IEEE Computer*, 2007.

[8] C. Belady. In the Data Center, Power and Cooling Costs More than the IT Equipment it Supports. *Electronics Cooling*, 23(1), 2007.

[9] K. G. Brill. Data Center Energy Efficiency and Productivity. In *The Uptime Institute - White Paper*, 2007.

[10] C. Clark, K. Fraser, S. Hand, J. G. Hansen, E. Jul, C. Limpach, I. Pratt, and A. Warfield. Live Migration of Virtual Machines. In *Symposium on Networked Systems Design and Implementation*, 2005.

[11] D. J. DeWitt. The Wisconsin Benchmark: Past, Present, and Future. In J. Gray, editor, *The Benchmark Handbook for Database and Transaction Systems (2nd Edition)*. Morgan Kaufmann, 1993.

[12] C. Gunaratne, K. Christensen, B. Nordman, and S. Suen. Reducing the Energy Consumption of Ethernet with Adaptive Link Rate. *IEEE Transactions on Computers*, 2008.

[13] S. Gurumurthi, A. Sivasubramaniam, M. J. Irwin, N. Vijaykrishnan, M. Jane, I. N. Vijaykrishnan, M. Kandemir, T. Li, and L. K. John. Using Complete Machine Simulation for Software Power Estimation: The SoftWatt Approach. pages 141–150, 2001.

[14] J. Hamilton. Where Does Power Go In DCs and How To Get It Back? *http://mvdirona.com/jrh/TalksAndPapers/JamesRH_DCPowerSavingsFooCamp08.ppt*, 2008.

[15] J. Hamilton. Cooperative Expendable Micro-slice Servers (CEMS): Low Cost, Low Power Servers for Internet-Scale Services. In *CIDR*, 2009.

[16] S. Harizopoulos, M. A. Shah, J. Meza, and P. Ranganathan. Energy Efficiency: The New Holy Grail of Database Management Systems Research. In *CIDR*, 2009.

[17] J. G. Koomey. Estimating Total Power Consumption by Servers in the US and the World. *http://enterprise.amd.com/Downloads/svrpwrusecompletefinal.pdf*, 2007.

[18] W. Lang, R. Kandhan, and J. M. Patel. Rethinking Query Processing for Energy Efficiency: Slowing Down to Win the Race. *http://cs.wisc.edu/~jignesh/eopt_full.pdf*, 2011.

[19] W. Lang and J. M. Patel. Towards Eco-friendly Database Management Systems. In *CIDR*, 2009.

[20] W. Lang and J. M. Patel. Energy Management for MapReduce Clusters. In *VLDB*, 2010.

[21] W. Lang, J. M. Patel, and J. F. Naughton. On Energy Management, Load Balancing and Replication. In *SIGMOD Record*, 2009.

[22] W. Lang, J. M. Patel, and S. Shankar. Wimpy Node Clusters: What About Non-Wimpy Workloads? In *DaMoN*, 2010.

[23] J. Leverich and C. Kozyrakis. On the Energy (In)efficiency of Hadoop Clusters. In *HotPower*, 2009.

[24] S. Manegold, P. A. Boncz, and M. L. Kersten. Generic Database Cost Models for Hierarchical Memory Systems. In *VLDB*, 2002.

[25] C. D. Patel and A. J. Shah. Cost Model for Planning, Development and Operation of a Datacenter. *http://www.hpl.hp.com/techreports/2005/HPL-2005-107R1.pdf*.

[26] P. Ranganathan, P. Leech, D. Irwin, and J. Chase. Ensemble-level Power Management for Dense Blade Servers. In *ISCA*, 2006.

[27] S. Rivoire, M. A. Shah, P. Ranganathan, and C. Kozyrakis. JouleSort: a balanced energy-efficiency benchmark. In *SIGMOD*, 2007.

[28] L. D. Shapiro. Join processing in database systems with large main memories. *ACM Trans. Database Syst.*, 11(3):239–264, 1986.

[29] M. E. Tolentino, J. Turner, and K. W. Cameron. Memory MISER: Improving Main Memory Energy Efficiency in Servers. In *IEEE Transactions on Computers*, 2009.

[30] N. Tolia, Z. Wang, M. Marwah, C. Bash, P. Ranganathan, and X. Zhu. Delivering Energy Proportionality with Non Energy-Proportional Systems - Optimizing the Ensemble. In *HotPower*, 2008.

[31] D. Tsirogiannis, S. Harizopoulos, and M. A. Shah. Analyzing the Energy Efficiency of a Database Server. In *SIGMOD*, 2010.

[32] Z. Xu, Y.-C. Tu, and X. Wang. Exploring Power-Performance Tradeoffs in Database Systems. In *ICDE*, 2010.

Energy Efficient Data Intensive Distributed Computing

Weiwei Xiong
University of California San Diego
La Jolla, CA 92093

Aman Kansal
Microsoft Research
Redmond, WA 98052

Abstract

Many practically important problems involve processing very large data sets, such as for web scale data mining and indexing. An efficient method to manage such problems is to use data intensive distributed programming paradigms such as MapReduce and Dryad, that allow programmers to easily parallelize the processing of large data sets where parallelism arises naturally by operating on different parts of the data. Such data intensive computing infrastructures are now deployed at scales where the resource costs, especially the energy costs of operating these infrastructures, have become a significant concern. Many opportunities exist for optimizing the energy costs for data intensive computing and this paper addresses one of them. We dynamically right size the resource allocations to the parallelized tasks such that the effective hardware configuration matches the requirements of each task. This allows our system to amortize the idle power usage of the servers across a larger amount of workload, increasing energy efficiency as well as throughput. This paper describes why such dynamic resource allocation is useful and presents the key techniques used in our solution.

1 Introduction

Data intensive distributed computing platforms such as MapReduce [4], Dryad [7], and Hadoop [5], offer an effective and convenient approach to solve many problems involving very large data sets, such as those in web-scale data mining, text data indexing, trace data analysis for networks and large systems, machine learning, clustering, machine translation, and graph processing. Their usefulness has lead to widespread adoption of such data intensive computing systems for data analysis in many large enterprises, and server clusters consisting of tens of thousands of servers now host these platforms. At these scales, the cost of resources consumed by the data parallel computing system becomes very significant. Both the operational costs such as energy consumption, and the capital expense of procuring the servers and supporting infrastructure are thus crucial to optimize.

Several opportunities exist in data intensive distributed computing systems to maximize the amount of work performed by a given set of resources and to reduce the energy consumed for performing a given amount of work. A first opportunity is to schedule the processing jobs in a manner that is best suited to the data placement and network requirements of each job, resulting in improved utilization of the infrastructure [8]. Another possibility is to reduce the amount of energy spent on storage. While redundant copies of data are required for reliability, machines hosting some of the redundant copies can be shut down if the throughput requirement and data consistency constraints permit [9]. A third opportunity arises in adapting the resource configuration to the

Bulletin of the IEEE Computer Society Technical Committee on Data Engineering

resource needs of the currently processing jobs. In this paper we consider this last opportunity in detail and show how energy use can be reduced and throughput increased for a given set of tasks and cluster size.

If the server cluster was to always run a single, pre-specified, data processing job, then it would be straightforward, though possibly tedious, to configure the server hardware to match the requirements of that processing task efficiently. However, in practice a single cluster is set up for multiple data analysis jobs. In fact, sharing the cluster yields significant advantages in efficient resource utilization since one job is unlikely to always have sufficient workload to justify a dedicated cluster. The complication that arises due to sharing is that the server configuration is not optimized to meet the resource requirements of any single job. At run time, depending on the characteristics of the job, certain resources such as the number of disks spinning or active servers, would be left unused, wasting valuable power and also the capital expense of the infrastructure itself. Since it is impractical to change the hardware dynamically, we explore software mechanism to optimize resource utilization and improve overall throughput, using Dryad as the underlying data intensive computing platform.

Specifically, we introduce a mechanism, referred to here as Energy Efficient Dryad (e-Dryad, for short), for fine grained allocation of cluster resources to incoming jobs that *right-sizes* the resource configuration to a job's requirements. Current job schedulers for data intensive distributed computing platforms allocate a set of servers (sometimes referred to as nodes) to each job. The number of servers allocated depends on the job's requirements, its priority, and the number of currently unallocated servers. However, the hardware configuration of the servers may not be matched to the processing demands of the job. Our proposed mechanism divides the server resources among jobs to match their hardware resource usage characteristics, leading to a more efficient and better balanced resource configuration for each job.

While e-Dryad improves the overall throughput compared to existing schedulers, the amount of instantaneous resources allocated to a job may in fact be smaller in our proposed approach. This does cause the latency of processing to to be higher in certain situations. However, the increased throughput implies that the jobs spend less time waiting for their turn in the queue and hence the overall completion time can in fact still be lower depending on the position in the queue.

We describe the design of e-Dryad, along with the challenges addressed and mechanisms used, in Section 2. The proposed system is evaluated with real workloads on an experimental server cluster, and the evaluation results are presented in Section 3. Section 4 discusses related problems and possible extensions. Section 5 summarizes the prior works in the area and Section 6 concludes.

2 Resource Allocation System Design

The problem of inefficient resource utilization arises because the hardware is not configured to match the resource requirement of every job. We explain this problem in more detail below and then present a solution to address it.

2.1 Problem Description

We describe the problem in the context of Dryad, since that is the platform used in our prototype, but the problem conceptually exists in most data intensive computing platforms. A common operational aspect of such platforms includes a scheduler that accepts computing jobs and assigns them to servers. A *job* here refers to a data processing program that includes at least one processing phase that can be massively parallelized. It may be followed by additional processing stages that further process or integrate the results from the parallelized phase or phases. In Dryad, each job is comprised of a root task that manages the actual processing tasks, referred to as worker tasks. The worker tasks accesses data stored on the server cluster's constituent computers themselves. The *scheduler* queues incoming jobs and as servers become available after finishing previously running jobs, they are assigned to new jobs from the queue. A good scheduler will assign servers to a job's worker tasks such

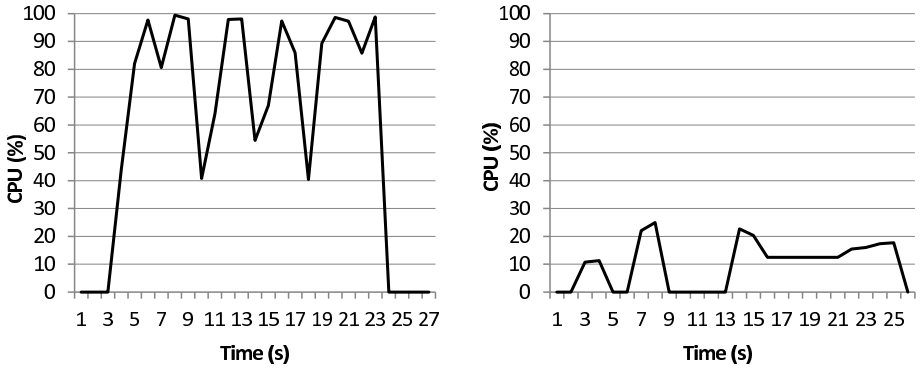
that the data they access from storage is on the assigned server itself or relatively close by within the network topology [8].

The parallelized phase is the one that requires the most resources and we primarily focus on the worker tasks comprising this phase. The worker tasks within a job are largely homogeneous in terms of computational resource requirement but the worker tasks from different jobs can be very different. Consider the following two examples of real world applications that are well suited to parallelized data intensive computation:

Word Statistics. This application computes the number of times different words occur in a very large corpus of text, such as a large number of web pages, stored across multiple machines [11]. Such tasks are commonly required for web scale indexing.

Terasort. The Terasort application sorts a very large number (billions) of records using case insensitive string comparisons on a 10-byte key [10]. This is a benchmark used for parallel computing.

Figure 1 shows the resource utilization for CPU, averaged across multiple servers, for a job from each of the above applications.



(a) Word Statistics.

(b) Terasort.

Figure 1: Resource usage of two Dryad jobs, averaged across multiple machines allocated during the job's execution.

It is clear from the figure that the Word Statistics job has a significantly higher CPU utilization. A similar figure may be plotted for IO bandwidth usage and that would show that the Terasort job has a higher IO usage and is not bottlenecked on CPU. Each job thus leaves some resource wasted. This is a problem due to two reasons. Firstly, the cost of the infrastructure directly depends on the number of servers required, and hence if the servers are left unused, that results in wasted resources. In the above example, the unused CPU could potentially have been used by other jobs waiting in the queue. Secondly, the above usage causes energy wastage. A large fraction of the power usage of a server, often as high as 60%, is spent to simply power on a server, and is referred to as idle power. This power is spent even if the CPU or other resources are not used by any job. Clearly, if the CPU resource left unused by Terasort were to be used by another job, the idle energy use will not increase and only a small amount of additional energy would be required. When resource utilization is high, the idle energy is amortized over greater amount of work done, resulting in more efficient operation.

Our goal in building e-Dryad is to address precisely the above two problems by improving resource utilization leading to better amortization of idle energy, and higher throughput from the given number of servers.

2.2 Resource Allocation Methodology

The basic idea for resource allocation in e-Dryad is to place worker tasks from multiple jobs on each machine, in a manner that improves the resource utilization. This implies that the worker tasks placed together must be carefully chosen. If two worker tasks are both bottlenecked on the resource, such as CPU, then placing them together would simply increase the run time of each task without improving resource utilization significantly. Instead, if two worker tasks with complimentary resource utilizations are placed on the same server, then the overall resource utilization would improve. Suppose one task is bottlenecked on CPU, while another is bottlenecked on the storage bandwidth, then placed together, they could utilize both resources maximally, leading to better amortization of the idle power. An important consideration here is that each task needs both storage and CPU resource. The CPU intensive task is also ultimately a data intensive program and would need some storage bandwidth and the storage intensive task obviously needs some CPU capacity to perform its work. This implies that the resource allocation mechanism should ensure that each task gets resources in its desired proportion. This may be viewed as re-balancing the hardware configuration to match the job requirements.

The e-Dryad resource allocation mechanism, based on the above basic idea, is shown in Figure 2, and described below.

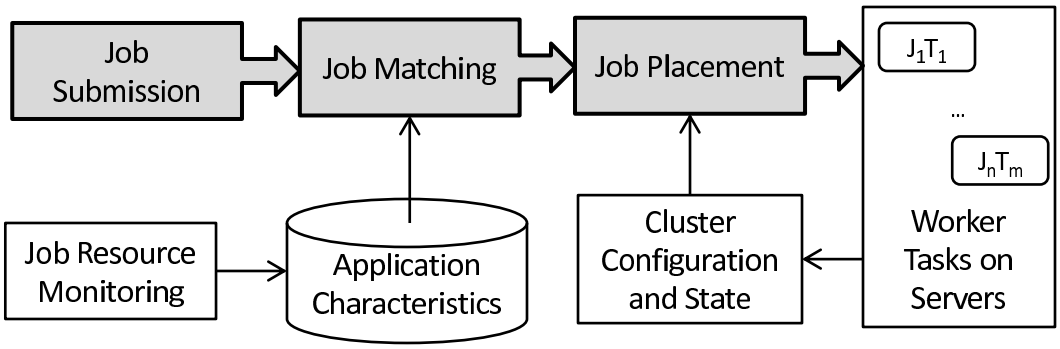


Figure 2: e-Dryad resource allocation block diagram

All incoming job requests are accepted through the *job submission* system as usual. In Dryad, jobs are often submitted through a web interface provided for the Dryad cluster, by uploading a job specification.

2.2.1 Job Matching

The first crucial step for e-Dryad is matching job resource usage characteristics to determine complementary jobs that are well suited to be placed together. In data intensive computing clusters, while new jobs are submitted everyday to process new data sets, the nature of the jobs themselves does not change as frequently. Rather, a slowly evolving set of applications is hosted by the cluster. We assume that repeat instances of the same application each have a unique job ID but are labeled with an application ID that allows identifying them as different instances of the same application. The e-Dryad resource allocation system maintains a set of *application characteristics*, shown as an input to the job matching block in the figure.

The application characteristics are acquired by monitoring the resource usage of multiple jobs belonging to the same application. The characteristics may be initialized based on the first instance, and updated over time as additional jobs from that application are observed. Monitoring of job resource usage is performed by the *job resource monitoring* block shown in the figure. This block itself is a distributed monitoring system that uses the Dryad infrastructure to correctly track the distributed resource usage of each job across multiple machines. The implementation of this system in our prototype is discussed in Section 2.3.

Based on the application characteristics, the job matching block identifies the resource for each application that shows the heaviest usage. This resource is likely the bottleneck for this application and hence the other resources on a machine running this application jobs would have spare capacity. Note that the determination of the bottleneck is only an estimate since the heavy resource usage may be an effect of data placement and network topology that may change based on job placement. The estimate improves over time as a well designed scheduler attempts to place each job's tasks efficiently in terms of data placement and the application characteristics are then dominated by the true bottlenecks. Matched jobs are ones that have a different bottleneck resource, and can hence be placed together to reduce the idle power consumption per unit work.

2.2.2 Job Placement

Placement of jobs determines the actual allocation of resources to the worker tasks of the jobs. Once the job matching block has determined which jobs are suitable to be placed together to optimize resource utilization, the job placement block assigns the worker tasks to the servers in the cluster based on the current state of the cluster in terms of server availability and other scheduling considerations. The cluster configuration and state information is directly measured from the cluster management interfaces.

Multiple different job placement heuristics may be employed based on the nature of jobs and desired performance and energy optimization objectives. We assume that the scheduling mechanism is the same as followed in Dryad and abstract it out, focusing only on the placement. A key resource allocation decision for placement after matching jobs are identified, is to determine how the resources of the server should be divided among the tasks. In our prototype we focus on the two resources that affect throughput the most for data intensive computation in a well scheduled system: processor time and storage bandwidth. The following two heuristics are employed for resource allocation:

Equal Sharing. In the equal sharing approach, two tasks from different jobs that are placed on a common server, are each assigned an equal share of the CPU time. One of these tasks is CPU intensive and leaves significant storage bandwidth unused. The other task being storage intensive, uses up a large fraction of the storage bandwidth even with half its usual CPU allocation since it is not bottlenecked on the CPU. The overall amount of work performed increases leading to improved amortization of idle energy.

Proportional Sharing. In the proportional sharing approach, rather than dividing the CPU resource equally, it is divided in a ratio proportional to the average CPU utilization known for the two jobs. While the CPU usage can vary across tasks and over the run time of a job, the division in proportion to the average usage captures the resource requirements better than equal sharing. In this case, resource utilization is further improved since the tasks bottlenecked on CPU can use up a greater portion of the CPU while the other task can likely get a sufficient amount of storage bandwidth even with a smaller share of the CPU. The proportional sharing does not guarantee that resource will be maximally utilized since it is unlikely that the resource requirements of different tasks are matched exactly in different resource dimensions to use each resource to a 100%.

The relative performance of the two approaches is studied experimentally in Section 3.

2.3 Implementation

The implementation of e-Dryad is based on a Dryad cluster installed on Windows Server 2008 systems. The operation of e-Dryad requires detailed resource usage tracing for job characterization as well as experimental verification. The job resource monitoring component, referred to in Figure 2 earlier, is implemented as follows. Tracking the resource usage of a single job requires understanding the placement of the tasks comprising the job on various servers, and then tracking the resource utilization of each task on each of those servers. The

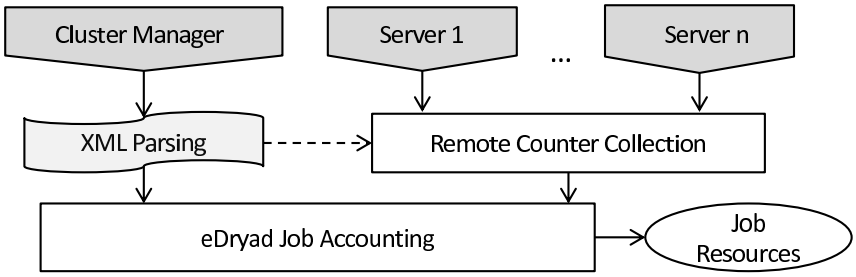


Figure 3: e-Dryad job resource monitoring.

monitoring system in e-Dryad is implemented as shown in Figure 3. The cluster manager in Dryad produces an XML formatted description of job layout across the servers. e-Dryad fetches that description to infer the servers allocated to each job at each time step. Resource usage at each server is monitored for the Dryad processes using Windows kernel provided performance counters. Windows system management enables these performance counters to be remotely accessed and e-Dryad accesses the counters for each of the processes from multiple remote servers. Based on the job to server mapping, it accounts for the resource usage of each job.

The system mechanism used to assign the right amount of resource is based on server virtualization. We create virtual machines on each server and divide the physical resources among virtual machines in a proportion that suits the job placement across servers. For instance, going back to the example shown in Figure 1, suppose quad core servers were used, one virtual machine could be assigned three processor cores and allocated to Word Statistics job, while the other one could be given one processor core and allocated to the Terasort job. Clearly, the bandwidth on the machine would also get split among the two VMs. Terasort may use a greater share of the bandwidth based on its needs. The virtualization based mechanism integrates seamlessly with the Dryad scheduler since from the perspective of the scheduler, each virtual machine is a different server to be assigned to a job. However, virtualization is only one of the options to allocate resources in the correct proportion. Equivalent functionality could also be achieved by simultaneously assigning the same server to multiple jobs and creating separate processes to host each job. The Dryad scheduler may be modified to incorporate e-Dryad based job characteristics in determining the job allocations across servers. The operating system on each allocated server could then be informed to allocate resources in the correct proportion to the different processes running the worker tasks from different jobs on each server.

3 Evaluation

We evaluate e-Dryad on a 32 node cluster, with Intel Xeon 2.5GHz processors. We use data intensive applications found in Dryad literature [11], including the two applications mentioned in Section 2.1, that are generally representative of data indexing tasks. The jobs are specified using DryadLINQ [10].

The energy use and savings for the equal and proportional sharing policies is shown in Table 1, and also compared to that of placement without e-Dryad. The improved job placement leads to significant energy savings, that for large scale data intensive computing clusters result in significant cost savings.

While the energy savings are definitely a desirable feature of e-Dryad and come along with increased overall throughput, they are also associated with a potential increase in latency, in terms of job execution time measured from the instant that a job is allocated on the cluster. The overall latency, including the queuing delay and processing delay may not be increased for most jobs. However, high priority jobs, that were not subject to significant queuing delays in the default design, could suffer increased latency. This is illustrated in Figure 4

Placement	Energy Use (J)	Energy Saving (%)
Dryad-default	35295	–
Equal Sharing	29303	17.01
Proportional Sharing	28282	20.40

Table 1: Energy Savings achieved by e-Dryad in experimental tests.

taking two jobs, one from the word statistics application and the other from the Terasort application. The figure

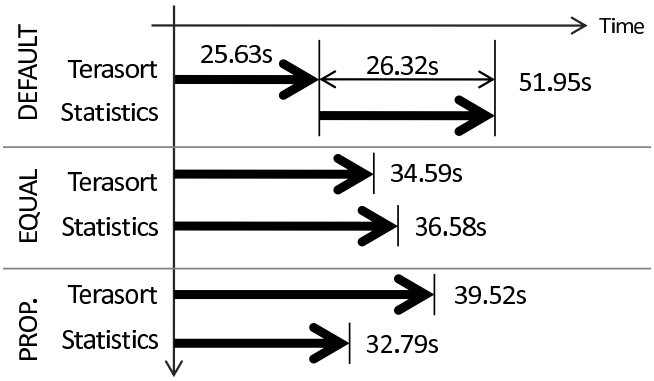


Figure 4: Illustration of effect on job latency with e-Dryad resource allocation. The "Prop." label refers to the proportional sharing approach.

shows the execution of these two jobs along the time line. The top segment shows how execution proceeds in the default system and the lower two segments show how execution proceeds with e-Dryad policies. Each job consumes a shorter processing time in the default system but the total execution time on the system is longer. For the first job the latency is smaller but the second job suffers a longer latency due to longer queuing delay. The simultaneous execution in the other two approaches allows the two jobs to exit the system sooner, leading to a lower total energy cost and higher throughput. The latency of the Terasort job has however increased by 34.9% in the equal sharing case and 54.2% in the proportional sharing case, due to reduced instantaneous resources granted.

The resource utilization also improves in the system. Taking the above run as an example, the CPU utilization is plotted in Figure 5. The utilizations shown are averaged across the 8 worker tasks comprising each job. As seen, the utilization is higher during the run with e-Dryad, and the jobs finish sooner (at 39s instead of 52s).

4 Discussion

The e-Dryad approach for resource allocation in data intensive computing highlights an important opportunity for improving energy costs and infrastructure expense. However, there are several additional features and design extensions that may be considered to the initial design presented above. We discuss some of these extensions below.

Temporal Profiling: In e-Dryad, we considered job level characteristics. A job has multiple stages of processing. The worker tasks perform different types of computation in each stage and more accurate resource utilization characteristics would emerge if the worker tasks of different stages were characterized individually. Further, even among the homogeneous parallel tasks of a single stage, resource usage may differ because of

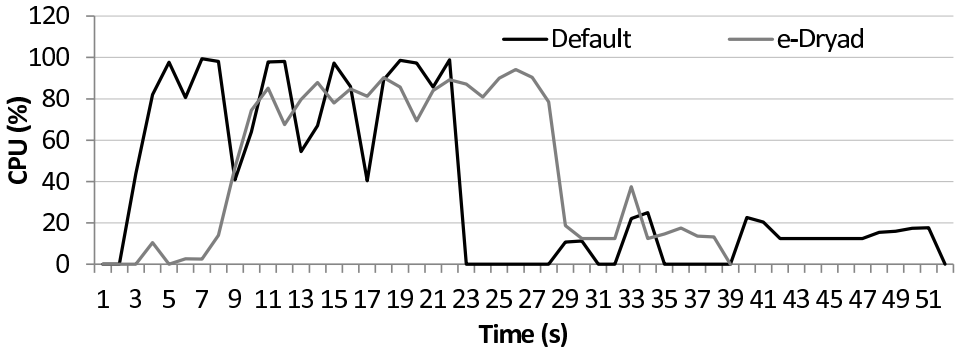


Figure 5: Resource utilization with and without e-Dryad.

placement. Some tasks may be accessing only local data, some could be fetching their data from remote machines, and some could be using a mix of local and remote data. Differences in network data rates available to different tasks will lead to differences in each task's server resource usage and considering such effects can help further optimize the e-Dryad resource allocation. However, in pactice, acquiring such detailed characteristics, some of which depend on run time placement, is often difficult.

Latency Constraints: Joint placement of jobs for improved resource efficiency sometimes requires reducing the instantaneous resource allocation for a job, and we showed in evaluations that the processing latency of a job, measured from the time it is allocated on the cluster, may be increased. This may happen for instance, when the storage bandwidth on the machines containing a job's data is shared with another job to improve the CPU utilization. In certain scenarios, we may wish the first job to have the entire storage bandwidth to minimize its latency. To address such scenarios, e-Dryad resource allocation may be modified to incorporate priorities and the high priority jobs may be allocated greater resources, at the cost of overall cluster efficiency.

Throughput Trade-off: We saw for the experimental system configurations and resource allocation policies that energy efficiency also improved throughput, though sometimes at the cost of higher latency. There are resource allocation policies when energy efficiency may be improved further by reducing the throughput. Given that the idle power of keeping a server powered on is high, we optimized energy efficiency by maximally utilizing available disk throughput. However, consider a policy where some of the servers are turned off, to ensure that all data is available but multiple redundant copies may not be on line. This will allow for reducing the idle energy costs but also reduce the available IO bandwidth. If all the jobs are processor intensive (IO bandwidth was not the bottleneck) and the reduced IO bandwidth does not cause the processors on powered on servers to stall on data access, then this reduced server configuration will be better aligned to job resource requirements and hence more energy efficient. In such a policy, that exploits additional resource management knobs such as server shutdown, both throughput and latency may be traded-off for improved energy efficiency.

Dynamic Allocations: The job allocation policy discussed in our prototype assumes that the resource allocation is static for each job. This is reasonable when the job queues are typically never empty and jobs are served in the order received. However, in cases where high prioity jobs require pre-emption of currently allocated jobs, or the arrival of a new job opens up opportunities for more efficient resource allocations than the current allocation, it becomes worthwile to consider dynamic resource allocations. If the tasks corresponding to a running job are stopped, the work performed until that time would be wasted. Dynamic allocation would thus require the capability to suspend a running task by saving the intermediate results and task state.

Multiple Resources: The heuristics presented for job matching and placement considered only two resources and shared a server among at most two worker tasks from matched jobs. The design may be extended to additional resources and may place more than two tasks on the shared server to further improve resource usage in even more dimensions leading to lower energy use per unit throughput.

5 Related Work

The problem of energy efficiency in data intensive computing platforms is a relatively new area of research. Another work directly addressing this problem, though through a different approach is presented in [9]. The placement of data among the servers is optimized such that certain servers may be placed in low power states when feasible.

Simultaneous placement of multiple workloads on the same servers is also considered in several works focused on virtual machine consolidation, such as [1,6]. However, the key problem addressed in virtual machine consolidation is efficient packing along pre-specified resource dimensions such as CPU time and memory space. The bottleneck resources of data intensive computing tasks, such as storage bandwidth, are not considered. Also, the resource requirements are assumed specified, while in our work we employ distributed monitoring to characterize the relevant resource characteristics of applications. The e-Dryad method also trades off latency for energy efficiency while virtual machine consolidation typically focuses on optimizing performance.

Another set of related works is found in optimizing the performance of data intensive computing platforms such as by improving the schedulers for Dryad or MapReduce [3, 8, 12] for addressing various challenges such as efficient handling of failed tasks, adaptation to data and network topology, and better utilization of cache hierarchy. The optimization of the jobs themselves through static analysis was proposed in [2]. Our work is complementary to the above works and could potentially be added on as a resource re-configuration layer with many of their scheduling policies. e-Dryad improves resource efficiency by matching the resource allocation to the resource configurations best suited to the computational tasks.

6 Conclusions

We described an important opportunity for improving the energy efficiency of data intensive computing platforms. Hardware configurations are rigid and cannot be right sized for the characteristics of each application that may be executed on the platform. We introduced a software mechanism to dynamically right size the hardware configuration by modifying the resource allocations made to each of the worker tasks of the jobs. The resource allocation method learns the characteristics of the jobs and determines the appropriate allocations to efficiently utilize the resources. This results in both lower energy consumption as well as higher throughput, and we saw through evaluations that the savings were significant. The changes in resource allocation do affect the latency performance of the jobs and we illustrated this trade-off with an example execution. The proposed methods demonstrate an important opportunity to improve energy efficiency and help reveal several additional challenges and extensions that may help improve data intensive computing infrastructures.

References

[1] Norman Bobroff, Andrzej Kochut, and Kirk Beaty. Dynamic Placement of Virtual Machines for Managing SLA Violations. In *Integrated Network Management, 2007. IM '07. 10th IFIP/IEEE International Symposium on*, pages 119–128, 2007.

[2] Michael J. Cafarella and Christopher Ré. Manimal: relational optimization for data-intensive programs. In *Procceedings of the 13th International Workshop on the Web and Databases*, WebDB '10, pages 10:1–10:6, 2010.

[3] Rong Chen, Haibo Chen, and Binyu Zang. Tiled-mapreduce: optimizing resource usages of data-parallel applications on multicore with tiling. In *Proceedings of the 19th international conference on Parallel architectures and compilation techniques*, PACT '10, pages 523–534, 2010.

[4] Jeffrey Dean and Sanjay Ghemawat. Mapreduce: a flexible data processing tool. *Commun. ACM*, 53:72–77, January 2010.

[5] Apache hadoop. `http://hadoop.apache.org/`.

[6] Fabien Hermenier, Xavier Lorca, Jean-Marc Menaud, Gilles Muller, and Julia Lawall. Entropy: a consolidation manager for clusters. In *Proceedings of the 2009 ACM SIGPLAN/SIGOPS international conference on Virtual execution environments*, VEE '09, pages 41–50, 2009.

[7] Michael Isard, Mihai Budiu, Yuan Yu, Andrew Birrell, and Dennis Fetterly. Dryad: distributed data-parallel programs from sequential building blocks. *SIGOPS Oper. Syst. Rev.*, 41:59–72, March 2007.

[8] Michael Isard, Vijayan Prabhakaran, Jon Currey, Udi Wieder, Kunal Talwar, and Andrew Goldberg. Quincy: fair scheduling for distributed computing clusters. In *Proceedings of the ACM SIGOPS 22nd symposium on Operating systems principles*, SOSP '09, pages 261–276, 2009.

[9] Jacob Leverich and Christos Kozyrakis. On the energy (in)efficiency of Hadoop clusters. *SIGOPS Oper. Syst. Rev.*, 44:61–65, March 2010.

[10] Yuan Yu, Michael Isard, Dennis Fetterly, Mihai Budiu, Úlfar Erlingsson, Pradeep Kumar Gunda, and Jon Currey. Dryadlinq: a system for general-purpose distributed data-parallel computing using a high-level language. In *Proceedings of the 8th USENIX conference on Operating systems design and implementation*, OSDI'08, pages 1–14, 2008.

[11] Yuan Yu, Michael Isard, Dennis Fetterly, Mihai Budiu, Ulfar Erlingsson, Pradeep Kumar Gunda, Jon Currey, Frank McSherry, and Kannan Achan. Some sample programs written in dryadlinq. Technical Report MSR-TR-2009-182, Microsoft Research, December 2009.

[12] Matei Zaharia, Andy Konwinski, Anthony D. Joseph, Randy Katz, and Ion Stoica. Improving mapreduce performance in heterogeneous environments. In *Proceedings of the 8th USENIX conference on Operating systems design and implementation*, OSDI'08, pages 29–42, Berkeley, CA, USA, 2008. USENIX Association.

Power Based Performance and Capacity Estimation Models for Enterprise Information Systems

Meikel Poess
Oracle Corporation
Meikel.Poess@oracle.com

Raghunath Nambiar
Cisco Systems, Inc.
rnambiar@cisco.com

Abstract

Historically, the performance and purchase price of enterprise information systems have been the key arguments in purchasing decisions. With rising energy costs and increasing power use due to the ever-growing demand for compute capacity (servers, storage, networks etc.), electricity bills have become a significant expense for todays data centers. In the very near future, energy efficiency is expected to be one of the key purchasing arguments. Having realized this trend, the Transaction Processing Performance Council has developed the TPC-Energy specification. It defines a standard methodology for measuring, reporting and fairly comparing power consumption of enterprise information systems. Wide industry adaption of TPC-Energy requires a large body of benchmark publications across multiple software and hardware platforms, which could take several years. Meanwhile, we believe that analytical power estimates based on nameplate power is a useful tool for estimating power consumption of TPC benchmark configurations as well as enterprise information systems. This paper presents enhancements to previously published energy estimation models based on the TPC-C and TPC-H benchmarks from the same authors and a new model, based on the TPC-E benchmark. The models can be applied to estimate power consumption of enterprise OLTP and Decision Support systems.

1 Introduction

In the last decades, performance and purchase price of hardware and software were the dominant concerns of data center managers. Even though the performance of hardware and software have improved substantially, and their price have dropped significantly over the years, the competitive business environment continued to demand more and more performance and compute capacity. Consequently, over the last few years, especially due to the increase in energy prices, the cost of owning and maintaining large data centers has become a serious concern for data center managers.

Hardware and software vendors have been investing in the development of energy efficient versions of their systems. Low power versions of system components have been developed or made power aware, such as processors that are equipped with demand-driven clock speed adjustments. Reducing power consumption is also at the top of the priority list of government agencies as they challenge data center managers and system developers to reduce power consumption globally. The U.S. Environmental Protection Agency (EPA) has been

Bulletin of the IEEE Computer Society Technical Committee on Data Engineering

Table 1: Adaption Rate of Energy Reporting in TPC Results.

Benchmark	Total number of results	Results since December 2009	Results with energy Information	Adaption Rate
TPC-C	750[1]	10	3	1.3 %
TPC-E	42	14	2	14 %
TPC-H	168	10	4	40 %

1. This includes benchmark results from revisions 1, 2, 3 and 5.

working with various organizations to identify ways in which energy efficiency can be measured, documented, and implemented, not only in data centers, but also in the equipment they house [3]. Furthermore, standard organizations have responded to the growing demand for energy benchmarks such as the Transaction Processing Performance Council (TPC), the Standard Performance Evaluation Corporation (SPEC) and the Storage Performance Council (SPC) [18]. All major computer and system vendors are members of these organizations. Each of these consortia addresses different, often unique aspects of computer system performance. While SPEC and SPCs benchmarks focus on subsets of large enterprise information systems, TPCs benchmarks address complex On Line Transaction Processing (OLTP) and Decision Support (DS) systems, often involving hundreds of processors and thousands of disk drives.

The TPC offers currently two benchmarks to measure OLTP systems, namely TPC-C [23] and TPC-E [9], and one to measure DS performance, TPC-H [18]. TPC benchmarks are widely accepted in the industry for disseminating objective and verifiable performance data using well designed, long lasting benchmarks. In the last 18 years over 750 TPC-C benchmark results were published across a wide range of hardware and software platforms, representing the evolution of transaction processing systems [15]. TPC-C results were published by over two dozen unique vendors and over a dozen database platforms. In its first three years 42 TPC-E were published. In the last eight years 168 TPC-H results were published. System vendors publishing benchmark results are referred to as benchmark sponsors.

In 2009 the TPCs step to add an optional power measurement methodology (TPC-Energy [5]) uniformly to all its benchmarks marked a significant milestone towards designing and deploying standards to measure energy efficiency in servers and storage sub-systems. So far three TPC-C, two TPC-E and four TPC-H results have been published with power consumption information (see Table 1). Depending on the benchmark specification the adaption rate of benchmark publications, which include energy measurements, varies between 1.3 % and 40 %. It is expected that the adaption rate will increase over the next years, as seen with previous benchmarks.

As of now, however, the majority of TPC-C, TPC-E and TPC-H benchmark results are still published without any energy consumption information. This is not surprising because it usually takes several years for system vendors to implement and tune their systems for new benchmarks. Wide industry adaption of TPC-Energy requires a large number of benchmark publications across multiple software and hardware platforms. Meanwhile, the authors believe that previously published analytical power estimation models for x86 based systems [14],[13], which are based on nameplate power consumption, are useful tools for TPC-C and TPC-H benchmark publications, whose full disclosure report (FDR) supplies all necessary information for these models.

Each server and storage sub-system is tagged with a nameplate power rating, which is typically estimated by its manufacturer simply by adding up the worst case power, drawn by all components in a fully configured system. The purpose of nameplate rating is to aid the buyer of a particular component in provisioning the power infrastructure for it. In general, the nameplate rating is a very conservative number that is guaranteed not to be reached. Because of safety and economic reasons components commonly use only a percentage of the power reported in their nameplate specification during maximum load. Estimates of this percentage vary, but 20 % to 30 % are not uncommon. Personal computers are even reported to use two to four times less power than specified in their nameplates. Hence, provisioning power based solely on the nameplate specification results in drastically over sizing power infrastructure and supporting systems [11].

In this paper we present updated versions of our analytical power estimation models for OLTP and DS sys-

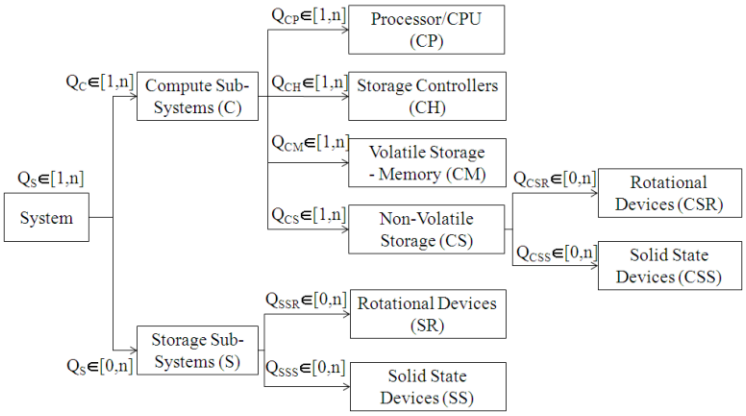


Figure 1: Hierarchy of System Components of the Analytical Power Estimation Model

tems, which are based on TPC-C and TPC-H benchmarks. Recent trends, such as the use of solid state memory technologies and in-memory database technologies due to the deployment of very large memory configurations require some modifications to our models, which were previously published in [14] and [13]. Additionally, we extend our suite of analytical power estimation models with one that is based on the TPC-E benchmark. We verify our models with measurements taken from fully scaled, optimized and published TPC-C, TPC-H and TPC-E configurations, including client systems, database server, and storage sub-system. Although estimated, the numbers obtained from our models are very close to those of the measured configurations. Hence, they can be applied to historical TPC data to perform power consumption trend analysis, they can help identifying the most power intensive system components, they can enhance dozens of performance and sizing tools that are based on TPC workloads and, to some extent, they can also be applied by data center designers to estimate power consumption of OLTP and DS systems that show system utilizations similar to those of TPC-C/E and TPC-H.

The remainder of this paper is structured as follows. Section 2 introduces a generalized power consumption model that can be applied to all current TPC benchmarks. This model generalizes and refines the previously published power estimation models [14] and [13]. Section 3 shows how this model can be applied to OLTP workloads and Section 4 shows how it can be applied to decision support workloads. Section 5 validates the power estimation models using TPC benchmark publications. The paper is summarized in Section 6.

2 Power Estimation Using Nameplate Information

Our analytical power consumption models, presented in [14] and [13], are based on the assumption that the peak power consumption of an entire system during steady state can be derived from the aggregate of the nameplate power consumptions of its individual components. Each model follows the same general approach: The nameplate power information of major system components, such as processor (CPU), volatile storage, internal non-volatile storage devices, i.e. rotational disks and solid state memory, and external storage sub-systems, i.e. enclosures with non-volatile storage devices, are aggregated discounting the nameplate overhead. Additional power of supporting components, such as motherboards and fans, is calculated with a combination of a fixed overhead and a percentage of the power consumption of the components they support. The models do not account for the power necessary for the air conditioning systems of data centers.

Figure 1 shows the hierarchy of system components that are used in our power estimation models. Each component in this hierarchy is abbreviated with up to three capital letters, as indicated in parenthesis. TPC

systems may consist of two types of sub-systems, namely compute sub-systems (C) and storage sub-systems (S). In case of a clustered system and systems that have multi-tier architectures, there can be multiple compute sub-systems. We also refer to the compute sub-systems as servers. Each server consists of one or more compute units (CP), i.e. processors/CPUs, a number of storage controllers to connect to external storage enclosures (CH), some sort of volatile storage, usually memory DRAM DIMMs (CM), some non-volatile memory (CS), traditionally rotational devices (CSR), but recently also Solid State Devices (CSS) and supporting components, such as the main board and cooling fans. We also refer to the supporting devices of servers as chassis. The storage sub-system (S), which is used to store data persistently, consists of non-volatile storage devices, traditionally rotational devices (SR), but recently also Solid State Devices (SS). We also refer to the storage sub-systems as storage enclosures.

Each of these components may occur multiple times in a system and each occurrence may have different nameplate characteristics. Hence, we enumerate them with an index on each level. For instance, the second CPU in the first compute sub-system is labeled $\text{CP}_{1,2}$. The 5th rotational device in the second supporting component of the first storage sub-system is labeled $\text{SSR}_{1,2,5}$. We refer to the quantities and power consumptions of these components with Q and P respectively. E.g., the number of CPUs in the first compute sub-system is $Q(C, P)$ and the power consumption of the second CPU in the first compute sub-system is $P(CP_{1,2})$.

There are two key requirements that need to be met for our power models to correctly estimate power consumption. Firstly, only workloads that observe steady state can be used. The second requirement is system balance. Depending on the application and system, an optimal component ratio has to be maintained to keep all components (CPU, disks, controllers etc.) utilized during the measurement interval. If a system does not have the optimal ratio between these components, the power consumption model will not produce accurate estimates for the same reason that the system needs to be fully utilized. For each of the components listed in Figure 1 we determine its peak power consumption as follows.

2.1 Power Consumption of Compute Units

We obtain the peak power consumption of compute units, i.e. processors/CPUs, from their manufacturers specifications [10]. The power consumption is usually specified as Thermal Design Power (TDP). Table 2 shows the peak power consumption of selected CPUs. They range from 50W to 150W. The power consumption of all processors in compute sub-system i can be calculated with the aggregate of the nameplate power consumption of all individual CPUs:

$$P(CP_i) = \sum_{j=1}^{Q(CP_i)} Q(CP_{i,j})$$

2.2 Power Consumption of Storage Controllers

The power consumption of all storage controllers can be estimated using the maximum power consumption as defined in the Peripheral Component Interconnect (PCI) standard. It specifies that a PCI card draws at most 25 W of power. The power consumption of all storage controllers in compute sub-system i can then be calculated multiplying the number of storage controllers by 25:

$$P(CH_i) = Q(CH_i) * 25$$

2.3 Power Consumption of Volatile Storage

Similarly to the compute node nameplate power consumption, we obtain the peak power consumption of volatile storage, usually DRAM memory DIMMs, from the manufacturers website. 3 shows the power consumption for

CPU Description	TDP [W]	CPU Description	TDP [W]
Intel Pentium III Xeon - 900 MHz	50	AMD Opteron 2.2GHz Dual Core - 2.2 GHz	93
Intel E5420	50	AMD Opteron Dual Core 1 MB L2 - 2.4 GHz	95
Intel Pentium Xeon MP - 1.6 GHz	55	Intel Xeon X5650	95
Intel Xeon MP - 1.6 GHz	55	Intel Itanium2 - 1 GHz	100
Intel Xeon MP - 2.0 GHz	57	Opteron 6176	105
Intel E5506	60	Intel Itanium 2 Processor 6M - 1.5 GHz	107
Intel E5530	60	Intel DC Itanium2 Processor 9050 - 1.6 GHz	130
Intel Xeon MP - 2.8 GHz	72	Intel Dual-Core Itanium2 1.6Ghz	130
Intel Xeon MP - 2.7 GHz	80	Intel Itanium2 - 1.6 GHz	130
AMD Opteron - 2.2 GHz	85	Intel Xeon 7350 2.93GHz	130
AMD Opteron - 2.4 GHz	85	Intel Xeon X5680	130
Intel Xeon MP - 3.0 GHz	85	Intel Xeon X5670	130
AMD 8220SE 2.8 GHz	93	Intel Xeon X5560	130
AMD Opteron - 2.6 GHz	93	Intel X7560	130
AMD Opteron - 2.8 GHz	93	Intel Xeon 7140 3.4GHz	150

Table 2: Thermal Design Power (TDP) of x86 CPUs (Intel and AMD)

DDR	Size [GByte]	Density [GByte]	TDP [W]
2	4	1	9.3
2	4	2	5.4
2	8	2	5.5
3	8	2	5.6
3	16	1	8

Table 3: Power Consumption of Volatile Storage (Memory)

various types of DRAM used in TPC benchmarks. The power consumption of the entire volatile storage in compute sub-system i can be calculated as the aggregate power consumption of all individual memory DIMMs as follows:

$$P(CM_i) = \sum_{j=1}^{Q(CM_i)} Q(CM_{i,j})$$

2.4 Power Consumption of Non-Volatile Storage

We distinguish non-volatile storage between rotational storage, i.e. disk drives, and solid state storage, i.e. SSDs. Peak power consumption levels of disk drives vary widely with the disks form factor, size, and rotational speed. 5 summarizes the peak power consumption of disk drives used in TPC benchmark publications. The energy consumption of SSDs depends on the underlying technology. Most SSDs use NAND-based flash memory, a non-volatile chip that can be electrically erased and reprogrammed. There are two different types of SSDs: Single-level cell (SLC) and multi-level cell (MLC). One major difference between these technologies is that SLC hold one data bit while MLC hold two data bits. SLC provides higher write performance and reliability while MLC allow for higher storage density and lower cost. Most of todays SSDs use the same interface as traditional hard disk drives, namely SAS or SATA, so supported in traditional SAS/SATA storage arrays. Another type of SSDs that is becoming popular is PCI Express-based flash storage card (e.g. ioDrive from Fusion-io and WrapDrive from LSI. Todays PCI Express-based flash storage cards can hold between 160 GByte to 1.2 TByte. Their nameplate power consumption is 25 Watts, equal to the nameplate power consumption of the PCI slot. The nameplate power of SSDs used in TPC benchmarks are listed in Table 4. They are obtained from manufacturers web sites [20]. FF refers to the form factor of the drive. The power consumption of the entire non-volatile storage in compute sub-system i can be calculated as the aggregate power consumption of its rotational devices and solid state devices:

Type of Device	Size [GByte]	Power [W]
SATA 2.5 inch Form Factor Solid State Drive	60	2
SATA 2.5 inch Form Factor Solid State Drive	120	2
PCI Express-based flash storage cards	160-1200	25

Table 4: Power Consumption of Solid State Devices

FF=2.5 RPM=10K		FF=2.5 RPM=15K		FF=3.5 RPM=7.2K		FF=3.5 RPM=10K		FF=3.5 RPM=15K	
GByte]	[W]	[GByte]	[W]	[GByte]	[W]	[GByte]	[W]	[GByte]	[W]
36	17	36	10.0	240	11.35	9	9.7	18	13.2
36	7.2	36	9.2	465	13	9	10.0	18	10.0
72	8.4	72	9.2	500	3.5	36	12.5	18.2	9.7
73	10.5	146	5.7			36	10.0	32	10.0
146	10.0					72	12.6	36	14.5
146	9.0					73	11.0	36	15
300	4.8					146	14.2	72	13.2
						146	11.4	73	16.2
						160	12.8	146	14.2
						250	11.35	146	19.0
						300	16.4	300	17.6
								300	14.4

Table 5: Power Consumption of Rotational Devices

$$P(CS_i) = \sum_{j=1}^{Q(CSR_i)} P(CSR_{i,j}) + \sum_{j=1}^{Q(CSS_i)} P(CSS_{i,j})$$

Similarly, the power consumption of the entire non-volatile storage in storage sub-system i can be calculated as the aggregate power consumption of all disk drive devices and solid state devices in all storage enclosures:

$$P(SS_i) = \sum_{j=1}^{Q(CSR_i)} P(CSR_{i,j}) + \sum_{j=1}^{Q(CSS_i)} P(CSS_{i,j})$$

2.5 Power Consumption of Compute Sub-System (servers)

In addition to compute units, storage controllers, volatile and non-volatile memory, we need to account for the supporting components of the compute sub-system to estimate their total power consumption. Supporting components are the main board, cooling fans, caches etc. They are also referred to as the Server Chassis. Recent studies [7] and [21] suggest that the power consumption of the server chassis can be expressed as a percentage (30 %) of the nameplate power consumption of its main components plus a fixed overhead (100W). Hence, we compute the power consumption of the entire compute sub-system i as follows:

$$P(C_i) = [P(CP_i) + P(CM_i) + P(CS_i)] * 1.3 + 100$$

2.6 Power Consumption of Storage Sub-Systems (Enclosure)

Similar to the power estimate of the compute sub-systems chassis case, we approximate the power consumption of the disks enclosures as a percentage of the nameplate power consumption of its main components. In our model we express it as 20 percent of the aggregate power consumption of all non-volatile storage devices. [7] and

[21] assume that the enclosures are fully populated. This is true for the majority of TPC results. Consequently the power consumption of an entire storage sub-system that contains only fully populated enclosures can be calculated as follows:

$$P(S_i) = [P(SS_i)] * 1.2$$

With the introduction of solid state drive technologies the above assumption is not true anymore and the power overhead of storage sub-systems might be under estimated using Formula 6. This is due to the increased performance of SSDs compared to rotational devices. Benchmark sponsors are able to substitute up to 12 rotational devices with one SSD using existing disk enclosures. Since the bandwidth of disk enclosures is limited by their controllers, only few of their slots are filled. Still the power infrastructure of these storage enclosures has been sized for many more devices. In order to accurately estimate the overhead of disk these enclosures, we need to assume that each enclosure is fully populated (Q_{max}) with rotational devices using the maximum allowable power for this storage device (P_{max}). The power consumption of storage enclosures that use some or only SSDs, can be estimated as:

$$P(S_i) = P(SS_i) + 0.2 * \sum_{j=1}^{Q_{max}(SS_i)} P_{max}(SS_i)$$

2.7 Power Consumption of the Entire System

Finally, the total power consumption of the entire system (P_S) can be estimated as the aggregated power consumption of the compute and storage sub-systems discounted by a factor of 0.2, which has also been validated in [14] and [13].

$$P = [\sum_{i=1}^{Q(C)} P(C_i) + \sum_{i=0}^{Q(S)} P(S_i)] * 0.2$$

3 Energy Consumption of On Line Processing Systems

Online Transaction Processing (OLTP) systems facilitate and manage transaction-oriented applications, typically for data entry and retrieval systems, used in industries such as banking, airlines, mail-order, supermarkets, and manufacturing. While some understand a transaction in the context of computer or database transactions, the TPC defines it as a business or commercial transactions. The TPC defines two benchmarks, TPC-C and TPC-E to measure the performance of large scale transactional systems.

3.1 The TPC-C Benchmark

TPC Benchmark C (TPC-C) [23] models an On Line Transaction Processing (OLTP) workload. In order to keep up to date with technology and system requirements in general, TPC-C has undergone three major revisions since its establishment in 1992. The first two revisions, published in the first 18 month were comparable due to their minimal effect on existing results. While revision 4 failed to get the necessary support revision 5 of TPC-C was approved in October 2000. This revision contained substantial changes with a large impact on existing results, which made it non-comparable to previous revisions under the rigid TPC rules. However, all revisions fulfill the requirements of the power consumption model. During its tenure each revision has been accepted in the industry as the most credible transaction processing benchmark with a large body of results across all major hardware and database platforms. Modeled after actual production applications and environments, it

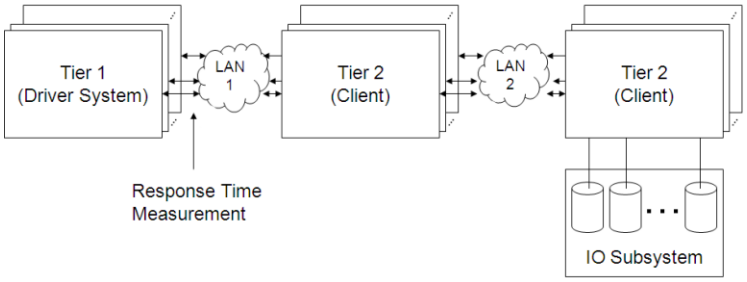


Figure 2: Typical TPC-C System Setup

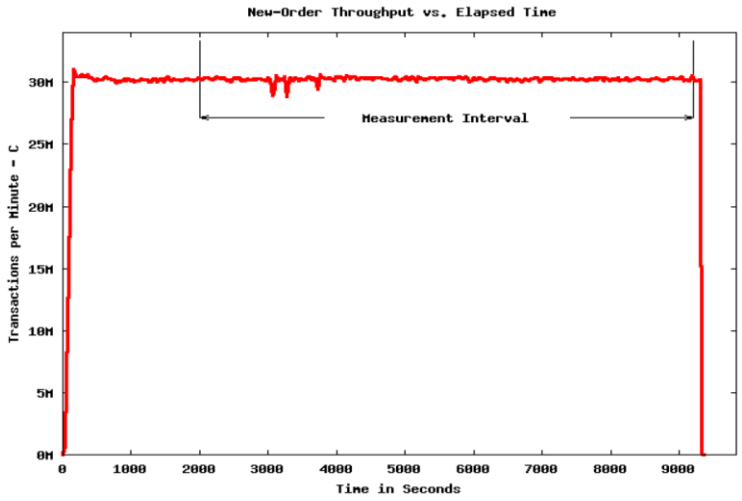


Figure 3: Throughput versus Time: TPC-C Publication 107111201

evaluates key performance factors such as user interface, communications, disk I/Os, data storage, and backup and recovery using a mixture of read-only and update-intensive transactions.

The typical TPC-C system is designed in three tiers: Tier 1, Driver System, Tier 2 Client and Tier 3 Database Server (see Figure 2). The Driver System emulates the user load.

The TPC-C performance reported in a benchmark publication is the transaction throughput of new orders during steady state condition [tmpC]. The performance is measured during the measurement interval, which must begin after the system reaches steady state, be long enough to generate reproducible throughput results that would be representative of the performance that would be achieves during a sustained eight hour period and extend uninterrupted for a minimum of 120 minutes. Another important metric of TPC-C benchmarks is price-performance. The price-performance metric [$/tpmC] is calculated by dividing the three-year cost of ownership of all components by the tpmC. See TPCs pricing specification [22] for how the three-year TCO is calculated.

Figure 3 graphs the number of new order transaction [tpmC] during the measurement interval as achieves by the current[2] performance leader.

[2]as of 1/4/2011 (see http://www.tpc.org/tpcc/results/tpcc_perf_results.asp)

3.2 The TPC-E Benchmark

TPC-E [30], approved in February 2007, is the next generation OLTP benchmark in TPCs benchmark suite. It implements a database-centric benchmark that:

1. Provides comparable performance. That is, performance from different vendors can be compared.
2. Implements an easy to understand business model.
3. Reduces the cost and complexity of running the benchmark, compared to the TPC-C benchmark.
4. Implements a complex database schema
5. Encourages database uses that are representative of client environments.
6. Leads to realistic benchmark implementations. That is, the software and hardware configuration used in the benchmark are similar to those actual end-users would use.

The TPC-E benchmark simulates the OLTP workload of a brokerage firm. Like TPC-C, the focus of TPC-E is the performance measurement of the central database that executes transactions related to the firms customer accounts. Although the underlying business model of TPC-E is a brokerage firm, the database schema, data population, transactions, and implementation rules have been designed to be broadly representative of modern OLTP systems.

TPC-E is similar to TPC-C in the following ways. The TPC-E configuration is a 3-tier model similar to TPC-C, with similar configuration and run rules (see 2). The primary metrics for TPC-E are tpsE, $/tpsE and availability date, which correspond to TPC-Cs tpmC, $/tpmC, and availability date. As in TPC-C the performance of TPC-E is measured during the measurement interval, which must begin after the system reaches steady state. TPC-C and TPC-E use a continuous scaling model and portions of the database scale in a linear fashion while the transaction profile is held constant.

Figure 4 graphs the throughput versus elapsed wall clock time of the current[3] performance leader of TPC-E [29].

3.3 Power Consumption Models for TPC-C and TPC-E

The general power estimation model, described in Section 2, can be applied to TPC-C and TPC-E benchmark publications because their measurement intervals are taken during steady state performance and because all components are fully utilized. The typical business objective of any TPC benchmark publication is to demonstrate performance and price-performance. Hence all TPC publications maintain optimal component ratios. This is because no vendor can afford to over-configure one part of the system because all parts that are used in a benchmark publication need to be disclosed and priced. And price-performance is widely being used by system vendors to showcase their advantages over those of their competitors. For instance, if a vendor over-configures a database server with 50 % more CPUs, those CPUs need to be priced, and, since the number of CPUs is disclosed, the result will be used by competitors to show that they can achieve the same performance with fewer CPUs. Lastly, some database vendors tie their pricing model to the number of CPUs, while some tie it to the number of disks. This inconsistency makes it even more unattractive to publish unbalanced TPC performance results.

In order to apply the power estimation model of Section 2 to TPC-C and TPC-E systems, we need to assign the different parts of TPC-Cs and TPC-Es three tier architectures (see 2) to the components of our power estimation model (see 1). Users of todays transaction systems are most often connected through the Internet rather than through closed circuit systems. Hence, in most cases, the deployment of a transaction system does not

[3] as of 1/4/2011 (see http://www.tpc.org/tpce/results/tpce_perf_results.asp)

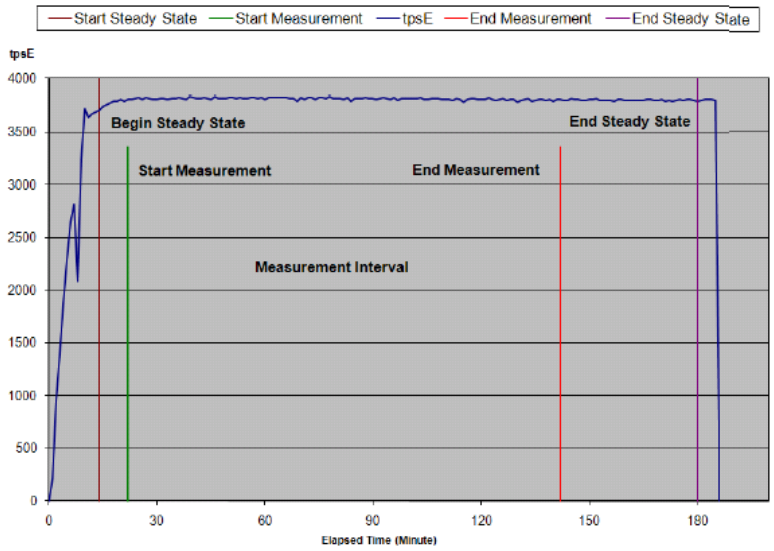


Figure 4: Throughput versus Time: TPC-E Publication 110092401

include the driver systems (Tier 1). In benchmark publications, the load imposed by users of the Tier 1 systems are emulated with far viewer systems than used in real life. Consequently, we only need to map Tier 2 and Tier 3 systems to our model. In TPC terminology, Tier 2 and Tier 3 systems are referred to as the System Under Test (SUT). This is true for TPC-C and TPC-C.

Systems that implement Tier 2 and Tier 3 can be mapped directly to the compute sub-systems of our model. The storage sub-system, which is usually connected to the database server, is mapped to the storage sub-system of our model.

4 Energy Consumption of Decision Support Systems

Generally, Decision Support (DS) workloads can be subcategorized into three distinct types of operations: Initial load, incremental load, and queries. They can be run in single- and multi-user modes. The single-user mode stresses a systems ability to parallelize operations such that the answer for a given request can be obtained in the least amount of time, as desired for overnight batch job processing. The multi-user mode stresses the systems ability to schedule concurrent requests from multiple users to increase overall system throughput. Furthermore DS workloads differ in the degree to which the queries are known in advanced (ad-hoc vs. reporting queries). The TPC currently has one decision support benchmark, called TPC-H. It is an ad-hoc benchmark, which executes each of the operations, initial load, incremental load, and queries in both single and multi-user mode.

4.1 The TPC-H Benchmark

TPC-H uses a 3rd normal form schema of eight base tables, which are populated with uniform data, i.e. without any data skew. Each TPC-H result is obtained on a database with a specific size, indicated by the scale factor (SF). The scale factor, specified in GByte, equals the raw data outside the database. The TPC rules prohibit comparing benchmark results between scale factors. The primary performance metric is the composite performance metric (QphH). It equally weights the contribution of the single-user and the multi-user runs by calculating the geometric mean of the single- and multi-user performances. With SF being the scale factor, T(qi) the elapsed

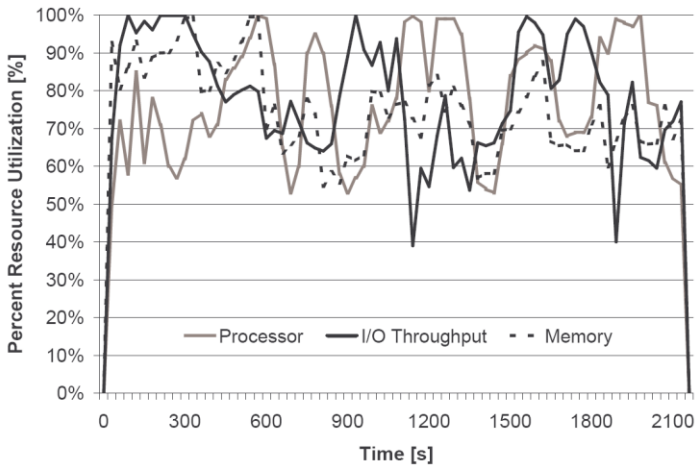


Figure 5: Oscillating Nature of the Resource Utilization During Decision Support Queries

time of Query i and T(ui) the elapsed time of Update Operation I, S the number of emulated users and Ts the elapsed time of the multi user run, the single user performance $P_{SingleUser}$ and multi-user performances $P_{MultiUser}$ are defined as follows:

$$P_{SingleUser} = \frac{3600 * SF}{\sqrt[24]{\Pi_{i=1}^{22} T(q_i) * \Pi_{i=1}^{2} T(u_i)}}$$

$$P_{MultiUser} = \frac{22 * S * 3600 * SF}{T_s}$$

$$P_{Composite} = \sqrt{P_{SingleUser} * P_{MultiUser}}$$

TPC-H 3rd Normal Form allows query execution of various execution paths. They are often dominated by large hash or sort-merge joins, but conventional index driven joins are also common. Large aggregations, which often include large sort operations, are widespread. This diversity imposes challenges both on hardware and software systems. High sequential I/O-throughput of large I/O operations is critical to excel in large hash-join operations. At the same time, index driven queries stress the I/O sub-systems ability to perform small random I/O operations. Due to the complex nature of the TPC-H queries, not all system resources, e.g. I/O, CPU, and Memory, are exhausted at any given point during query execution of the single user test. For instance, a hash join is typically CPU bound during the build phase of its hash table and, usually, I/O bound during its probe phase. Consequently, a system consumes more power in the storage sub-system during some time of the single user test and more CPU power during other times of the single user test. However, the magnitude of oscillating resources is alleviated during the multi-user mode because operations across users are usually not synchronized and, therefore, resource consumptions of concurrent queries do not align. For instance when multiple user issue queries performing hash-join operations one query might be in the build phase while another query is in the probe phase. Figure 5 shows the resource consumption of processor (CPU), I/O throughput and memory during a multi-user execution of typical decision support queries of 8 concurrent users. The CPU, I/O and memory resources are normalized by percent of their theoretical maximum value that can be achieved in this configuration. The graphs show that on average each resource is utilized about 80 percent.

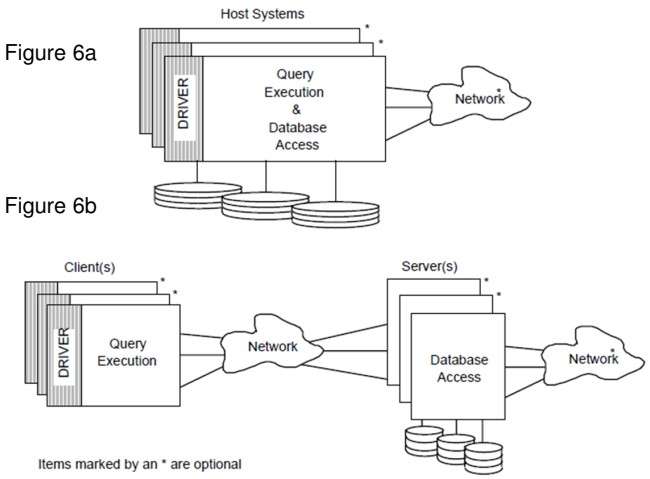


Figure 6: Typical TPC-H Configuration (source: TPC-H Specification 2.13.0)

4.2 Typical TPC-H Systems

The TPC-H specification allows two types of configurations, a host-based configuration in which the query execution and database access happens on the same system and a client/server configuration in which the query execution and database access happen on two systems that are connected via a network.

Figure 6 illustrates these two options. Figure 6a depicts the host-based solution and Figure 6b depicts the client/server solution. In both figures the driver is shown in the shaded area. The System Under Test (SUT) consists of the following components: 1

- The host and server systems (hardware and software)
- Client processing units
- Communication systems (hardware and software)
- Data storage

4.3 Power Consumption Model TPC-H

TPC-H fulfills both requirements of the power estimation model. The multi-user test of TPC-H is in a steady state. As shown in Figure 4 resources in a typical TPC-H system are used about 80 percent during the entire measurement interval. Additionally, as with TPC-C and TPC-E benchmark publications, the business objective of a TPC-H benchmark publication is to demonstrate performance and price-performance. Therefore, TPC-H systems are very well balanced.

Our generalized power estimation model can be used for both, the host-based and the client/server solutions of TPC H. For the host-based solution we only need to account for one compute systems, or multiple compute subsystems in case of clustered solutions, and one storage subsystems. In case of a client/server based solution, we have to distinguish between compute notes for the clients and server plus the storage subsystem.

Result	[26]	[25]	[24]
Measured tpmC	1,807,347	290,040	1,193,472
Measured Watts per ktpmC	2.46	4.22	5.93
Measured Power [W]	4,441.0	1,223.9	7,077.0
Estimated Power [W]	4,356.3	1,137.4	7,412.7
Total Difference [W]	-84.7	-86.5	335.7
Relative Difference [%]	-1.9	-7.1	4.7

Table 6: Power Consumption: Measured vs. Estimated values. Watts per ktpmC is a metric required by TPC-Energy. It shows how much energy is needed to run 1000 TPC-C transactions.

Result	[28]	[27]
Measured tpsE	2,001.12	1,400.14
Measured Watts per tpsE	5.84	6.72
Measured Power [W]	11683	9403
Estimated Power [W]	12065.7	9414.5
Total Difference [W]	382.7	11.5
Relative Difference [%]	3.3	0.1

Table 7: Power Consumption: Measured vs Estimated Values

5 Verification of Analytical Power Consumption Models With Fully Configured TPC-C/E and H Systems

The analytical power consumption models are verified against all TPC benchmark publications with energy metrics as of 1/21/2011: three TPC-C results, [26], [25], and [24], two TPC-E results, [28] and [27], and four TPC-H, [32],[33],[31] and [34]. Tables 6, 7 and 8 summarize the measured and reported power consumption of the entire SUT of the above nine TPC publications. Details of each benchmark configuration are summarized below.

The benchmark configuration of result [26] achieves 1,807,347 tpmC with one database server, populated with four Intel Xeon X7560 processors, 64 16 GByte memory modules, four internal disk drives, 18 disk enclosures with a total of 132 disk drives and 256 SSDs. The middle tier consists of four servers, each with two Intel Xeon X5670 processors, twelve 2 GByte memory modules and two internal disk drives. The benchmark configuration of result [25] achieves 290,040 tpmC with one database server, populated with one Intel Xeon X5650 processors, 16 16 GByte memory modules, 16 internal SSDs, three disk enclosures with a total of 25 disk drives and eight SSDs. The middle tier consists of three servers, each with one Intel Xeon E5506 processors, two 1 GByte memory modules and one internal disk drive. The benchmark configuration of result [24] achieves 1,193,472 tpmC with one database server, populated with two AMD Opteron 6167 SE processors, 16 16GByte and 32 8 GByte memory modules, two internal disk drives, 13 disk enclosures with a total of 166 disk drives and 120 SSDs. The middle tier consists of 24 servers, each with one Intel Xeon E5530 processors, two 1 GByte memory modules and 1 internal disk drive.

Each result uses a combination of rotational storage devices and SSD devices in their storage sub-system. Interestingly, although result [25] uses the least number of rotational devices compared to the others, it is not the most energy efficient result. Result [26], which is the most energy effective system with 2.46 Watts per ktpmC, uses 132 rotational devices, more than 5 times the number of rotational devices of result [25].

The power consumptions calculated with the power estimation model for TPC-C are within -7.1 % to 4.7 % of the reported power numbers.

The benchmark configuration of result [28] achieves 2,001.12 ptsE with one database server populated with four Intel Xeon X7560 processors, 64 16 GByte memory modules, four internal disk drives, 40 disk enclosures with a total of 990 disk drives. The middle tier consists of four servers, each with one Intel Xeon X5420

Result	[32] SF=100	[33] SF=100	[31] SF=300	[34] SF=300
QphH	73,975	71,438	107,561	121,346
Watts per QphH	5.93	6.48	9.58	10.33
Measured Power [W]	438	463	1031	1254
Estimated Power [W]	476.9	514.72	992.7	1141.8
Total Difference [W]	38.9	51.7	-38.3	-112.2
Relative Difference [%]	8.9	11.2	-3.7	-8.9

Table 8: TPC-H Power Consumption: Measured vs Estimated Values

processors, 12 x 2 GByte memory modules and four internal disk drives. The benchmark configuration of result [27] achieves 1,400.14 tpsE with one database server, populated with two AMD Opteron 6167 SE processors, 32 8 GByte memory modules, two internal disk drives, 28 disk enclosures with a total of 700 disk drives. The middle tier consists of four servers, each with one Intel Xeon E5420 processors, two 1 GByte memory modules and four disk drives. Result [28], which uses about 20 percent more energy (2280 Watts) than Result [27], is more energy efficient, because it achieves about 30 percent higher performance. Hence, its Watts per transaction metric is 0.88 Watts less. Overall the estimated power is very close compared to the measured power. Result [28] is 3.3 percent off (382.7 Watt), while Result [27] is 0.1 percent off (11.5).

For TPC-H we have the most number of published results available, namely four. The benchmark configuration of result [32] achieves 74,975 QphH with one database server populated with two Intel Xeon X5680 processors, 12 x 16 GByte memory modules, two internal disk drives and four SSDs. The benchmark configuration of result [33] consists of a database server populated with two AMD Opteron 6167 SE processors, 24 8 GByte memory modules three internal disk drives and four SSDs. It achieves 71,438 QphH. The benchmark configuration of result [31] achieves 107,561 QphH with one database server populated with four AMD Opteron 6167 SE processors, 16 16 GByte and 32 8 GByte memory modules, ten internal disk drives, and four PCI based flash devices. The benchmark configuration of result [34] achieves 121,346 QphH with one database server populated with four Intel X5670 processors, 16 16GByte and 48 8 GByte memory modules and eight internal disk drives. Even for TPC-H the power estimations are very close to the measured values. The relative difference ranges between -8.9 and 11.2 percent (-112.2 Watts to 38.9 Watts). The relative error margins for TPC-H seem to be higher compared to those of TPC-C and TPC-E. However, the TPC-H systems are far small compared to those of TPC-C and TPC-E. In most cases they are 10 times larger. Looking at the absolute difference rather than the relative difference the estimates of the TPC-H systems are small compared to those of the TPC-C and TPC-E systems. The estimates of TPC-C and TPC-E systems are between -86.5 and 382.7 Watts off while the estimates of the TPC-H systems are only -112.2 to 51.7 Watts off.

6 Conclusion

Historically vendors have optimized computer systems for performance and cost of ownership to be competitive in the market place. The TPC benchmarks have played a vital role by providing architecture and platform neutral metric and methodologies to measure these aspects. Recently announced TPC-Energy benchmark specification enables measurement and reporting of energy impact of performance and cost. While this new specification is expected to take several years to mature with publications across vendors and platform, we believe that analytical power estimation models based on nameplate data, presented in this paper are a useful tool for estimating and sizing TPC configurations and similar enterprise database systems.

As shown in 6 and 7 the power consumption estimates for OLTP benchmarks (TPC-C and TPC-E) are within 10 % of actual published numbers. For larger configurations, i.e. larger than 4kW peak power consumption, estimates are within 5 %. Power consumption estimates for TPC-H benchmarks are within 10 % of their measured power consumption, accept for one result (result [33]), which is 11.2 % larger than its measured power

consumption (8). The authors believe that estimates within 10 % of actual power consumption meet most estimation requirements. The authors plan to keep a close eye of future benchmark publications and enhance the estimation model.

Acknowledgement

The authors would like to acknowledge Satinder Sethi and and Ray Glasstone for their invaluable comments and feedback and Heinrich Poess for his infinite support.

References

[1] 60 and 120 GB Solid State Drive http://h18000.www1.hp.com/ products/quickspecs/13415_div/13415_div.pdf

[2] A. Fanara, E. Haines, A. Howard. The State of Energy and Performance Benchmarking for Enterprise Servers. pp 52-66 TPCTC 2009

[3] A. Fanara, E. Haines, A. Howard: The State of Energy and Performance Benchmarking for Enterprise Servers. TPCTC 2009: 52-66

[4] AMD 6100 processors series specification: http://products.amd.com/en-us/opteroncpuresult.aspx?f1=AMD+Opteron%E2%84%A2+6100+Series+Processor&f2=&f3=Yes&f4=&f5=512&f6=&f7=D1&f8=45nm+SOI&f9=&f10=6400&f11=&f12=Active&

[5] E. Young, P. Cao, M. Nikolaiev. First TPC-Energy Benchmark: Lessons Learned in Practice TPCTC (LNCS Vol. 6417) 2010 136-152

[6] ENERGY STAR Program Requirements for Computer Servers (2009) available at: http://www.energystar.gov/ia/partners/product_specs/ program_reqs/computer_server_prog_req.pdf

[7] F. Xiaobo, W. Weber, and L. A. Barroso. 2007. Power Provisioning for a Warehouse-sized Computer. Proceedings of the 34th International Symposium on Computer Architecture in San Diego, CA. Association for Computing Machinery, ISCA '07. http://labs.google.com/papers/power_provisioning.pdf

[8] Fusion IO drive http://kb.fusionio.com/KB/a13/iodrive-and-iodrive-duo-power-consumption.aspx

[9] Hogan, T. "Overview of TPC Benchmark E: The Next Generation of OLTP Benchmarks" TPCTC (LNCS Vol. 5895) 2009 84-98

[10] Intel Xeon Processor Family specification: http://ark.intel.com/ProductCollection.aspx?familyID=594

[11] J. Mitchell-Jackson, J. G. Koomey, B. Nordman, and M. Blazek. Data center power requirements: measurements from Silicon Valley. Energy (Energy) ISSN 0360-5442, 28(4):837 850, 2003.

[12] M. Kunz. Energy consumption of electronic network components (English Version). Zurich: Bundesamt fr Energiewirtschaft Forschungsprogramm Elektrizitt, Basler & Hofman, November 26, 1997.

[13] M. Poess, R. O. Nambiar (2008, September). Energy cost, the key challenge of today's data centers: a power consumption analysis of TPC-C results. PVLDB 1(2). pp. 1229-1240

[14] M. Poess, R. O. Nambiar (2010, January). A power consumption analysis of decision support systems. WOSP/SIPEW 2010 pp. 147-152

[15] M. Poess, R. O. Nambiar (2010, September).Transaction Processing vs: Moores Law: A Trend Analysis. TPCTC (LNCS Vol. 6417) 2010 110-120

[16] M. Poess, R. O. Nambiar, K. Vaid, J. M. Stephens, K. Huppler, E. Haines (2009, March). Energy benchmarks: a detailed analysis. e-Energy. pp. 131-140

[17] Overview of the TPC Benchmark C: The Order-Entry Benchmark: http://www.tpc.org/tpcc/detail.asp

[18] Poess, M. and Floyd, C., "New TPC Benchmarks for Decision Support and Web Commerce". ACM SIGMOD RECORD, Vol 29, No 4 (Dec 2000)

[19] Specifications of disk drive from Hitachi: http://www.hitachigst.com/tech/techlib.nsf/techdocs/

[20] Specifications of disk drives form Seagate http://www.seagate.com/staticfiles/support/disc/manuals/, http://www.seagate.com/docs/pdf/datasheet/

[21] Suzanne Rivoire, Mehul Shah, Parthasarathy Ranganathan, Christos Kozyrakis, Justin Meza: Modeling and Metrology Challenges for Enterprise Power Management, IEEE Computer, December 2007

[22] TPC Pricing v1.5: http://www.tpc.org/pricing/spec/Price_V1.5.0.pdf

[23] TPC-C Benchmark Revision 5.9: http://www.tpc.org/tpcc/default.asp

[24] TPC-C Result 110062201 http://www.tpc.org/tpcc/results/ tpcc_result_detail.asp?id=110062201

[25] TPC-C Result 110081701 http://www.tpc.org/tpcc/results/ tpcc_result_detail.asp?id=110081701

[26] TPC-C Result 110083001 http://www.tpc.org/tpcc/results/ tpcc_result_detail.asp?id=110083001

[27] TPC-E Result 11006210 http://www.tpc.org/tpce/ results/tpce_result_detail.asp?id=110062103

[28] TPC-E Result 110062202 http://www.tpc.org/tpce/results/ tpce_result_detail.asp?id=110062202

[29] TPC-E Result 110092401 http://www.tpc.org/results/FDR/tpce/ fujitsu.RX900S1.100924.01.fdr.pdf

[30] TPC-E Version 2.12.0 http://www.tpc.org/tpce/spec/v1.12.0/TPCE-v1.12.0.pdf

[31] TPC-H Result 110062104 http://www.tpc.org/tpch/results/ tpch_result_detail.asp?id=110062104

[32] TPC-H Result 110070201 http://www.tpc.org/tpch/results/ tpch_result_detail.asp?id=110070201

[33] TPC-H Result 110071401 http://www.tpc.org/tpch/results/ tpch_result_detail.asp?id=110071401

[34] TPC-H Result 110091501 http://www.tpc.org/tpch/results/ tpch_result_detail.asp?id=110091501

[35] Transaction Processing Performance Council. http://www.tpc.org/information/about/abouttpc.asp

[36] United States Environmental Protection Agency. http://www.epa.gov/aboutepa/index.html

ACM SIGMOD/PODS 2011 Conference

Athens, Greece
June 12-16, 2011
http://www.sigmod2011.org/
2011 ACM SIGMOD International Conference on Management of Data
2011 ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems

Call For Participation

For decades, the joint ACM SIGMOD/PODS Conference has established itself as the top data management conference in the world for researchers, practitioners, developers, and users to report and share cutting-edge ideas and results, and to exchange techniques, tools, and experiences. We are delighted to invite you to attend ACM SIGMOD/PODS, to be held in Athens, Greece from June 12-16, 2011.

The conference will take place at the Divani Caravel Hotel, Athens, located at 2, Vas. Alexandrou Ave., 16121 Athens, Greece. Attendees who wish to book via the phone may call +30-210-7207000 and mention the ACM SIGMOD / PODS Conference. Reservation forms are also available at the conference website. Registration information will be available at: http://www.sigmod2011.org

SIGMOD

SIGMOD 2011 will include two **Research Plenary Sessions** where all of the over 80 research papers accepted to the SIGMOD 2011 proceedings will be presented as research posters. All research papers will be invited to participate in our **Experimental Repeatability** effort, the goal of which is to ensure that SIGMOD papers stand as reliable, archival work for future research.

The SIGMOD **Demonstration Program** has become an important venue to share cutting-edge, prototype data management systems with the greater SIGMOD community. This year, SIGMOD will feature 32 demonstrations showcasing the most exciting new data management technologies. We will again be holding a **Best Demonstration Award Competition** to recognize the most innovative demonstrations.

The SIGMOD **Industrial Program** is a forum for high quality presentations on innovative commercial software, systems and services for all facets of information technology. This year's talks will include both refereed presentations and invited presentations on trend-setting deployments in cloud data management, business analytics, new uses of SSD, and massive parallel processors.

The SIGMOD **Tutorial Program** will include 6 tutorials on new applications of Datalog, flash memory, copy detection, data privacy, web data management and statistical relational models. SIGMOD will host the **Third Annual SIGMOD Programming Contest.** Student teams from degree granting institutions are invited to compete in this annual contest. This year, the task is to implement a high-throughput main-memory index that is made durable using a flash-based SSD.

The **SIGMOD Undergraduate Research Poster Competition** provides undergraduate students an opportunity to showcase their research accomplishments in a poster competition. Travel stipends are available for top students. The SIGMOD program also includes a **New Researcher's Symposium** (the goal of which is to provide graduate students and junior researchers advice on careers in data management research), and a **panel discussion on Health Informatics**.

SIGMOD will also host the following eight **Workshops**: Semantic Web Information Management (SWIM), Networking Meets Databases (NetDB), International Workshop on Testing Database Systems (DBTest), Data Engineering for Wireless and Mobile Access (MobiDE), Web and Databases (WebDB), Data Management on New Hardware (DaMoN), Innovative Database Research (IDAR), Databases and Social Networks (DBSocial).

PODS

PODS 2011 will include a **Technical Program** of 25 research papers, a **keynote presentation** by Prof. Tova Milo (Tel Aviv University) and two **tutorials** by Prof. Marcelo Arenas (PUC Chile) and Prof. S. Muthu Muthukrishnan (Rutgers University).

Call for Participation

The **27th IEEE International Conference on Data Engineering** addresses research issues in designing, building, managing, and evaluating advanced data-intensive systems and applications. It is a leading forum for researchers, practitioners, developers, and users to explore cutting-edge ideas and to exchange techniques, tools, and experiences. The mission of the conference is to share research solutions to problems of today's information society and to identify new issues and directions for future research and development work.

Keynotes

- Anastasia Ailamaki (EPFL, Switzerland)
- Johannes Gehrke (Cornell University, USA)
- Georg Gottlob (Oxford University, UK)

Workshops

- 6th International Workshop on Self Managing Database Systems (SMDB 2011)
- 1st International Workshop on Managing Data Throughout its Lifecycle (DaLi 2011)
- 2nd International Workshop on Graph Data Management: Techniques and Applications (GDM 2011)
- 3rd Workshop on Hot Topics in Software Upgrades (HotSWUp 2011)
- 1st International Workshop on Data Engineering Applications in Emerging Areas: Health Care Informatics and Energy Informatics (DEAEA 2011)
- 2nd International Workshop on Data Engineering meets the Semantic Web (DESWeb 2011)
- 1st International Workshop on Data Management and Analytics for Semi-Structured Business Processes (DMA4SP)
- ICDE 2011 PhD Workshop

Venue

ICDE 2011 will be held in the Hannover Congress Centrum, Hannover, Germany. Hannover is the capital of the federal state of Lower Saxony (Niedersachsen), Germany. Hannover is known as a trade fair city (e.g. CeBIT) but also for their Royal Gardens of Herrenhausen and many other places of interest.

For more information please regular visit the conference web site:

http://www.icde2011.org/

Microsoft hp YAHOO! LABS

IEEE computer society

IEEE Technical Committee on Data Engineering

Leibniz Universität Hannover L3S Forschungszentrum Research Center DFG

IEEE Computer Society
1730 Massachusetts Ave, NW
Washington, D.C. 20036-1903

Non-profit Org.
U.S. Postage
PAID
Silver Spring, MD
Permit 1398