Energy-proportional Query Execution 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

Because energy use of single-server systems is far from being *energy proportional*, we explore whether or not better energy efficiency may be achieved by a cluster of nodes whose size is dynamically adjusted to the current workload demand. As data-intensive workloads, we submit specific TPC-H queries against a distributed shared-nothing DBMS, where time and energy use are captured by specific monitoring and measurement devices. We configure various static clusters of varying sizes and show their influence on energy efficiency and performance. Further, using an *EnergyController* and a load-aware scheduler, we verify the hypothesis that energy proportionality can be well approximated by dynamic clusters.

1. INTRODUCTION

Energy efficiency is becoming more important in database design. A substantial number of scientific contributions examined and optimized the energy consumption of database servers and their components. Recently, the research focus shifted from inflexible single-server DBMSs to distributed clusters running on lightweight nodes [4]. Although distributed systems impose some performance degradation compared to a single, brawny server, they offer higher energy saving potential in turn.

Current hardware is not energy proportional, because a single server consumes, even when idle, a substantial fraction of its peak power [1]. Because typical usage patterns lead to a server utilization far less than its maximum, energy efficiency of a server aside from peak performance is reduced [6]. In order to achieve energy proportionality using commodity hardware, we have chosen a clustered approach, where each node can be powered independently. By turning on/off whole nodes, the overall performance and energy consumption can be fitted to the current workload [2]. Unused servers could be either shut down or made available to other processes. If present in a cloud, those servers could be leased to other applications.

DaMoN'13, June 24 2013 New York, NY, USA

So far, only the contribution by Lang et al. [4] delivered initial results to this problem, however, only using experiments on *static clusters* with differing server configurations where Vertica was running some TPC-H queries. Although they did not explain how the unused servers in the cluster could be turned on/off, they provided at least a kind of *existence proof* that research in DB clusters may lead to enhanced energy efficiency for DB applications.

Focusing on the DBMS's storage system, we have already checked the feasibility of approaching energy proportionality by dynamically migrating whole storage segments (32 MB) across nodes to balance between performance and energy consumption [5]. In this paper, we rely on a statically configured storage system with fixed segment/page allocation and aim at optimizations based on flexible turning on/off processing nodes to approximate an energy-proportional query-execution layer. We are running TPC-H queries on a cluster of nodes, in which we dynamically adjust the number of nodes to fit to the current workload. By measuring the energy consumption, we verify that energy-proportional query processing can be achieved.

2. ENERGY PROPORTIONALITY

Single-server DBMSs exhibit poor energy-proportional behavior, because most of the built-in hardware components are more or less static energy consumers, e. g., main memory, storage disks, and mainboard. Only the processor is capable of scaling down when idle [1].

A key observation made by Tsirogiannis et al. [8] concerning the energy efficiency of single servers, the best performing configuration is also the most energy-efficient one, because power use is not proportional to system utilization and, for this reason, runtime needed for accomplishing a computing task essentially determines energy consumption. Hence, the system must be fully utilized to be most energy efficient. However, real-world workloads do not stress servers continuously. Typically, their average utilization ranges between 20 and 50% of peak performance [1]. Therefore, traditional DB-servers are chronically underutilized and operate below their optimal energy-consumption-per-query ratio. As a result, there is a big optimization opportunity to decrease power consumption during off-peak times.

The main drawback of a traditional server is its high amount of static power consumption. Typically, an idle server, ready to accept queries, is already consuming more than 50% of its peak power, without performing any work. In a cluster of lightweight nodes, single servers can be powered independently, thus allowing more fine-grained control

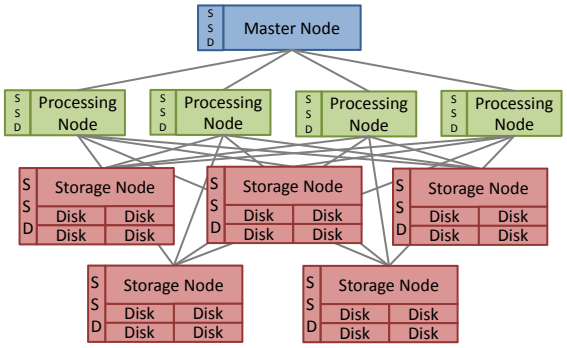
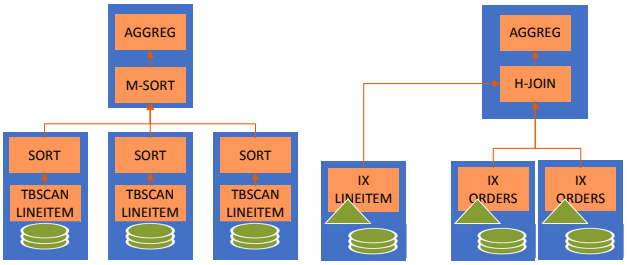


Figure 1: Overview of the WattDB cluster



(a) TPC-H Query 1 (b) TPC-H Query 4

Figure 2: TPC-H queries

over the overall energy consumption. We are taking this approach with *WattDB*, expecting energy savings under realistic workloads.

3. CLUSTER DESIGN

We have developed a research prototype of a distributed DBMS called *WattDB* on a scale-out architecture, consisting of n (currently 10) nodes, interconnected by an 1GBit/s Ethernet switch. The cluster consists of 10 identical nodes, composed of an Intel Atom D510 CPU, 2 GB DRAM and an SSD. The configuration is considered Amdahl-balanced [7], i. e., balanced between I/O and network throughput on one hand and processing power on the other.

Compared to InfiniBand, the bandwidth of the interconnecting network is limited but sufficient to supply the lightweight nodes with data. More expensive, yet faster connections would have required more powerful processors and more sophisticated I/O subsystems. Such a design would have pushed the cost beyond limits, especially because we would not have been able to use commodity hardware. Furthermore, by choosing lightweight components, the overall energy footprint is low and the smallest configuration, i. e., the one with the fewest number of nodes, exhibits low power consumption. Moreover, experiments running on a small cluster can easily be repeated on a cluster with more powerful nodes.

A dedicated node is the *master node*, handling incoming queries and coordinating the cluster. Some of the nodes have each four hard disks attached, and act as *storage nodes*, providing persistent data storage to the cluster. The remaining nodes (without hard disks drives) are called *processing nodes*. Due to the lack of directly accessible storage, they can only operate on data provided by other nodes (see Figure 1).

All nodes can evaluate (partial) query plans and execute DB operators, e. g., sorting, aggregation, etc., but only the *storage nodes* can access the DB storage structures, i. e., tables and indexes. Each storage node maintains a DB buffer to keep recently referenced pages in main memory, whereas a processing node does not cache intermediate results. As a consequence, each query needs to always fetch the qualified records from the corresponding storage nodes.

Hence, our cluster design results in a *shared-nothing architecture* where the nodes only differentiate to those which have or have not direct access to DB data on external storage. Each of the nodes is additionally equipped with a 128GB Solid-State Disk (Samsung 830 SSD). The SSDs do not store the DB data, they provide swap space to support external sorting and to provide persistent storage for configuration files. We have chosen SSDs, because their access latency is much lower compared to traditional hard disks; hence, they are better suited for temp storage.

In WattDB, a dedicated component, running on the master node, controls the energy consumption, called *EnergyController*. This component monitors the performance of all nodes in the cluster. Depending on the current query workload and node utilization, the *EnergyController* activates and suspends nodes to guarantee a sufficiently high node utilization depending on the workload demand. Suspended nodes do only consume a fraction of the idle power, but can be brought back online in a matter of seconds. It also modifies query plans to dynamically distribute the current workload on all running nodes thereby achieving balanced utilization of the active processing nodes.

3.1 Power Consumption

As we have chosen lightweight nodes, the power consumption of each node is rather low. A processing node with an SSD attached, but no additional hard drives, consumes ~22 Watts. Each hard drive adds about 1.5 Watts to the power consumption, hence each of our storage nodes use ~28 Watts. The interconnecting switch needs 17 Watts and is included in all measurements. The minimal configuration of the cluster consists of the storage nodes, as they cannot be switched off without losing access to the data, and the coordinating master, necessary to receive and distribute queries and to manage the cluster. All functionalities (storage, processing, and coordination) could be centralized on one node, further minimizing the static energy use. For the results reported in this paper, we have chosen to run the tasks on separate nodes to minimize interferences of the functions, e. g., memory contention.

3.2 Storage Structures and Indexes

In WattDB, data is stored in tables, which are subdivided into partitions using the physical tuple order. Hence, tuple distribution among the partitions is not controlled by logical predicates. Partitions organized as heaps consist of a set of segments, where a segment specifies a range of pages on a hard drive. To preserve physical clustering of a partition, it is always assigned to a single node and their segments are only distributed across the hard disks of that node. Segments can be moved to other disks and even to other nodes, but that technique is not used in this paper [5]. Indexes implemented as B*-trees can be created on a partition to speed up query evaluation.

In this paper, we have incorporated the TPC-H data generator directly into WattDB. The generator runs on the master node and inserting tuples on the storage nodes. The main reason to incorporate the data generator, instead of running an external application, was to reduce data generation times. Our decision did not interfere with query evaluation. Some data types of the TPC-H specification are yet unsupported by WattDB and therefore replaced by equivalent types. For example, the *DATE* type was replaced by an *INTEGER*, storing the date as YYYYMMDD, which is functionally identical. Key constraints were not enforced because of the same reason. Yet, the data generator automatically adheres to these constraints.

To support different query characteristics, i. e., I/O- and CPU-intensive workloads, we have created indexes on the columns used in query *Q4*. As a result, the query optimizer prefers an IX-Scan to a TABLESCAN, thus reducing the I/O requirements of the query. We did not create further indexes, especially ones to support query *Q1* to keep that query I/O-bound.

3.3 Load-aware Scheduler

The challenge of optimizing and scheduling queries in single-node systems becomes even more important in distributed environments. In addition to the two traditional optimization criteria, i. e., I/O and CPU usage, network utilization plays a crucial factor in distributed query plans. Furthermore, in a dynamic cluster, the number of computational resources is not fixed, i. e., additional nodes can be turned on to provide more CPU power and main memory if necessary. Hence, the choice where to execute a query operator is non-trivial, especially while trying to minimize energy consumption and/or execution times. Data access operators cannot be re-assigned, because table and index scans only occur at the storage level, i. e., performed by the storage nodes. But join, sort, and aggregate operators can be freely placed among the nodes.

We have implemented a cost-based scheduler, working on predefined statistics to estimate I/O and CPU costs of each operator. As dynamic decision information, the scheduler receives monitoring data from all nodes to estimate their current utilization. After annotating each operator in a query plan with its expected cost and the estimated number of result tuples, the operators are distributed among the nodes according to the nodes' utilization and the estimated network delay. The scheduler's estimated overall utilization of the cluster acts as an indicator for the power management component whether or not to adjust the number of active nodes. In turn, the total number of active nodes directly influences the scheduler's decisions.

Distributing query plans is the final step in query planning, before the execution of the physical operators begins. Because the scheduler is called for every new query, reacting to changing workloads can be performed very quickly, on a per-query-basis.

As we will show in the following, energy efficiency does not depend on query performance, instead we have to trade-off between the two. The scheduler can therefore be tuned to optimize for performance, energy consumption, or a mix of both. Optimizing for performance increases the estimated utilization of all nodes, hence, the power management will power-up nodes more aggressively.

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 number of active database nodes and the total throughput of queries during each test. This device sends the stream of measurements to a connected PC, running the monitoring software. This computer is also 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 the power consumption in high resolution. A more detailed description of the measurement device can be found in [3].

4. EXPERIMENTAL SETUP

We run all experiments using DB data and queries from the well-known TPC-H benchmark. We have generated a TPC-H database with a scale factor of 1 on the storage nodes and provided indexes (as conventional B*-trees) for the most important attributes.

In our experiments, we use a varying number $k \leq 7$ of statically assigned storage nodes. In case of $k = 1$, all TPC-H tables are allocated across the disks of that single storage server. In case of $k > 1$, the larger tables LINEITEM and ORDERS are divided into k equal-sized partitions and uniformly assigned across the k storage nodes, whereas each of the smaller tables (PART, SUPPLIER, PARTSUPP, CUSTOMER, NATION, REGION) is always fully allocated on a single storage node. Up to $n-k$ nodes are used as processing nodes.

As our workload, we have selected the TPC-H queries *Q1* and *Q4*, because these queries enable us to check and demonstrate important performance-critical aspects of a DBMS—primarily the scan/sort/aggregate and join performance of WattDB.

4.1 Q1

The query plan for *Q1* is depicted in Figure 2(a). This is an I/O- and CPU-intensive query, scanning all data in the LINEITEM relation and selecting some tuples. The tuples are then sorted and aggregated. To leverage the high number of CPU cores in the cluster, sorting is split into two phases: First, all tuples are pre-sorted on the originating storage node. This done using an external sort algorithm, spilling the intermediate sort runs to the Solid-State Disk. Second, the pre-sorted tuples are streamed to a processing node, where all runs from the partitions are sorted using merge sort. Additionally, the now sorted tuples are grouped and aggregated.

4.2 Q4

Figure 2(b) sketches the query plan for *Q4*. In the TPC-H specifications, *Q4* contains an EXISTS subquery, which is unnested by the optimizer and replaced by an equi-join, which produces identical results with superior performance. The tuples of the inner (ORDERS) and outer (LINEITEM) relation are accessed via an index range scan. We have chosen a hash join, because the inner relation fits well into main memory. Using the result of the join, the tuples are aggregated as defined by TPC-H. Both queries are parameterized according to the specifications.

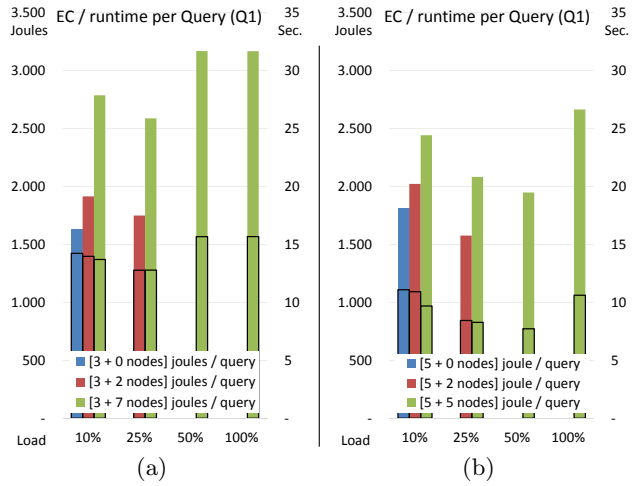


Figure 3: (a) Varying query load (from 10% to 100%) using TPC-H query Q1 on 3 storage nodes and 0 to 7 processing nodes. (b) Same query load on 5 storage nodes with up to 5 processing nodes.

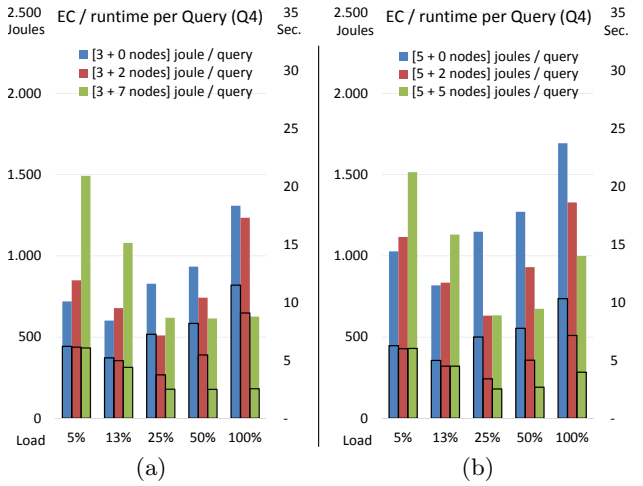


Figure 4: (a) Varying query load of TPC-H query Q4 using 3 storage nodes and up to 7 processing nodes. (b) Same query load on 5 storage nodes with 0 to 5 processing nodes.

5. EXPERIMENTAL RESULTS

After having generated the data, we evaluate a series of queries concurrently issued by a number of DB clients. Each test runs for 600 seconds.

5.1 Static Cluster

First, we present measurements run on a fixed number of nodes to demonstrate that energy consumption and performance can be tuned to our needs. For this reason, we select the most energy-efficient configuration for each workload beforehand. To carve out the differences between the two queries, *Q1* and *Q4*, we have run them separately on the cluster. Each workload in our experiments has a differing number of parallel DB clients running on the benchmark/monitoring PC. The number of active DB clients, continuously sending queries to the database, is controlled by the benchmark specification. Thus, the utilization of the database changes with the amount of DB clients.

Figures 3 (a) and (b) plot the energy consumption and performance running *Q1* on a cluster of nodes. Depending primarily on the number of nodes, the power consumption of a given cluster is more or less constant and, therefore, not shown in the graphs. The X-axis depicts the load of the cluster—starting with a very low load on the left and increasing it to the right. To characterize how the cluster is utilized, we define its load level x% by the number x of the DB clients. Hence, a load of 100% represents 100 DB clients. To enable better comparability, we normalized all results to *quantity per query*. The solid bars in the graphs illustrate the average *energy consumption per query* (Y-axis on the left), whereas the framed (unfilled) bars report the average *runtimes per query* (Y-axis on the right). The (up to) three bars (belonging to experiments characterizing the same utilization) represent clusters, where the number of nodes increase going from left to right.

We have evaluated identical workloads on different cluster sizes to demonstrate the dependency between energy consumption and performance. The smallest cluster, running the benchmark only on 3 storage nodes, has the lowest power consumption (∼114 Watts), but also the lowest performance (compare the left-most bars in Figure 3(a)).

For low utilizations (see X-axis), the energy consumption per query is therefore minimal, although the runtime per query is higher, compared to larger cluster configurations. With an increasing number of DB clients, the small cluster comes to its limits and is unable to handle the workload, i. e., the available main memory is exhausted. Therefore, more powerful—yet more power-consuming—configurations take over.

The next, larger configuration includes two more processing nodes (one of them is the already present master node). This configuration has a higher energy consumption under low utilization, but is able to process bigger workloads.

Finally, we have combined 3 storage nodes with 7 processing nodes, creating an even more power-consuming configuration (∼246 Watts). Of course, its high number of active nodes leads to a waste of energy under low utilization, as the query processing time stays almost the same. The (3 +7 nodes) configuration is the only one powerful enough to handle all the workloads. Still, the query runtimes increased under high utilization, possibly due to a bottleneck in the I/O subsystem.

We have repeated the same benchmark on different cluster configurations having 5 storage nodes, depicted in Figure 3(b). Although the idle power consumption is higher and low-utilization workloads result in worse energy efficiency, the 5-storage-node cluster exhibits better energy efficiency under high loads. Increasing the number of processing nodes results in the same behavior as previously described, although the query runtimes at 50% load are further decreasing.

On one hand, adding two more storage nodes instead of processing nodes and therefore increasing the I/O bandwidth pays off at high load. On the other hand, as the two additional nodes increase the lowest possible power consumption, the energy efficiency at low utilization is worse. Therefore, we can already conclude that performance does not necessarily correlate with energy efficiency.

Next, we have executed the same benchmark for query *Q4*. The six cluster configurations were identical to the previous experiments. Figures 4 (a) and (b) illustrate the results for query *Q4*, where a utilization of 100% represents 200 DB clients. The left figure depicts measurements on the cluster using 3 storage nodes ((3 + x) cluster), whereas the right figure shows those for the cluster with 5 storage nodes. All configurations are able to process the workloads, but with rising load, the query runtime gets worse. With a higher number of processing nodes, query performance is improved, whereas, however, energy consumption is increased. This result indicates that I/O bandwidth in the *Q4* experiments is sufficient. Now, the number of processing nodes is a performance-critical factor, as the cluster with (3 + 7) nodes exhibits a better performance than the 5-storage-node cluster, having only 5 processing nodes.

To further explore the influence of the I/O bandwidth in the cluster, we have measured the performance/energy outcome for the (1 + x)-nodes and (7 + x)-nodes cluster similar to the experiments in Figures 3 and 4. Due to space limitations, the graphs cannot be shown here. In the (1 + x)-nodes cluster, bandwidth to disks is too low, thus, I/O latency for the individual tasks and, in turn, the entire processing times are strongly increased. As a consequence, query response times, throughput, and energy efficiency are impaired. Hence, we may not reach given performance goals, while we unnecessarily waste energy due to prolonged query runtimes.

In the (7 + x)-nodes cluster, I/O bandwidth is not in short supply. Obviously, its static fraction of power consumption is higher, because storage nodes can't be turned off. As a result, the energy efficiency of this configuration at low utilization is worse, compared to smaller clusters. Although the 7 storage nodes provide enough disk bandwidth for all benchmarks, i. e., queries are not slowed down by I/O latencies, the overall energy efficiency suffers from the steadily high power consumption. Additionally, high workloads of *Q4* are afflicted with the limited availability of processing nodes.

These experiments clarify the importance of an adequate I/O subsystem. Because the two latter configurations (1 and 7 storage nodes) do not provide more insights, we have focused on the two middle-sized configurations to present in this paper. For each query workload, an optimal configuration w.r.t. energy efficiency exists: The lower utilizations are handled best by the smallest cluster, as its performance is sufficient and it consumes the least power. With rising load, the next bigger cluster shows better energy efficiency. Finally, at full utilization, the most powerful cluster offers the best efficiency, although it needs the most power.

These experiments on a statically configured cluster already reveal the opportunity of trading performance for energy savings. Yet, manually configuring the database to fit the expected workload is not optimal. For highly varying workloads, it is not even possible to select a single, fixed configuration with balanced performance and power consumption.

5.2 Dynamic Cluster

After having evaluated the energy/performance behavior of static clusters, we wanted the cluster to dynamically adjust to a given workload, without needing predefined configurations. The cluster should tune the number of active processing nodes to fit the current load, and power up/down such nodes when the workload changes. To reach this goal, we use our power management component (*EnergyController*), running on the master node.

This component monitors the current state of the cluster and is able to startup and shutdown (suspend) nodes. We already introduced and explained power management for storage nodes in [5], where we experienced prolonged provisioning times due to the physical re-assignment of data segments and the resulting copy times. In this work, power management involves processing nodes, which do not have any additional startup cost and can come online in a matter of seconds. Similarly, suspending an underutilized processing node can occur immediately after finishing the last running query on it.

By running benchmarks on static configurations (see previous section), we have determined the maximum number of parallel queries per node and other limiting factors like CPU and memory bottlenecks. We feed this data to the EnergyController to improve the quality of its decisions, whether the cluster is over- or underutilized.

After setting up the cluster, we run a series of workloads with a varying number of parallel queries of both, Q1 and Q4. In this experiment, we have run queries of both types in parallel, to generate a more complex workload with I/O- and CPU-intensive parts. Each workload, with a fixed number of DB clients, runs for 10 minutes; hence, the database load changes every 600 seconds. The DB cluster adjusts itself to satisfy the current workload by powering up/down nodes.

Figure 5 plots the results in sequence. Each data point represents a number of DB clients running in parallel and sending either *Q1* or *Q4* queries to the cluster. The X-axis depicts the number of concurrent queries of each kind. The upper part of Figure 5 consists of three graphs, plotting *performance* as query throughput, *power consumption* of the cluster, and *energy efficiency* expressed in number of queries per 10 Watts.

To compare the measurements using a dynamically adjusting cluster, we have re-run the same benchmark on a fixed number of nodes, and also included the results in this graph. The small-dashed (blue) line represents the dynamic cluster, the large-dashed (red) line the fixed cluster with 5 processing nodes, and the dotted (black) line the (5 + 1)-nodes cluster.

The fixed cluster configurations mark the bounding box in which the dynamic cluster can adjust. While the small cluster shows the lowest power consumption, its query capabilities are limited. Therefore, it is unable to exceed a certain performance, even at high system utilization. The big cluster has the highest power consumption of all three configurations and, naturally, the highest performance. But under low system load, that performance lays waste and the energy efficiency is impaired.

The dynamic cluster is able to adjust between both extremes and quickly adapts to the current workload. Therefore, its power consumption and performance covers the entire range between the two static configurations. Note, however, it more or less matches the performance of the big cluster, while it saves a substantial amount of energy. Gain in energy efficiency is particularly high for moderate cluster utilization where maximum performance can be achieved with much less power consumption (see middle part of the benchmark in Figure 5).

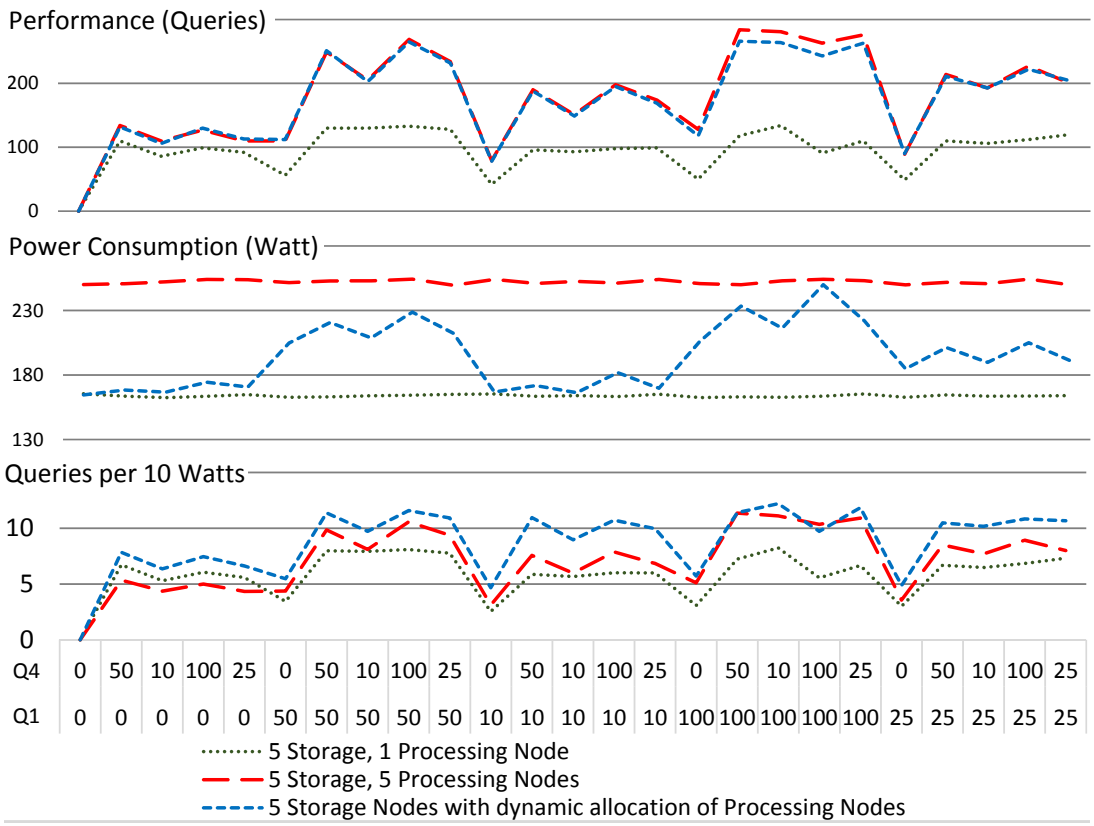


Figure 5: Experiments on three cluster configurations

6. CONCLUSION AND FUTURE WORK

In this paper, we have exposed the opportunity to trade performance for energy savings by manually selecting an adequate number of nodes to process the workloads. We have also shown that the best performing configuration is not always the most energy-efficient one. Instead, performance and energy efficiency are competing goals to be balanced. Static configurations require prior knowledge of the upcoming workloads and exhibit drawbacks under varying workloads, as a predefined configuration cannot exhibit ideal behavior for all load situations. By scaling the number of active nodes to the current need, we are able to dynamically adjust power consumption and performance. Therefore, we can select the best configuration, either in terms of power consumption, energy efficiency, or performance.

These experiments also emphasize the importance of an adequate I/O-subsystem. As we have shown in [5], adapting the storage to changing workloads—although possible and energy-saving—is a cumbersome and slow task. Because of the timespan needed to copy and move data, adjustments cannot be made every few seconds; hence, pre-selecting a robust storage configuration for the expected workload is essential. Changing the number of processing nodes as done in these experiments is a far more lightweight operation, as it does not require to restart queries or change data placement. WattDB is therefore able to react to workload variations within a few seconds. Combining both approaches to form a dynamically adjusting storage and processing layer seems promising for future work.

7. REFERENCES

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

[2] T. Härder, V. Hudlet, Y. Ou, and D. Schall. Energy efficiency is not enough, energy proportionality is needed! In *DASFAA Workshops, 1st Int. Workshop on FlashDB, LNCS 6637*, pages 226–239, 2011.

[3] V. Hudlet and D. Schall. Measuring energy consumption of a database cluster. In *BTW, LNI 180*, pages 734–737, 2011.

[4] W. Lang, S. Harizopoulos, J. M. Patel, M. A. Shah, and D. Tsirogiannis. Towards energy-efficient database cluster design. *PVLDB*, 5(11):1684–1695, 2012.

[5] D. Schall and T. Härder. Towards an energy-proportional storage system using a cluster of wimpy nodes. In *BTW, LNI 214*, pages 311–325, 2013.

[6] D. Schall, V. Höfner, and M. Kern. Towards an enhanced benchmark advocating energy-efficient systems. In *TPCTC, LNCS 7144*, pages 31–45, 2012.

[7] A. S. Szalay, G. C. Bell, H. H. Huang, A. Terzis, and A. White. Low-power Amdahl-balanced blades for data-intensive computing. *SIGOPS Oper. Syst. Rev.*, 44(1):71–75, 2010.

[8] D. Tsirogiannis, S. Harizopoulos, and M. A. Shah. Analyzing the energy efficiency of a database server. In *SIGMOD Conference*, pages 231–242, 2010.