Towards an Energy-Proportional Storage System using a Cluster of Wimpy Nodes

Daniel Schall, Theo Härder

Databases and Information Systems Group
University of Kaiserslautern, Germany
{schall,haerder}@cs.uni-kl.de

Abstract: Previous DB research clearly concluded that the most energy-efficient configuration of a single-server DBMS is typically the highest performing one. This observation is certainly true if we focus in isolation on specific applications where the DBMS can steadily run in the peak-performance range. Because we noticed that typical DBMS activity levels—or its average system utilization—are much lower and that the energy use of single-server systems is far from being *energy proportional*, we came up with the hypothesis that better energy efficiency may be achieved by a cluster of nodes whose size is dynamically adjusted to the current workload demand. We will show that energy proportionality of a storage system can be approximated using a cluster of nodes, built of commodity hardware. To simulate data-intensive workloads, synthetic benchmarks submit read/write requests against a distributed DBMS (WattDB) and, in turn, its HDD- and SSD-based storage system, where time and energy use are captured by specific monitoring and measurement devices. The cluster dynamically adjusts its configuration such that energy consumption and performance are tuned to fit the current workload. For each benchmark setting, an optimal number of nodes is processing the queries in the most energy-efficient way, which does not necessarily correspond to the best performing configuration. The chosen workload is rather simple and primarily serves the purpose to deliver a proof of existence that energy proportionality can be approximated for certain kinds of query processing and, especially, for storage systems.

1 Introduction

The need for more energy efficiency in all areas of IT gained interest in research recently and ideas to increase the energy efficiency of stand-alone servers were proposed. Due to their narrow power range [BH07], i.e., the power spectrum between idle and full utilization, the goal of increased energy efficiency cannot be reached using todays hardware. Besides reducing the energy consumption of servers, other ideas like improving the cooling infrastructure and reducing its power consumption help reducing the energy footprint of data centers, although they do not decrease the energy consumption at the server level.

The original problem—reducing the energy consumption of installed servers—leads to a demand for energy-proportional hardware. *Energy proportionality* describes the ability of a system to reduce its power consumption to the actual workload, i.e., a system, that is utilized only 10% of its peak performance, must not consume more than 10% of its peak

power consumption. For specific applications, this goal can be approached by hardware-intrinsic properties, e.g., CPUs automatically entering sleep states or hard disks spinning down when idle. So far, automatically scaling systems down when idle—thus preventing high idle power consumption of todays servers—is the main focus of energy proportionality. Unfortunately, current hardware is not energy proportional for data-intensive applications as studies have shown (see [BH09]).

Several components such as CPUs are able to quickly change into sleep states, requiring less energy, when idle. Other components, especially the two main energy consumers of a DBMS, main memory and storage, exhibit bad energy characteristics. DRAM chips consume a constant amount of power—regardless of their use—and it is not possible to turn off unused memory chips in order to reduce energy consumption. Spinning down hard disks when idle conflicts with long transition times and results in slow query evaluation. For these reasons, such mechanisms are not very useful in case of DB applications where reference locality of data in large main-memory-resident DB buffers has to be preserved and low-latency accessibility of storage devices has to be guaranteed.

Todays servers are not energy efficient and approaches focusing on single machines cannot achieve energy proportional behavior either. This conclusion shifted the research focus from single-node approaches to clusters of servers, which appear more promising. Tsirogiannis et al. [THS10] observed in an extensive study based on empirical DBMS measurements that "*within a single node intended for use in scale-out (shared-nothing) architectures, the most energy-efficient configuration is typically the highest performing one*". In an independent study based on DBMS buffer management [OHS10], we came up at the same time with a similar conclusion concerning performance and energy efficiency of database systems and storage managers. Therefore, we want to improve energy efficiency of a DBMS by enabling the software side to explicitly power up/down resources as needed. Many clustered database systems exist, yet none has the ability to flexibly *scale up and down* in order to save energy.

We have shown in [SHK12] that real-world workloads usually do not stress database systems 24/7 with peak loads. Instead, the workloads alternate in patterns between high and near-idle utilization. But, database systems have to be tailored to the peak performance to satisfy incoming queries and potential users, i. e., customers. Therefore, database servers usually come with big DRAM buffers and a number of disks as external storage—both components that consume a lot of energy. The majority of these resources is only needed in rare time intervals to handle peak workloads. All other times, they lie waste, thereby substantially decreasing the overall energy efficiency of the server. During times of underutilization, overprovisioned components are not needed to satisfy the workload. By adjusting the database systems to the current workload's needs, i.e., making the system energy proportional, energy consumption could be lowered while still allowing the maximum possible performance.

This paper is structured as follows: Section 2 sketches WattDB and explains the power management algorithm. In Section 3, we describe the benchmarking model we used in this paper and explain our measurement setup for performance and energy consumption, before we discuss the results of our empirical benchmark runs in Section 4. Finally, we conclude and give an outlook in Section 5.

2 The WattDB Approach

For the reasons outlined above, we raised the hypothesis whether or not overall energy-efficiency optimization or energy-proportional system behavior could be better approached by a cluster of DB servers and redirected our work towards this research goal [GHP+10]. Hence, we want to achieve optimal DBMS energy efficiency—independent of its level of activity. For this purpose, we use a cluster of wimpy (lightweight) nodes, which can be powered on and off individually, allowing the cluster to scale. By dynamically adjusting the number of nodes in the cluster, the overall performance and energy consumption can be tailored to the current workload.

Our research project [SH11] focused on approaching energy-proportional runtime behavior for database management. So far, a commercially available DBMS does not exist, which can dynamically support powering up and down the nodes of a server cluster. Therefore, we have decided to build *WattDB* as a research prototype from scratch. The system is still under development and does not yet provide a full-fledged query execution engine. By providing limited querying capabilities and a primary-key index for tables, WattDB keeps queries close to the data. For this reason, it is not necessary to ship data pages to remote locations, which would burden the rather high network latency with each page read/write request.

2.1 Cluster Hardware

The cluster hardware consists of identical nodes, interconnected by a Gigabit-Ethernet switch. Each node is equipped with 2 GB of DRAM, an Intel Atom CPU D510 and (optionally) two storage disks. The hardware components are chosen to balance processing power and I/O bandwidth, making the nodes Amdahl-balanced [SBH+10]. All components running at 100% utilization result in a peak power consumption of about 30 Watts, hence they are ”wimpy”, compared to typical DB servers. This is the 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, typically used in servers, consumes ∼10 Watts per DIMM. Likewise, our HDDs need less power than high-performance drives, which makes them more energy efficient.

By choosing commodity hardware with limited data bandwidth, Ethernet wiring is sufficient for interconnecting the nodes. Currently, we have up to ten nodes running in the cluster. The nodes are connected to the same ethernet segment and can all communicate with each other. The total power consumption of the cluster can be broken down to roughly 20 Watts for the backplane switch, connecting the nodes; another 23 Watts for each active node, and 2.5 Watts for standby nodes. Fully utilized nodes (disks and CPU) consume about 30 Watts, mostly accounted to the CPU, as the power consumption of the disk drives is more or less steady. Replacing the magnetic storage disks with SSDs does not affect the power consumption of the storage nodes.

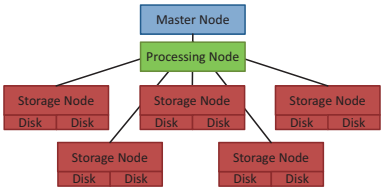


Figure 1: Overview of the cluster

2.2 Software

WattDB is a single-master DBMS, where one dedicated node accepts client connections, distributes incoming queries and keeps metadata for all cluster nodes. This node is also responsible for controlling the power consumption of the cluster by turning nodes on and off. The master is not processing queries, but distributing query plans and collecting results. This decision was made to prevent interferences between query processing and maintenance tasks on the master. Figure 1 sketches an exemplary cluster of seven nodes with the master on top. The remaining nodes can be classified as either storage nodes or processing nodes, whether they have disks attached or not. *Storage nodes* provide storage space to the cluster and act as page servers for other nodes. *Processing nodes* are executing queries by requesting pages from the storage devices, evaluating the content and writing them back. These nodes also hold the DB buffer to mitigate latency and limited bandwidth to the storage nodes. As mentioned, the query capabilities are still limited. The minimal configuration of the cluster requires the master node, at least one processing node, and also one storage node. It would be possible to allocate all three functions on a single physical node, but we decided to keep them separate for easier debugging and analysis.

2.3 Database Functionality

WattDB is based on a hybrid storage architecture. At the hardware level, the database can be considered as a shared-disk system; each processing node can access all storage disks remotely by connecting to the corresponding node and requesting pages. Characterized by its processing behavior, the shared-disk architecture is restricted to a logical shared-nothing DBMS. Each processing node has limited access to the storage layer and may only work on pages whose accessibility is previously defined by the master node. This restriction is enforced by the database software. By combining both approaches, *shared disk and shared nothing*, WattDB gains the flexibility to re-assign pages to processing nodes while avoiding synchronization cost that would come with a plain shared-disk architecture.

Database tables can span multiple processing nodes, each responsible for one of the table's partitions. Data is physically partitioned to the storage devices present, logical partitioning is currently not supported. Therefore, queries have to be evaluated on all processing nodes having partitions of the table in question. Re-distributing data blocks on storage devices is always possible and does not require any logical partitioning scheme, because it oper-

ates on physical data blocks. The mapping from logical to physical pages is done in the processing nodes, redistributing storage blocks only requires an update of the mapping.

2.4 Power Management

To approach energy-proportional processing behavior, WattDB is intended to dynamically scale the number of active nodes based on the current workload.

The master node is responsible for managing the cluster, switching nodes on and off, and redistributing the storage load. Each node monitors its disk and CPU usage and reports the readings periodically to the master to allow informed decisions based on the actual utilization of each node. The master is using the monitoring results for cluster orchestration—in particular, to estimate the overall cluster performance and to react to changing utilization.

Based on the monitoring data, the master node is running a kind of scheduling algorithm to adjust the number of nodes. This algorithm runs every minute and takes the past five minutes into account for calculating the IOPS. Listing 1 sketches this process in pseudo-code. First, all active storage devices in the cluster are examined and the current IOPS are compared to the threshold of max. allowed IOPS for this device (line 7 of the listing).[1]

Algorithm 1 Power-management pseudo code

```
1  ForEach(Storage storage in Cluster.Storages) {
2
3     If(!storage.PoweredOn) {
4        continue;
5     }
6
7     If(storage.IOPS > MAX_IOPS_PER_DISK) {
8        // Storage overloaded, acquire new storage and distribute data
9
10       Storage storageNew = PowerUpAnotherStorage();
11       Storage storageOld = GetStorageWithHighestLoad();
12
13       distributeBlocks(storageOld, storageNew);
14    }
15
16    If(storage.IOPS < MIN_IOPS_PER_DISK) {
17       // Storage underutilized, consolidate data to other active storages
18
19       consolidateStorage(storage);
20       storage.Suspend();
21    }
22 }
23 // Suspend unused nodes
24 ForEach(Node node in Cluster.Nodes) {
25    If(node.ActiveStorages == 0 &&
26       node.Partitions == 0 &&
27       !node.IsMaster) {
28       node.Suspend();
29    }
30 }
```

[1] The threshold is set to 90% of the peak IOPS for the drive, which was determined beforehand.

If the current utilization of a device exceeds the threshold, it is considered overloaded and the data is distributed to other storages devices. Not depicted in this listing is the selection of the distribution targets (line 13): The algorithm tries to move blocks to active, non-overloaded storage devices attached to the same node first, to minimize network traffic. In case these storage devices are already utilized too much, additional disks on the same node are powered up and used as a target, if possible. Lastly, when all the storage devices identified above are not sufficient to handle the load, other nodes are taken into account as well, and data blocks are shipped over the network to re-distribute the load. In case no other eligible nodes are found, additional storage nodes have to be powered up first.

After analyzing the overutilized storage disks and distributing their load, the algorithm now examines underutilized storage devices and tries to consolidate data blocks to other storage devices (line 16). This step performs the opposite work as sketched above and aims to reduce the number of storage disks, while still maintaining sufficiently high IOPS. Consolidating storage disks (line 19) follows a similar logic as before: First, disks on the same node are selected as target devices, if they are not overloaded already. Second, remote locations are involved and blocks have to be sent via the network. In both cases, *all* blocks are moved to other locations in order to shutdown the originating disk. After redistributing the disk load, the algorithm takes a final step and suspends all nodes, which do currently not serve a purpose (line 24–30).

3 Benchmarking for Energy Efficiency

So far, we have sketched the essential components of WattDB responsible for approaching our overall goal. However, testing a system for energy efficiency also requires a new benchmarking paradigm. Established procedures to measure DBMS performance cannot be used to get meaningful estimates of the system's power consumption (see below).

3.1 Benchmark Procedure

As we have shown in [SHK12], typical server workloads strongly vary in their utilization. In this study, we have examined the servers of an ERP service provider and monitored their workloads. We observed that servers are typically not fully utilized all the time. Instead, the servers are usually loaded between 20 and 30% of their peak performance; but these servers are not overdimensioned for their workloads: During some (short) period of the day, their full capacities are needed to satisfy the performance demands, either for processing the incoming OLTP workload or for generating OLAP reports. This observation was made by other studies as well. Barroso and Hölzle [BH07] have shown that a cluster of nodes is more than half of the time below 50% load. Yet, its energy consumption during underutilization is comparable to its peak.

Current database benchmarks (primarily TPC-*) lately incorporated power measurements to deliver additional runtime characteristics for their evaluation. TPC-Energy was defined as such an addition to the performance-centric measurements. Nevertheless, the power

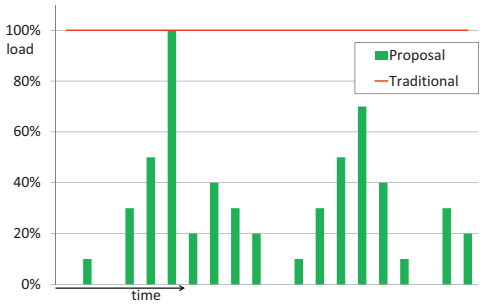


Figure 2: Traditional benchmarking compared to our proposal

consumption is only reported for idle and full utilization, leaving the typical DB server usage disregarded.

We will employ the benchmarking paradigm we proposed earlier [SHK12], which addresses typical usage patterns of databases and tries to mimic them in a benchmark run. Figure 2 depicts an exemplary benchmark run compared to traditional benchmarks. While traditional benchmarks only stress the system under test at 100%, the proposed benchmark will burden the system at different utilization levels to simulate such typical usage patterns.

Each benchmark step is scheduled to run for 30 minutes. If the DB cluster is able to finish the benchmark earlier, due to overprovisioned resources, the energy consumption of the time remaining is still recorded. In case the cluster cannot fulfill all queries in time, the total additional processing time and energy consumption is measured. These constraints are introduced in accordance to the benchmark definition in [SHK12].

3.2 Simulating OLTP Queries

Complex query capabilities are not yet implemented in WattDB, therefore, OLTP queries cannot be used to benchmark the database. Instead, the benchmark consists of a set of threads, executing an OLTP simulation. Each thread is representing one database client, running a series of OLTP queries.

Each query consists of a series of page reads, (artificial) processing steps and writes to simulate an OLTP query trace and to generate load at the storage layer; the processing nodes are not utilized much. Hence, this benchmark is heavily IO-intensive to empirically evaluate especially the storage layer and its energy-efficiency potential.

The benchmark operates on a 128 GB database with a primary-key index stored as a B*-tree. The database is preallocated on a single storage node which also contains the index. To circumvent the OS file system buffers and minimize the management overhead, no file system is used; instead, WattDB operates on raw disk devices. The database pages contain multiple records (which currently consist of an ID column and additional columns, filled with *junk* data to increase size). The inner leaves of the index fit into 2 GB of main memory, hence, after warming up, the buffer should contain a large fraction of the index.

The (simulated) OLTP clients randomly select IDs for reading records. For each request, the DBMS traverses the primary-key index to fetch the respective leaf page and locates the requested record inside the page. To emulate data processing, the threads generate CPU load by spin-locking. Finally, with a 1:4 chance, the page gets marked dirty in the buffer and, hence, has to be written back to the storage node at some time.[2] Afterwards, the benchmark thread goes to sleep for a specified time interval, before commencing the next read-process-write cycle. Such breaks are necessary when running an energy benchmark—as opposed to a performance benchmark. The system-under-test utilization can be tuned by adjusting the number of concurrently running clients, i. e., threads.

3.3 Measuring Energy Consumption

To measure the energy consumption of the cluster during the benchmark runs, we developed and installed a measurement framework. The basic power measurements are done using a custom measurement hardware. This device is capable of keeping track of each node in the cluster and the additional peripherals (i. e., the network switch). It can read the power consumption at a frequency of 300 Hz and report the values digitally over a standard USB connection. A software component is reading the measurements and aggregates all values to the cluster's total power consumption. This aggregate is then reported along with the benchmark runtime to a log file. Furthermore, the framework consists of a hardware device, capable of monitoring the energy consumption of up to ten nodes and infrastructure devices like ethernet switches as well, and a software component which aggregates and logs the measured data. Our framework can be integrated into the benchmark component and allows combining performance measurements with energy data. Although it would be possible to monitor the energy consumption of each node separately, we aggregated all energy measures to show the consumption of the cluster for the sake of clarity (and not their differing distribution in ten separate plots). A detailed description of the hardware device can be found in [HS11].

3.4 Experimental Setup

Figure 3 depicts the experimental setup. The database cluster, consisting of ten nodes and the ethernet switch[3], is powered by the measurement device, allowing us to monitor the power consumption of each server. The measured (analog) values (AC) are converted to digital readings and streamed to a connected computer executing the benchmarks based on a configuration file (Cfg) defining the specific usage patterns and sends queries to WattDB. By combining information from the benchmark runs (i. e., start and stop signals, duration) and the power measurements, the energy consumption for each benchmark run can be exactly determined. All data is written into a log file for later evaluation.

[2] The buffer decides which pages and when to write back.

[3] A single server would not need a dedicated ethernet switch, therefore we think it is fair to include it as a required hardware component into the measurements.

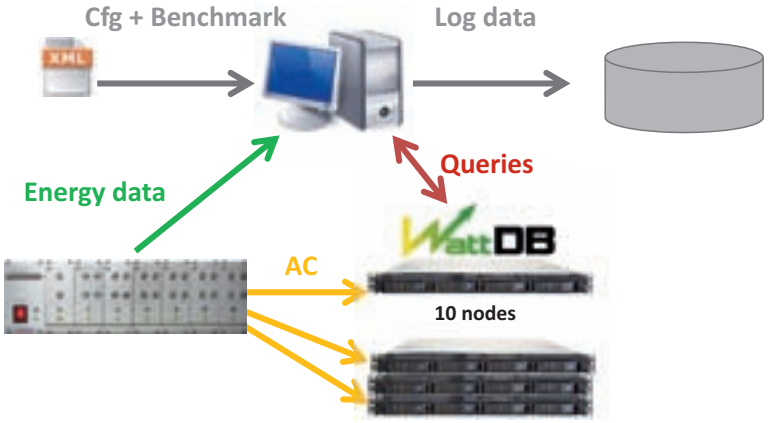


Figure 3: Experimental measurement environment

4 Measurement Results

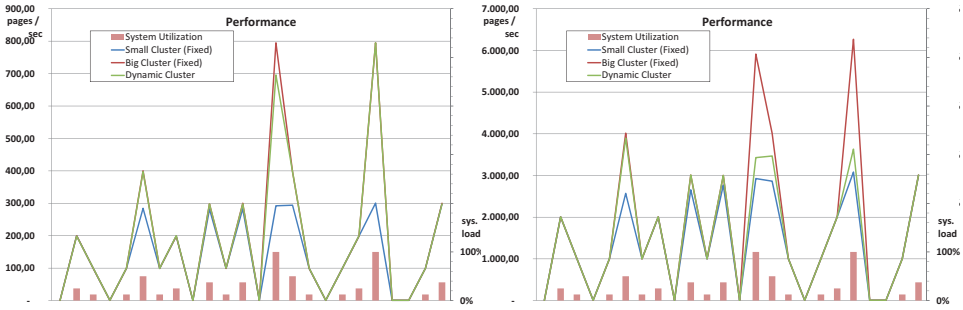
In this section, we explain the benchmarking methodology we used to verify our claims on a cluster of wimpy nodes, as previously described, and show the results of the runs. We have deployed the WattDB software with an energy management component in the cluster, connected to an energy measurement device. By running the benchmark against a cluster configuration, we expect the software to react to the changing workloads and power up/down nodes as needed. To make results comparable, we have run the identical load profile (as explained by Figure 2) three times.

For the first cluster configuration, we distributed the DB pages to two storage disks on one node and disabled the power management algorithm. Therefore, the number of nodes was fixed to the bare minimum of 3 (master, processing node and storage node) and the cluster was fixed to its most power-saving configuration delivering the lowest performance.

As next cluster configuration, we distributed the DB pages to all available disks and started the same benchmark, again with disabled power management. This time, all nodes (the master, one processing node and five storage nodes) were active and the cluster was able to work with maximum performance and, as a consequence, maximum power consumption.

The results of these two cluster configurations were used as baselines to estimate the performance and power coverage the cluster can achieve. Finally, we set up an unrestricted cluster configuration, with the power management component in full control of the cluster and its current workloads. We expected the cluster to adapt to the current workloads, as the benchmark runs proceeded.

During each of the runs, we measured the runtime and the energy consumption of the query phase from the start of the first query until the last query finished. Initially, the benchmarks were run on a cluster of seven nodes with 10 magnetic disks attached. Five nodes were acting as storage nodes, each having two disks attached, one was used as processing node and the remaining one was the coordinating master node. Later, we replaced the magnetic



(a) HDD **(b)** SSD

Figure 4: Performance: benchmark results for three cluster configurations

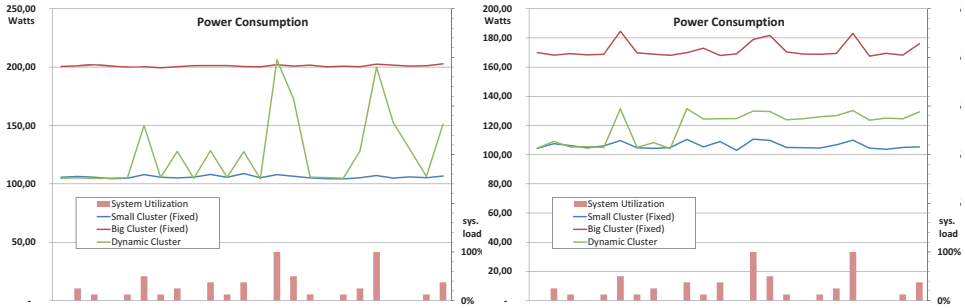
disks with 8 *Solid State Disks* (SSDs), thus reducing the number of storage nodes to four, and ran the benchmarks again. Since SSDs provide much more IOPS than traditional disks, we have increased the workload for the SSD benchmark by the factor of 10.

While running the benchmark against the three cluster configurations, energy consumption and runtime were reported to file. At the bottom of the result graphs, load profile or system utilization is depicted on the secondary axis. This is similar as in Figure 2 and only included for reference. Each graph plots the results of three cluster configurations: *Small Cluster* is showing results for the first run, where only two storage disks were active, thus forming the smallest possible configuration, *Big Cluster* is referencing to the second configuration with 10 (8) storage disks, and *Dynamic Cluster* depicts the measurements for an unrestricted run with the power management component active. In addition to the storage nodes, the master node and a single processing node were used in all three configurations.

On the left of each figure, the results of the configurations using magnetic disks are shown. The graphs on the right depict the results using SSDs. In all graphs, *sys load* refers to the utilization of the storage system, where 100% represent the maximum throughput the system could achieve (using all storage nodes and disks).

4.1 Performance

Figure 4 shows the performance graph for each run. As expected, the big cluster delivered the best performance and was even able to handle the highest utilization. The small cluster's performance broke down, due to it's constrained number of storage disks and the limited maximum IOPS. Finally, the dynamic cluster showed more or less identical performance to the big one, with small limitations in a case where dynamic adaptation caused some blocks to be moved between storage devices, which decreased the maximum performance. (It took only a few minutes to redistribute several Gigabyte of storage via Gigabit-Ethernet. By using compression, we were able to further reduce network traffic.) Compared to the total runtime of at least 30 minutes, redistribution cost was acceptable.



(a) HDD **(b)** SSD

Figure 5: Power consumption: benchmark results for the three cluster configurations[5]

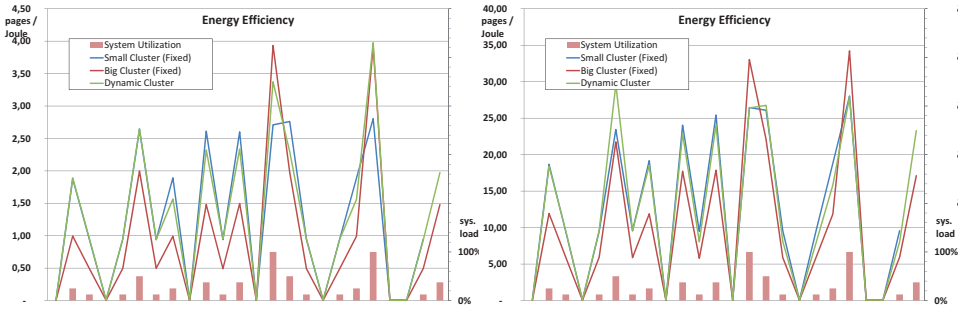
The performance using SSDs was much better, compared to magnetic disks. Still, the relative results are comparable, except for the dynamic configuration, which did not deliver the same peak performance as the pre-configured big cluster. When stressing the SSD cluster with heavy load, the performance of the cluster did not increase as expected. This might indicate optimization potential in the power management component or a bottleneck which was not monitored, e. g., CPU or network.

4.2 Power Consumption

Figure 5 visualizes the power consumption during the benchmark runs. Both fixed configurations exhibit a mainly static power consumption, because the number of nodes was fixed. The big cluster delivers no measurable difference for the HDD configuration between idle and full utilization.[4] Compared to idle, the SSDs exhibit a slightly increased power consumption under load. The dynamic configuration oscillates between the lowest and highest power consumption, as the cluster adapts to the workload. Using HDDs, the power management decided for our benchmark to use all available storage devices to share the load. For SSD configurations, however, not all storage devices were used, possibly because the storage was not over-utilized and some other component of the cluster was the bottleneck. Our power management decided to distribute the load to two storage disks on two separate nodes, instead of two disks on the same node. This is another indicator that the network was the limiting factor in the benchmarks, and not the IOPS of the SSDs.

[4]CPU-bound benchmarks might reveal different power characteristics.

[5]The use of enterprise server hardware would enlarge the relative distance between the curves of the small and big cluster. As a consequence, the overall saving of the dynamic cluster would have been amplified.



(a) HDD **(b)** SSD

Figure 6: Variation of energy efficiency for the three cluster configurations

4.3 Energy Efficiency

If we relate performance to energy consumption (which is power consumption times benchmark duration), we can calculate the energy efficiency for each of the runs, which is shown in Figure 6. Energy efficiency is expressed in pages per Joule, i. e., how many pages can be processed by consuming one Joule of energy. Not surprisingly, the small cluster exhibits the best energy efficiency during low utilizations. The big cluster is simply overprovisioned to satisfy the workload and consumes more energy to process the same amount of work, hence, its energy efficiency is worse.

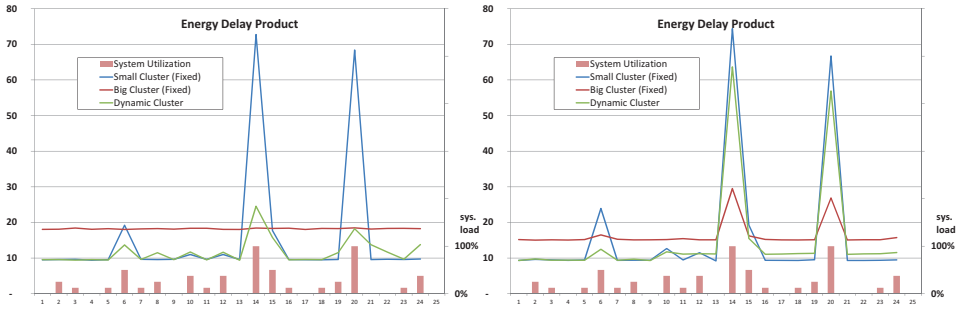
At full utilization, the situation turns in favor of the big cluster. The small cluster is not suited to handle the high utilization and needs almost 3 times as long as the big cluster to process the workload (not depicted here). As a consequence, the energy consumption of the small cluster is much higher and the energy efficiency accordingly lower.

The dynamic cluster powers storage nodes up and down according to the current workload. Therefore, under low utilization, its energy efficiency is identical to the small cluster. With rising load, the dynamic cluster powers up additional storage devices; hence, its energy efficiency gets comparable to that of the big cluster. Again, transition costs to move storage blocks decrease the energy efficiency in the dynamic case.

Using SSDs, energy efficiency is roughly 10 times better, although the difference between the small and the big cluster is not as prominent as with magnetic disks. The small cluster is still the most energy efficient one at low utilization and the big one only pays off at full load, but the reduced performance of the dynamic cluster affects its energy efficiency.

4.4 Energy Delay Product

When calculating energy efficiency, a system that is twice as slow, but consumes only half the power, will get the same score, because the same amount of work can be done using the same amount of energy. This calculation disregards the user expectation, as the



(a) HDD **(b)** SSD

Figure 7: Illustration of the EDP for the three cluster configurations

user/consumer is interested in getting results quickly. Therefore, we added another metric, which takes power consumption and query delay (i. e., the inverse of the performance) into account; the *Energy Delay Product* (EDP), a metric stemming from chip and microprocessor design [GH96]. EDP is defined as follows:

$$EDP = \text{ execution time} \times \text{ energy consumption } \textit{or}$$
$$EDP = \text{ execution time}^2 \times \text{ power consumption}$$

Hence, faster query execution times get rewarded more than power consumption, i. e., a system with twice the execution time must consume less than 1/4th of the power to get the same EDP rating. A lower EDP is generally favorable.

Figure 7 shows the EDP for the three benchmark runs. The small cluster exhibits the lowest EDP under low utilization, but does not perform well under heavy load. Starting at 50% utilization, the EDP of the small cluster outgrows the big cluster's EDP, because the load is too high for the small cluster and the execution time nearly triples. The big cluster shows a stable EDP, regardless of the workload. In most cases, the cluster is underutilized and, thus, more energy is consumed and the EDP is higher, compared to the small cluster. Only in peak-load situations, the additional performance of the big cluster pays off. The dynamic cluster shows the best overall EDP, with a similar score to the small cluster when not fully utilized and a slightly higher EDP in the peak-performance benchmarks.

Running the benchmark on SSDs, even the big cluster seems to have trouble handling the heavy workloads, hence, the rising EDP. The dynamic cluster is also unable to adjust the configuration to score a low EDP. As previously mentioned, this indicates a bottleneck beyond the reach of the current monitoring.

In summary, the measurements clearly illustrate that no fixed cluster configuration, neither a small one, nor a big one, is able to process the given workload in the most energy-efficient way. Hence, the results of the dynamic cluster can be considered as a proof of existence that, in specific cases, energy proportionality can be approximated for DBMS processing and that the increased effort pays off in terms of energy saving—without sacrificing too much performance.

5 Related Work, Conclusion & Outlook

As highlighted, a key result for energy efficiency of single-server DBMSs was published by Tsirogiannis et al. [THS10]. An exploration of a clustered DB system—close to our approach—was reported by Lang et al. [LHP+12]. Their contribution identified ways to improve energy efficiency in database applications by using a *static server cluster* instead of a single-server DBMS. Using a COTS (commercial off-the-shelf) parallel DBMS, they ran a single TCP-H query as workload and repeated their experiments on clusters where the cluster size/server configurations were set up for each test run. First, they reduced the number of beefy (powerful) nodes step-by-step from 16 to 8 nodes. Then, they gradually replaced the beefy nodes by wimpy (lightweight) ones. Both experiments resulted in decreased energy consumption, because the reduced cluster size or the lightweight nodes replacing beefy nodes needed less power, and in reduced performance. By analyzing the results, they revealed opportunities leading to improved energy efficiency. Although the experiments in [LHP+12] used only static server configurations and did not explain how the unused servers in the cluster could be turned on/off, they also delivered at least a kind of *existence proof* that research in DB clusters may lead to enhanced energy efficiency.

Our results are a step forward towards dynamically achieving energy proportionality. We have shown that energy proportionality can be approximated by using a cluster of commodity hardware and that tuning the system to a certain performance level comes with the drawbacks of limiting either the maximum processing power or the possible energy savings. By automatically adjusting the number of nodes to a running workload, which results in a balanced trade-off between performance and energy consumption, we demonstrated that it is possible to configure a cluster to process data in the most energy-efficient way. By dynamically reconfiguring the storage to fit the workload, a cluster of nodes reveals substantial energy-saving potential compared to big servers and also compared to statically configured clusters. The workload explored was rather simple and served as a starting point to reveal in our future work the data clusters and workload patterns requiring only limited reorganization/reallocation where such a storage server—while preserving its intended energy-proportional behavior—can be used. Nevertheless, our findings represent a milestone towards an energy-proportional DBMS, because the storage in traditional database systems accounts for more than half of the power consumption [PN08].

By using SSDs, IO is not the only limiting factor in the cluster. In the future, we will include CPU, main memory, and network into the monitoring scope as well to allow dynamic adjustments of all DBMS-relevant components. While scaling at the storage side was rather easy—it only required to copy/move data from one disk to another, while keeping the logical pointers up-to-date—, scaling at the processing side is more complex.

The experiments show that re-distribution of storage blocks and cluster balancing cannot be done frequently, due to the cost of shipping storage blocks via the network. By including historical measurements and forecast data, we are planning to extend the reactive power management component to become proactive [KHH12]. Workloads usually follow an easy-to-predict pattern, e.g., workdays are similar to each other, workloads in December keep rising for e-commerce back-end DBMSs, and so on. Therefore, we expect even better energy savings with a proactive cluster.

The granularity of control over performance and energy is a single server. With n nodes in the cluster, the finest grain of adaptation is $1/n$-th of the total power. In this work, we have run the experiments on a cluster of seven nodes (5 storage nodes). This configuration implies that the finest grain of control is 1/5th of the total, or 20%. For more fine-grained control, the number of nodes should be increased. Lastly, there is still optimization potential for power management, and many other factors besides forecast data can be included in the decision process such as CPU/network/memory utilization, information about the workloads, repeating patterns, etc. More specific data can lead to more intelligent use of resources and even better energy efficiency than we have shown in this paper.

References

[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.

[GH96] R. Gonzalez and M. Horowitz. Energy dissipation in general purpose microprocessors. *IEEE Journal of Solid-State Circuits*, 31(9):1277 –1284, 1996.

[GHP+10] Clement Genzmer, Volker Hudlet, Hyunjung Park, Daniel Schall, and Pierre Senellart. The SIGMOD 2010 Programming Contest - A Distributed Query Engine. *SIGMOD Record*, 39(2):61–64, 2010.

[HS11] Volker Hudlet and Daniel Schall. Measuring Energy Consumption of a Database Cluster. In *Proc. 14-th GI-Conf. on Database Systems for Business, Technology and Web, LNI - P 180*, pages 734–737, 2011.

[KHH12] Christopher Kramer, Volker Höfner, and Theo Härder. Lastprognose für energieeffiziente verteilte DBMS. *Proc. 42. GI-Jahrestagung 2012, LNI 208*, pages 397–411, 2012.

[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.

[OHS10] Yi Ou, Theo Härder, and Daniel Schall. Performance and Power Evaluation of Flash-Aware Buffer Algorithms. In *DEXA, LNCS 6261*, pages 183–197, 2010.

[PN08] Meikel Poess and Raghunath Othayoth Nambiar. Energy Cost, The Key Challenge of Today's Data Centers: A Power Consumption Analysis of TPC-C Results. *PVLDB*, 1(2):1229–1240, 2008.

[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.

[SH11] Daniel Schall and Volker Hudlet. WattDB: An Energy-Proportional Cluster of Wimpy Nodes. In *SIGMOD Conference*, pages 1229–1232, 2011.

[SHK12] Daniel Schall, Volker Höfner, and Manuel Kern. Towards an Enhanced Benchmark Advocating Energy-Efficient Systems. In *TPCTC, 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 Conference*, pages 231–242, 2010.