Scaling the Memory Power Wall With DRAM-Aware Data Management

Raja Appuswamy
EPFL
raja.appuswamy@epfl.ch

Matthaios Olma
EPFL
matthaios.olma@epfl.ch

Anastasia Ailamaki
EPFL
anastasia.ailamaki@epfl.ch

ABSTRACT

Improving the energy efficiency of database systems has emerged as an important topic of research over the past few years. While significant attention has been paid to optimizing the power consumption of tradition disk-based databases, little attention has been paid to the growing cost of DRAM power consumption in main-memory databases (MMDB).

In this paper, we bridge this divide by examining power– performance tradeoffs involved in designing MMDBs. In doing so, we first show how DRAM will soon emerge as the dominating source of power consumption in emerging MMDB servers unlike traditional database servers, where CPU power consumption overshadows that of DRAM. Second, we show that using DRAM frequency scaling and power-down modes can provide substantial improvement in performance/Watt under both transactional and analytical workloads. This, again contradicts rules of thumb established for traditional servers, where the most energy-efficient configuration is often the one with highest performance.

Based on our observations, we argue that the long-overlooked task of optimizing DRAM power consumption should henceforth be considered a first-class citizen in designing MMDBs. In doing so, we highlight several promising research directions and identify key design challenges that must be overcome towards achieving this goal.

1. INTRODUCTION

Over the past few years, a significant reduction in cost/GB of DRAM, coupled with an ever-increasing demand for low-latency query processing, has led to the rise in popularity of main-memory databases (MMDB) [12, 6, 5, 19]. Unlike traditional disk-based databases, MMDB are optimized to operate on data that resides entirely in DRAM. As MMDB installations continue to scale in size, servers hosting MMDBs are being increasingly populated with a large number of high-density DIMMs.

Research on power consumption of traditional disk-based database servers has shown storage and CPUs to be the major source of power consumption [25, 16, 23]. Consequently, the majority of research in improving the energy efficiency of database systems has focused on reducing CPU power consumption using Dynamic Voltage and Frequency Scaling (DVFS) [13, 24], or shifting to power-proportional storage systems based on SSDs [23].

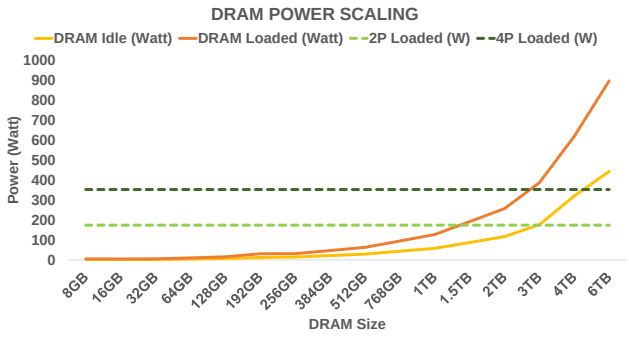


Figure 1: DRAM, CPU power consumption as reported by HP Power Advisor.

The server hardware that is used to house MMDBs, however, has very different price, performance, and power consumption characteristics compared to a traditional database server, as large amounts of persistent storage media, and associated RAID/SAN controllers, are replaced by a proportionate amount of high-density DRAM DIMMs. Thus, it is important to understand if the conclusions drawn by research on traditional databases still hold for MMDBs.

MMDB power breakdown In order to understand the power consumption of DRAM, CPU, and other system components in state-of-the-art database servers, we used the HP Power Advisor [8] to derive power consumption estimates for HP Proliant DL580 Gen8– the server recommended by HP for running a mid-sized (500 to 1000 tenants) database server [9]. We configured the server to use six-core Intel® Xeon® E7-4809v2 in both two-socket (2P) and four-socket (4P) configurations. A 2P configuration has 48 DIMM slots while 4P has 96 DIMM slots. Thus, the maximum possible memory capacity that can be provisioned is 3 TB for 2P setup and 6 TB for 4P setup.

Figure 1 shows power consumption of DRAM and CPUs as reported by the HP Power Advisor from the minimum (8 GB) to maximum (6 TB) permissible memory capacity. For each of the sixteen memory sizes shown, we populated the server by using all valid DIMM types and recorded the reported power consumption values. For instance, for the memory size of 128 GB, we gathered power reports for 32×4 GB, 16×8 GB, 8×16 GB, 4×32 GB, 2×64 GB

DaMoN'15, June 01 2015, Melbourne, VIC, Australia
2015 ACM. ISBN 978-1-4503-3638-3/15/06...$15.00
DOI: http://dx.doi.org/10.1145/2771937.2771947.

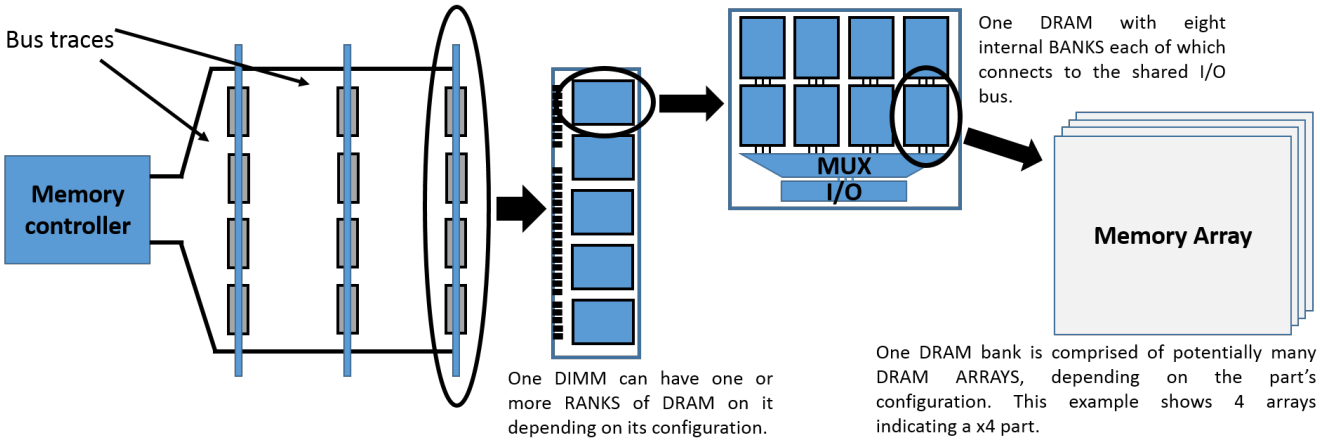


Figure 2: The memory subsystem as found in modern servers that follows the DIMM, rank, chip, bank, array, cell hierarchy.

DIMM configurations. Figure 1 shows the power consumed by the optimal configuration and there are three important observations to be made.

First, DRAM power consumption is no longer negligible in a loaded system as it matches CPU power consumption at 1.5 TB for the 2P setup, and at 3 TB for the 4P setup. At 6 TB, DRAM consumes 2.5×–5.15× more power than CPUs in 2P–4P setup respectively. In fact, we found that even with 15-core Xeon E7-4870 CPUs in a 4P setup (total of 60 cores), DRAM matched the power consumption of all CPUs put together (700 W) at 4 TB and consumed 30% more at 6 TB.

Second, even the idle DRAM power consumption exceeds that of fully loaded CPUs above 3 TB for 2P, and 4 TB for 4P setups. In addition, comparing idle power consumption of DRAM and CPUs (not shown), we observed that at 6 TB, DRAM consumes 2.4× to 4.7× more power than idle CPUs. These two observations clearly indicate that, contrary to the studies on traditional disk-based database servers [23], DRAM in MMDB servers no longer lags behind CPUs in the power consumption race.

Third, the results clearly indicate how the slope of DRAM power consumption curve changes from sublinear at low capacity to superlinear at high capacity. For instance, DRAM loaded power increases a meager 3× as we scale capacity 16× from 8 GB to 128 GB. But the next 4× capacity increase (to 1 TB) brings about a 10× rise in power consumption. As we will describe later in Section 2, this happens due to two reasons. First, high capacities in a single server can only be achieved using high-density DRAM DIMMs due to a strict upper limit on the number of DIMMs that can be populated in a server. Second, high-density DRAM DIMMs consume much more power than their low-density counterparts.

Towards energy-efficient MMDB Given these observations, we believe that MMDBs should be designed with optimizing DRAM power consumption as a primary target. In this paper, we take the first steps towards meeting this goal by identifying promising hardware features that could help in reducing DRAM power consumption (Section 3). Then, we present experimental results that quantify the pros and cons of using these features with a state-of-the-art MMDB under transactional and analytical benchmarks (Section 4). Finally, we outline design extensions to databases that are necessary for achieving the new optimization goal and identify open research problems that merit more attention from the database community (Section 5).

2. BACKGROUND

Figure 2 shows the hierarchical organization of a modern memory subsystem. The Memory Controller (MC), which is typically integrated on the same package as the CPU, issues commands to DRAM chips on a Last-Level Cache (LLC) miss. The memory bus that interconnects MC with DRAM chips is split into multiple channels in order to enable disjoint requests across channels to be serviced in parallel.

Each channel can be populated with one or more *Dual Inline Memory Modules* (DIMM). Each DIMM is a Printed Circuit Board (PCB) that consists of several DRAM chips and other peripheral circuitry like Phase Lock Loop (PLL) and register devices. Each DRAM chip produces/consumes multiple bits on each access and is classified into types (×4, ×8) depending on access granularity (4 or 8 bits). In order to satisfy a memory access request, a group of DRAM chips, referred to as a *rank*, are accessed in parallel. For instance, given a bus width of 64 bits, a rank of eight ×8 DRAM chips would be accessed in parallel to transfer 64 bits. Each DIMM can pack several such DRAM chips, and hence, typically consists of one, two, or four disjoint ranks.

Each DRAM chip internally contains multiple *banks*, each of which, in turn, is composed of several two-dimensional storage arrays. The basic unit of storage in the array, also referred to as the *DRAM cell*, is a capacitor capable of storing a bit.

2.1 DRAM Scaling Tradeoff

Capacity Limitation Table 1 shows the different DDR3 memory types used in server systems today. Standard DRAM DIMMs, also referred as to an Unbuffered DIMM (UDIMM), outshine other memory types with respect to performance and power consumption. However, as each UDIMM places an electrical load on the memory bus, using multiple UDIMMs per channel cause signal integrity issues at high clock frequencies. Hence, servers limit UDIMMs to just 2 dual-rank DIMMs per channel. Thus, a 2P server with four memory channels per socket would be limited to a maximum of 128 GB with UDIMMs (2 sockets × 4 channels/socket × 2 DIMMs/channel × 2 ranks/DIMM × 4 GB/rank).

Capacity–Performance Tradeoff: Registered (RDIMM) and Load Reduced DIMMs (LRDIMM) reduce electrical loading by buffering and lining up address, control, and data signals using an on-board buffer chip. Due to reduced electrical loading, it is possible to use 3 quad-rank LRDIMMs per channel increasing the

Memory Type	Max Capacity	Latency (nsec)	Bandwidth (GB/s)	Loaded W/GB	Idle W/GB
UDIMM	128	153.7	72	0.2	0.02
RDIMM	384	161.4	60	0.2	0.04
LRDIMM	768	235	40.4	0.15	0.07
HCDIMM	768	161.9	63.9	0.74	0.37

Table 1: Maximum capacity, performance, and power characteristics obtained by using various server DIMM types to populate a 2P HP Proliant Gen8 server equipped with Intel® Xeon® E5 processors [10, 17].

maximum capacity to 768 GB[1]. However, the increase in capacity offered by LRDIMMs comes at the expense of performance. Compared to UDIMMs which can be clocked to run at, or in some cases even above, 1600 MT/s, LRDIMMs have to be throttled at 1066 MT/s to achieve full capacity utilization. This, in combination with longer I/O trace lengths (in PCB), and added component delays caused by the centralized memory buffer, have a significant impact on performance, as LRDIMMs reduce bandwidth by 45% and increase latency by 53%.

Capacity–Power Tradeoff: HCDIMMs [17] regain the lost performance by replacing LRDIMM's centralized buffer with multiple distributed on-board buffers (one buffer per column of DRAM chips). Further, as HCDIMMs can be clocked at 1333 MT/s, they provide 17% increase in bandwidth compared to LRDIMMs. Unfortunately, the performance improvement comes at the expense of power consumption as the use of distributed buffers produces 3.7× increase in loaded power consumption with respect to UDIMMs. Worse yet, LRDIMMs and HCDIMMs incur an unwarranted 3.5×–18.5× increase in idle power consumption.

DDR4: Recent trends indicate that this situation is likely to continue in the future. With DDR4 DRAM, servers can be provisioned with either a single DDR4 DIMM per channel clocked at 3200 MT/s or 3 DIMMs at 1333 MT/s. In addition, DDR4 is also expected to introduce a new point-to-point topology that can be used in place of the traditional stub-bus topology used today. With the new topology, only one DIMM is used per channel and it interfaces with the memory controller using high-speed point-to-point links. As the new topology eliminates stubbing and reduces the load factor to a single DIMM, it is expected to improve performance significantly as DIMMs can be clocked to run at high data rates without any noticeable signal degradation. However, as just one DIMM per channel is clearly too little for modern servers, DDR4 is expected to increase capacity by interfacing the memory controller with digital switches that, in turn, connect to DDR4 LRDIMMs using the traditional bus topology. The digital switch is itself expected to add latency in addition to the LRDIMM buffering latency. Thus, the capacity–performance tradeoff continues with DDR4.

DDR4 introduces lower operating voltages, and hence lower power consumption, compared to DDR3. However, DDR4 LRDIMMs borrow and build upon the distributed buffering architecture used by HCDIMMs for increasing density and reducing load. Hence, it is likely that the capacity–power tradeoffs that exist today with UDIMM/HCDIMM will continue to exist with DDR4.

Given these tradeoffs, and given the ever-increasing in-memory dataset sizes, it is evident that DRAM will soon emerge as one of major (if not the primary) sources of power consumption in MMDB servers.

[1]The maximum capacity shown here differs from Figure 1 as the latter was obtained using Intel® Xeon® E7 series of processors which support eight memory channels per socket.

3. MOTIVATION

The high idle and loaded power consumption of DRAM causes more harm than the mere increase in operational expenses incurred by powering and cooling.

Today, enterprise servers can only be configured with a single DRAM type (UDIMM, LRDIMM or HCDIMM). As mixed-mode provisioning is not permitted, the choice of DRAM type determines the amount of memory that can be provisioned in a server. Since a single server is limited to 128 or 256 GB (depending on the processor generation) with UDIMMs, the only option to scale memory with UDIMMs is to opt for a scale out configuration. However, such a setup increases operational expenses, as one needs to manage, power, and cool two or more servers, and also impacts performance as distributed transactions incur the overhead of two-phase commit. While the scale-up configuration avoids such penalties, one is limited to using performance-limited LRDIMMs or power-hungry HCDIMMs in such a setup. Thus, in either case, DRAM not only imposes a huge power penalty on database installations that are memory intensive, but also plays a crucial role when it comes to choosing between scaling up or out.

In addition, given that server provisioning is done with peak utilization in mind, it is likely that MMDB servers will be over provisioned to accommodate peak load. Thus, a significant fraction of memory in the server is likely to remain under utilized. Given DRAM's high idle power usage (Figure 1, Table 1), the only way to reduce under utilization is to consolidate more clients in a database (multitenancy) or more database instances in a single server (virtualization). However, this creates a vicious cycle, as consolidation cannot be done without reprovisioning for the new peak capacity, and the added DRAM capacity cannot be utilized fully without more consolidation.

Given these effects, we believe that it is more important now, than ever before, to revisit database design decisions with the goal of optimizing DRAM power consumption. In this section, we will describe recent hardware extensions that could be used by databases to achieve this goal.

3.1 Hardware Support

Modern DRAM hardware supports two mechanisms that could be used for reducing DRAM power consumption, namely, frequency scaling and power-down modes.

Frequency Scaling: All server systems, and even several high-end desktops, support scaling the frequency of memory bus and DRAM devices. Reducing the frequency has a direct impact on power consumption as it reduces the background power caused by transistor leakage and DRAM refresh operations (DRAM cells need to be refreshed at regular intervals to prevent data loss due to charge leakage).

Despite its benefits, frequency scaling is not used today to reduce power consumption. Rather, it is extensively used by gaming enthusiasts who over-clock their PCs to achieve higher perfor-

mance. For instance, DRAM vendors have already released DDR4 devices that can be clocked from 1600 MT/s to well over 3400 MT/s [7]. Server hardware vendors, on the other hand, use this ability to under-clock DRAM frequency to increase memory capacity, as more DIMM slots can be populated in a channel by lowering the frequency as we mentioned earlier.

Power-down modes: As shown in Table 2, modern DRAM devices support a wide-range of power-down states. Under normal conditions, an idle DRAM is in the Active Standby state and can service requests without any penalty. As the DRAM transitions to "deeper" sleep states, both power savings and exit latencies increase presenting an interesting power–performance trade off.

Mode	**Power** (W)	**Latency** (ns)
Active standby	5.36	0
Precharge standby	4.66	14
Active powerdown	3.28	6
Fast exit powerdown	2.79	19.75
Slow exit powerdown	1.60	24
Self Refresh	0.92	768
Self Refresh (reg off)	0.52	6700

Table 2: Table shows various power-down states supported by modern DRAM devices and lists the power consumption and exit latencies associated with each state [1, 14]

Given the high penalty associated with self-refresh power states, they are used only for reducing power consumption during whole-system standby in portable PCs. Modern DDR devices and OSes have been extended explicitly to optimize standby power with extensions like Power-Aware Self Refresh (PASR), where the OS groups memory pages in as few banks as possible and shares this information with hardware, thereby enabling the DRAM to selectively refresh only a few banks.

In servers, MCs minimize the performance impact by adopting a conservative approach towards using the low-power states. First, they monitor ranks for idleness and transition DRAM to a low-power state only if the rank idle time exceeds a configurable threshold. In a few Intel® Xeon® processors, the minimum recommended threshold is as high as fifteen idle memory cycles [11, 14]. Second, MCs limit the transition to one of the Precharge powerdown states (fast or slow exit). A transition to the Self-Refresh state happens only on an explicit request from the Power Control Unit (PCU) (like the standby case) or when the entire package is in a C0 power state.

4. SCALING DRAM POWER WALL

In this section, we will highlight the utility of the two hardware features presented in Section 3 by presenting experimental results that quantify the performance impact and potential power savings observed while using these features to optimize power consumption of a MMDB.

4.1 Experimental Setup

Hardware All our experiments were conducted on a Dell Poweredge r720 server. The server has two sockets, each housing a 2 GHz, eight-core Intel® Xeon® E5-2640 v2 CPU with 32 KB L1, 256 KB L2 and 20 MB shared last-level caches. The server is provisioned with 256 GB of memory, populated using sixteen 16 GB RDIMMs evenly spread across all eight memory channels (two DIMMs per channel, eight per socket). Unless otherwise mentioned, system memory is clocked to run at 1600 MT/s. We disable hyper-threading to minimize interference of threads running on the same core, which introduces variability.

Profiling Methodology We use Intel's Performance Counter Monitor [3] as well as Intel's RAPL (Running Average Power Limit) for profiling the power consumption and performance characteristics of the memory subsystem. Using these tools, we capture the average power for each socket and DRAM rank separately. We also capture the "CKE off" residencies to identify the amount of time DRAM spends in power-down modes, and per-channel DRAM memory bandwidth to identify performance bottlenecks.

Software Stack RHEL 6.5 with Linux kernel version 2.6.32 is used as the base OS in all the experiments. We use HyPer [12] (online demo version) as the MMDB in our analysis. We picked HyPer as it represents a state-of-the-art hybrid MMDB that can support both OLTP and OLAP workloads, and uses several optimization techniques (physical data partitioning, aggressive JIT compilation).

Benchmarks We used TPC-C (SF=100) and TPC-H (SF=10) as our transactional and analytical macrobenchmarks. We chose these scale factors to limit the data size under both benchmarks to 10 GB. In addition, we also used two microbenchmarks to analyze the behaviour of two operations that are common in MMDBs, namely, partitioned scans and parallel aggregations.

Concurrent partitioned scans: In our concurrent partitioned-scan benchmark, (application written in C++), each thread continuously scans 128 MB of 64-bit integers, allocated on the local memory node. Each run lasts 5 seconds and we measure the number of scans performed.

Parallel aggregation: We evaluate the performance of a variant of STREAM[15], a synthetic benchmark that measures sustainable memory bandwidth and computation rate for simple vector kernels. Our variant builds two arrays b and c, and performs the aggregation: $a = \sum(b(i) + c(i))$. Elements are of *double* data type. In our experiments, we use 8 GB arrays. Furthermore, STREAM supports the OpenMP API[18] for parallelism, and we express the operation:

```
#pragma omp parallel for reduction(+:a)
for (i=0; i< STREAM_ARRAY_SIZE; i++)
   a = a + b[i] + c[i];
```

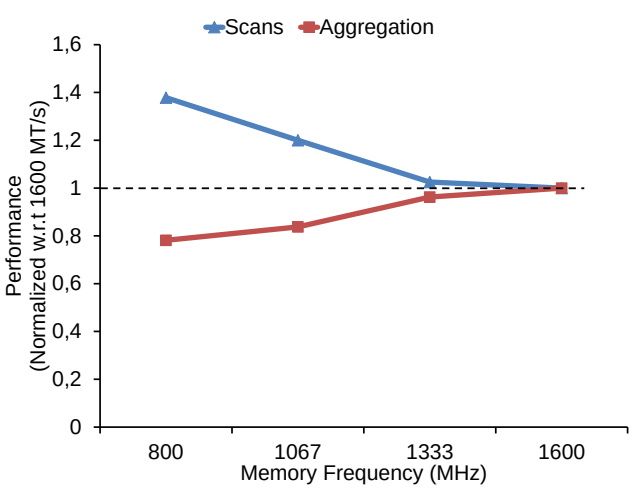
The range is partitioned and distributed among the given number of threads. Thus, we can easily assess the energy efficiency for different levels of parallelism. In order to avoid creating non-uniform memory access (NUMA), we limit the thread count to eight and affinitize all threads to the same socket.

Metric We use performance–power ratio as the primary metric of energy efficiency for comparing different configurations. For TPC-C, we use the transactions completed per second (TPS), as reported by HyPer, as our performance metric. For TPC-H, we use the total execution time (for 22 queries) as reported by HyPer to derive the per second query completion rate, which we use as our performance metric. For scan and aggregation microbenchmarks, we use the throughput reported by the application at completion time as our performance metric.

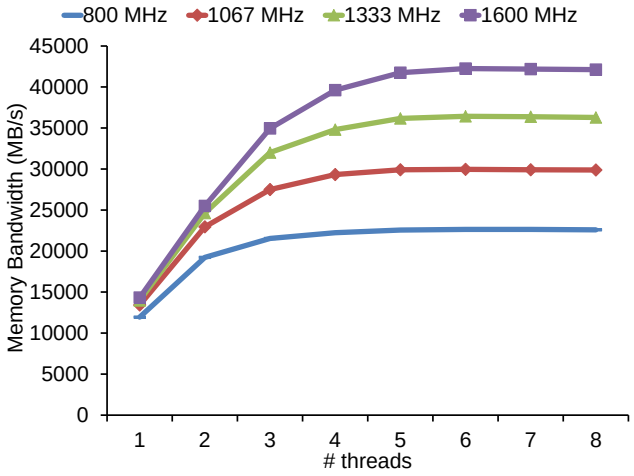
As we will show later, varying the frequency or power-down modes of DRAM has no impact on CPU power consumption. As CPUs always consume a constant amount of power in all our experiments, we use only the DRAM power consumed as our power metric.

4.2 Microbenchmarks

In this section, we present the energy-efficiency impact of var-



(a) Performance/DRAM-Watt for scans and aggregation (8 threads).



(b) Memory bandwidth utilization for parallel aggregation (variable thread count).

Figure 4: Impact of DRAM frequency scaling.

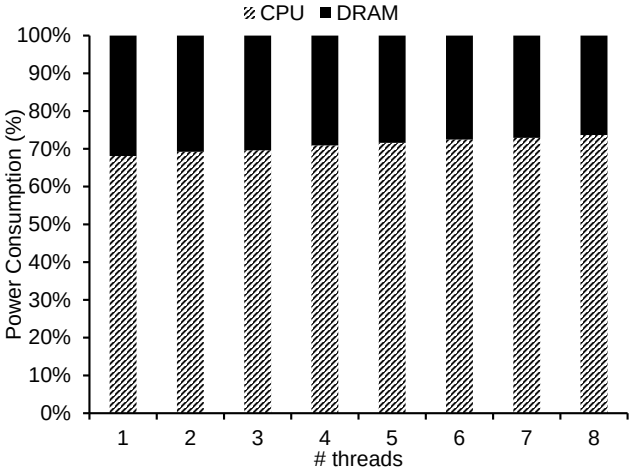


Figure 3: CPU–DRAM power consumption for partitioned scan (variable thread count).

ious DRAM power-saving techniques under our microbechmarks. To isolate the impact of frequency scaling, we fix the DRAM to operate in high-performance mode by disabling transitions to power-down modes in the BIOS. As support for dynamic, fine-grained scaling of DRAM frequencies does not exist today, we ran these experiments by setting the BIOS variable that configures the global frequency of system memory at boot time.

Support for software-driven, application-controlled power-mode transitioning does not exist today. However, Intel® Xeon® processors support hardware-driven power-down transitions, also referred to as "Dynamic CKE", which can be enabled or disabled using a static configuration parameter in the system BIOS. We used this parameter to run two sets of experiments, one with CKE enabled, and the other with CKE disabled, to evaluate the impact of global DRAM power-down on performance and power consumption.

4.2.1 *Power Breakdown*

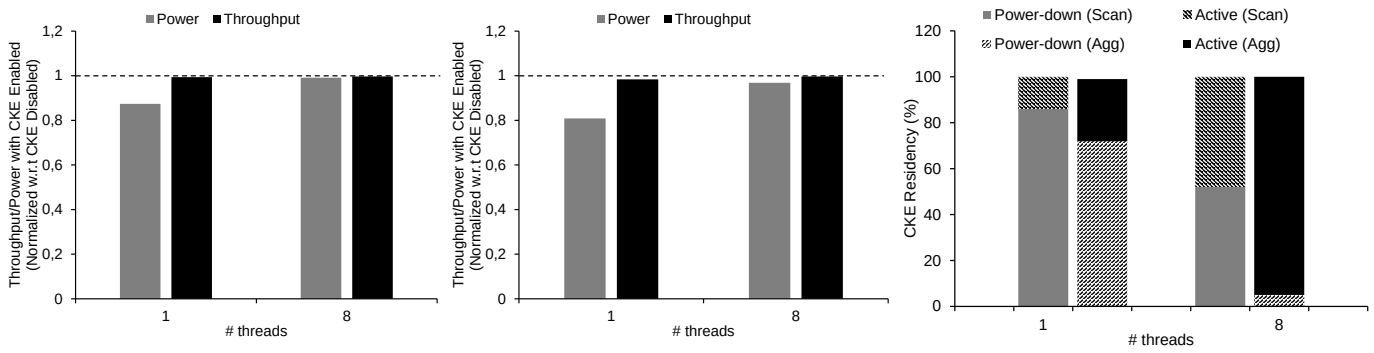
Figure 3 shows the contributions of the CPU package and DRAM to total power consumption under the partitioned-scan microbenchmark at various thread counts. There are two important observations to be made. First, DRAM is responsible for 32% of total power consumption under the single-threaded scan benchmark. This fraction is an optimistic lower-bound on DRAM power consumption as our server is equipped with only 256 GB of memory using energy-efficient RDIMMs. Should we replace the RDIMMs with LRDIMMs and scale into TBs, this fraction would be substantially different. Second, even when we scale up the benchmark, saturating the CPU with eight threads, DRAM continues to contribute a non-negligible 26% to the overall power consumption. Thus, contrary to traditional disk-based database studies [23], CPU is not the only dominating contributor to power consumption in MMDB servers.

4.2.2 *Impact of Frequency Scaling*

Figure 4a shows the performance/DRAM-Watt achieved at various DRAM frequencies under both our microbenchmarks configured to run with eight threads. The values are normalized with respect to the configuration with the highest performance (1600 MT/s). As can be seen, the two benchmarks exhibit different trends, as the most energy-efficient configuration clocks the DRAM at 800 MT/s under the partitioned-scan microbenchmark, and at 1600 MT/s under the parallel-aggregation microbenchmark. This divergence is directly related to the impact of DRAM frequency scaling on memory bandwidth.

Figure 4b shows the observed memory bandwidth under the aggregation benchmark at various thread counts. Clearly, the benchmark saturates memory bandwidth in all configurations, albeit at different thread counts. Thus, the performance of aggregation benchmark deteriorates by over 46% as we scale down from 1600 MT/s to 800 MT/s. This sharp drop in performance is not offset by the 31% drop in DRAM power consumption. Hence, the performance/Watt of the 800 MT/s configuration lags behind that of the fastest configuration by 21%.

Unlike the aggregation benchmark, we observed (not shown) that the peak bandwidth utilization of the scan benchmark was 14 GB/s even with eight parallel threads. As even the 800 MT/s is able to meet this bandwidth requirement, performance drops by just 3% as we scale down from 1600 MT/s to 800 MT/s. Hence, the



(a) Impact of CKE on performance and power under the scan microbenchmark.

(b) Impact of CKE on performance and power under the aggregation microbenchmark.

(c) CKE power-down residency under scan and aggregation benchmarks.

Figure 5: Impact of power-down modes.

31% drop in DRAM power consumption overshadows the negligible decrease in throughput, thereby, making the 800 MT/s the most energy-efficient configuration.

4.2.3 Impact of Low-Power Modes

Figures 5a, 5b show the performance and DRAM power consumption under both microbenchmarks when power-down transitions are enabled (CKE-enabled case) normalized by values corresponding to the CKE-disabled case. There are three important observations to be made.

First, CKE power down modes have negligible impact on performance and power consumption with eight parallel threads under both scan and aggregation benchmarks. This is due to the fact that with eight threads, these microbenchmarks are extremely memory intensive. Thus, as the hardware does not observe enough rank idleness, it prevents DRAM ranks from transitioning into power-down modes. Figure 5c confirms this by showing the amount of time spent by DRAM ranks in power-down modes as reported by pcm (CKE residency). As can be seen, with eight parallel threads, DRAM ranks spend as high as 95% of time in high-power mode.

Second, single-threaded benchmark runs, however, exhibit very different behaviour, as enabling power-down transitions saves 20% power under the scan benchmark, and 13% under the aggregation benchmark. This is due to the fact that under single-threaded benchmarks, most DRAM ranks are able to transition to power-down modes, as shown by the 72%–84% CKE residency in Figure 5c.

Third, despite spending 86% of time in power-down modes, the effective power saving is only 20% under the scan benchmark. Based on this observation, it is clear that the hardware adopts a conservative approach towards power-down transition by limiting DRAM ranks to power-down modes with low exit latency, and hence, low power saving (Table 2).

4.3 Macrobenchmarks

Given the microbenchmark results from the previous section, it is clear that the effectiveness of DRAM power saving techniques varies widely depending on the workload. Thus, in this section, we present the results obtained by using the same power saving techniques to improve the energy efficiency of a MMDB under both transactional and analytical macrobenchmarks.

4.3.1 Impact of Frequency Scaling

Figure 6a demonstrates the performance/DRAM-Watt achieved at various frequencies under both TPC-H and TPC-C benchmarks. Again, the values are normalized with respect to the configuration with the highest performance (1600 MT/s).

Clearly, for both TPC-C and TPC-H, the most energy-efficient configuration is the one with the lowest DRAM frequency (800 MT/s). Under both benchmarks, the 800 MT/s configuration consumes 20% less DRAM power than the 1600 MT/s configuration. Under TPC-H, the 800 MT/s configuration also incurs a 20% drop in performance due to the bandwidth-intensive nature of the benchmark. Thus, the TPC-H performance/Watt curve stays relatively flat as we scale the frequency. Under TPC-C, however, the 800 MT/s configuration incurs only a 7% drop in performance, as TPC-C is latency sensitive and not bandwidth intensive. Thus, the power savings overshadow performance loss, improving energy efficiency by a factor of 1.15.

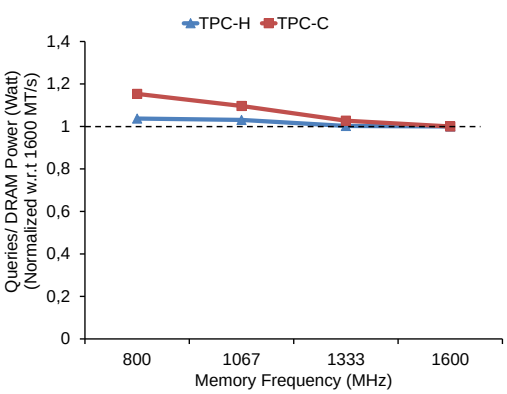
Thus, contrary to disk-based databases [23], the most energy-efficient configuration (800 MT/s) is not the one with the best performance (1600 MT/s).

4.3.2 Impact of Low-Power Modes

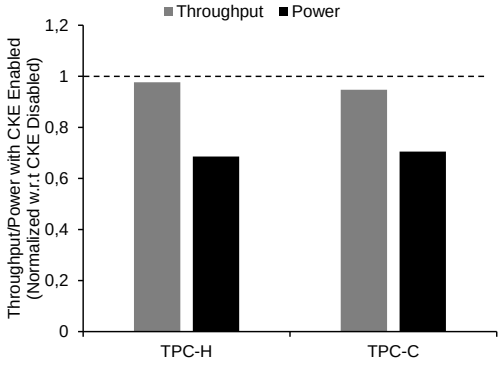
Figure 6b shows both performance and DRAM power consumption of HyPer under both macrobenchmarks, when power-down transitions are enabled (CKE–enabled case) normalized by values corresponding to the CKE–disabled case. Again, contrary to disk-based DB studies, the energy-efficient configuration which powers down idle DRAM to low-power states improves performance/Watt over the high-performance, albeit power-hungry, configuration by a factor of a 1.34 under TPC-C, and a factor of 1.42 under TPC-H. Enabling CKE power down has a negligible 3%–5% impact on the performance of TPC-H and TPC-C benchmarks. However, in both cases, it reduces power consumption by 30%, thereby, improving energy efficiency.

In order to further explain this drop in power consumption, we show the power breakdown between CPUs and DRAM on a per socket basis under the TPC-C benchmark in Figures 6c, 6d. As expected, the CPU power consumption remains unaltered. DRAM power consumption, however, drops by 31% when power-down transitions are enabled. We also observed that the CKE power-down residency of DRAM ranks was 89% under TPC-C. This indicates that even highly-optimized MMDBs like Hyper provide sufficient rank idleness under complex workloads like TPC-C, thereby, enabling energy savings through power-down state transitions.

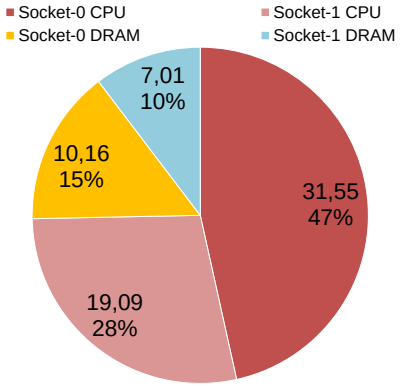
We would like to mention here explicitly that although our benchmarks limit the data size to 10 GB, we believe that power savings reported here will extend to larger data sizes due to two reasons.



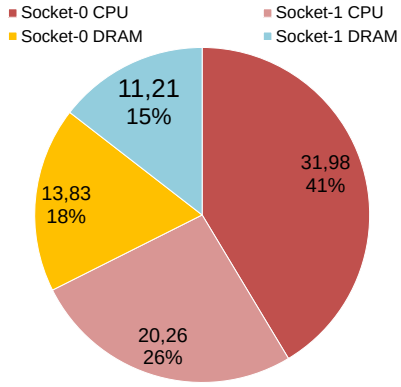
(a) Performance/DRAM-Watt under TPC-C and TPC-H.



(b) Impact of CKE on performance and power under TPC-C and TPC-H.



(c) Power breakdown for CKE enabled.



(d) Power breakdown for CKE disabled.

Figure 6: Impact of frequency scaling and power-down modes on TPC-C and TPC-H.

First, irrespective of dataset size, TPC-C transactions are short lived and touch very few records. Second, TPC-C transactions exhibit data access locality which has been used to improve other design aspects, like partitioning and task scheduling. Nonetheless, as a part of future work, we intend to use detailed memory-system simulators to investigate the sensitivity of these results to variations in both exit latencies and data access localities.

5. TOWARDS ENERGY-EFFICIENT MMDB

The results from Section 4 show how power-down modes and frequency scaling can provide substantial reduction in power consumption even without software support. In this section, we will describe MMDB extensions that we believe can further increase power savings while having minimal impact on performance.

DRAM-aware memory layout Hot–cold data classification has traditionally been used by disk-based database and storage systems to store data in the ideal storage tier depending on its "temperature". MMDBs could use similar classification techniques to identify and separate hot and cold data in different DRAM ranks. With such a classification, existing power-down techniques supported by hardware can be used unmodified to achieve higher power savings, as the hardware would automatically detect higher idle periods for "cold" ranks and transition them to deeper power-down states.

DB-driven gear shifting If the workload exhibits predictable locality, one could further improve performance by essentially breaking up the single DRAM tier into a "hot" tier and a "cold" tier explicitly. With such an approach, the DBMS could use workload information to predict the size of "hot" and "cold" DRAM tiers together with the ideal power states for each tier based on its knowledge of exit latencies. Then, the DBMS could reconfigure hardware and reorganize data to match the target setting.

Such a software-driven gear shifting approach has the ability to both reduce the performance impact and improve power savings at the same time. For instance, the DBMS could reduce performance impact by preventing "hot" ranks from ever transitioning deeper than a Standby state (Table 2) while simultaneously saving power by always keeping "cold" ranks in Self Refresh state.

Multitier memory and storage tiering Recently, MMDBs have used a similar approach to reduce DRAM capacity by pushing out "cold" data to block-based storage media (SSDs or HDDs). This approach, also referred as "Anti-caching" [2], has the disadvantage that access to "cold" data is often two to five orders of magnitude slower than DRAM access depending on the type of "cold" storage device used. Thus, the performance of anti-caching-based MMDB depends on the frequency with which the "cold" tier is accessed.

DRAM operating at deep power-down state essentially adds an additional tier to this two-tier hierarchy. For instance, DRAM operating at the lowest power-down mode (6 µs for Self Refresh state) is one to two orders of magnitude slower than DRAM operating in Active state, and one to two orders of magnitude faster than even state-of-the-art PCIe SSDs (like Micron RealSSD P320h) for reading (60 µs)/writing(1ms) data [22].

Thus, an important research challenge is to identify how an approach like anti-caching should be integrated into a multitier DRAM architecture, and to establish rules of thumb that can determine how

data waterfalls through various tiers.

Tiering-aware query optimization Query optimization for MMDBs continues to be an active area of research in itself. Bifurcation of DRAM into two tiers makes it even more challenging. To our knowledge, existing work on query optimization for improving energy efficiency focuses on using an optimizer to pick a power-optimal query plan [25], or instrumenting the query plan to transition DRAM into power modes [20]. We believe that such an approach might not be ideal as the query plan is statically generated using a cost-model at compile time which might not reflect the runtime execution environment. We believe that the optimizer should not be triggering mode transitions on each query. Rather, the MMDB should use optimizer input to determine the ideal memory layout and tiering configuration upfront. Once such a configuration has been deployed, the optimizer should explicitly take into account the differences in access latencies between DRAM tiers while producing an optimal query plan.

Hybrid workloads Recent research in the architecture community has shown that certain workloads are extremely hard to optimize using power-down modes due to two reasons [4]. First, under workloads that lack locality, there is not enough idleness at the DRAM rank level. Thus, the memory controller's idle-rank threshold is never exceeded and most ranks spend their time in Active Standby mode consuming a large amount of power. Second, forcing a transition to low-power states under such workloads results in a huge impact on performance as exit latencies in today's DRAM devices are too high.

This might also apply to latency-sensitive database workloads (OLTP) that do not have predictable locality and it is important to investigate if such workloads might experience a much higher drop in performance if deep power-down states (like Self Refresh) are used.

Frequency scaling provides an alternative for such workloads as it has the effect of reducing memory channel bandwidth without significantly increasing latency. However, frequency scaling might have a significant impact on the performance of bandwidth-sensitive analytical workloads. Thus, optimizing the energy efficiency of MMDBs requires pairing workloads with the ideal power saving mechanism. Such pairing can be achieved easily if a MMDB is used to service only one type of workload. Unfortunately, several state-of-the-art MMDBs today support both OLTP and OLAP workloads [6, 12]. Thus, further research is required to understand the pros and cons of using both these techniques in tandem.

CPU-DRAM synergy Given the dominance of CPU power consumption in traditional databases, there has been a lot of work in literature focusing on using CPU power states (C-states/P-states) and DVFS in combination with task/operator scheduling to reduce power consumption [13, 21, 24]. Given that modern DRAM also supports similar features, it is important to understand the synergy between CPU and DRAM techniques.

6. CONCLUSIONS

Today, there is no single server DRAM type that can meet the performance and power requirements of main-memory databases. While energy analysis of traditional databases has shown CPUs to be the dominating source of power consumption, given the capacity–performance–power tradeoffs we showed in this paper, we believe that DRAM will soon eclipse CPUs power consumption in emerging MMDBs. Thus, we believe that the oft-ignored DRAM power consumption should become the focus of further research on energy efficiency of MMDBs.

In this paper, we take the first steps towards this goal by 1) identifying hardware features that help in reducing DRAM power consumption, and 2) performing an experimental analysis of the impact of these features on power and performance of a state-of-the-art MMDB under both transactional and analytical workloads. Our results indicate that even without any software support, these features can provide, in the best case, 2× improvement in performance/Watt. Based on these results, we outlined several promising research directions that must be explored in order to identify an optimal design for energy-efficient MMDBs.

Acknowledgements

We would like to thank the reviewers for their invaluable feedback and members of the DIAS lab for their support. This work has been partially funded by the EU FP7 project BigFoot (Grant No 317858)

7. REFERENCES

[1] H. David, C. Fallin, E. Gorbatov, U. R. Hanebutte, and O. Mutlu. Memory Power Management via Dynamic Voltage/Frequency Scaling. In *ICAC*, 2011.

[2] J. DeBrabant, A. Pavlo, S. Tu, M. Stonebraker, and S. Zdonik. Anti-caching: A new approach to database management system architecture. *VLDB Endow.*, 6(14):1942–1953, 2013.

[3] R. Dementiev, T. Willhalm, O. Bruggeman, P. Fay, P. Ungerer, A. Ott, P. Lu, J. Harris, P. Kerly, and P. Konsor. Intel performance counter monitor 2.0, 2012. http://www.intel.com/software/pcm.

[4] Q. Deng, D. Meisner, L. Ramos, T. F. Wenisch, and R. Bianchini. MemScale: Active Low-power Modes for Main Memory. In *ASPLOS*, 2011.

[5] C. Diaconu, C. Freedman, E. Ismert, P.-A. Larson, P. Mittal, R. Stonecipher, N. Verma, and M. Zwilling. Hekaton: SQL Server's Memory-optimized OLTP Engine. In *SIGMOD*, pages 1243–1254, 2013.

[6] F. Färber, S. K. Cha, J. Primsch, C. Bornhövd, S. Sigg, and W. Lehner. SAP HANA Database: Data Management for Modern Business Applications. *SIGMOD Record*, 2012.

[7] G.Skill. DDR4 SDRAM 3200. `http://www.newegg.com/Product/Product.aspx?Item=N82E16820231808`.

[8] HP. Power advisor. Available at `http://www8.hp.com/us/en/products/servers/solutions.html?compURI=1439951#.VRFbY4WRDFU`.

[9] HP. Server buying guide. Available at `http://www8.hp.com/us/en/prodserv/serverbuyingguide/overview.html`.

[10] HP. Configuring and using ddr3 memory with hp proliant gen8 servers, 2012.

[11] Intel. Intel xeon processor e3-1200 family datasheet, 2011.

[12] A. Kemper and T. Neumann. HyPer: A hybrid OLTP&OLAP main memory database system based on virtual memory snapshots. In *ICDE*, pages 195–206, 2011.

[13] W. Lang, R. Kandhan, and J. M. Patel. Rethinking query processing for energy efficiency: Slowing down to win the race. *IEEE DEBull*, 34(1):12–23, 2011.

[14] K. T. Malladi, I. Shaeffer, L. Gopalakrishnan, D. Lo, B. C. Lee, and M. Horowitz. Rethinking dram power modes for energy proportionality. In *MICRO*, pages 131–142, 2012.

[15] J. D. McCalpin. Memory bandwidth and machine balance in current high performance computers. *IEEE CS TCCA Newsletter*, pages 19–25, Dec. 1995.

[16] J. Meza, M. A. Shah, P. Ranganathan, M. Fitzner, and J. Veazey. Tracking the Power in an Enterprise Decision Support System. In *ISLPED*, pages 261–266, 2009.

[17] Netlist. HyperCloud HCDIMM: Scaling the High Density Memory Cliff, 2012. Available at http://www.netlist.com/media/blog/hypercloud-memory-scaling-the-high-density-memory-cliff/.

[18] OpenMP Architecture Review Board. OpenMP application program interface version 3.0, May 2008. http://www.openmp.org/mp-documents/spec30.pdf.

[19] Oracle. Oracle TimesTen In-Memory Database. Available at https://www.oracle.com.

[20] J. Pisharath, A. Choudhary, and M. Kandemir. Reducing Energy Consumption of Queries in Memory-resident Database Systems. In *CASES*, pages 35–45, 2004.

[21] I. Psaroudakis, T. Kissinger, D. Porobic, T. Ilsche, E. Liarou, P. Tözün, A. Ailamaki, and W. Lehner. Dynamic Fine-grained Scheduling for Energy-efficient Main-memory Queries. In *DAMON*, 2014.

[22] S. Review. Micron RealSSD P320h Enterprise PCIe Review. http://www.storagereview.com/micron_realssd_p320h_enterprise_pcie_review.

[23] D. Tsirogiannis, S. Harizopoulos, and M. A. Shah. Analyzing the energy efficiency of a database server. In *SIGMOD*, 2010.

[24] Y.-C. Tu, X. Wang, B. Zeng, and Z. Xu. A System for Energy-efficient Data Management. *SIGMOD Record*, 2014.

[25] Z. Xu, Y.-C. Tu, and X. Wang. Exploring power-performance tradeoffs in database systems. In *ICDE*, pages 485–496, 2010.