On the Path to Sustainable, Scalable, and Energy-efficient Data Analytics: Challenges, Promises, and Future Directions

Sriram Lakshminarasimhan*‡, Prabhat Kumar†, Wei-keng Liao†, Alok Choudhary†, Vipin Kumar§, Nagiza F. Samatova‡*††

* *North Carolina State University, Raleigh, NC*
‡ *Oak Ridge National Laboratory, Oak Ridge, TN*
† *Northwestern University, Evanston, IL*
§ *University of Minnesota, Minneapolis, MN*
†† *Corresponding author: samatova@csc.ncsu.edu*

***Abstract*—As scientific data is reaching exascale, scalable and energy efficient data analytics is quickly becoming a top notch priority. Yet, a sustainable solution to this problem is hampered by a number of technical challenges that get exacerbated with the emerging hardware and software technology trends. In this paper, we present a number of recently created "secret sauces" that promise to address some of these challenges. We discuss transformative approaches to efficient data reduction, analytics-driven query processing, scalable analytical kernels, approximate analytics, among others. We propose a number of future directions that could be pursued on the path to sustainable data analytics at scale.**

I. Introduction

The projected energy costs of scaling machines to exascale forms a nontrivial barrier to realization. Currently, the energy bills from system operation and hardware cooling exceed the cost of hardware ownership within just a few years of operation [9]. In fact, this trend of rising energy costs is expected to continue over the next decade [1], [2], [7]. Furthermore, uncontrolled energy generation negatively affects hardware reliability. Therefore, increasing energy efficiency is a top notch priority for extreme-scale computing, in general, and data-intensive computing, in particular. Creating a sustainable path to exascale is challenged by emerging trends related to data, hardware, and energy issues. These issues call for the next breed of data analysis algorithms and software libraries to be designed to achieve high operational efficiency on datasets growing in size and complexity and on computational resources increasing in their heterogeneity.

A. Data Trend: Scientific data sets grow not only in size but in complexity.

Scientific data sizes are expected to reach hundreds of petabytes and beyond. But the algorithmic and energy challenges of handling this data due not just to the data size alone, but also to its complexity, i.e., high-dimensional, non-stationary, multi-scale, multi-variate, spatio-temporal nature. Naturally, analysis algorithms on these sets are becoming increasingly sophisticated, requiring an increasing number of power-demanding computational and memory resources.

Arguably, traditional approaches to developing scalable algorithms and software for data analytics and mining are not suitable at exascale, and a paradigm shift may be necessary to address the challenge. Current algorithms often assume that data sizes can fit in the memory of small-scale systems. This could be true for local context mining that constrains the data to a single time step or a single variable over a few time steps. However, simulations that are driven by local space-time relationships are largely performed with the purpose of discovering or explaining non-local, large-scale, multi-variate, space-time relationships through interactive "what-if" data exploration that often requires the full context of the available data. An analytics algorithm restricted to work on a smaller working set to minimize I/O and finish in a reasonable time may miss important relationships or may not even be suitable for finding particular phenomena in large-scale data. In the end, although spatio-temporal data sets can be mined at various scales, many phenomena of interest become accessible only at a finer scale. Thus, the fundamental differences in data context and heterogeneity of access patterns call for analysis algorithms that are co-designed with efficient memory and data management solutions.

B. Energy Trend: Performance-energy tradeoffs are becoming an essential part of the design and implementation of architectures, system software, and algorithms.

In parallel computing, as computation scales, efficiency tends to be diminished, and if the power consumed in resource usage is not reduced proportionately, then the overall energy usage of the algorithm can increase in a super-linear fashion. Therefore, performance-energy tradeoffs must be considered in a systematic way to develop appropriate power-aware algorithms and software.

Although this observation can be generally applied, analytics provides particular opportunities, such as approximate algorithms, fixed-point computations, index-based analytics, knowledge priors-driven search, hierarchical exploration, and others (as discussed later) in order to develop energy-performance optimizations. A framework for analysis of performance/energy tradeoffs for large-scale parallel systems thus becomes necessary for designing and implementing energy-efficient scalable analytics and mining algorithms.

As part of this framework, identifying the right configuration among multiple resources for different classes of problems is essential to reduce the high-energy consumption. Given the number of applications, the task of breaking them down to be applicable on different architectures becomes tedious, if not impossible. However, some general solutions, when incorporated into the software framework, can help increase the energy-efficiency—minimizing data movement (e.g., via data compression) and generating approximate results (e.g., via partial-precision indexing and querying as well as fixed-point arithmetic), as described later.

C. Architecture Trend: Emerging HPC systems are adopting increasingly heterogeneous architectures.

The newer HPC systems will comprise heterogeneous components with multi-core processors along with general-purpose graphic process units (GPGPUs) accelerators and possibly FPGAs. Algorithms for scalable data analysis kernels must be capable of acceleration on hybrid multi-node, multi-core HPC architectures comprised of a mix of GPUs and FPGA, which is a significant challenge in its own right.

Furthermore, as systems scale, due to power and cost considerations, memory sizes are not expected to increase linearly. However, at the same time, technologies such as solid-state device (SSD) memories (e.g., FLASH, phase-change memories) have the potential to provide much higher capacities compared to DRAMs, at a lower power-cost point, while providing much lower latencies as compared to disks, particularly, for random accesses. Clearly, out-of-core algorithms, which can exploit SSDs will be needed to deal with such systems with the above constraints, but these newer features provide an opportunity for different design optimizations for algorithms and software.

II. Data Reduction

The data flow in scientific analysis pipelines proceeds through several resources, starting from computational nodes that run simulations and produce their output data, through local and shared storage devices, to visualization clusters and accelerators for analysis, and display. The movement of data not only results in a high-energy cost, but the discrepancy in the bandwidth of the "pipes" leads to significant idle energy atrophy. For example, visualization routines for analyzing data are primarily rate limited due to I/O. As both I/O and network throughput lack in bandwidth compared to the computational power, compute nodes on visualization clusters are forced to wait for data to be ingested. To increase resource utilization, one could add more storage devices, and build higher capacity networks. However, this does not scale, as energy and maintenance costs increase steeply, while offering diminishing returns. A systematic solution is needed, where the data is reduced at the earliest possible stage of the pipeline, such that the benefits of reduced data movement compound at each level of in transit data processing.

Data compression can thus be a viable solution for reducing data movement and the strain on storage resources. However, this "viabilty" is contingent upon high lossless reduction achieved by a compression method on scientific datasets, which is a significant challenge. This is primarily due to the hard-to-compress nature of scientific datasets that consist of predominantly real-valued numbers containing a significant amount of high-entropy fractional components. Standard compression routines are unable to extricate the parts of the data that contain the most information, and hence they deliver a low compression ratio and low (de-)compression throughput. Under both lossless and error-bounded lossy compression (needed to maintain high simulation fidelity), the majority of the current data reduction utilities are found to be lacking in compression performance.

Although reduction of data is essential, without supporting high-throughput (de-)compression routines, achieving energy-efficiency becomes an even harder problem. Compression routines expend additional CPU cycles that can consume almost an order of magnitude more power than storage accesses. Compression routines must, therefore, deliver high-performance on all three fronts– compression throughput, decompression throughput, and compression ratios—to achieve high energy efficiency. This is illustrated by ISOBAR'S method [4], [5], as briefly described next.

For a majority of the scientific datasets, the exponent and the higher-order bytes of the floating-point data tend to exhibit more repetition than the mantissa bytes. Based on this observation, ISOBAR applies a pre-conditioner to identify those byte columns that are "compressible." Instead of wasting CPU cycles trying to compress incompressible bytes in the data, ISOBAR groups the compressible parts together and performs compression only on those parts. By operating on a lower memory footprint, this method results in higher energy-efficiency, while simultaneously offering better throughput. A preliminary result of ISOBAR compression performance, along with its power consumption is shown in Figure 1. Apart from the higher throughput, the higher compression ratios offered by ISOBAR decrease the data movement, thus resulting in higher energy gains. In fact, ISOBAR achieves $\approx 20\%$ more reduction in storage, with a factor of $1.8 - 5.5$ reduction in energy consumption.

Compression is expected to play a vital role not only for

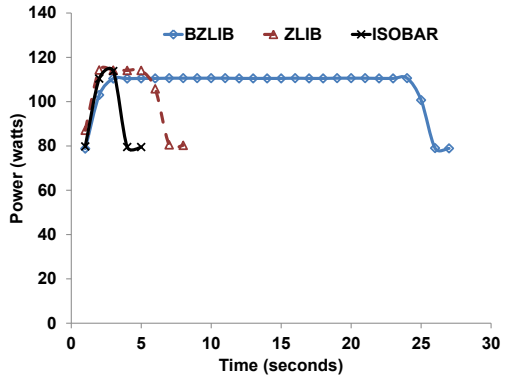

Figure 1. Power profile for ISOBAR, ZLIB, and BZLIB compression.

reducing the amount of data stored but also for addressing the data movement and I/O bottlenecks. Unfortunately, state-of-the-art I/O middleware does not have native support for write compression in a parallel context, due to the complexity of handling the resultant variable-sized compressed data buffers that require synchronization between all nodes performing shared-file I/O. The proper placement of (de-)compression routines for maximal I/O throughput and minimal data movement is a non-trivial issue.

Arguably, interleaving lossless compression and parallel I/O is a "secret sauce" for addressing the I/O bottleneck [5]. Through fast, dynamic identification of highly-compressible bytes, ISOBAR [4] can asynchronously write the remaining, incompressible bytes to storage, effectively hiding the cost of compression and I/O synchronization. This renders parallel write compression viable under the full resource utilization scenario.

Such an interleaving method naturally fits into data staging architectures [6], [3], where various data transformations occur "in transit" or in situ, between compute nodes and disks. With interleaved compression and I/O, we perform compression as an in situ transfer and storage optimization. Using various performance metrics about the system (e.g., network bandwidth and latency, disk read/write throughput) and the application (e.g., compression ratio and throughput, amount of data per core), ISOBAR's theoretical model accurately predicts the optimal balance among the placement of compression (on compute, I/O, or staging nodes), the data movement, and I/O. ISOBAR exhibits both read and write performance gains proportional to the degree of data reduction, which ranges as high as 46% on scientific datasets, in addition to reducing the total amount of data that is being stored and accessed. It is worth noting that merely compressing the data without interleaving, and then writing it to storage, results in only marginal gains (6%).

As application of compression methods becomes more ubiquitous, wherein they are no longer employed merely for archival purposes, maintaining read and query efficiency becomes of paramount importance, as described next.

III. Query-driven Data Analytics

Query processing is a driving force behind many of the statistical and visual data analytics routines on scientific data. Within an exploratory setting, processing ad-hoc queries at high-throughput requires the use of precomputed indexes. However, indexing scientific data at scale presents a number of storage, computational, and energy challenges.

First, the prime impediment stems from the unreasonable storage requirements of indexing high-cardinality floating-point variables. Second, index construction for scientific databases is traditionally performed in an expensive post-processing manner, once all of the data has been generated and global descriptive statistical information has become available for optimal index parameter selection. In contrast, next-generation scientific data management will likely demand high-throughput, parameter-free index building while the data is being produced. Third, processing the majority of the queries run on scientific data is typically rate limited by I/O, not only due to the massive size of the index and data, but also due to heterogeneous access patterns induced by various types of queries. Finally, query operations have not traditionally been optimized for energy-efficiency, which is critical for a sustainable scientific data management solution.

I/O-bound operations, such as those induced by query processing, typically deliver unacceptable response times, but are reasonably energy-efficient for a number of reasons (e.g. low power consumption by disks). The inverse relationship between performance and energy efficiency necessitates re-evaluating the existing query processing and indexing designs.

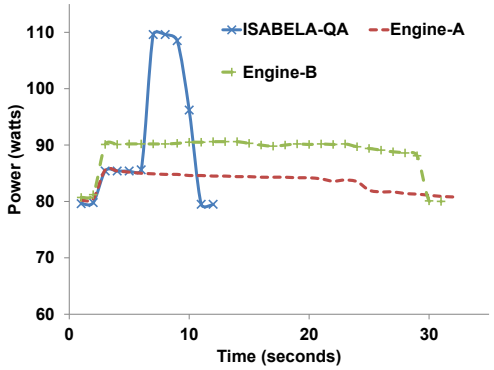

Figure 2. Example energy profiles for ISABELA-QA in comparison with popular scientific database technologies.

Scientific query processing is used in performing a wide variety of analysis functions, ranging from identifying correlations across spatial and temporal scales to detecting anomalies, clustering, dimension reduction, and other mining operations. For example, consider the following query on a dataset produced from a climate simulation:

"Which cities in California experienced abnormally high temperatures (over $100^{\circ}F$) during the month of May in the year of 2000?" This query requires efficient retrieval of data across space, time, and value. To efficiently accelerate queries of this nature, several indexing schemes pioneered in databases can be used. However, building B-Trees and bitmap indexes [10] on these datasets can require storage of approximately 100%-300% of the raw data. Considering the fact that indexes tend to be auxiliary data structures, the total storage requirement can become over 300% of the original data. Moreover, traditional databases do not natively support array-based retrieval, which is highly common in scientific data analytics. Analytics not only needs index-efficient operations, like retrieving small-sized "interesting" regions, but also fast large-scale data retrieval. Such non-trivial challenges often lead to performance rate-limited by I/O, incurring higher data movement than necessary, with additional heavy-weight indexes.

Arguably, several classes of queries on exascale scientific datasets would thus be more effective if the query engines were tailored around coarse-grained and storage light-weight indexes. ISABELA-QA [14] operates by building indexes on the metadata generated by the ISABELA compression methodology [15], instead of the actual data. Data in the compressed form can be used to quickly extract the relevant information needed to answer queries efficiently. Figure 2 shows a promising result with ISABELA-QA, a query processing engine over compressed data, which results in better overall performance as well as energy-efficiency. Its overall storage requirement (compressed scientific data+index) was observed to be around 40% of the original data with a bounded per-point relative error of less than 0.1%. With the above result in mind, we next describe some future research directions for building efficient querying schemes.

Redesigning Indexing and Storage: Future solutions can look towards techniques that have tightly coupled implementations of data reduction, layout optimizations, and indexing for delivering a high-throughput light-weight query processing engine. By alleviating the performance gap on I/O, one can move away from the traditional notion of query processing as being an I/O-bound problem, and design faster I/O-CPU balanced solutions, with minimal data movement. Further investigation is required into effectively partitioning the data for querying over different access patterns, parallelizing the task of building indexes for efficient parallel execution, and load-balancing queries from an energy perspective.

Approximate Querying: Application scientists constantly perform exploratory querying over data, where they are interested in a quick, approximate result of a query rather than in a slow, complete result. One method to enable such an approach would be to employ querying only the indexes, rather than the entire dataset. This could, in fact, be an effective solution for visualization routines, as they can usually tolerate some amount of loss in precision without appreciable decrease in quality of output. Approximate query processing, as employed in ISABELA-QA, or by querying the precomputed indexes, can result in significant improvement in query throughput. Then, if the user requests more accurate results, extra disk accesses can be made to iterate over full-precision data. The above is only one example of approximate querying, and it becomes evident that this can be extended beyond the precision levels offered by data aggregation or sampling, in a manner, similar to multi-resolution analysis commonly applied in visualization.

Cooperative Computing: Query optimizers in databases typically use a cost-based model to identify the most efficient execution plan for a given query. Recent works [16], [17] additionally incorporate the energy cost when picking the ideal query execution plan and achieve appreciable savings in energy consumption by picking the most energy-efficient plan. The scope for research in this area is expanded when factoring in the hetoregenous platform. Query optimizers should be able to break down the execution plan, to incorporate co-processing on GPU and CPU simultaneously. Such optimizers should consider different configuration options, and intelligent execution pipeline to minimize both energy consumption and query response time.

IV. Scalable Analytical Kernels

As both the size and complexity of scientific data grow, the demands for more sophisticated analytics increases. While domain scientists will likely remain the primary conceptual designers of how their application-specific data get analyzed, they must be relieved from the burden of optimizing the analytical pipelines on the next-generation HPC systems.

By examining a number of data analysis pipelines in a particular area, it is possible to determine to what extent they have similar performance characteristics and share computational kernels. Highly optimized versions of these kernels can then be used in building higher-level algorithms. By using these kernels, designers can achieve efficient implementations of their algorithms easily.

Fortunately, the execution of many data mining algorithms is dominated by a small number of kernels. As an illustration, Table I [12] lists important kernels for representative algorithms in classification, association rule mining, and clustering. Note that the fraction of execution by the top three kernels usually exceeds 90% and reaches up to 99% for this set of algorithms. Thus, a promising strategy would provide a generic and highly optimized set of core, or kernel analytics functions, from which a broad constellation of high performance analytical pipelines could be organically constructed.

Arguably, a comprehensive library of such analytical kernels could eventually bring analytics algorithms to the next level. NU-MINEBENCH utilizes this strategy in the context

Table I
THE FREQUENCY OF KERNEL OPERATIONS IN ILLUSTRATIVE DATA MINING ALGORITHMS AND APPLICATIONS

Application	Top 3 Kernels (%)		
	Kernel 1 (%)	Kernel 2 (%)	Kernel 3 (%)
K-means [1]	Distance (68)	Center (21)	minDist (10)
Fuzzy K-means [1]	Center (58)	Distance (39)	fuzzySum (1)
BIRCH	Distance (54)	Variance (22)	redist.(10)
HOP [1]	Density (39)	Search (30)	Gather (23)
Naïve Bayesian [1]	probCal (49)	Variance (38)	dataRead (10)
ScalParC [1]	Classify (37)	giniCalc (36)	Compare (24)
Apriori	Subset (58)	dataRead (14)	Increment (8)
Eclat	Intersect (39)	addClass (23)	invertC (10)
SVMlight [1]	quotMatrix(57)	quadGrad (38)	quotUpdate(2)

[1] Application has the sum of top 3 kernel frequencies exceeding 90%

of heterogeneous HPC computing. It currently consists of 20+ analytics and mining algorithms and software [12], [8]. The use of such analytical kernels as plug-ins in the existing environments for statistical computing such as R and WEKA could offer a desirable ease-of-use. In fact, the APIs for such kernels, as implemented within RSCALAPACK and parallel R libraries mimick R-like serial functions and enable software reuse with minimal or no changes to the current analysis codes [13].

V. APPROXIMATE ANALYTICS

Data mining and analytics algorithms are approximate in many ways. First, discovery of patterns, clustering, predictive modeling and learning relationships provide results within an error bound (e.g., the convergence error for clustering is within a pre-specified bound), meet certain confidence constraints (e.g., the probability of a decision rule to be true is above 80%), or have certain confidence in model prediction (e.g., precision is above a threshold). Second, the data itself is typically noisy due to the collection process or assumptions in the underlying mathematical models used to generate the data. Third, the existence of many potential alternatives requires quick exploration of the space of mining models where it is sufficient to know relative ranking of the models rather than to have accurate results in order to allow a user to perform a more in-depth analysis on a smaller parameter space. Therefore, approximate algorithms for data analytics are needed for both performance and resource usage purposes.

Fixed-Point Arithmetic on Analytical Kernels: Several hardware architectures, including mobile devices and embedded systems, lack sufficient floating-point computation power and thus require algorithms that can deliver the required performance while operating with lower energy. Algorithms implemented using fixed-point arithmetic [11] trade precision for energy-and-computational efficiency, replacing expensive floating-point operations with a sequence of faster integer operations. Compute-intensive parts of kernels described in Table I depend heavily on floating-point computations, which consume a significant amount

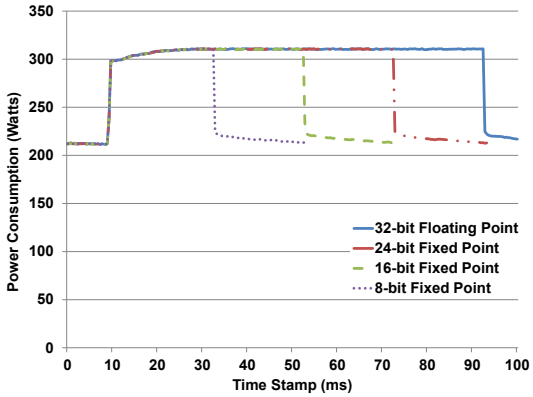

Figure 3. Power consumption on transferring varying fixed-point representations from the CPU to the GPU.

of resources on accelators like FPGAs, making them prime candidates for fixed-point optimization. Gains in energy result since fewer precision bits would need to be transferred to the FPGAs as shown in Figure 3.

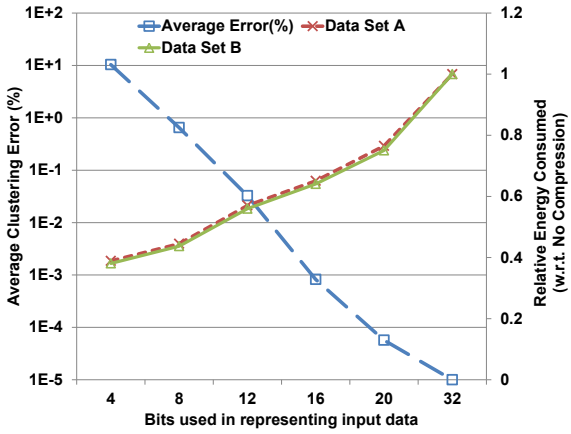

Figure 4. Comparison of energy consumption and k-means clustering error with varying fixed-point representations.

Fixed-point representations use a virtual decimal point in an integer data that seperates the inegral and fractional

parts of a real number. A fixed-point variable represented using **Q.i.f** uses **i** bits for the integer part and **f** bits for the fractional part of the data. For implementation efficiency, the values of **i** and **f** are generally chosen as powers of two, and the fixed-point number is represented in base 2, rather than base 10. The target analytical kernel is analyzed and those functional blocks that would ideally benefit with a fixed-point representation are chosen for conversion. For these blocks, a range analysis is applied to identify variables susceptible to an overflow or underflow error for various combinations of integer and fractional part sizes in the fixed-point representation; i.e., the values of **i** and **f**.

The ensuing accuracy analysis phase details the gradual loss in accuracy between fixed-point and floating-point representations. While gradual loss in accuracy is expected, care must be taken so that critical errors do not arise. In those highly-sensitive parts, native implementations floating-point have to be retained. Figure 4 shows the performance benefits of a reduced-precision implementation of $k-$means clustering algorithm. Using a fixed-point 12-bit representation results in 44% energy saving, with just 0.03% relative error in mis-classification. Similar results are seen with Pearson correlation, with speedups of ≈ 2.5 and a $50-70\%$ reduction in energy consumption.

VI. Acknowledgements

We would like to thank David Boyuka II and John Jenkins for insightful comments and discussions on the paper. This work was supported in part by the U.S. Department of Energy, Office of Science and the U.S. National Science Foundation (Expeditions in Computing). Oak Ridge National Laboratory is managed by UT-Battelle for the LLC U.S. D.O.E. under contract no. DEAC05-00OR22725.

References

[1] EPA Report to Congress on Server and Data Center Energy Efficiency Public Law 109-431. *Public Law*, 109:431, 2007.

[2] http://iee.ucsb.edu/greenscale. The Institute for Energy Efficiency, UC Santa Barbara. 2008.

[3] H. Abbasi, M. Wolf, G. Eisenhauer, S. Klasky, K. Schwan, and F. Zheng. DataStager: Scalable data staging services for petascale applications. In *Proceedings of the 18th International Symposium on High Performance Distributed Computing*, HPDC '09, pages 39–48. ACM, 2009.

[4] E. R. Schendel, Y. Jin, N. Shah, J. Chen, C.S. Chang, S-H. Ku, S. Ethier, S. Klasky, R. Latham, R. Ross, and N. F. Samatova. ISOBAR preconditioner for effective and high-throughput lossless data compression. In *IEEE 28th International Conference on Data Engineering (ICDE)*, 2012.

[5] E. Schendel, S. Pendse, J. Jenkins, D. Boyuka, Z. Gong, S. Lakshminarasimhan, H. Kolla, J. Chen, Q. Liu, S. Klasky, R. Ross, and N. F. Samatova. ISOBAR hybrid compression-I/O interleaving for large-scale parallel I/O optimization. In *Proceedings of the 21st international symposium on High performance distributed computing*, HPDC, 2012.

[6] H. Abbasi, G. Eisenhauer, M. Wolf, K. Schwan, and S. Klasky. Just in time: Adding value to the IO pipelines of high performance applications with JITStaging. In *Proceedings of the 20th International Symposium on High Performance Distributed Computing*, HPDC '11, pages 27–36. ACM, 2011.

[7] J. G. Koomey. Estimating Total Power Consumption by Servers in the US and the World, 2007.

[8] J. Pisharath, J. Zambreno, B. Ozisikyilmaz, and A. Choudhary. Accelerating Data Mining Workloads: Current Approaches and Future Challenges in System Architecture Design. In *International Workshop on High-Performance and Distributed Mining (HPDM)*, 2006.

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

[10] K. Wu, E. Otoo and A. Shoshani. On the Performance of Bitmap Indices for High Cardinality Attributes. In *Proc. of the Thirtieth international conference on Very large data bases (VLDB) - Volume 30*, pages 24–35, 2004.

[11] R. Narayanan, B. Ozisikyilmaz, G. Memik, A. Choudhary, J. Zambreno. Quantization Error and Accuracy-Performance Tradeoffs for Embedded Data Mining Workloads. In *Proceedings of the 7th international conference on Computational Science*, ICCS 2007, pages 734–741, 2007.

[12] R. Narayanan, B. Ozisikyilmaz, J. Zambreno, G. Memik, and A. Choudhary. MineBench: A Benchmark Suite for Data Mining Workloads. In *IEEE International Symposium on Workload Characterization*, pages 182 –188, Oct 2006.

[13] S. B. Yoginath, N. F. Samatova, D. Bauer, G. Kora, G. Fann and A. Geist. RScaLAPACK: High-Performance Parallel Statistical Computing with R and ScaLAPACK. In *ISCA PDCS*, pages 61–67, 2005.

[14] S. Lakshminarasimhan, J. Jenkins, I. Arkatkar, Z. Gong, H. Kolla, S-H Ku, S. Ethier, J. Chen, C.S. Chang, S. Klasky, R. Latham, R. Ross and N. F. Samatova. ISABELA-QA: Query-driven Data Analytics over ISABELA-compressed Scientific Data. In *International Conference for High Performance Computing, Networking, Storage and Analysis (SC 2011)*, Seattle, Washington, 2011.

[15] S. Lakshminarasimhan, N. Shah, S. Ethier, S. Klasky, R. Latham, R. Ross, and N. F. Samatova. Compressing the Incompressible with ISABELA: In-situ Reduction of Spatio-temporal Data. In *17th European International Conference on Parallel Processing (Euro-Par)*, pages 366–379. 2011.

[16] W. Lang, R. Kandhan and J. M. Patel. Rethinking Query Processing for Energy Efficiency: Slowing Down to Win the Race. *IEEE Data Eng. Bull.*, 34(1):12–23, 2011.

[17] Z. Xu, Y-C. Tu and X. Wang. Exploring Power-Performance Tradeoffs in Database Systems. In *IEEE 26th International Conference on Data Engineering (ICDE)*, pages 485 –496, March 2010.