Energy and Performance—Can a Wimpy-Node Cluster Challenge a Brawny Server?

Daniel Schall, Theo Härder
{schall, haerder}@cs.uni-kl.de
DBIS Group, University of Kaiserslautern, Germany

Abstract: Traditional DBMS servers are often over-provisioned for most of their daily workloads and, because they do not provide energy proportionality, waste more energy than necessary. A cluster of wimpy servers, where the number of nodes can dynamically adjust to the current workload, might offer better energy characteristics for these workloads. Yet, clusters suffer from *friction losses* and cannot quickly adapt to the workload, whereas a single server delivers maximum performance instantaneously.

Designed for a cluster of nodes, our WattDB system primarily aims at energy proportionality for a wide range of DB applications. In this paper, we check this system under OLTP and OLAP workloads against a single-server DBMS in terms of throughput/response time and energy efficiency. To test the system's ability to adjust to changing workloads, we execute several benchmark at differing system activity levels. To quantify possible energy saving and its conceivable drawback on query runtime, we evaluate our WattDB implementation—to obtain maximum accuracy possible—on a cluster of wimpy nodes as well as on a single, brawny server and compare the results w.r.t. performance and energy consumption. Our findings confirm that—especially for OLAP workloads—energy can be saved without sacrificing too much performance.

1 Introduction

Saving energy is a concern in all areas of IT. Studies have shown that single servers have only limited potential for energy optimizations and that, in general, the best performing configuration is also the most energy-efficient one [THS10]. This is due to the narrow power spectrum between idle and full utilization of a single server [BH07].

Today's server hardware is not energy-proportional; even at low utilization levels, it consumes a significant amount of power rapidly approaching peak power [HHOS11]. Scaling systems down automatically when idle, thus preventing high idle power consumption, is the main focus of today's servers to achieve energy efficiency. Of course, several components such as CPUs are able to quickly change into sleep states, requiring less energy, when idle. But other components, such as two main energy consumers of DBMSs—main memory and storage drives—exhibit bad and more or less load-independent energy characteristics. Hence, especially for DB servers, ~50% of its power is already wasted when idle.

Therefore, better energy proportionality cannot be achieved with current, centralized solutions. This observation also holds for traditional DBMSs, composed of a single server with huge main memory and lots of storage drives attached. In contrast to centralized, brawny servers, a scale-out cluster of lightweight (wimpy) servers has the ability to shutdown single nodes independently. At an abstract level, this enables the cluster to dynamically add or remove storage and processing power based on the cluster's utilization.

With cloud computing, elastic systems have emerged that adapt their size to the current workload. While stateless or lightweight systems can easily increase or reduce the number of active computing nodes in a cluster, a database faces much more challenges due to the need of high interactions among the nodes and fast reachability of DB data. Similar to cloud-based solutions, we hypothesize that a cluster of nodes may adjust the number of active nodes to the current demand and, thus, approximate energy proportionality.

Based on these observations, we developed WattDB, a research prototype of a distributed DBMS cluster, running on lightweight, Amdahl-balanced nodes using commodity hardware. The cluster is intended to dynamically shrink and expand its size, dependent on the workload. However, reconfiguring a cluster to balance node utilization requires data to be moved from node to node. But, copying data is time-consuming and adds overhead to the already loaded cluster. Hence reducing both, time and overhead, is crucial for an elastic DBMS. Although the cluster may not be as powerful as a monolithic server, we expect our system to consume significantly less energy for typical DBMS workloads.

In this paper, we compare a single, brawny server with a cluster of wimpy nodes under OLTP and OLAP workloads, running TPC-H and TPC-C benchmarks, respectively. In Sect. 2, we give an overview of recent research addressing *partitioning, elasticity, and energy efficiency* of DBMSs. Sect. 3 introduces important aspects of our energy-proportional database cluster, whereas Sect. 4 contains the results of several empirical experiments and compares energy use and performance of our cluster to those of a brawny server. In Sect. 5, we summarize the main issues of our work and give some conclusions.

2 Related Work

Reducing energy consumption of servers and enabling dynamic reconfiguration are all subject to a variety of research approaches. For the reason, we give a short overview of related works in three fields that serve as a building blocks of our research.

2.1 Dynamic Clustering

Traditional clustered DBMSs do not dynamically adjust their size (in terms of the number of active nodes) to their workload. Hence, scale-out to additional nodes is typically supported, whereas the opposite functionality, shrinking the cluster and centralizing the processing—the so-called scale-in—, is not. Recently, with the emergence of clouds, a change of thinking occurred and dynamic solutions became a research topic.

In his PhD thesis [Das11], Sudipto Das implemented an elastic data storage, called Elastras, able to dynamically grow and shrink on a cloud. As common in generic clouds, his work is based on decoupled storage where all I/O involves network communication. *Key Groups*, application-defined sets of records frequently accessed together, can be seen as dynamic partitions that are often formed and dissolved. By distributing the partitions among nodes in the cluster, both performance and cost can be controlled.

A lot more data management systems working on a cloud have been proposed. In [BFG$^+$08], Brantner et al. designed a DBMS using Amazon S3 as storage and running on top. Lomet et al. [LFWZ09] divided the database into two layers, one transactional and one persistence component that can run independently.

In [AFP$^+$09], Armbrust et al. proposed a scalable storage layer supporting consistency and dynamic scale-out/in called SCADS. Objects in SCADS are stored in logical order. Hot, i. e., frequently accessed objects are distributed among disks to improve access latencies and mitigate bottlenecks. The system was also extended to automatically adjust to workload changes and autonomously redistribute data.

Besides relational approaches, other implementations relax traditional DBMS properties to gain performance and simplify partitioning. Yahoo PNUTS [CRS$^+$08], Bigtable [CDG$^+$08], and Cassandra[1], are example of systems sacrificing transaction or schema support and query power [AFP$^+$09]. Instead of arbitrary access patterns on the data, only primary key accesses to a single record are supported [VCO10].

Amazon's SimpleDB[2] allows transactions to access multiple records, but limits accesses to single tables. Moreover, most current scalable data storage systems lack the rich data model of an RDBMS, which burdens application developers with data management tasks. Yet, no *fully-autonomous, clustered DBMS* exists providing ACID properties for transactions and SQL-like queries while dynamically adjusting its size to the current workload.

2.2 Energy Optimizations

Lang et al. [LHP$^+$12] have shown that a cluster suffers from friction losses due to coordination and data shipping overhead and is therefore not as powerful as a comparable heavyweight server. On the other hand, for moderate workloads, i. e., the majority of real-world database applications, a scale-out cluster can exploit its ability to reduce or increase its size sufficiently fast and, in turn, gain far better energy efficiency.

In [SH13a], we already explored the capabilities and limitations of a clustered storage architecture that dynamically adjusts the number of nodes to varying workloads consisting of simple *read-only page requests* where a large file had to be accessed via an index. We concluded that it is possible to approximate energy proportionality in the storage layer with a cluster of wimpy nodes. However, attaching or detaching a storage server is rather expensive, because (parts of) datasets may have to be migrated. Therefore, such events (in appropriate workloads) should happen on a scale of minutes or hours, but not seconds.

In [SH13b], we focused on the query processing layer—again for varying workloads consisting of two types of *read-only SQL queries*—with similar conclusions. In this contribu-

[1]`http://cassandra.apache.org/`
[2]`http://aws.amazon.com/simpledb/`

tion, we revealed that attaching or detaching a (pure) processing node is rather inexpensive, because repartitioning and movement of data is not needed. Hence, such an event can happen in the range of a few seconds—without disturbing the current workload too much.

We substantially extended the kind of DBMS processing supported by WattDB to *complex OLAP / OLTP workloads consisting of read-write transactions* in [SH14]. For this purpose, we refined and combined both approaches to get one step closer to a fully-featured DBMS, able to process OLTP and OLAP workloads simultaneously. In this work, we were able to trade performance for energy savings and vice versa. Yet, we found out that the adaptation of the cluster and the data distribution to fit the query workload is time-consuming and needs to be optimized.

As discovered before, a single-server-based DBMS is far from being energy-proportional and cannot process realistic workloads in an energy-efficient way. Our previous research indicates that a cluster of lightweight (wimpy) servers, where nodes can be dynamically switched on or off, seems more promising. In this paper, we compare our scale-out cluster to a big server to quantify possible energy savings and to discover promising workloads.

3 Cluster vs. Big Server

Our cluster hardware consists of n (currently 10) identical nodes, interconnected by a Gigabit-ethernet switch. Each node is equipped with an Intel Atom D510 CPU (with two threads using HyperThreading) running at 1.66 GHz, 2 GB of DRAM and an SSD for data storage. The configuration is considered Amdahl-balanced [SBH$^+$10], i. e., balanced w.r.t. I/O and network throughput on one hand and processing power on the other. By choosing commodity hardware with limited data bandwidth, Gigabit-Ethernet wiring is sufficient for interconnecting the nodes. All nodes can communicate directly.

To compare performance and energy savings, we ran the same experiments again on a single, brawny server. This server has two *Intel Xeon X5670* processors with 24 GB of RAM and 10 SSDs.[3] Each CPU has 12 cores and 24 threads (using HyperThreading), running at 2.93 GHz.

Fig. 1 sketches the cluster with 10 nodes and the big server. For comparison, we have highlighted the main components (CPU cores, main memory and disk) inside the nodes as well as the communication network. Each wimpy node consumes ~22 – 26 Watts when active (based on utilization) and ~2.5 Watts in standby. The interconnecting network switch consumes ~20 Watts and is included in all measurements.

In its minimal configuration—with only one node and the switch running and all other nodes in standby—the cluster consumes ~65 Watts. This configuration does not include any disk drives, hence, a more realistic minimal configuration requires ~70 Watts. In this state, a single node is serving the entire DBMS functionality (storage, processing, and cluster coordination). With all nodes running at full utilization, the cluster will consume

[3]For a fair comparison with the cluster of wimpy nodes, we reduced RAM to 24GB, although the server can handle much more. Yet, with more main memory, the power use of the server would also be much higher.

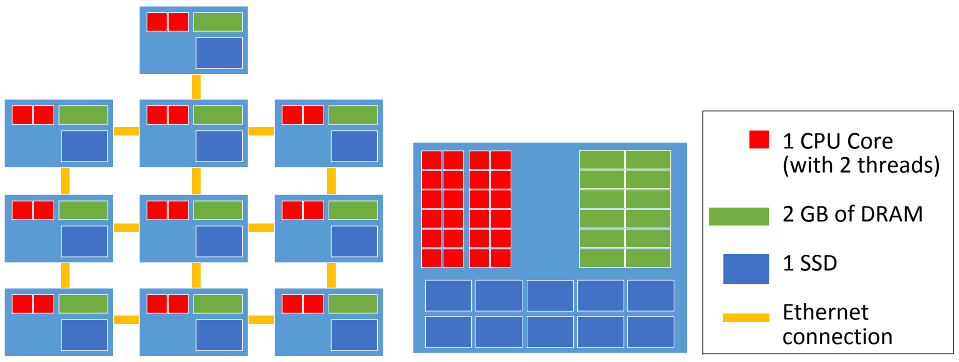


Figure 1: The 10-node cluster compared with the brawny server

~260 to 280 Watts, depending on the number of disk drives installed. This is another reason for choosing commodity hardware which uses much less energy compared to server-grade components. For example, main memory consumes ~2.5 Watts per DIMM module, whereas ECC memory, used in the brawny server, consumes ~10 Watts per DIMM.

The power consumption of the big server (with 10 SSDs) ranges from ~200 Watts when idle to ~430 Watts at full utilization.[4] In theory, the systems should show similar performance. All nodes in the cluster come with 16.6 (10x1.66) GFLOPS, whereas the performance of the big server is rated with 17.6 GFLOPS. Furthermore, L2 caches and memory bandwidth of both systems are similar and the same number of disks is installed.

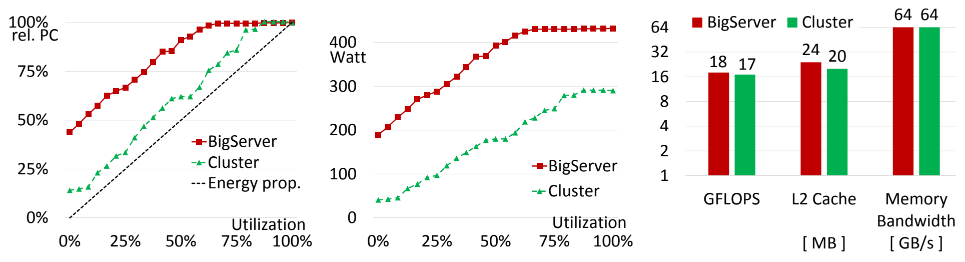
Fig. 2 visualizes power consumption and theoretical performance figures, as given in the product sheets, for both systems. Of course, the relationships shown for the power consumption are idealized, because power consumption of the big server and the full cluster may differ to a certain amount. Fig. 2a shows how energy proportionality drawn for the big server is much better approximated by the cluster where at least 1 and—depending on the workload—up to 10 nodes are active. Fig. 2b gives an impression about the power consumption of both systems when running at a specific activity level.

3.1 DBMS Software

By the time, research gained interest in energy efficiency of database servers, no state-of-the-art DBMS was able to run on a dynamically adapting cluster. To test our hypothesis (see Sect. 1), we developed *WattDB* that supports SQL query processing with ACID properties, but is also able to adjust to the workload by scaling out or in, respectively.

To enable a fair comparison, the same software is running on the cluster and the big server. On the latter, the dynamic features of WattDB are not needed and are therefore disabled. The smallest configuration of WattDB is a single server, hosting all database functions and

[4]These measurements include only 24 GB of DRAM as previously explained.



(a) Relative power consum. (b) Absolute power consum. (c) Performance figures

Figure 2: Power consumption and theoretical performance figures for both systems

acting as endpoint to DB clients. This server is called *master node*. DB objects (tables, partitions) and query evaluation can be offloaded to arbitrary nodes in the cluster to relieve the node, but it will always act as the coordinator and client endpoint. Some of the key features and design considerations of WattDB are explained in the following.

3.2 Dynamic Query Processing

To run queries on a cluster of nodes, distributed query plans are generated on the master node. Except data access operators which need local access to the database's records, all query operators can be placed on remote nodes. Running query operators on a single node does not involve network communication among query operators, because all records are transferred via main memory. Distributing operators implies shipping of records among nodes and, hence, introduces network latencies. Additionally, the bandwidth of the Gigabit Ethernet, which we are using for our experiments, is relatively small, compared to memory bandwidth.

To mitigate the negative effects of distribution, WattDB is using *vectorized volcano-style query operators* [Gra94, BZN05]; hence, operators ship a set of records on each call. This reduces the number of calls between operators and, thus, network latencies. To further decrease network latencies, buffering operators are used to prefetch records from remote nodes. *Buffering operators* act as proxies between two (regular) operators; they asynchronously prefetch sets of records, thus, hiding the delay of fetching the next bunch of records.

In WattDB, the query optimizer tries to put pipelining operators[5] on the same node to minimize latencies. Offloading pipeline operators to a remote node has little effect on workload balancing and, thus, does not pay off. Instead, blocking operators[6] may be placed on remote nodes to equally distribute query processing. They generally consume more resources (CPU, main memory) and are therefore prime candidates for workload balancing in the cluster.

[5]Pipelining operators can process one record at a time and emit the result, e. g., projection operators.
[6]Blocking operators need to receive all records before emitting the first result record, e. g., sorting operators.

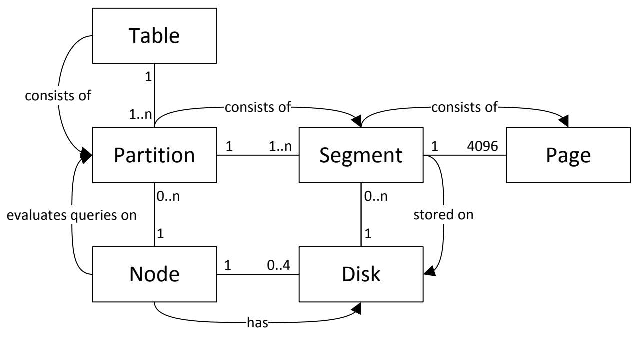


Figure 3: Database schema

3.3 Dynamic Reorganization

The master node is coordinating the whole cluster. It is globally optimizing the query plans, whereas regular nodes can locally optimize their part of the plan. Furthermore, it takes nodes on- and offline and decides when and how DB tables are (re)partitioned.

Every node is monitoring its utilization: CPU, memory consumption, network I/O, and disk utilization (storage and IOPS). Additionally, performance-critical data is collected for each database partition, i. e., CPU cycles, buffer page requests and network I/O. With these figures, we can correlate the observed utilization of cluster components to (logical) database entities. Hence, both types of data are necessary to identify sources of cluster imbalance. We use the performance figures of the components to identify their over- or under-utilization. In addition, activity recording of database entities is needed to determine the origin of the cluster's imbalance. For this reason, the nodes send their recording every few seconds to the master node.

The master checks the incoming performance data to predefined thresholds—with both upper and lower bounds. If an overloaded component is detected, it will decide where to distribute data and whether to power on additional nodes and resume their cluster participation. Similar, underutilized nodes trigger a scale-in protocol, i. e., the master will distribute the data (processing) to fewer nodes and shutdown the nodes currently not needed. Decisions, what data to migrate and where, are done based on the current utilization of the nodes, the expected query workload, and the estimated cost, it will take to migrate data between nodes.

In WattDB, we have implemented different policies regarding the scale-out behavior. First, each node in the cluster stores data on local disks to minimize network communication. If storage space of a node is in short supply, database partitions are split up on nodes with free space.

Second, WattDB tries to keep the I/O rate for each storage disk in a certain range. Underutilized disks are eligible for additional data—either newly generated by INSERT operations or migrated from overloaded disks. Utilization among storage disks is first locally balanced on each node, before an allocation of data from/to other nodes is considered.

Third, each node's CPU utilization should not exceed the upper bound of the specified threshold (80%). As soon as this bound is violated on a node, for more than one minute, WattDB first tries to offload query processing to underutilized nodes.[7] If the overload situation cannot be resolved by redistributing the query load, the current data partitions and their node assignments are reconsidered. When a partition causing overload is identified, it is split according the partitioning scheme applied, where affected segments are moved to other nodes [SH15]. Fig. 3 sketches the relationships between tables, partitions, nodes, and segments.

For underutilized nodes, an inverse approach is needed. A scale-in protocol is initiated, which quiesces the involved nodes from query processing and shifts their data partitions to nodes currently having sufficient processing capacity, as soon as the average utilization over one minute falls below 30%.[8]

Similar rules exist for network and memory utilization, e. g., if the working sets of the transactions become too big for the DB buffer, repartitioning is triggered. WattDB makes decisions based on the current workload, the course of utilization in the recent past, and the expected future workloads [KHH12]. Additionally, workload shifts can be user-defined to inform the cluster of an expected change in utilization.

Cost of reorganization Moving data is an expensive task, in terms of energy consumption and performance impact on concurrently running queries. Data reorganization binds some computing resources, which would be needed to optimally process the query workload. This competition leads to fewer resources for running database workloads and, in turn, reduces query throughput. However, the reorganization cost should amortize by reducing the energy consumption of subsequent queries. Though it is difficult to calculate the exact energy consumption of a data move operation with respect to the impact of running queries, the energy cost can be estimated with the duration of the move operation and the (additional) power consumption. Hence, moving 1 GByte of data to a dedicated node with 25 Watts power consumption will require approximately 10 seconds and 250 Joules.

To save energy, reconfiguration overhead needs to pay off by reducing future query runtimes. Likewise, scale-in must trigger when the cluster is able to handle the workload with less nodes. To estimate the impact of reorganization, WattDB relies on a simplified cost model where upcoming workload predictions and maintenance costs are calculated.

3.4 Power Measurement

We have developed a measurement device, capable of monitoring the power and energy consumption of each node in the cluster. The device is also able to monitor these metrics of the big server. This device sends the stream of measurements to a connected PC, running the monitoring software. The monitoring software can further capture the number of active database nodes and the total throughput and response times of queries during the tests.

[7]This works well for operators like **SORT**, **GROUP**, and **AGGREGATE**.

[8]The rules implemented are a little more sophisticated, depending on additional factors, e. g., overall utilization and intensity of workload shifts. In this work, we illustrate a simplified model to keep things short.

This computer is controlling the benchmark execution by submitting queries to the master node; thus, it enables fine-grained monitoring in correlation with the benchmark runs. The measurement frequency of the device reaches up to 100 Hz; hence, we are able to determine power use in high resolution. A detailed description of the measurement device can be found in [HS11].

The energy measurement device is only used for external monitoring and not connected to the master node. Internally, the DBMS is working with estimates to determine overall power consumption.

4 Experiments

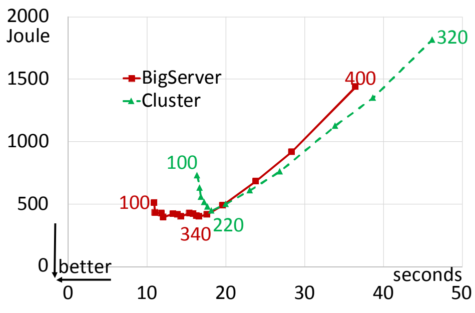
To compare the energy consumption of our cluster to that of a traditional DB server, we have processed OLAP and OLTP workloads on both platforms. First, we have run performance-centric benchmarks, i. e., DB clients continuously submit queries to the full cluster or the big server, to assess peak performance of both systems. Next, we have evaluated energy-centric benchmarks [SHK12], i. e., requests of DB clients are issued only at timed intervals thereby delaying query submission, to identify the energy-efficiency potential of the big server and the cluster. In the following, we first describe the experimental setup, before we present our results.

4.1 Experimental Setup

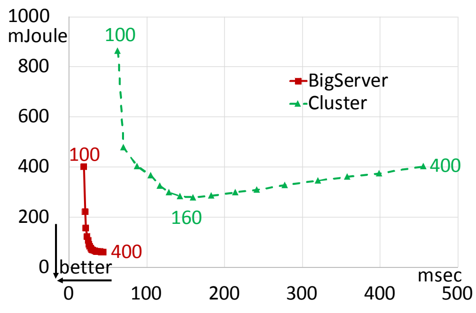
For all experiments, using OLTP and OLAP, we have set up the systems as previously described. A separate server, directly connected to the master node and the big server, respectively, is used as the benchmark driver, submitting queries to the cluster as well as monitoring response time and throughput. The previously introduced power measurement device is also hooked up to the benchmark driver to correlate all measurements with energy consumption.

OLAP workloads: For measuring OLAP performance and energy efficiency, we are using the well-known TPC-H benchmark with a scale factor of 300; hence, 300 GB of raw data are generated. Due to additional indexes and storage overhead, the final DB has approx. 460 GB of raw data. On the centralized server, small tables are stored on a single disk, whereas larger ones, e. g., the *LINEITEM* and *ORDERS* tables, are partitioned and distributed among all disks to increase access bandwidth and to parallelize processing on partitions.

On the cluster, the *REGION* and *NATION* tables are replicated to all nodes, while the other tables are partitioned and distributed equally among the nodes. In static benchmarks, no repartitioning occurs, even if the initial distribution leads to hotspots in the data, that impact the node's performance. If dynamic features of WattDB are enabled, the DBMS will automatically repartition as previously described.



(a) OLAP: energy/query over response time



(b) OLTP: energy/query over response time

Figure 4: Peak performance and energy consumption for both systems

OLTP workloads: For online transaction processing, we are running the TPC-C benchmark on the systems with a scale factor of 1000. Hence, a thousand warehouses were generated on the cluster, consisting of about 100 GB of data. Due to additional indexes and storage overhead, the final DB has ~200 GB of raw data at the start of the experiments.

4.2 Performance-Centric Benchmark

First, to evaluate the *peak performance* of both systems, we run performance-centric benchmarks similar to TPC-C and TPC-H on the cluster and the big server. We repeated the experiments with varying numbers of parallel DB clients (between 100 and 400) to estimate a saturation point, i. e., how many parallel queries the systems can process—without entering an overload state. Summarizing the OLAP results in Fig. 4a, the x-axis and y-axis illustrate response time in seconds resp. energy consumption per query. Numbers on the individual graphs annotate the amount of parallel DB clients for this curve progression.

Compared to the cluster, we can conclude that the big server handles queries for the same number of clients generally faster and with lower energy consumption per query, i. e., better energy efficiency. Up to 340 DB clients, the response times on the big server increase only slightly. Then, the server seems saturated and runtimes start to build up. Consequently, energy consumption per query rises.

Even for medium-sized workloads (up to 220 clients), the cluster cannot meet the energy and performance figures of the big server. Already 220 clients seem to overload the cluster, where runtimes and energy consumption for the same number of clients start to increase faster than for the big server. When stressing the cluster with more than 340 clients, WattDB crashes due to shortage of main memory.

Fig. 4b illustrates the results of the same experiments repeated using OLTP queries. Obviously, the big server is much better suited for OLTP than the cluster, as is exhibits lower query response times and also less energy consumption. Response times on the big server increase only slightly with the number of DB clients and the system does not show saturation at all. Consequently, energy consumption per query improves continuously. In contrast, the cluster is saturated with 160 clients; when further increasing the number of parallel queries, response times start to increase faster.

Analyzing the access patterns of both, OLTP and OLAP queries, the different performance figures are explainable: OLAP queries read huge amounts of records, join them with (small) fact tables, and then group and aggregate the results to satisfy analytical inquiries. Hence, the reading part of these queries can run in parallel on all partitions, speeding up the query linearly with the number of disks, CPUs, and/or nodes. After having fetched the qualified records, the joins with the fact tables can also run concurrently. The final grouping and aggregation steps can be pre-processed locally for each of the parallel streams and quickly aggregated into a final result. Hence, this kind of access pattern seems to well fit both, a single, multi-core machine with lots of disks and memory but also a cluster of independent nodes, exchanging query results via network. In [SH15], we have analyzed the abilities of a cluster—with an emphasis on dynamic repartitioning of DB data—to process that kind of workloads in greater detail.

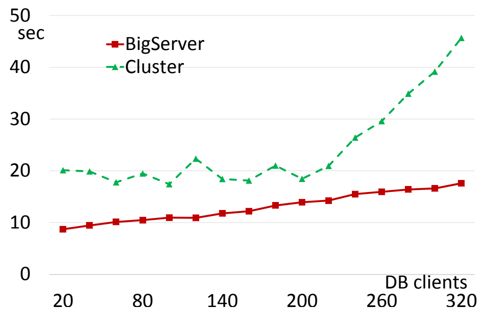
In contrast, OLTP queries touch very little data, but update records frequently. Since writers need to synchronize to avoid inconsistencies, lock information must be shared among all nodes involved. A single server with a main-memory-resident lock table can synchronize transactions much faster than a cluster, needing to exchange lock tables among nodes. Further, OLTP query operators modifying records cannot be offloaded to other nodes. Therefore, a query plan for transactional workloads is much more rigid than that of OLAP queries. In summary, it is comprehensible that a cluster of nodes is better suited for OLAP kinds of workloads than for typical transaction processing with ACID guarantees.

4.3 Energy-Centric Benchmark

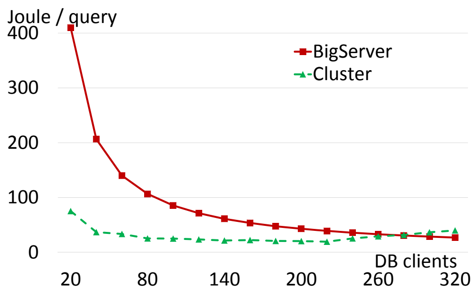
After evaluating the peak performance of both configurations, we ran experiments representing average, real-world workloads. Because DB servers are often heavily underutilized, as mentioned earlier, we modified the benchmark driver to submit queries *at timed intervals* [SHK12].

Workload scaling: In each experiment, we have spawned the number of OLTP or OLAP clients between 20 and 320, sending queries to the database. Each client sends a query in a specified interval. If the query is answered within the interval, the next query is not initiated immediately, but at the start of the subsequent interval. If the query is not finished within the interval, the client waits for the answer until sending the next query. In this way, each DB client generates its share of utilization. The DBMS has to answer queries quickly enough to satisfy the DB clients, but there is no reward for even faster query evaluation. It is important to delay query submission of the clients, since we are not interested in maximizing throughput, but want to adjust the DBMS to a given workload instead, using an optimal number of nodes.[9]

[9]Otherwise, the whole benchmark would degenerate to a simple performance-centric evaluation, which is not what we intended.



(a) Query response time



(b) Energy consumption per query

Figure 5: Performance and energy consumption for varying OLAP activity levels

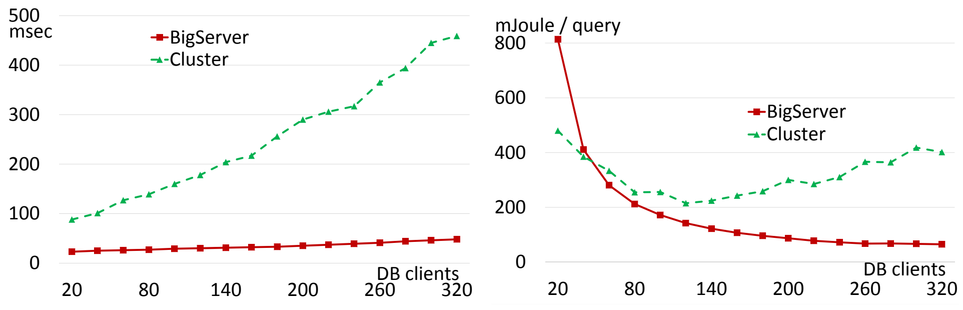
Before each workload changes, the cluster is *manually* reconfigured to best match the expected workload. We let the benchmarks run for a short warm-up time prior to measuring energy efficiency and performance to eliminate start-up cost and to identify maximum energy savings potential.

OLAP: In this experiment, we vary the number of parallel clients between 20 and 320. As before, we are using TPC-H queries on a SF300 dataset. To control utilization, the clients send queries at an interval of at least 20 seconds. Whatever comes last, the query result or the end of the interval, is the trigger for the next query.

Fig. 5 illustrates the results for the energy-centric OLAP benchmark. The left side depicts query response times of the brawny server and the wimpy cluster. As expected, the centralized machine handles queries faster than the cluster, even faster than the (pre-specified) target response time of 20 seconds per query. Therefore, the server is idle for longer time periods, still consuming energy. The cluster is meeting the target response times quite well, except for higher utilization, as observed earlier. After about 220 parallel clients, query performance starts to drop and runtimes build up.

Comparing energy consumption per query of both systems, the cluster delivers far better results for *average utilization*. Due to the cluster's scale-out and adaptation to the necessary number of nodes, its energy consumption per query stays at the same level almost the entire time, regardless of utilization. Only at high workloads, energy consumption increases because of lengthy query runtimes. The big server, with more or less static power consumption over the whole utilization spectrum, delivers bad energy efficiency for low and moderate workloads. Only at high utilization, when all the processing power of the server is needed, its energy consumption per query pushes below that of the cluster. From this experiment, we conclude that the cluster seems to be better fit for moderate OLAP workloads than the big server.

OLTP: We repeated the same experiment using OLTP queries from the TPC-C benchmark. Prior, the corresponding dataset was generated with a scale factor of 1,000. Identical to the OLAP benchmark, we scaled the DB clients between 20 and 320. Each client was waiting 40 ms between queries to simulate low and moderate workloads too.



(a) Query response time (b) Energy consumption per query

Figure 6: Performance and energy consumption for varying OLTP activity levels

Fig. 6 plots the results of the energy-centric OLTP benchmark run. Whereas the big server exhibits query response times between 30 and 50 milliseconds, the cluster performs with processing times between 50 and 450 ms much worse. Apparently, the cluster is not well suited to process update-intensive OLTP workloads. Likewise, energy efficiency of the cluster is only better at very low workloads, below 50 parallel DB clients. While the big server consumes between ~100 and ~800 mJoule/query, the cluster needs between ~200 and ~400 mJoule/query. The big server's energy efficiency is much better at more challenging workloads, unlike OLAP workloads, where the cluster was more efficient for most scenarios. In conclusion, we have constituted a tradeoff between performance and energy consumption on the cluster. By reducing the number of nodes, both power consumption and peak performance are lowered. For moderate workloads, lower performance is tolerable and, thus, energy efficiency can be improved.

4.4 Dynamic Workloads

As previously described, the limiting factor for dynamic repartitioning is the migration cost, i. e., the performance impact and time it takes to move data between nodes. To estimate its impact on the cluster's elasticity, we have run experiments on a *dynamically adapting* cluster. Similar to the previous tests, we are running a mix of workloads against the cluster, ranging from low utilization up to heavy workloads. In this experiment, the cluster is given no warm-up times to adjust itself to a given task; instead, we are monitoring performance and energy consumption continuously.

Workloads change every 5 minutes, starting with a moderate workload of 20 DB clients, sending OLTP or OLAP queries respectively. The number of DB clients sending queries in parallel over time, i. e., the workload pattern (used in the experiments of Sect. 4.4.1), is depicted in Fig. 7.

To quantify the importance of workload forecasting, we processed the same benchmark on the cluster twice—and once on the big server for comparison. The first run on the

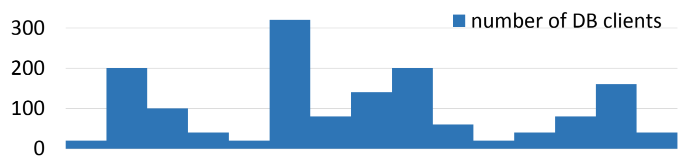


Figure 7: Dynamic workload over time

cluster hits the system unprepared; WattDB has to reactively adjust to the changing workload. During the second run, the DBMS was informed of upcoming workload changes (30 minutes in advance). Hence, the DBMS could proactively adjust to such a new load situation.

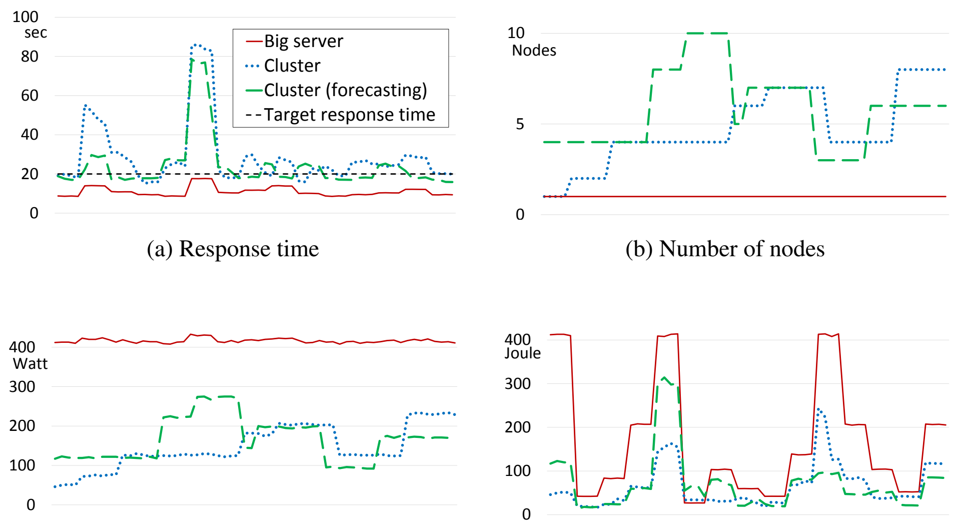
In [KHH12], we have developed a load forecasting framework to predict upcoming database workloads based on historic performance data. We observed that workloads are often repetitive and, therefore, quite easy to forecast.

4.4.1 OLAP and OLTP Processing

In the following, results of the benchmark runs are discussed separately for OLAP and OLTP. Results for the big server are visualized as *solid red* lines in Figures 8 and 9. *Dotted blue* lines represent benchmark results measured on the cluster without workload forecasting, while *dashed green* lines illustrate results of the same experiment using forecasting. The top-left plot in both figures draws the average query response time. The target response time (20 seconds for OLAP and 200 msec for OLTP drawn as *short-dashed black* lines) is included to expose the load-dependent response time deviations in the various experiments. A target response time is needed for dynamic adaption, to give the scaling algorithm a bounded optimization goal. To characterize the varying size of the cluster, the number of active nodes is depicted in the top-right chart. Underneath, the course of the overall power consumption is shown for all three experiments on the left. The resulting average energy consumption per query is plotted on the lower-right—to contrast it to the power consumption of the corresponding system.

OLAP (big server): Fig. 8 shows the results for TPC-H queries on the big server as solid red lines. The big server does not exhibit transition times between workload changes, since reconfiguration is not needed on this single-node system. Query runtimes are fast, always beating the target response time. Yet power consumption is constantly high, regardless of utilization, as already observed in earlier experiments. Average energy consumption per query is comparably high, although query runtimes are low. Because this benchmark is energy-centric, faster query runtimes do not lead to better results.

OLAP (cluster): The dotted blue lines in Fig. 8 depict the TPC-H results for the cluster without workload forecasting. The number of nodes in the cluster jitters heavily, as the system tries to adjust itself to the current workload. Reconfiguration takes time, e. g., migrating from 2 to 4 nodes requires each of the two source nodes to ship about 100 GB of data to one of the targets, hence, it takes up to 20 minutes to complete repartitioning. Therefore, query response times in this benchmark experiment are highly fluctuating and,



(a) Response time (b) Number of nodes

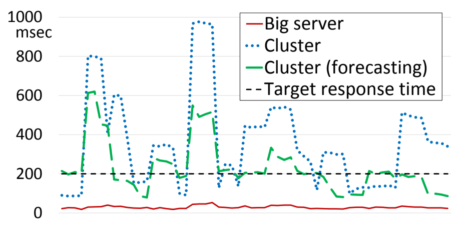
(c) Power consumption (d) Energy per query

Figure 8: Dynamic OLAP workload on the big server and the cluster

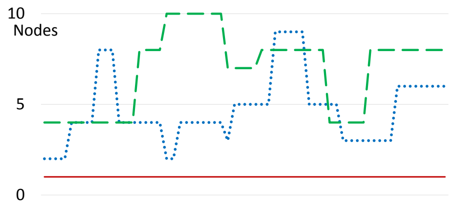
in turn, often missing the predefined deadline (target response time). Yet, as we have shown in previous experiments, the cluster, in theory, should be able to handle most of the workloads within the deadline. Due to high additional reconfiguration overhead, the nodes are overloaded. Therefore, query runtimes and avgerage energy per query remain high.

OLAP (cluster with forecasting): In this benchmark, we inform the cluster of workload changes occurring in the next 30 minutes. Hence, instead of only reacting to load changes, WattDB can now prepare for upcoming load. The dashed green lines in Fig. 8 illustrate the results. In comparison to the first run on the cluster, response times are generally lower and pass the deadline more often. Since the cluster prepares for heavy workloads in advance by scaling out to more nodes, the number of nodes is also larger in average, resulting in increased average power consumption. The energy consumption per query, however, shows a mixed picture. For low utilization, but more nodes running to prepare for upcoming events, energy consumption is higher compared to the version without forecasting. On the other hand, for higher utilization, thanks to in-advance preparations related to the forecasting approach, query runtimes are lower and exhibit better overall energy efficiency.

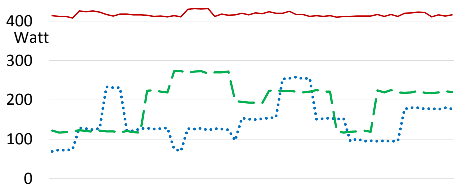
When comparing the big server with the cluster, we can conclude that the server is performing better, i.e., it exhibits lower query response times. On the other hand, the cluster is more energy efficient, especially during low and moderate utilization, due to its adaptation to the workload. The cluster benefits from scale-in, when performance is not needed. This translates to a steadily varying power consumption (according to the cluster size), whereas the server displays a more or less constant one. For OLAP workloads, the cluster seems an eligible alternative to a big server.



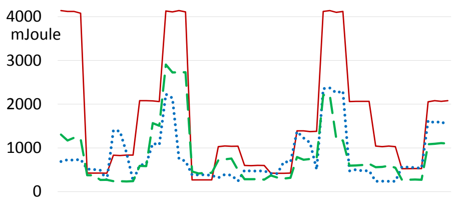
(a) Response time



(b) Number of nodes



(c) Power consumption



(d) Energy per query

Figure 9: Dynamic OLTP workload on the big server and the cluster

OLTP (big server): After running OLAP benchmarks, we have repeated the same dynamic workload with OLTP clients on the TPC-C dataset. Fig. 9 illustrates our results for experiments based on the energy-centric OLTP benchmark. As before, solid red lines characterize the behavior of the big server, which exhibits low query runtimes, but also high power consumption.

OLTP (cluster): The dotted blue lines of Fig. 9 summarize our results for the benchmark run on a cluster without load forecasting. Obviously, the response times shown are high. Because the cluster is forced to permanently repartition, response times and, in turn, energy efficiency are further worsened. Because the cluster can only react to the current workload, rebalancing starts after a sufficient change of the workload is detected. As discussed for the OLAP benchmark, this puts too much stress on the nodes and notably slows down query processing. Compared to the big server, query response times are much longer for high activity levels of the cluster. Yet, power and energy consumption are lower. Therefore, the cluster delivers better energy efficiency overall—if longer query response times are deemed acceptable.

OLTP (cluster with forecasting): The dashed green plots of Fig. 9 depict the OLTP benchmark results on the cluster using workload forecasting. Compared to the previous benchmark, the average number of nodes is higher, because WattDB is preparing for workloads in advance. As a results, query runtimes are more stable and more often pass the deadline. However, power consumption is often higher. Again, overall energy efficiency is characterized by a mixed picture: Due to preparations related to the forecasting approach, lower workloads have worse energy efficiency, but more intense workloads benefit by achieving lower energy consumption per query.

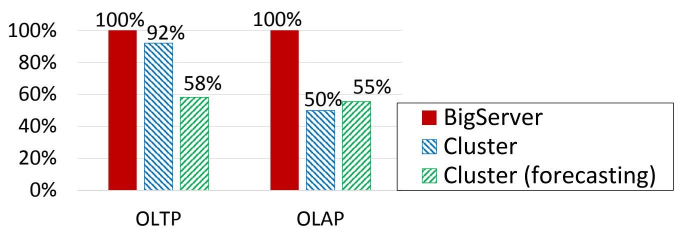


Figure 10: Energy Delay Product of OLAP & OLTP workloads

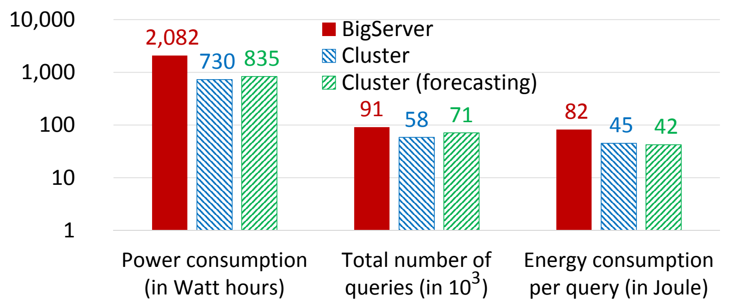
4.4.2 Energy Delay Product

The product of energy consumption (Joules) and runtime (seconds) is called *Energy Delay Product* (EDP), a metric originally stemming from chip and microprocessor design [GH96]. EDP is used as a metric combining energy consumption with performance of alternate architecture design choices. As rationale behind this metric, a system—running twice as fast and with twice the power consumption—exhibits the same EDP, hence, it is not better or worse than the original system. Another system, consuming the same amount of energy but delivering results faster will have a lower EDP. Likewise, lowering energy consumption while keeping runtimes unchanged results in a better EDP. Therefore, the Energy Delay Product is a good measure to compare the energy consumption/performance of heterogeneous systems, given the same task.

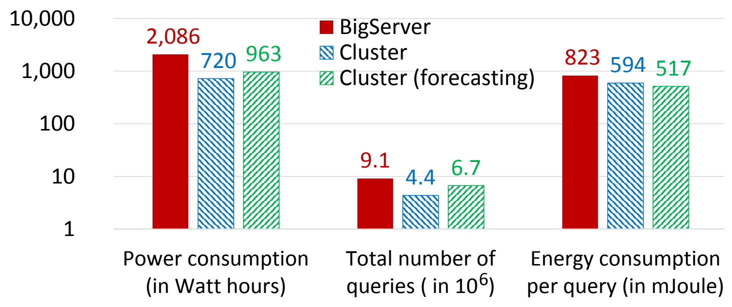
Despite different hardware platforms and power consumption figures, by calculating the EDP, we can make a square comparison of the big server and the cluster. In Fig. 10, we have summarized the EDP results from our dynamic experiments on OLTP and OLAP workloads for all three configurations. To illustrate the differences, all values are relative to the EDP of the big server. Hence, the forecasting cluster's EDP is about half of that of the big server, regardless of OLTP or OLAP workloads. For OLTP workloads, the cluster without forecasting exhibits nearly the same EDP as the big server, obviously, forecasting really payed off here. In contrast, the cluster without forecasting shows the lowest EDP under OLAP queries, even lower than the forecasting version. Apparently, preparing the cluster for upcoming workloads was a (little) waste of time and energy here. However, the cluster's EDP is lower than that of the big server; hence, with the same energy budget, we should be able to perform more work—although it might take some extra time.

4.4.3 Summary

Reviewing the results from all benchmarks, we want to extract some condensed numbers to facilitate high-level comparison and to gain a few key observations. For this reason, we have separately computed indicative numbers for the dynamic OLAP and OLTP experiments: total energy consumed (in Watt hours), total number of queries represented in units of 10^3 resp. 10^6, and average energy consumption per query in Joule resp. mJoule. These Key figures are shown in Fig. 11, where the logarithmic y-axis should be regarded.



(a) OLAP workload



(b) OLTP workload

Figure 11: Overall energy use, throughput, and avg. energy consumption per query

First, the cluster is no match for the big server considering pure performance. The centralized system does not require network communication or synchronization among nodes. Therefore, it can deliver much better query throughput than the cluster, where queries have to be distributed, results have to be collected, and the overall execution of concurrent queries on multiple nodes needs some form of synchronization to ensure ACID properties. All these factors lead to friction losses which slow down query processing.

Second, the cluster handles low and moderate workloads quite well, although the big server is still faster. Yet, the cluster requires less than half of the server's power (left-most bars in the figures). Therefore, the cluster needs less energy per query and is more energy efficient, as depicted by the right-most bars in the figures.

Third, dynamic workloads with varying utilization require preparation to adjust the number of nodes to the needs. If workloads are predictable, the cluster exhibits better energy efficiency than the single server while delivering comparable performance. Although, energy consumption of a forecasting cluster is higher, its query performance outweighs the additional wattage.

5 Conclusion

In this paper, we have examined the energy-saving potential of a clustered DBMS compared to a traditional DBMS based on a single server. An important goal of this paper was to compare performance and energy efficiency of our WattDB cluster to those of a big server. Of course, if peak DBMS performance is required during almost the entire operating time, a single-server approach has no alternative as our performance-centric benchmarks clearly reveal. However, as stated in various studies [BH09, SHK12], average utilization figures are far from continuous peak loads. A large share of database or data-intensive applications runs less than an hour close to peak utilization on workdays and is resilient w.r.t. somewhat slower response times. During the remaining time, their activity level is typically in the range of 20% – 50% and often lower. Therefore, from low- to mid-range workloads, a dynamically adjusting cluster of nodes will consume significantly less power without sacrificing too much performance. Hence, their throughput/response time requirements could be conveniently satisfied by the performance characteristics of our cluster with much less energy consumption, as confirmed by Fig. 8. Hence, the application range, where the cluster's energy efficiency largely dominates that of a single server, has quite some practical benefit.

Especially for OLAP workloads, where lots of records need to be read and aggregated without much coordination effort, a cluster seems to be a viable alternative to a single server. On the other hand, when processing OLTP workloads, where transactions need to synchronize continuously, a cluster suffers from high friction losses and is a magnitude slower than the centralized approach.

As shown, predictability of workloads and data elasticity are crucial for our approach. Fortunately, typical usage patterns are predictable and a cluster can therefore prepare for upcoming workloads. Thus, dynamically adjusting a cluster to the workload—although time-consuming—is possible.

References

[AFP$^+$09] Michael Armbrust, Armando Fox, David A. Patterson, Nick Lanham, Beth Trushkowsky, Jesse Trutna, and Haruki Oh. SCADS: Scale-Independent Storage for Social Computing Applications. *CoRR*, abs/0909.1775, 2009.

[BFG$^+$08] Matthias Brantner, Daniela Florescu, David Graf, Donald Kossmann, and Tim Kraska. Building a Database on S3. In *SIGMOD Conf.*, pages 251–264, 2008.

[BH07] Luiz André Barroso and Urs Hölzle. The Case for Energy-Proportional Computing. *IEEE Computer*, 40(12):33–37, 2007.

[BH09] Luiz Andre Barroso and Urs Hölzle. *The Datacenter as a Computer: An Introduction to the Design of Warehouse-Scale Machines*. Morgan & Claypool Publishers, 2009.

[BZN05] Peter A. Boncz, Marcin Zukowski, and Niels Nes. MonetDB/X100: Hyper-Pipelining Query Execution. In *CIDR Conf.*, pages 225–237, 2005.

[CDG$^+$08] Fay Chang, Jeffrey Dean, Sanjay Ghemawat, Wilson C. Hsieh, Deborah A. Wallach, Mike Burrows, Tushar Chandra, Andrew Fikes, and Robert E. Gruber. Bigtable: A Distributed Storage System for Structured Data. *ACM TOCS*, 26(2), 2008.

[CRS$^+$08] Brian F. Cooper, Raghu Ramakrishnan, Utkarsh Srivastava, Adam Silberstein, Philip Bohannon, Hans Jacobsen, Nick Puz, Daniel Weaver, and Ramana Yerneni. PNUTS: Yahoo!'s Hosted Data Serving Platform. *Proc. VLDB Endow.*, 1(2):1277–1288, 2008.

[Das11] Sudipto Das. *Scalable and Elastic Transactional Data Stores for Cloud Computing Platforms*. PhD thesis, Univ. of California at Santa Barbara, 2011.

[GH96] R. Gonzalez and M. Horowitz. Energy Dissipation in General-Purpose Microprocessors. *IEEE Journal of Solid-State Circuits*, 31(9):1277 –1284, 1996.

[Gra94] G. Graefe. Volcano—An Extensible and Parallel Query Evaluation System. *IEEE Trans. on Knowl. and Data Eng.*, 6(1):120–135, 1994.

[HHOS11] Theo Härder, Volker Hudlet, Yi Ou, and Daniel Schall. Energy Efficiency is not Enough, Energy Proportionality is Needed! In *DASFAA Workshops, LNCS 6637: 226-239*, 2011.

[HS11] Volker Hudlet and Daniel Schall. Measuring Energy Consumption of a Database Cluster. In *BTW, LNI 180*, pages 734–737, 2011.

[KHH12] Christopher Kramer, Volker Höfner, and Theo Härder. Load Forcasting for Energy-Efficient Distributed DBMSs (in German). In *Proc. 42. GI-Jahrestagung 2012, LNI 208*, pages 397–411, 2012.

[LFWZ09] David B. Lomet, Alan Fekete, Gerhard Weikum, and Michael J. Zwilling. Unbundling Transaction Services in the Cloud. *The Computing Research Repository*, abs/0909.1768, 2009.

[LHP$^+$12] Willis Lang, Stavros Harizopoulos, Jignesh M. Patel, Mehul A. Shah, and Dimitris Tsirogiannis. Towards Energy-Efficient Database Cluster Design. *PVLDB*, 5(11):1684–1695, 2012.

[SBH$^+$10] Alexander S. Szalay, Gordon C. Bell, H. Howie Huang, Andreas Terzis, and Alainna White. Low-Power Amdahl-Balanced Blades for Data-Intensive Computing. *SIGOPS Oper. Syst. Rev.*, 44(1):71–75, 2010.

[SH13a] Daniel Schall and Theo Härder. Towards an Energy-Proportional Storage System using a Cluster of Wimpy Nodes. In *BTW, LNI 214*, pages 311–325, 2013.

[SH13b] Daniel Schall and Theo Härder. Energy-Proportional Query Execution Using a Cluster of Wimpy Nodes. In *SIGMOD Workshops, DaMoN*, pages 1:1–1:6, 2013.

[SH14] Daniel Schall and Theo Härder. Approximating an Energy-Proportional DBMS by a Dynamic Cluster of Nodes. In *DASFAA Conf., LNCS 8421*, pages 297–311, 2014.

[SH15] Daniel Schall and Theo Härder. Dynamic Physiological Partitioning on a Shared-nothing Database Cluster. In *ICDE Conf.*, 2015.

[SHK12] Daniel Schall, Volker Höfner, and Manuel Kern. Towards an Enhanced Benchmark Advocating Energy-Efficient Systems. In *TPCTC Conf., LNCS 7144*, pages 31–45, 2012.

[THS10] Dimitris Tsirogiannis, Stavros Harizopoulos, and Mehul A. Shah. Analyzing the Energy Efficiency of a Database Server. In *SIGMOD Conf.*, pages 231–242, 2010.

[VCO10] Hoang Tam Vo, Chun Chen, and Beng Chin Ooi. Towards Elastic Transactional Cloud Storage with Range Query Support. In *VLDB Conf.*, number 1-2, pages 506–514, 2010.