An Analysis of Memory Power Consumption in Database Systems

Alexey Karyakin
University of Waterloo
200 University Avenue West
Waterloo, Ontario N2L 3G1
alexey.karyakin@uwaterloo.ca

Kenneth Salem
University of Waterloo
200 University Avenue West
Waterloo, Ontario N2L 3G1
kmsalem@uwaterloo.ca

ABSTRACT

The growing appetite for in-memory computing is increasing memory's share of total server power consumption. However, memory power consumption in database management systems is not well understood. This paper presents an empirical characterization of memory power consumption in database systems, for both analytical and transactional workloads. Our results indicate that memory power optimization will be effective only if it can reduce background power through more aggressive use of low power memory idle states.

ACM Reference format:
Alexey Karyakin and Kenneth Salem. 2017. An Analysis of Memory Power Consumption in Database Systems. In *Proceedings of DaMoN'17, Chicago, IL, USA, May 15, 2017,* 9 pages.
DOI: http://dx.doi.org/10.1145/3076113.3076117

1 INTRODUCTION

CPUs have generally been considered to be the dominant power consumers in database management systems (DBMS). For this reason, much of the previous work on DBMS power optimization has focused on CPUs [17, 18, 21, 25, 27]. Various studies [9, 19, 23] that break down power consumption have attributed anywhere from 5% to over 40% of server power consumption to memory, depending on system configuration and workload. However, the growing appetite for large-memory servers to support fast analytics over large volumes of data, combined with the falling cost of memory over time, are conspiring to increase memory's share of total server power consumption. For example, a recent analysis [5] using the HP Power Advisor suggested that DRAM power consumption may exceed CPU power consumption in servers with a few terabytes of memory. Such servers are already a reality.

As memory becomes responsible for a greater share of the total system power consumption, techniques for memory power optimization will become more critical. However, before attempting to optimize memory power, it is important to understand how it is consumed. The goal of this paper is to contribute to that understanding. We focus on in-memory databases, since the demand for fast in-memory data processing is one of the drivers of memory growth.

We are interested in understanding how database systems consume memory power under a variety of conditions. The dynamic range of memory power is significant. Understanding memory power under peak load is of limited value if such peaks are rare. Similarly, it is important to consider situations in which the system's memory is not fully utilized, since systems are often overprovisioned to allow for growth. Thus, in this paper, we address two questions. First, *how does memory power consumption vary with load in database systems?* Second, *how does it vary with database size?* We answer these questions experimentally, by subjecting database systems to controlled workloads and measuring memory power consumption. We use a test server with a custom-built power measurement apparatus that allows us to directly measure the power consumed by individual DIMMs. We also use a memory power model, calibrated using the same measurement apparatus, to explain the power measurements we obtain.

Our results show that memory power consumption is load-dependent, but is highly non-proportional. That is, large increases in load result in relatively small increases in memory power consumption. Furthermore, most memory power consumption is background power, rather than active power (Section 3), even for the most memory-intensive workloads we tested. Finally, memory power consumption is largely independent of database size. This is because default system configurations distribute memory allocations across all memory modules. As a result, a DBMS that uses only 30% of the available memory on a server will use 30% of every DIMM, rather than fully utilizing 30% of the DIMMs and leaving the remainder idle. Because of DIMM load/power characteristics, the result is that all DIMMs consume a lot of power. These observations strongly suggest that DBMS memory power optimizations will not be effective unless they focus on reducing background power.

2 RELATED WORK

There is a growing body of work on improving the energy efficiency of database systems. Much of this work focuses on the energy consumed by CPUs [17, 18, 21, 25, 27] or secondary storage [25], using tools such as dynamic voltage and frequency scaling (DVFS) to manage energy consumption. Other work [12, 24] has taken a more holistic view of the problem, and tries to improve the overall energy efficiency of a system, e.g, by controlling the query execution plan or the scheduling of database operators.

Relatively little work has focused specifically on memory. Bae and Jamel [6] consider a heuristic technique for adjusting the size of a DBMS buffer pool to control a power/performance tradeoff.

DaMoN'17, Chicago, IL, USA

DOI: http://dx.doi.org/10.1145/3076113.3076117

Korkmaz et al [17] report preliminary results on the use of rank-aware memory allocation to reduce memory power consumption in DBMS under an OLTP workload. They estimate memory power reductions of up to 40% in scenarios in which memory is overprovisioned. Most recently, Appuswamy et al. [5] consider the effects of memory frequency scaling and low-power modes on energy efficiency. Their results suggest that memory frequency scaling has little effect on energy efficiency, while low-power modes did improve energy efficiency. However, results are reported for only a single (unspecified) load condition and database size for each workload. Our results complement theirs by exploring a variety of test conditions and by analyzing the various components of memory power consumption.

Tsirogiannis et al [24] characterized database server power consumption as a function of workload characteristics, such as types of query operators and degree of parallelism. The breakdown of total energy consumption across server components was analyzed. However, this analysis did not consider how memory power varies with load, and assumed it to be constant. Using per-DIMM power measurement, we find that memory consumption does depend on workload and characterize this dependency.

There has also been work on optimizing memory power consumption for systems other than DBMS. Delaruz et al. [10] proposed to use process scheduling to maximize memory idle intervals. Huang et al. [13] propose to consolidate cold memory regions ones in a few memory modules, to boost their low power state residency. Malladi et al. [22] argue that software-based techniques alone cannot achieve enough memory idleness and propose hardware improvements to reduce power state transition latencies. Zhang at al. [28] envision a system that can power off individual components, such as DRAM modules of whole NUMA nodes, to further reduce power consumption during periods of lower utilization, compared to low power modes.

3 BACKGROUND

The memory subsystem is organized hierarchically. Each CPU has one or more *memory controllers*. Each memory controller exposes several (typically between 1 and 4, or 8 if an extender is used) memory *channels*. A number of DRAM *ranks* are connected to each channel, sharing its address and data signals, and most of its control signals. Because of channel sharing, only one rank can communicate to a memory controller at a time.

Physically, DRAM ranks are packaged into *modules*. Today, dual in-line memory modules, or *DIMMs*, built with DDR3 [2] and DDR4 [3] DRAM are common. DIMMs define the granularity of memory power measurement, because all ranks in a DIMM are powered using a single set of power supply rails. A DIMM may contain a *register* or *buffers*, which contribute to the total power consumption of the DIMM.

DRAM standards [2, 3] define a number of *power states*. To receive and execute commands, DRAM must be in the *Standby* state. In other power states, which we refer to collectively as *low power states*, some DRAM components are disabled. This reduces power consumption but introduces a delay in command processing (*exit latency*) because a transition to the Standby state is required before commands can be handled. Transitions into and out of the Standby

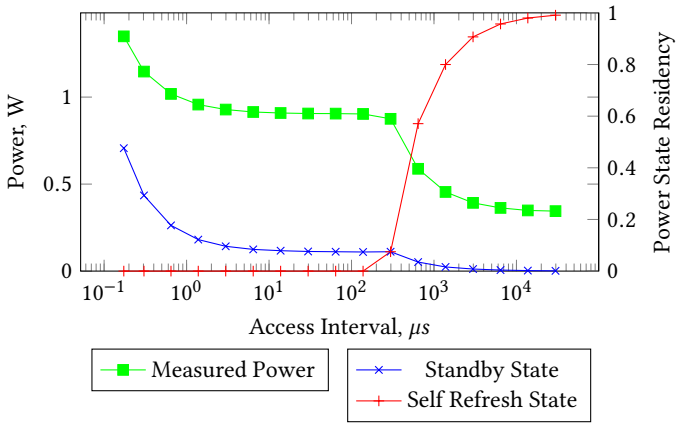

Figure 1: Single DIMM Power and Power State Residencies vs. access interval

state are controlled by the CKE (Clock Enable) signal generated by the memory controller. One important low power state is *Self Refresh*, in which DRAM consumes minimum power. During Self Refresh, the DRAM interface is completely disabled, except for the CKE line, and refresh cycles are generated internally. The CKE signal can go inactive for each rank separately. Therefore, most power states apply per-rank. Power state transitions are managed by the memory controller according to a *power policy*, and no direct controls are exposed to software.

We designed a simple microbenchmark to illustrate the power characteristics of a DIMM. In this microbenchmark, a single thread issues a sequence of read requests to a DIMM, with a controllable delay (referred to as the *access interval*) between requests. Figure 1 shows DIMM power consumption (on the left vertical axis) as a function of the access interval, for a DDR4 DIMM. DIMM power was measured using the measurement apparatus described in Section 4.1.

The power consumption curve shows several clear phases. When the DIMM is accessed infrequently (access intervals larger than 500 microseconds), the DIMM spends much of its time in Self Refresh state, and power consumption drops down almost to 0.2 watts. For access intervals in the range of 1-500 microseconds, the DIMM drops into a low power state between accesses, but not all the way down to Self Refresh. In this range, power consumption hovers just under 1 watt. Below 1 microsecond, the DIMM starts to spend more time in the Standby state, and power consumption climbs with the access rate. At peak load (with is not shown in the figure as it cannot be achieved with this benchmark design), power consumption for this DIMM tops out at about 3 watts.

4 POWER MEASUREMENT

To understand how database systems consume memory power, we need a way to measure it. A common solution to this problem is to rely on memory power estimates reported by the server. RAPL is a hardware mechanism used to estimate and limit power consumed by CPU and DRAM in Intel processors [9]. RAPL uses a model

Table 1: Major sources of error in memory power measurement

Type of error	Magnitude	Description
Current calibration resistor tolerance	1%	Mostly affects sensitivity and applies equally and linearly to all measurements
Current sensor nonlinearity	1.5%	Computed over the range 0 to approximately 5A
Current sensor supply voltage drop	1%	Non-linear, reduces the difference between high and low power values

to estimate memory energy consumption. The model's inputs include the values of performance counters in the CPU and memory controller, e.g., counts of the number of memory read and write operations. Software can read RAPL's memory power estimates.

There are two problems with this approach. First, the actual RAPL energy model and its calibration procedure are not documented, and its accuracy is unspecified. A recent validation of RAPL DRAM power estimates reported mixed results [11]. Estimates generally tracked measured power trends, but the magnitude of the estimation error varied depending on the processor and type of DIMM that was tested. In addition, estimation error was reported to be larger when there was more idleness, which is problematic since the exploitation of low power states is one opportunity for memory power optimization. A second problem with RAPL estimates is that they are difficult to explain, since the model on which they are based is undocumented.

To address the first problem, we customized a server with an apparatus to allow us to directly measure the power consumed by individual DIMMs. To address the second, we developed a simple model that estimates memory power consumption as a function of processor-measurable workload statistics, such as power state residencies and memory operation counts. We validate the model's power predictions using measurements, and use the model itself to explain those measurements. We describe the customized test server in Section 4.1, and the power model in Section 4.2

4.1 Test Server

To allow for memory power measurement, we used customized test server for all of our experiments. The server has dual 8-core Intel Xeon E5-2640 v3 ("Haswell") processors, running at 2.60 GHz, on the Asus Z10PE-D16 motherboard with AMI BIOS dated 01/25/2016. Each processor has four memory channels, with two DIMM slots in each channel. We populated one DIMM slot in each channel with a dual-rank 16GB DDR4-1866 DIMM, totalling 128 GB. We used default BIOS settings in all experiments.

We measure memory power by installing a DIMM riser with current sense capability in each of the memory slots and measuring the current on DIMM power supply rails. DDR4 memory has multiple power supplies, but only VDD and VPP are directly related to memory power, so we measured only these two. Current in each power supply line was measured by a Hall effect sensor (Allegro ACS725 [20]) that produces an analog voltage signal proportional to current. We assumed VDD and VPP voltages to be constant (1.2 V and 2.5 V) as we have found that they change at most by 1% between the states with idle and full memory utilization. This allowed us to reduce the number of data acquisition channels twice. To register the analog signals, we use a pair of Microchip evaluation boards [15], each consisting of MCP3914 8-channel simultaneous sampling Delta-Sigma ADC [14] and a microcontroller. However, the integrated microcontroller was not powerful enough to collect data in all available sampling rates, so we read digital data directly from the serial ADC interface using another microcontroller board (STM32 Discovery [4]). Overall, the measurement platform can collect two current values from each of the 8 DIMMS, totalling 16 analog channels, at 78K samples per second. In practice, the signal bandwidth was limited by a low-pass filter in the input of the evaluation board, with the cut-off frequency of 1591 Hz. For measurement purposes, we were interested in average values over relative long intervals (seconds), therefore, we kept the filter and used higher oversampling rates of the Delta Sigma ADC, producing lower output sampling rate.

Unfortunately, we did not have access to high accuracy current calibration equipment, therefore we performed two-point calibration in both voltage and current domains. Overall, we believe the relative power measurement error is within 5%. For voltage calibration, we measured the output of a 2.5 V±0.02% voltage reference (Maxim MAX6325), divided with a 0.1% two-resitor voltage divider. We used 0 V as the second interpolation point by measuring the voltage with shorted ADC inputs. The value of his measurement is equal to the offset error of the ADC. To calibrate current to voltage transformation coefficients, we simultaneosly measured the output of the ACS725 current sensor and the voltage drop on a 1% current sense resistor with two channels of the MCP3914 ADC, for two different currents, 0 A and approximately 4 A.

There are a number of factors that affect measurement accuracy, especially given the complexity of the measurement platform. We list the major ones in Table 1.

4.2 Memory Power Model

We use a linear power model, similar to the model descibed by Korkmaz et al [17], to break down and explain our DIMM power consumption measurements. Total memory power consumption is commonly broken into two components: *background power* and *active power* [1, 7, 26]. Background power represents power consumption that depends only on the power state of the DIMM, and not on the operations (e.g., reads and writes) being performed by the DIMM. Although DRAM specifications define many power states, not all are implemented and used. For our model, we consider three states: CKEON, SR, and CKEOFF. CKEON is the Standby state of the DIMM, when the CKE signal is enabled and read and write operations are possible. SR represents the Self Refresh state. CKEOFF represents all low power (CKE disabled) states other than Self Refresh. For DIMMs with multiple ranks, we decompose the CKEON state into several sub-states, depending on how many of the ranks are active, i.e., have CKE enabled.

Table 2: Power Model Power/Energy Coefficients

Parameter	Value	Stdev	Units	Description
E_{act}	6.0	0.4	nJ	Energy of an activate/precharge cycle
E_r	6.6	0.15	nJ	Energy of a page read operation
E_w	8.7	0.17	nJ	Energy of a page write operation
P_{sr}	0.35	0.05	W	Background power in the Self Refresh state
P_{off}	0.89	0.05	W	Background power in the CKEOFF state, per DIMM
P_{on}	1.56	0.04	W	Background power in the CKEON state, per DIMM
P^i_{on}	0.098	0.006	W	Additional power, per rank i, in the CKEON state

We model background power, P_{bk}, as follows:

$$P_{bk} = T_{sr}P_{sr} + T_{off}P_{off} + T_{on}P_{on} + \sum_{i \in ranks} T^i_{on}P^i_{on} \quad (1)$$

In this expression, the T terms represent normalized state residencies, i.e., the percentage of time that a DIMM (or rank) spends in a power state. These state residencies are workload-specific model input parameters, values for which can be measured while the workload under test is executing. The P coefficients represent power consumption in each state. Values for these parameters are determined by a model calibration process in which we subject the system to controlled synthetic workloads, measure memory power, and then estimate power coefficients using regression.

When it is in the CKEON (Standby) state, a DIMM consumes additional power per operation performed. This is the active component of the total power consumption. Our model recognizes three operations: row activations, reads, and writes. We do not explicitly count refreshes, because the refresh rate is constant. The power consumed by refresh operations is implicitly included in the background power. Active power, P_{act}, is modeled as:

$$P_{act} = N_{act}E_{act} + N_rE_r + N_wE_w \quad (2)$$

Here, the Ns are model input parameters representing the number of operations of each type, and the Es coefficients are calibrated values representing the energy consumed per operation. The total power consumption for a DIMM is the sum of P_{bk} and P_{act}.

Tables 2 and 3 summarize the power coefficients and the model inputs. We also included the average model coefficient values and their standard deviation in Table 2. Values of the model coefficients shown in Table 2 will be specific to a particular model of DIMM. We obtain the coeffcients values by fitting a linear regression to a set of measured power values in a separate set of synthetic workloads. The set of workloads used in calibration was chosen such that workload characterics vary independently along the following dimensions: read/write ratio, sequential versus random access, access to local or remote CPU memory, frequency of random access, and whether one or two ranks were accessed in a module.

Table 3: Power Model Input Parameters

Variable	Description
N_{act}	Number of activate/precharge cycles
N_r	Number of page reads
N_w	Number of page writes
T_{sr}	Normalized Self Refresh state residency
T_{off}	Normalized CKEOFF state residency
T_{on}	Normalized CKEON state residency of DIMM
T^i_{on}	Normalized CKEON state residency of rank i

5 MEMORY POWER ANALYSIS

We characterize DBMS power consumption for two types of workloads: transaction processing (OLTP) and analytical (OLAP). In particular, we run the TPC-C and TPC-H benchmarks, respectively. These benchmarks differ in their data access patterns: in TPC-C records are accessed mostly by index look-up (point queries), while TPC-H queries scan large data ranges e.g. for aggregations.

We used Shore-MT [16] for the TPC-C experiments. Shore-MT is a research storage manager, optimized for multi-core systems. It implements a traditional buffer pool with a variant of CLOCK page replacement policy, ARIES transaction logging and recovery. Shore MT comes with a set of benchmarking tools, called Shore Kits. We used the Shore Kits implementation of the TPC-C workload for our experiments.

For OLAP workloads, we ran the TPC-H benchmark on MonetDB [8], which is an column store. MonetDB relies on OS file mapping mechanism to access persistent data. Therefore, it operates best when the mapped files are cached by the OS and degrades when the dataset size exceeds the amount of available memory.

We focus on memory-resident workloads, therefore, we limited database sizes to ensure that they fit in memory. In the TPC-C experiments, 96 GB of total 128 GB of physical memory was allocated to the buffer pool and in TPC-H the database was resident in the OS file system cache. We monitored disk activity to verify that read I/O was not significant during measurement.

We analyzed memory power for various combinations of database size and DBMS load. In a real setting, these two parameters are often correlated. However, we varied them independently to understand their individual effect on the energy consumption. For each combination of database size and offered load, we ran the workload, measured consumed power in each DIMM, and collected a number of DRAM counter values for each memory channel. We then used these counters as inputs to the memory power model, estimating the background and active power components.

In TPC-C, the database scale factor was varied between 100 and 600 warehouses, which translates to initial database size of approximately 12.5 to 81 GB. Each experiment consisted of a database generation step and three 30-minute runs. Before each run, the database was restored from a saved copy and the Shore Kits process was restarted. We divided each measurement run into 15-second

intervals and collected performance and power data for each interval separately. Since each run started in cold state, we considered intervals that did not reach 80% of the set throughput as warm-up and discarded the results obtained during these intervals. To reduce the effect of database growth in TPC-C due to data insertion, we only report the results of the first 10 "warm" intervals (2.5 minutes). The database grew between 0.6 and 4 GB during each experiment, depending on the transaction rate. Although database growth introduces a variability of the database memory footprint in each experiment, this effect is small because database growth during each run is small compared to the size of available memory.

For TPC-C, we modified the workload generator to insert uniform random think times, with controllable mean, between requests. First, we determined the a nominal maximum transaction rate by running the experiment with no think time using the smallest database size. On our system, this was approximately 300000 tpmC. In actual experiments, the think times were then calibrated to produce target throughputs ranging from 1/8 of the maximum to the maximum, in 8 steps. The load generator used 1000 client threads and 16 worker threads. When presenting measurement results as a function of throughput in Section 5.1, we normalize all throughput values to the nominal maximum value.

For TPC-H, we varied the database scale factor from 6 to 72, which resulted in the database sizes ranging from 7.2 GB to 86 GB. We control the database workload intensity by controlling the number of concurrent query sessions, while restricting each session to use a single core in the server. Each session executes a batch of all 22 queries of the TPC-H benchmark in a random order, to reduce the possibility of inter-query optimization. By varying the number of concurrent sessions from 1 to 16, we generate load from 1/16 to 16/16 of full CPU utilization in our system. We repeated each run 5 times, restarting the database and clearing the filesystem cache between runs, and take an average for each metric in the last four runs. To measure throughput, we first compute the reciprocal of the geometric mean of query batch run time of all client sessions, giving a measure of the query completion rate per session. We then multiply this by the number of sessions and the database scale factor, since TPC-H queries take longer for larger databases. This metric, which we refer to as *TPC-H scaled througput*, approximates the amount of work done by the database system per unit of time.

5.1 TPC-C Results

For our TPC-C experiments, Figure 2 shows the measured average memory power consumption as a function of workload intensity (transaction throughput), for three different database sizes. Memory power consumption is highly non-proportional with respect to workload intensity. The highest transaction rate we tested is about 8 times higher than the lowest rate. Over this range of workload intensity, memory power grows linearly, but only by about 23%.

Figure 2 also shows that, perhaps surprisingly, memory power consumption is not affected by the database size. That is, if we run TPC-C transactions at the same rate against databases of two different sizes, the total memory power consumption is the same. There are several reasons for this. First, the memory load imposed by TPC-C transactions doesn't depend strongly on database size. With a smaller database, one might expect the memory load to

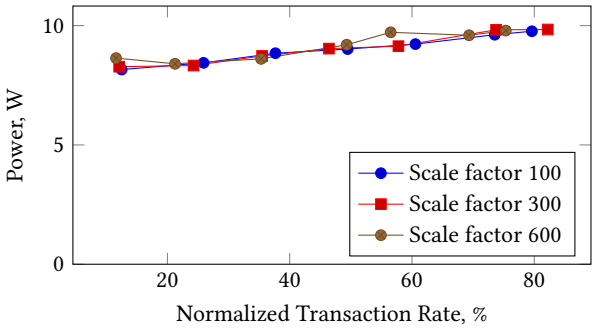

Figure 2: Average DRAM power in TPC-C vs. normalized transaction rate, for three database sizes

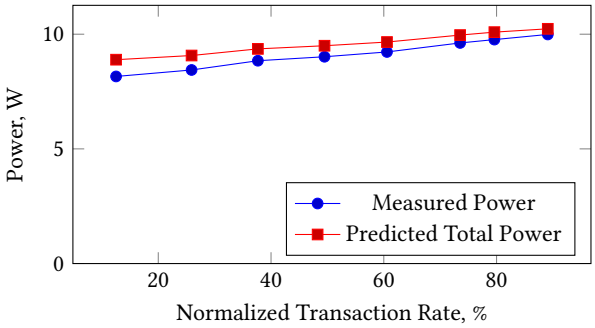

Figure 3: Actual and Predicted Average DRAM power in TPC-C vs. normalized transaction rate, small database (SF=100)

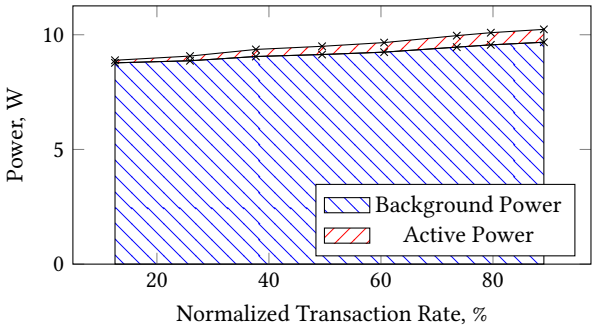

Figure 4: Active and Background DRAM power breakdown in TPC-C vs. normalized transaction rate, small database (SF=100)

be concentrated on fewer DIMMs. However, this is not true - all DIMMs have similar power consumption. This is because memory is allocated across all DIMMs, even for small database sizes. A primary cause of this is memory interleaving, which controls the mapping of the server's physical address space to the DIMMs. Interleaving results in a fine-grained distribution of physical addresses across all of the DIMMs for a given socket. Hence, the memory load is spread across the DIMMs as well.

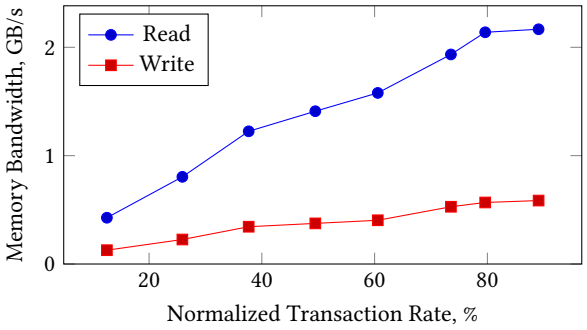

Figure 5: Memory read and write bandwidth vs. normalized transaction rate in TPC-C, small database (SF=100)

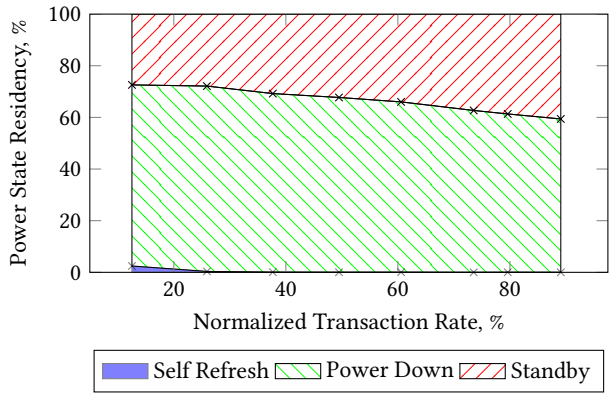

Figure 6: Power state residency vs. normalized transaction rate in TPC-C, small database (SF = 100)

To further analyze our results, we used the memory power model that was presented in Section 4.2. Figure 3 illustrates the accuracy of the power model by comparing the measured total memory power consumption with the consumption predicted by the model, for the small database size (SF 100). Comparisons of measured and estimated power for the other database sizes are not shown, but are very similar. For TPC-C, the model slightly overestimates the measured power throughout the workload intensity range, but the estimation error is small, with a maximum difference of about 10%.

Figure 4 breaks estimated total memory power consumption into background and active components, again for the small database size. Memory power consumption is almost entirely attributable to background power consumption. Both background power and active power increase with workload intensity, although the former increases even more than the latter. Even at the maximum load we tested, active power represents a very small fraction (less than 10%) of total power consumption. Active power is low because only a fraction of available memory bandwidth (approximately 120 GB/s) is utilized by TPC-C. Figure 5 shows measured memory read and write bandwidths, as a function of workload intensity. Background power increases with load because DIMMs spend more time in the Standby power state as the workload intensity increases. Figure 6 shows the residency in the Standby and Self Refresh states as functions of

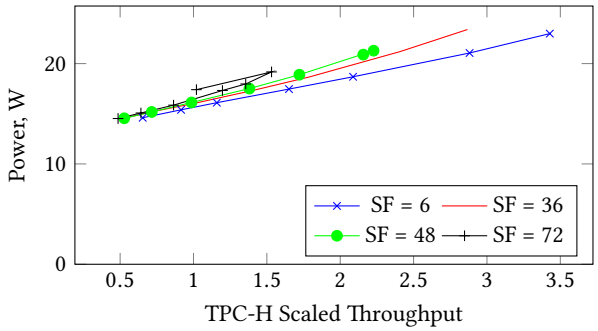

Figure 7: Memory power vs. Scaled Throughput in TPC-H

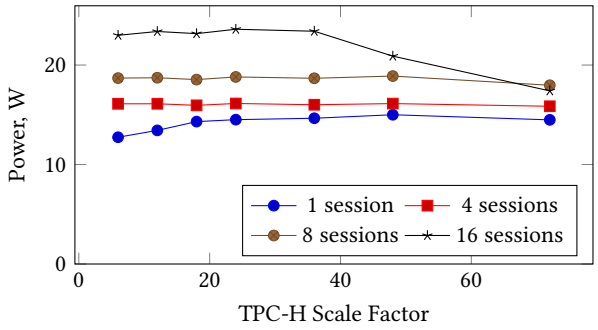

Figure 8: Memory power vs. database size in TPC-H

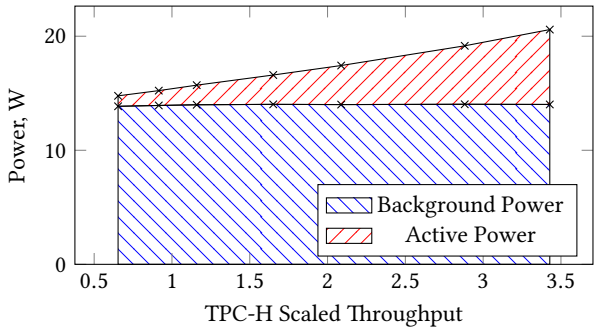

Figure 9: Active and Background DRAM power breakdown in TPC-H vs. Scaled Throughput, small database (SF=6)

workload intensity. At the lowest workload intensity, DIMMs on average spend only 27% of the time in the Standby state. This rises to 40% at the highest workload intensity. Unfortunately, Figure 6 also shows that the DIMMs almost never sink all the way into Self Refresh mode, even when the workload intensity is low. Although we have shown power state residencies for only the small database size, this observation is true for all database sizes we tested. Thus, even a moderate workload does not exhibit enough memory idle time to save power in the Self Refresh state. This represents a lost opportunity, as power consumption in Self Refresh is significantly lower than in other power states.

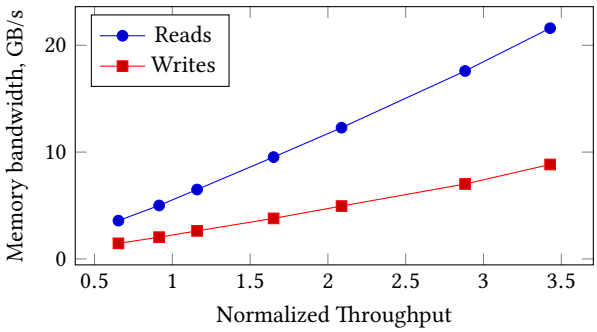

Figure 10: TPC-H Memory bandwidth vs. Scaled Throughput

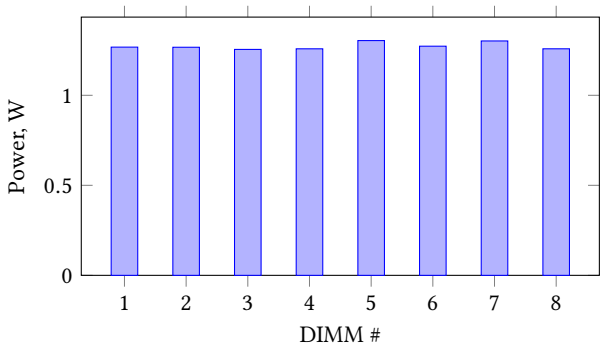

Figure 11: Power consumed by individual DIMMs in TPC-C runs with memory interleaving, SF=100, 100% load

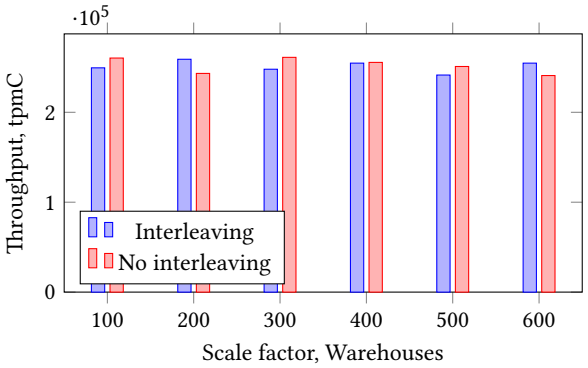

Figure 12: Maximum TPC-C throughput in runs with and without memory interleaving

5.2 TPC-H Results

Figure 7 shows measured average memory power consumption as a function of workload intensity (TPC-H scaled throughput), for a range of database sizes. Recall that, for a given database size, we vary throughput by varying the number of concurrent query sessions. As was the case for TPC-C, memory power increases linearly with workload intensity, but starts at a relatively high value. Hence, power proportionality is poor. TPC-H is also more memory power hungry than TPC-C. Memory power consumption under our most intensive TPC-H workloads is almost twice as high as that under our peak TPC-C workload.

Figure 8 shows the same data as Figure 7, but plotted against database size, rather than workload intensity. In general, memory power largely independent of the database size, for the same reasons that it is independent for TPC-C. As Figure 8 shows, database size does have some effect of memory power in two situations. First, when there is only a single session and the database is very small, most of the memory traffic goes to one processor's DIMMs. The other processor's DIMMs are lightly loaded, reducing the overall memory power consumption. The other anomalous case is when the number of concurrent sessions is high and the database is very large. In this case, the system is overloaded and throughput drops, lowering memory usage.

We used the power model to further explain the power measurements shown in Figures 7 and 8. Because of space constraints, we do not present a comparison of estimated and measured power. However, for TPC-H we found that the model's power estimates were very accurate for low workload intensities. As workload intensity increased, the model tended to underestimate the actual power consumption. However, the underestimation was never more than 15%.

Figure 9 shows the model's estimated breakdown of memory power into background and active components, for the small database. This breakdown is similar for all other database sizes except for the largest one, for which active power falls off at high workload intensities when the system is overloaded. The active power component is much larger than it was for TPC-C, because the TPC-H benchmark is much more memory intensive. Nonetheless, most of the power consumption is still due to the background component, as was the case for TPC-C. Background power consumption does not grow with workload intensity, as it does for TPC-C. The reason for this is is that DIMMs are almost always in the Standby power state under TPC-H, even with a single query session. Standby power state residency was at least 98% in all runs.

6 NON-INTERLEAVED MEMORY

As discussed in Section 5, memory interleaving tends to distribute memory accesses uniformly over each processor's DIMMs. For example, Figure 11 shows the power consumed by each DIMM for a TPC-C run with scale factor 100 and 100% load, with memory interleaving. DIMMs 1 to 4 are connected to CPU socket 1 and DIMMs 5 to 8 are connected to CPU socket 2. All DIMMs consume similar power, despite the fact that the database occupies only about 12.5 GB of the 96 GB available on the server.

Since memory power consumption is dominated by background power, power optimization techniques should try to increase the time that DIMMs spend in low power states. However, by distributing memory accesses across DIMMs at a fine granularity, interleaving tends to keep all DIMMs busy. Thus, memory interleaving is likely to doom any attempt to optimize DIMM power consumption.

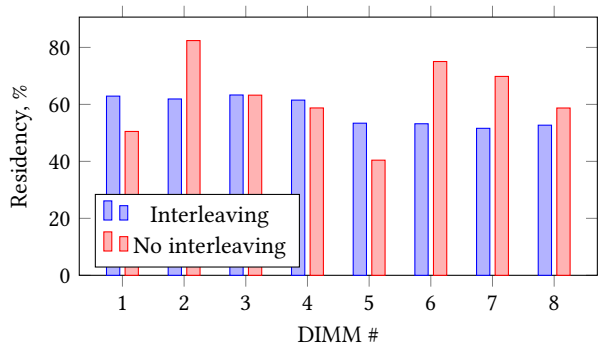

Figure 13: CKE OFF state residency in individual DIMMs in TPC-C runs with and without memory interleaving, SF=100, 100% load

In this section, we consider the implications of disabling memory interleaving, since doing so appears to be a prerequisite for memory power optimization. Memory interleaving is a performance optimization, as it allows memory access to be parallelized across DIMMs. Therefore, they key question we wish to answer is how much of a performance impact memory interleaving has on our TPC-C and TPC-H workloads. If it is large, then memory power optimization may require a substantial performance trade-off. If not, then memory power optimization may be possible "for free".

To answer this question, we repeated the TPC-C and TPC-H experiments described in Section 5, but with memory interleaving disabled. Interleaving was disabled by changing interleaving settings in the system BIOS. In the non-interleaved experiments, we did not attempt to control how the database systems made use of virtual memory. Neither did we attempt to control kernel's use of physical memory or the mapping of physical memory to DIMMs.

6.1 TPC-C Without Memory Interleaving

Figure 12 shows TPC-C performance (at 100% load, with no client think times) with and without interleaving, for databases of different sizes. For TPC-C, memory interleaving has no significant effect on performance, regardless of the database size. This is because the TPC-C workload is not very memory intensive, as was shown in Section 5.

Since disabling interleaving does not hurt TPC-C performance, it can be disabled to enable memory power optimization techniques. Our experiments also found that disabling memory interleaving does not, by itself, result in significant memory power savings. It does lead to uneven use of the DIMMs, because of the way data happen to map to DIMMs in our test server. For example, Figure 13 shows the CKE OFF power state residency for each individual DIMM for TPC-C runs with and without interleaving. The variance across the DIMMs is significantly higher in the non-interleaved case. However, these differences do not translate into substantial differences in power consumption. At maximum load, for the small TPC-C database, total memory power consumption was about 10 watts with interleaving, and 9.6 watts without interleaving, a difference of less than 5%. Thus, disabling interleaving should be viewed as a prerequisite for the use of application of memory power optimizations (such as rank-aware allocation [17]), and not as a power-saving technique in its own right.

Table 4: TPC-H total run time, seconds, with and without memory interleaving

Scale factor	With interleaving	Without interleaving	Relative slowdown
6	22.4	24.6	10%
48	232.2	254.7	9.7%

6.2 TPC-H Without Memory Interleaving

Table 4 shows the total running time of all 22 TPC-H queries, for small (scale factor 6) and large (scale factor 48) databases, with and without memory interleaving. TPC-H is a much more memory intensive workload than TPC-C, and in this case we do see some negative performance impact from disabling interleaving. As the table shows, the performance hit was about 10% on average over all of the TPC-H queries. However, we also found that some TPC-H queries were more sensitive than others to interleaving. Figure 14 shows the run time of each individual TPC-H query, with and without interleaving, for the large database. Six queries suffered slowdowns of 20% to 40%, and one query (Query 6) showed about 75% performance degradation. These results suggest that for more memory-intensive database workloads, like TPC-H, memory power optimization is likely to require a performance tradeoff.

7 CONCLUSION

Our experiments show that background power dominates memory power consumption for in-memory databases, for both TPC-C and TPC-H workloads. Active power does increase with load, but it represented only 30% total memory power in the most intensive OLAP workload we tested, and did not exceed 10% in OLTP workloads. Our results also show that memory power consumption is not reduced when memory is not fully used, i.e., when the database is small.

We believe that the implications of this are clear: memory power optimization for DBMS must focus on background power. That is, the goal must be to increase the amount of time that DIMMs spend in low power states, especially Self Refresh. Unfortunately, today's systems, which are typically configured with interleaved memory and which employ memory controllers with conservative power policies, work against such optimizations. Providing DBMS with better control over memory configuration parameters that affect background power consumption will allow them to exploit memory power/performance tradeoffs to save energy.

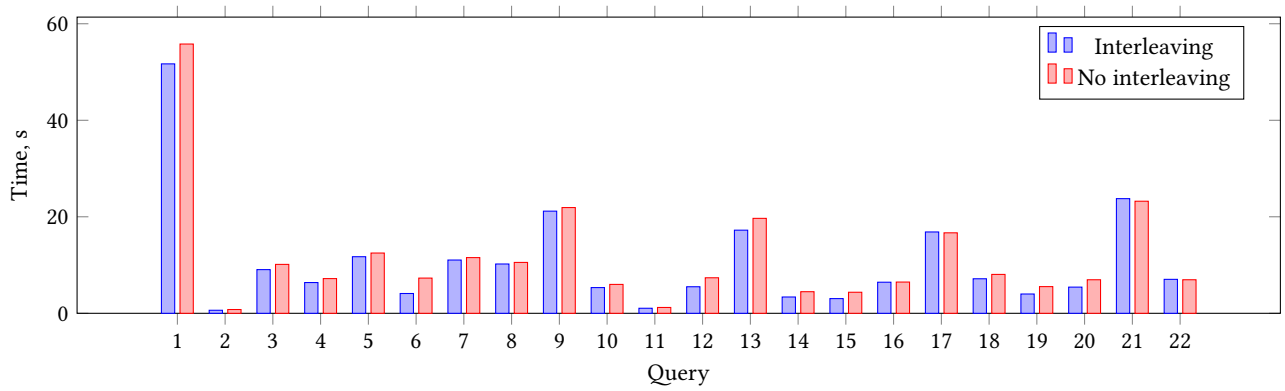

Figure 14: Average query run times in TPC-H with and without memory interleaving, SF=48

REFERENCES

[1] 2007. TN-41-01: Calculating Memory System Power for DDR3. Technical Note. (Aug. 2007).

[2] 2012. JESD79-3F. DDR3 SDRAM. JEDEC Standard. (July 2012).

[3] 2013. JESD79-4A. DDR4 SDRAM. JEDEC Standard. (Nov. 2013).

[4] 2014. STM32F4DISCOVERY. Discovery kit with STM32F411VE MCU. Data Brief. (Nov. 2014). http://www.st.com/en/evaluation-tools/32f411ediscovery.html

[5] Raja Appuswamy, Matthaios Olma, and Anastasia Ailamaki. 2015. Scaling the Memory Power Wall With DRAM-Aware Data Management. In *Proceedings of the 11th International Workshop on Data Management on New Hardware (DaMoN'15)*. ACM, New York, NY, USA, 3:1–3:9. DOI: http://dx.doi.org/10.1145/2771937.2771947

[6] Chang S Bae and Tayeb Jamel. 2011. Energy-aware Memory Management through Database Buffer Control. In *Proc. Workshop on Energy-Efficient Design*.

[7] Ishwar Bhati, Zeshan Chishti, and Bruce Jacob. 2013. Coordinated Refresh: Energy Efficient Techniques for DRAM Refresh Scheduling. In *Proceedings of the 2013 International Symposium on Low Power Electronics and Design (ISLPED '13)*. IEEE Press, Piscataway, NJ, USA, 205–210. http://dl.acm.org/citation.cfm?id=2648668.2648720

[8] Peter A. Boncz, Martin L. Kersten, and Stefan Manegold. 2008. Breaking the memory wall in MonetDB. *Commun. ACM* 51, 12 (2008), 77–85. DOI: http://dx.doi.org/10.1145/1409360.1409380

[9] Howard David, Eugene Gorbatov, Ulf R. Hanebutte, Rahul Khanna, and Christian Le. 2010. RAPL: Memory Power Estimation and Capping. In *Proceedings of the 16th ACM/IEEE International Symposium on Low Power Electronics and Design (ISLPED '10)*. ACM, New York, NY, USA, 189–194. DOI: http://dx.doi.org/10.1145/1840845.1840883

[10] V. Delaluz, A. Sivasubramaniam, M. Kandemir, N. Vijaykrishnan, and M. J. Irwin. 2002. Scheduler-based DRAM energy management. In *Proc. Design Automation Conference*. ACM Press, 697. DOI: http://dx.doi.org/10.1145/513918.514095

[11] Spencer Desrochers, Chad Paradis, and Vincent M. Weaver. 2016. A Validation of DRAM RAPL Power Measurements. In *Proceedings of the Second International Symposium on Memory Systems (MEMSYS '16)*. ACM, New York, NY, USA, 455–470. DOI: http://dx.doi.org/10.1145/2989081.2989088

[12] S. Gotz, T. Ilsche, J. Cardoso, J. Spillner, T. Kissinger, U. Assmann, W. Lehner, W.E. Nagel, and A. Schill. 2014. Energy-Efficient Databases Using Sweet Spot Frequencies. In *2014 IEEE/ACM 7th International Conference on Utility and Cloud Computing (UCC)*. 871–876. DOI: http://dx.doi.org/10.1109/UCC.2014.142

[13] Hai Huang, Kang G. Shin, Charles Lefurgy, and Tom Keller. 2005. Improving Energy Efficiency by Making DRAM Less Randomly Accessed. In *Proceedings of the 2005 International Symposium on Low Power Electronics and Design (ISLPED '05)*. ACM, New York, NY, USA, 393–398. DOI: http://dx.doi.org/10.1145/1077603.1077696

[14] Microchip Technology Inc. 2013. MCP3914: 3V Eight-Channel Analog Front End. (Aug. 2013). http://www.microchip.com/wwwproducts/en/MCP3914

[15] Microchip Technology Inc. 2013. MCP3914 ADC Evaluation Board for 16-bit MCUs. User's Guide. (Oct. 2013). http://ww1.microchip.com/downloads/en/DeviceDoc/50002176A.pdf

[16] Ryan Johnson, Ippokratis Pandis, Nikos Hardavellas, Anastasia Ailamaki, and Babak Falsafi. 2009. Shore-MT: A Scalable Storage Manager for the Multicore Era. In *Proceedings of the 12th International Conference on Extending Database Technology: Advances in Database Technology (EDBT '09)*. ACM, New York, NY, USA, 24–35. DOI: http://dx.doi.org/10.1145/1516360.1516365

[17] Mustafa Korkmaz, Alexey Karyakin, Martin Karsten, and Kenneth Salem. 2015. Towards Dynamic Green-Sizing for Database Servers. In *International Workshop on Accelerating Data Management Systems Using Modern Processor and Storage Architectures - ADMS 2015, Kohala Coast, Hawaii, USA, August 31, 2015*. 25–36. http://www.adms-conf.org/2014/adms15_korkmaz.pdf

[18] Willis Lang, Ramakrishnan Kandhan, and Jignesh M. Patel. 2011. Rethinking Query Processing for Energy Efficiency: Slowing Down to Win the Race. *IEEE Data Eng. Bull.* 34, 1 (2011), 12–23. http://adrem.ua.ac.be/sites/adrem.ua.ac.be/files/eopt.pdf

[19] C. Lefurgy, K. Rajamani, F. Rawson, W. Felter, M. Kistler, and T.W. Keller. 2003. Energy management for commercial servers. *Computer* 36, 12 (Dec. 2003), 39–48. DOI: http://dx.doi.org/10.1109/MC.2003.1250880

[20] Allegro Microsystems LLC. 2015. ACS725: Automotive-Grade, Galvanically Isolated Current Sensor IC With Common-Mode Field Rejection in a Small Footprint. (2015). http://www.allegromicro.com/en/Products/Current-Sensor-ICs/Zero-To-Fifty-Amp-Integrated-Conductor-Sensor-ICs/ACS725.aspx

[21] David Lo, Liqun Cheng, Rama Govindaraju, Luiz Andre Barroso, and Christos Kozyrakis. 2014. Towards Energy Proportionality for Large-scale Latency-critical Workloads. In *Proceeding of the 41st Annual International Symposium on Computer Architecuture (ISCA '14)*. IEEE Press, Piscataway, NJ, USA, 301–312. http://dl.acm.org/citation.cfm?id=2665671.2665718

[22] Krishna T. Malladi, Ian Shaeffer, Liji Gopalakrishnan, David Lo, Benjamin C. Lee, and Mark Horowitz. 2012. Rethinking DRAM Power Modes for Energy Proportionality. In *Proceedings of the 2012 45th Annual IEEE/ACM International Symposium on Microarchitecture (MICRO-45)*. IEEE Computer Society, Washington, DC, USA, 131–142. DOI: http://dx.doi.org/10.1109/MICRO.2012.21

[23] Justin Meza, Mehul A. Shah, Parthasarathy Ranganathan, Mike Fitzner, and Judson Veazey. 2009. Tracking the Power in an Enterprise Decision Support System. In *Proceedings of the 2009 ACM/IEEE International Symposium on Low Power Electronics and Design (ISLPED '09)*. ACM, New York, NY, USA, 261–266. DOI: http://dx.doi.org/10.1145/1594233.1594295

[24] Dimitris Tsirogiannis, Stavros Harizopoulos, and Mehul A. Shah. 2010. Analyzing the Energy Efficiency of a Database Server. In *Proceedings of the 2010 ACM SIGMOD International Conference on Management of Data (SIGMOD '10)*. ACM, New York, NY, USA, 231–242. DOI: http://dx.doi.org/10.1145/1807167.1807194

[25] Yi-Cheng Tu, Xiaorui Wang, Bo Zeng, and Zichen Xu. 2014. A System for Energy-efficient Data Management. *SIGMOD Rec.* 43, 1 (May 2014), 21–26. DOI: http://dx.doi.org/10.1145/2627692.2627696

[26] Donghong Wu, Bingsheng He, Xueyan Tang, Jianliang Xu, and Minyi Guo. 2012. RAMZzz: Rank-aware Dram Power Management with Dynamic Migrations and Demotions. In *Proceedings of the International Conference on High Performance Computing, Networking, Storage and Analysis (SC '12)*. IEEE Computer Society Press, Los Alamitos, CA, USA, Article 32, 11 pages. http://dl.acm.org/citation.cfm?id=2388996.2389040

[27] Zichen Xu, Yi-Cheng Tu, and Xiaorui Wang. 2010. Exploring power-performance tradeoffs in database systems. In *2010 IEEE 26th International Conference on Data Engineering (ICDE)*. 485–496. DOI: http://dx.doi.org/10.1109/ICDE.2010.5447840

[28] Dongli Zhang, Moussa Ehsan, Michael Ferdman, and Radu Sion. 2014. DIMMer: A Case for Turning off DIMMs in Clouds. In *Proceedings of the ACM Symposium on Cloud Computing (SOCC '14)*. ACM, New York, NY, USA, Article 11, 8 pages. DOI: http://dx.doi.org/10.1145/2670979.2670990

.