A Survey of Architectural Techniques for DRAM Power Management

Sparsh Mittal

Electrical and Computer Engineering
Iowa State University
Iowa, USA, 50014
Email: sparsh@iastate.edu

Abstract:

Recent trends of CMOS technology scaling and wide-spread use of multicore processors have dramatically increased the power consumption of main memory. It has been estimated that modern data-centers spend more than 30% of their total power consumption in main memory alone. This excessive power dissipation has created the problem of "memory power wall" which has emerged as a major design constraint inhibiting further performance scaling. Recently, several techniques have been proposed to address this issue. The focus of this paper is to survey several architectural techniques designed for improving power efficiency of main memory systems, specifically DRAM systems. To help the reader in gaining insights into the similarities and differences between the techniques, this paper also presents a classification of the techniques on the basis of their characteristics. The aim of the paper is to equip the engineers and architects with knowledge of the state of the art DRAM power saving techniques and motivate them to design novel solutions for addressing the challenges presented by the memory power wall problem.

Keywords: Architectural techniques, power efficiency, energy saving, DRAM, main memory, survey, review, classification

Reference to this paper should be made as follows: Mittal, S. 'A Survey of Architectural Techniques for DRAM Power Management', Int. J. of High Performance System Architecture, Vol. 4, Nos. 2, pp. 110-119

Biographical notes: Sparsh Mittal received his B. Tech. degree in Electronics and Communications Engineering from the Indian Institute of Technology, Roorkee, India. He was the graduating topper of his batch and his major project was awarded Institute Silver medal. At present, he is pursuing his PhD degree in Electrical and Computer Engineering at Iowa State University, USA. He has been awarded scholarship and fellowship from IIT Roorkee and ISU. His research interests include memory system power efficiency, cache architectures in multicore systems and real-time systems.

.

1 Introduction

Recent years have witnessed a dramatic increase in power consumption of computing systems. As an example, in year 2006 alone, the data-centers and servers in U.S. consumed 61 billion kilowatt hours (kWh) of electricity [1]. Further, the energy consumption of main memory is becoming an increasing fraction of total energy consumption of the system, in processors ranging from low-end to high-end. It has been estimated that as much as 40% of the total power consumed in smartphones and datacenters is attributed to the memory system [2–4]. Studies conducted on real server systems show that memory system consumes as much as 50% more power than the processor cores [3].

There are several architectural and technological trends that mandate the use of high amount of main memory resources and thus, contribute to the increase in memory power consumption. Firstly, in modern processors, the number of cores on a single chip is on rise [5, 6] and hence, the pressure on the memory system has been increasing. Secondly, as we move to the exascale era, the requirements of data storage and computation are growing exponentially [7–10] and to fully optimize the value of such data, modern processors are using main memory (as opposed to persistent storage) as the primary data storage for critical applications. Thirdly, modern data-centers exhibit low average utilization [2], but frequent, brief bursts of activity, and thus, to meet

the requirements of service-level-agreements (SLAs), operators are forced to provision high amount of main memory resources, suitable to meet the worst case requirement. To cater to the above mentioned demands, modern processors are using main memory with high bandwidth, frequency and capacity. Finally, CMOS technology scaling has enabled higher transistor packing density, which have further exacerbated the problems of power consumption and heating and have also inhibited the effectiveness of cooling solutions. Thus, improving the power efficiency of memory systems has become extremely important to continue to scale performance [8] and also achieve the goals of sustainable computing.

Recently, many techniques have been proposed for optimizing the power efficiency of memory systems. In this paper, we review several of these techniques. As it is practically infeasible to review all the techniques proposed in the literature, we take the following approach to limit the scope of the paper. We only include techniques proposed for saving energy in DRAM systems and do not discuss other emerging storage technologies (e.g. phase change memory). We consider DRAM, since it has been traditionally used as main memory because of its properties such as high density, high capacity, low cost and device standardization etc. [11]. Also, we focus on *architecture-level* techniques which allow runtime power management and do not discuss *circuit-level* innovations for reducing power consumption. Further, although the techniques aimed at improving memory performance (e.g. reducing latency) are also likely to improve memory power efficiency, we only include those techniques that have been *evaluated* for improving memory power efficiency. Finally, since different techniques have been evaluated using different experimentation methodologies, we do not present their quantitative results; rather, we only discuss the key ideas of those techniques.

The remainder of the paper is organized as follows. In section 2, we briefly discuss the terminology used in DRAM systems and the sources of power consumption in them. Understanding the sources of power consumption also helps in gaining insights into the opportunities available for improving power efficiency. In section 3, we present a classification of the techniques proposed for managing DRAM power consumption to highlight the similarities and differences among them. A more detailed discussion of these techniques is provided in section 4. Finally section 5 provides the concluding remarks.

2 Background

In this section, we briefly discuss DRAM terminology [12] and the sources of power consumption in DRAM systems, to aid in discussion of DRAM power management techniques discussed in the next sections.

2.1 DRAM terminology

In DRAM terminology, a *column* is the smallest addressable portion of the DRAM device and a *row* is a group of bits in the DRAM array that are sensed together at the instance of receiving an activate signal. A DRAM *bank* is an array of DRAM cells, which can be active independently and has same data bus width as external output bus width. A DRAM *rank* is a group of DRAM devices which operate together to service requests from the memory controller. A *DIMM* (Dual in-line memory module) is a printed circuit board containing one or more DRAM ranks on it, which provides an interface to the memory bus. A DRAM *channel* is a group of one or more DIMMs of DRAM that handle requests from the memory controller. An typical DRAM can have 2 channels, 2 DIMMs per channel, 2 ranks per DIMM, 8 banks per rank, for a total of 64 ($= 2 \times 2 \times 2 \times 8$) banks.

2.2 Sources of Power Consumption

The power consumption in DRAM memory is broadly classified in three categories, namely activation power, read/write power, background power (for a more detailed analysis, see [11, 13, 14]). Activation power refers to the power dissipated in activating a memory array row and in precharging the arrays bitlines. The read/write power refers to the power which is consumed when the data moves either into or out of the memory device. The background power is independent of the DRAM access activity and is due to the transistor leakage, peripheral circuitry, and the data refresh operations. Note that the DRAM memory cells store data using capacitors that lose their charge over time and must be periodically recharged; this is referred to as data refreshing.

3 A Classification of DRAM Power Management Techniques

In this section, we present a classification of the DRAM power management techniques based on their characteristics.

As shown by Barroso et al. [2], modern servers operate most of the time between 10% and 50% of maximum utilization. Thus, considerable opportunities exist to transition idle inactive memory banks into low-power modes. These power modes could be either state-preserving modes (i.e. the data is retained) or state-destroying modes (i.e. the data is not retained). For this purpose, DRAM chips provision several modes of operation. Each mode is characterized by its power consumption and the time that it takes to transition back to the active mode. This time is referred to as resynchronization latency or exit latency. Typically, the modes with lower energy consumption also have higher reactivation time and vice versa. Also, a DRAM module may enter a low-power state-preserving mode when it is idle, but must return to the active mode to

service a request. A large number of techniques have been proposed which utilize the adaptive power saving capability offered by the modern multi-banked memory systems [4, 15–59].

Some techniques perform memory access redistribution (also called memory traffic reshaping), which involves changing the address mapping in DRAM, migrating data within DRAM etc., for increasing the idle period of certain memory banks or increasing memory reference locality [15, 17, 24, 34, 38–40, 48–50, 60–64].

Several other techniques reduce the power consumed in each DRAM access by accessing only part of DRAM in each memory access [11, 54, 57, 65–68]. Thus, these techniques provision activating a much smaller portion of DRAM circuit component than what is activated in conventional DRAMs.

Many techniques for reducing memory power consumption use mechanisms based on memory access scheduling [15, 32, 44, 69–76]. This includes mechanisms such as memory access throttling or buffering or coalescing. Also, techniques have been proposed which reduce the number of accesses to memory by smart management of last level caches or program level transformations etc. [15, 77–79].

Some techniques use data compression to reduce the memory footprint of the application [49, 80–82]. This helps in reducing the number of memory banks occupied by application data. The techniques based on data replication increase idle time duration of memory banks by duplicating their selected read-only data blocks on other active banks [49].

Like other CMOS circuits, DRAM power also depends on operating frequency and supply voltage and hence DVFS (dynamic voltage/frequency scaling) mechanism has been used in several techniques to save memory power [29, 64, 70, 83]. Some techniques reduce the DRAM refresh power, by trading off either performance or reliability [43, 78, 84–89]. Some techniques account for the effect of temperature while optimizing for memory energy [17, 70, 90–92]. Such techniques are referred to as thermal-aware memory energy saving techniques.

Also, while most of the techniques work to *reduce* total power consumption of DRAM system, a few techniques work to *limit* their average power consumption [23], while some other techniques work to limit their peak power consumption [22, 30].

4 DRAM Power management techniques

In this section, we review several DRAM power saving techniques. As discussed before, we only present the key ideas of each technique and do not discuss their qualitative results.

Lebeck et al. [40] propose a technique for turning off DRAM chips in low-power mode. Their technique works by controlling virtual address to physical address mapping such that the physical pages of an application are clustered into a minimum number of DRAM chips and the unused chips are transitioned to low power modes. In addition, their technique also monitors the time period between accesses to a chip as a metric for measuring the frequency of reuse of a chip. When this time is greater than a certain threshold, the chip is transitioned to the low-power mode.

Fan et al. [31] present an analytical model to approximate the idle time of memory chips. Based on this, they identify the threshold time, after which the memory chip can transitioned to low-power state. They observe that, for their experimentation framework, the simple policy of immediately transitioning a DRAM chip to a low-power mode when it becomes idle, performs better than more sophisticated policies that predict DRAM chip idle time. Li et al. [42] propose a technique for saving memory energy which adaptively transitions memory module to low-power modes while also providing guarantees on the maximum performance loss.

Delaluz et al. [26] propose software and hardware based approaches to save memory energy. Their hardware based approach works by estimating the time of next access to a memory bank and then, depending upon the time, switching the bank to a suitable low-power mode. The software-directed approach uses compiler analysis to insert memory module transition instructions in the program binary. To avoid the time overhead of resynchronization, they propose bringing the memory bank to active mode before its next use. Depending upon the break-even analysis of length of idle time and power saving opportunity of different power saving modes, a suitable power saving mode is chosen. Delaluz and Sivasubramaniam et al. [27] describe an OS scheduler-based power mode control scheme for saving DRAM energy. Their scheme tracks memory banks that are being used by different applications and selectively turns on/off these banks at context switch points.

A limitation of the approaches based on power mode control is that most of the idle times between different accesses to memory ranks is shorter than the resynchronization time between different power modes. To address this issue, Huang and Shin et al. [34] propose a method for saving memory energy by concentrating the memory access activities to merely few memory ranks, such that rest of the ranks can be switched to low-power modes. Their method migrates the frequently-accessed pages to "hot" ranks and the infrequently-used and unmapped pages to "cold" ranks. This also helps in elongating the idle periods of cold ranks.

Delaluz and Kandemir et al. [24] propose a technique to save memory energy by dynamically placing the arrays with temporal affinity into the same set of banks. This increases the opportunities of exploiting deeper sleep modes (more energy-saving operating modes) and keeping modules in low-power modes for longer durations of time. Using the same principle, they also propose an array interleaving mechanism [25] for clustering multiple arrays, which are accessed simultaneously, into a single common data space. Interleaving enhances

spatial locality of the program and reduces the number of accesses to the off-chip memory. Along with interleaving the arrays, their mechanism also transforms the code accordingly by replacing the original array references and declarations with their transformed equivalents.

Huang and Pillai et al. [33] propose a technique for saving memory energy using virtual memory management. Their technique works by using virtual memory remapping to reduce the memory footprint of each application and transitioning the unused memory modules to low-power modes. Zhou et al. [59] discuss a utility-based memory allocation scheme where different applications are allocated memory in proportion to their utility (i.e. the performance benefit gained by allocation of the memory). After allocation, the rest of the memory is transitioned to low-power modes for saving power. For estimating the utility of allocating memory to different applications, their scheme dynamically tracks page miss-rate curve (MRC) for virtual memory system using either hardware or software methods. Lyuh et al. [44] propose a technique which uses analytical model for saving memory power. Their technique selects a suitable low-power mode for a memory bank by synergistically controlling assignment of variables to memory banks and scheduling of memory access operations, such that total memory power consumption is minimized.

Bi et al. [19] propose methods to hide the latency of resynchronization of memory ranks to low-power modes by exploiting the knowledge of system input/output (I/O) calls. Their technique works on the observation that a majority of file-I/O accesses are made through system calls, the operating system knows the completion time of these accesses. Thus, using this knowledge, their technique transitions idle memory ranks into low-power modes. Further, to hide the resynchronization delay, their technique uses prediction mechanism to estimate the most likely rank to be accessed on a system call entry and speculatively turns on that rank. On a correct prediction, the rank transition completes before the memory request arrives and thus, the resynchronization latency is fully hidden.

Pandey et al. [50] propose a technique for saving energy in DMA (direct memory access) transfers. Since DMA transfers are usually larger than the transfers initiated by the processors, they are divided into multiple number of smaller transfer operations. However, due to the availability of only short time gaps between any two DMA-memory requests, the opportunity of transitioning the memory to low-power mode remains small. To address this, Pandey et al. propose temporally aligning DMA requests coming from different I/O buses to the same memory device. For this purpose, their technique delays DMA-memory requests directed to a memory chip which is in low-power mode and tries to gather enough requests from other I/O buses before transitioning that chip to the normal power mode. This helps in elongating the idle time of memory chips and also maximizes the utilization of active time of memory chips.

Koc et al. [39] discuss a data-recomputation approach for increasing idle time of memory banks to save their energy. When an access to a bank in low-power mode is made, their technique first checks the active banks. If the requested data can be recomputed by using the data stored in already active banks, their technique does not activate the bank in low-power mode. Rather, the data request is fulfilled based on the computations performed on the data obtained from the already active banks.

To reduce the refresh power of the DRAM devices, Ghosh et al. [84] propose adaptive refresh method. The conventional refresh mechanism of DRAM periodically refreshes all the memory rows for retaining the data. However, from the standpoint of data retention, an access to a memory row performs an operation equivalent to a regular refresh. The technique proposed by Ghosh et al. [84] uses this observation to avoid refreshing a memory row, if it has been recently read out or written to by the processor. To track the recency of memory access operation, they use counters for each row in the memory module. Using this technique, the number of regular row-sweeping refresh operations are greatly reduced, which results in saving of power.

J. Liu et al. [85] propose a technique for saving DRAM energy by avoiding unbeneficial refreshes. Their technique works on the observation that in the DRAM, only a small number of cells need to be refreshed at the minimum conservative refresh rate. The rest of the cells can be refreshed at a much higher rate, while still maintaining their charge. Based on this observation, their technique groups DRAM rows in multiple bins and uses different refresh interval for different bins. Thus, by refreshing most of the cells less frequently than the leaky cells, their technique reduces the number of refresh operations required and reduces memory power consumption.

Isen et al. [78] discuss a technique for utilizing program semantics to save memory energy. Their technique uses memory allocation/deallocation information to identify inconsequential data and avoids refreshing them. For example, the regions of memory which are free (unallocated and invalid) or freshly allocated (allocated but invalid) do not store meaningful data and hence, retaining the data of those regions is not important. Thus, their technique saves power by avoiding refreshing such data.

S. Liu et al. [43] propose an application level technique to reduce refresh level power in DRAM memories. They show that many applications are tolerant to errors in the non-critical data, and errors in non-critical data show little or no impact in the application's final result. Based on this observation, their technique works by using programmer supplied information to identify critical and non-critical data in the programs. Using this information, at runtime, these data are allocated in different modules of the memory. The memory modules containing critical data are refreshed at the regular refresh-rate, while the modules containing non-critical data are refreshed at

substantially lower rates. The use of lower refresh rates leads to saving in refresh power, however, it also increases the probability of data corruption. Thus, their technique exercises a trade-off between energy saving and data corruption.

Sudan et al. [63] propose a technique for saving memory power by using OS management approach. Their technique works by controlling the address mapping of OS pages to DRAM devices such that the clusters of cache blocks from different OS pages, which have similar access counts are colocated in a row-buffer. This improves the hit rate of the row-buffer and thus leads to saving of memory power. For co-locating pages, Sudan et al. propose two techniques. One of their technique reduces OS page size such that the frequently accessed blocks are clustered together in the new, reduced size page (called a "micro-page"). Then, the hot micro-pages are migrated in the same row-buffer. The second technique proposed by them uses a hardware scheme. This scheme introduces a layer of translation between physical addresses assigned by the OS and those used by the memory controller to access the DRAM devices. By taking advantage of this layer of mapping, hot pages are migrated in the same row-buffer.

Trajkovic et al. [73] propose a buffering based technique for reducing memory power consumption. Their technique works on the observation that if in a synchronous DRAM, two memory access (i.e. read/write) operation are done in a same activate-precharge cycle, the cost of activation and precharging can be avoided. This is because, the DRAMs allow the row to be left 'on' after a memory access. Based on this observation, on read accesses, their technique prefetches additional cache blocks. Similarly, for write accesses, combines multiple blocks which are to be written to the same DRAM row. To store the extra prefetched lines, their technique uses a small storage structure in the memory controller. Similarly, to buffer the writes to the same DRAM row also, a small storage structure is used. By adapting the above mentioned prefetching and write-combining scheme for each application, their technique achieves reduction in memory power consumption.

Zheng et al. [57] propose a technique for saving memory power consumption by reducing the number of memory chips involved in each memory access. This is referred to as "rank-subsetting" approach. Their technique adds a small buffer called "mini-rank buffer" between each DIMM and the memory bus. Using this, a DRAM rank, which normally provides 64-bit datapath, can be internally designed using either eight 8-bit ranks, or four 16-bit ranks or two 32-bit ranks, which are termed as mini-rank. With this support, on any memory access, only a single mini-rank is activated and the other mini-ranks can be transitioned to low-power modes.

Fang et al. [66] extend mini-rank approach to heterogeneous mini-rank design which adapts the number of mini-ranks according to the memory access behavior and memory bandwidth requirement of each workload. Based on this information, for a latency-sensitive application, their technique uses a mini-rank configuration which does not degrade application performance; while for a latency-insensitive application, their technique uses a mini-rank configuration which achieves memory power saving.

Yoon et al. [75] propose a technique for saving memory power consumption by intelligently utilizing low-power mobile DRAM components. Their technique uses buffering mechanism to aggregate the data outputs from multiple ranks of low frequency mobile DRAM devices (e.g. 400MHz LPDDR2), to collectively provide high bandwidth and high storage capacity equal to server-class DRAM devices (e.g. 1600MHz DDR3).

Yoon and Jeong et al. [74] propose a technique for saving memory power by dynamically changing the granularity of data transferred in each DRAM access. Their technique works by managing virtual memory such that a specific access granularity can be used for each page based on the spatial locality present in each application. For applications with high spatial locality, their technique uses coarse-grained data accesses, while for applications with low spatial locality their technique uses fine-grained data accesses.

Several researchers have proposed techniques which use DVFS mechanism to save memory energy. Deng et al. [28, 29] use memory DVFS to save memory energy. At the time of low memory activity, their technique lowers the frequency of DRAM devices, memory channels and memory controllers such that the performance loss is minimum. This leads to saving of memory power consumption. They also extend their technique for coordinating DVFS across multiple memory controllers, memory channels, and memory devices to minimize the overall system power consumption.

Diniz et. al. [30] propose a technique to limit the *instantaneous* (peak) power consumption of main memory to a pre-specified power budget. Their technique uses knapsack and greedy algorithms to decide the timings at which memory devices should be transitioned to suitable low-power modes such that the instantaneous power of memory is always within the power budget. David et al. [23] present a scheme for limiting the *average* power consumption of memory, by suitably transitioning memory devices into low-power modes. Chen et al. [22] propose a method for limiting the *peak* power consumption of the server (which includes power consumption of processor and main memory) system using control theoretic approach.

Amin et al. [15] propose a replacement policy for last-level cache, which tries to increase the idle time of certain pre-chosen DRAM ranks, called "prioritized ranks". This replacement policy tries to prevent the replacement of blocks belonging to the prioritized ranks. This reduces the conflict misses and writebacks to the prioritized rank, increasing the idle period between accesses made to those ranks. Amin et al. also propose a technique which buffers writeback requests sent to DRAM, to increase the idle period of the DRAM ranks. The requests are buffered as long as the target ranks remain idle or the buffer remains

full. When the targeted ranks become active (due to the demand misses), the buffered requests are sent to them.

Ozturk et al. [48] present a bank-aware cache miss clustering approach for saving DRAM energy. Their technique uses compiler analysis to restructure the code such that the cache misses from the last-level cache are clustered together. Clustering of the cache misses also leads to the clustering of cache hits. Thus, the memory accesses and memory idle cycles are also clustered. This increases the memory access activities in certain banks and the other banks become idle for a long time. By taking advantage of this, idle memory banks are transitioned to low-power modes.

As the computational requirements of state-of-the-art applications is increasing [93], the pressure on memory systems is also on rise and to mitigate this pressure, researchers have proposed techniques to intelligently manage the last level caches (LLCs) in the processors. Mazumdar et al. [79] propose a technique for reducing the number of memory accesses in multicore systems by cache aggregation approach. Their technique works on the observation that due to the availability of high-bandwidth point-to-point interconnects between sockets, a read from the LLC of a connected chip consumes less time and energy than an access to DRAM. Based on this, their technique uses the LLC of an idle processor in a connected socket for holding the evicted data from the active processor. This reduces the number of accesses to DRAM and thus reduces the power consumption of DRAM.

Phadke et al. [88] propose a heterogeneous main memory architecture which comprises of three different memory modules. Each memory module is optimized for latency, bandwidth, power consumption, respectively, at the expense of the other two. Their technique works by using offline analysis to characterize an application based on its LLC (last level cache) miss rate and memory level parallelism. Using this information, at runtime, the operating system allocates the pages of an application in one of the three memory modules that satisfies its memory requirements. Thus, their approach saves memory energy and also improves performance of the system.

Yang et al. [82] discuss a software-based RAM compression technique for saving power in embedded systems. Their technique uses memory compression only for those applications which may gain performance or energy benefits from compression. For such applications, their technique performs compression of memory data and swapped-out pages in online manner, thus dynamically adjusting the size of the compressed RAM area. Thus, their technique saves power by using compression to increase the effective size of the memory.

Ozturk et al. [49] integrate different approaches such as dynamic data migration, data compression, and data replication etc. to effectively transition a large number of memory banks into low-power modes. They formulate DRAM energy minimization problem as a integer linear programming (ILP) problem and solve it using an ILP solver. Using ILP formulation, they find the (nonuniform) bank architecture and accompanying data mapping strategy which best suits the application-data access patterns. Similarly, they use ILP formulation to find best possible data replication scheme which increases idle time of certain banks by duplicating their selected read-only data blocks on other active banks. They also use ILP formulation to find the best time to compress and/or migrate the data between banks.

Several researchers have used domain-specific optimizations to save DRAM power. Kim et al. [61] and Li et al. [62] propose techniques for reducing DRAM power consumption in video processing domain. Video processing applications are characterized by abundant spatial and temporal image data correlations, and unbalanced accesses to frames (e.g. certain image frames are accessed much more frequently than other image frames). Hence, to take advantage of these properties, their techniques map image data in DRAM in a way which minimizes the number of row-activations. Thus, the power consumption of DRAM is reduced.

Chen et al. [21] propose a technique for tuning the garbage collector (GC) in Java to reduce memory power consumption. GC is an tool used in Java virtual machine (JVM) for automatic reclamation of unused memory. Chen et al. propose using GC to turn off the memory banks that do not hold live data. They also observe that the pattern of object allocation and the number of memory banks available in the DRAM architecture crucially influence the effectiveness of GC in optimizing energy.

Pisharath et al. [51] propose an approach to reduce memory power consumption in memory-resident database management systems (DBMS). One of their techniques uses hardware monitors to detect the frequency of use of memory banks during query execution and based on this, switches the idle banks into low-power mode. Another technique uses a software approach. For DBMS systems, when the query is submitted, it is first parsed and then sent to the query optimizer which uses query tree to find the best suited plan for execution of the query [51]. At this point, query optimizer finds the database tables which will be accessed to answer the query. Based on this information, their technique changes the table-to-bank mapping such that memory accesses can be clustered. Also, the queries presented to the database are augmented with explicit bank turn off or turn on instructions. Using this support, at runtime, the memory banks are dynamically transitioned into low-power mode.

Since leakage (static) power varies exponentially with the temperature, the dissipation of power in DRAM leads to increase of device temperature, which further increases the leakage power dissipation. This may lead to thermal emergencies. Also, many of the above mentioned approaches move or map frequently accessed pages to merely a few active memory modules. This is also likely to increase the temperature of the active modules. To address this, Ayoub et al. [17] propose a technique

which monitors the temperature of the active modules. When the temperature reaches a threshold, it selectively migrates a small number of memory pages between active and dormant memory modules and transitions the active modules in the self-refresh mode. Since this approach spreads out the memory accesses to multiple modules, it reduces the power density of the active modules and thus avoids thermal emergencies.

C. Lin et al. [90] propose a technique for addressing memory thermal issues which works by orchestrating thread scheduling and page allocation. Their technique groups the program threads in multiple groups such that all the threads in a group can be active simultaneously. Then each group is mapped to certain DIMMs and at any time, only one group and its corresponding DIMMs remain active and the rest of the DIMMs are inactivated to reduce their temperature. Similarly, J. Lin et al. [70, 91] propose techniques to mitigate overheating in the memory system by adjusting memory throughput to stay below the emergency level.

5 Conclusion

Recent advances in CMOS fabrication and chip design have greatly increased the power consumption of main memory in modern computing systems. To provide a solution to this problem, several research efforts have been directed towards managing the power consumption of main memory. In this paper, we surveyed several architectural techniques which are designed for improving DRAM memory power efficiency. We also presented a classification of the proposed techniques across several parameters, to highlight their similarities and differences. We believe that this survey will help researchers and designers to understand the state of the art in approaches pursued for reducing memory power consumption. At the same time, it will also encourage them to design innovative solutions for memory systems of future green computing infrastructure.

References

[1] R. Brown, E. Masanet, B. Nordman, B. Tschudi, A. Shehabi, J. Stanley, J. Koomey, D. Sartor, P. Chan, J. Loper, *et al.*, "Report to congress on server and data center energy efficiency," *Public law 109-431*, 2007.

[2] L. Barroso and U. Hölzle, "The datacenter as a computer: An introduction to the design of warehouse-scale machines," *Synthesis Lectures on Computer Architecture*, vol. 4, no. 1, pp. 1–108, 2009.

[3] C. Lefurgy, K. Rajamani, F. Rawson, W. Felter, M. Kistler, and T. Keller, "Energy management for commercial servers," *Computer*, vol. 36, no. 12, pp. 39–48, 2003.

[4] M. Ware, K. Rajamani, M. Floyd, B. Brock, J. Rubio, F. Rawson, and J. Carter, "Architecting for power management: The IBM® POWER7 approach," in *HPCA*, pp. 1–11, IEEE, 2010.

[5] S. Borkar, "Thousand core chips: a technology perspective," in *Proceedings of the 44th annual DAC*, pp. 746–749, ACM, 2007.

[6] Intel. `http://ark.intel.com/products/53575/`.

[7] A. Agrawal *et al.*, "A new heuristic for multiple sequence alignment," in *IEEE EIT*, pp. 215–217, 2008.

[8] K. Bergman *et al.*, "Exascale computing study: Technology challenges in achieving exascale systems," tech. rep., DARPA Technical Report, 2008.

[9] S. Khaitan, J. McCalley, and M. Raju, "Numerical methods for on-line power system load flow analysis," *Energy Systems*, vol. 1, no. 3, pp. 273–289, 2010.

[10] M. Raju *et al.*, "Domain decomposition based high performance parallel computing," *International Journal of Computer Science Issues*, 2009.

[11] E. Cooper-Balis and B. Jacob, "Fine-grained activation for power reduction in DRAM," *Micro, IEEE*, vol. 30, no. 3, pp. 34–47, 2010.

[12] B. Jacob, S. Ng, and D. Wang, *Memory systems: cache, DRAM, disk*. Morgan Kaufmann Publication, 2007.

[13] "Calculating memory system power for DDR3." http://download.micron.com.

[14] T. Vogelsang, "Understanding the energy consumption of dynamic random access memories," in *MICRO*, pp. 363–374, 2010.

[15] A. Amin and Z. Chishti, "Rank-aware cache replacement and write buffering to improve DRAM energy efficiency," in *Proceedings of the 16th ACM/IEEE international symposium on Low power electronics and design*, pp. 383–388, ACM, 2010.

[16] V. Anagnostopoulou, S. Biswas, H. Saadeldeen, A. Savage, R. Bianchini, T. Yang, D. Franklin, and F. T. Chong, "Barely alive memory servers: Keeping data active in a low-power state," in *ACM Journal on Emerging Technologies in Computing Systems, Special issue on Sustainable and Green Computing Systems*, April 2012.

[17] R. Ayoub, K. Indukuri, and T. Rosing, "Energy efficient proactive thermal management in memory subsystem," in *International Symposium on Low-Power Electronics and Design (ISLPED)*, pp. 195–200, IEEE, 2010.

[18] H. Ben Fradj, C. Belleudy, and M. Auguin, "System level multi-bank main memory configuration for energy reduction," in *Integrated Circuit and System Design. Power and Timing Modeling, Optimization and Simulation* (J. Vounckx, N. Azemard, and P. Maurine, eds.), vol. 4148 of *Lecture Notes in Computer Science*, pp. 84–94, Springer Berlin / Heidelberg, 2006.

[19] M. Bi, R. Duan, and C. Gniady, "Delay-hiding energy management mechanisms for DRAM," in *HPCA*, pp. 1–10, IEEE, 2010.

[20] K. Chandrasekar, B. Akesson, and K. Goossens, "Run-time power-down strategies for real-time SDRAM memory controllers," in *Proceedings of the 49th Annual DAC*, pp. 988–993, ACM, 2012.

[21] G. Chen, R. Shetty, M. Kandemir, N. Vijaykrishnan, M. J. Irwin, and M. Wolczko, "Tuning garbage collection for reducing memory system energy in an embedded java environment," *ACM Trans. Embed. Comput. Syst.*, vol. 1, pp. 27–55, Nov. 2002.

[22] M. Chen, X. Wang, and X. Li, "Coordinating processor and main memory for efficient server power control," in *International Conference on Supercomputing*, ICS '11, (New York, NY, USA), pp. 130–140, ACM, 2011.

[23] H. David, E. Gorbatov, U. R. Hanebutte, R. Khanna, and C. Le, "RAPL: Memory power estimation and capping," in *International Symposium on Low-Power Electronics and Design (ISLPED)*, pp. 189 –194, aug. 2010.

[24] V. De La Luz, M. Kandemir, and I. Kolcu, "Automatic data migration for reducing energy consumption in multi-bank memory systems," in *DAC, 2002*, pp. 213–218, IEEE, 2002.

[25] V. Delaluz, M. Kandemir, N. Vijaykrishnan, M. Irwin, A. Sivasubramaniam, and I. Kolcu, "Compiler-directed array interleaving for reducing energy in multi-bank memories," in *ASPDAC*, pp. 288–293, IEEE, 2002.

[26] V. Delaluz, M. Kandemir, N. Vijaykrishnan, A. Sivasubramaniam, and M. Irwin, "DRAM energy management using software and hardware directed power mode control," in *HPCA*, pp. 159–169, IEEE, 2001.

[27] V. Delaluz, A. Sivasubramaniam, M. Kandemir, N. Vijaykrishnan, and M. Irwin, "Scheduler-based DRAM energy management," in *DAC*, pp. 697–702, ACM, 2002.

[28] Q. Deng, D. Meisner, A. Bhattacharjee, T. F. Wenisch, and R. Bianchini., "MultiScale: Memory System DVFS with Multiple Memory Controllers," in *International Symposium on Low power electronics and design (ISLPED)*, July 2012.

[29] Q. Deng, D. Meisner, L. Ramos, T. Wenisch, and R. Bianchini, "Memscale: active low-power modes for main memory," *ACM SIGPLAN Notices*, vol. 46, no. 3, pp. 225–238, 2011.

[30] B. Diniz, D. Guedes, W. Meira Jr, and R. Bianchini, "Limiting the power consumption of main memory," in *ACM SIGARCH Computer Architecture News*, vol. 35, pp. 290–301, ACM, 2007.

[31] X. Fan, C. Ellis, and A. Lebeck, "Memory controller policies for DRAM power management," in *Proceedings of the 2001 international symposium on Low power electronics and design*, pp. 129–134, ACM, 2001.

[32] M. Floyd, S. Ghiasi, T. Keller, K. Rajamani, F. Rawson, J. Rubio, and M. Ware, "System power management support in the IBM POWER6 microprocessor," *IBM Journal of Research and Development*, vol. 51, no. 6, pp. 733–746, 2007.

[33] H. Huang, P. Pillai, and K. Shin, "Design and implementation of power-aware virtual memory," *USENIX Annual Technical Conference*, pp. 57–70, 2003.

[34] H. Huang, K. Shin, C. Lefurgy, and T. Keller, "Improving energy efficiency by making DRAM less randomly accessed," in *Proceedings of the 2005 international symposium on Low power electronics and design*, pp. 393–398, ACM, 2005.

[35] I. Hur and C. Lin, "A comprehensive approach to DRAM power management," in *14th International Symposium on High Performance Computer Architecture, 2008. HPCA.*, pp. 305–316, IEEE, 2008.

[36] S. Irani, S. Shukla, and R. Gupta, "Online strategies for dynamic power management in systems with multiple power-saving states," *ACM Transactions on Embedded Computing Systems (TECS)*, vol. 2, no. 3, pp. 325–346, 2003.

[37] M. Kandemir, U. Sezer, and V. Delaluz, "Improving memory energy using access pattern classification," in *International Conference on Computer-Aided Design*, pp. 201–206, IEEE Press, 2001.

[38] B. Khargharia, S. Hariri, and M. S. Yousif, "Self-optimization of performance-per-watt for interleaved memory systems," in *14th international conference on High performance computing*, HiPC'07, (Berlin, Heidelberg), pp. 368–380, Springer-Verlag, 2007.

[39] H. Koc, O. Ozturk, M. Kandemir, and E. Ercanli, "Minimizing energy consumption of banked memories using data recomputation," in *International Symposium on Low Power Electronics and Design, 2006.*, pp. 358–361, 2006.

[40] A. Lebeck, X. Fan, H. Zeng, and C. Ellis, "Power aware page allocation," *ACM SIGPLAN Notices*, vol. 35, no. 11, pp. 105–116, 2000.

[41] X. Li, R. Gupta, S. Adve, and Y. Zhou, "Cross-component energy management: Joint adaptation of processor and memory," *ACM Transactions on Architecture and Code Optimization (TACO)*, vol. 4, no. 3, p. 14, 2007.

[42] X. Li, Z. Li, F. David, P. Zhou, Y. Zhou, S. Adve, and S. Kumar, "Performance directed energy management for main memory and disks," in *ACM SIGARCH Computer Architecture News*, vol. 32, pp. 271–283, ACM, 2004.

[43] S. Liu, K. Pattabiraman, T. Moscibroda, and B. Zorn, "Flikker: Saving DRAM refresh-power through critical data partitioning," *ACM SIGPLAN Notices*, vol. 46, no. 3, pp. 213–224, 2011.

[44] C. Lyuh and T. Kim, "Memory access scheduling and binding considering energy minimization in multi-bank memory systems," in *Proceedings of the 41st annual DAC*, pp. 81–86, ACM, 2004.

[45] K. T. Malladi, F. A. Nothaft, K. Periyathambi, B. C. Lee, C. Kozyrakis, and M. Horowitz, "Towards energy-proportional datacenter memory with mobile DRAM," in *ISCA*, pp. 37 –48, june 2012.

[46] D. Meisner, B. Gold, and T. Wenisch, "PowerNap: eliminating server idle power," *ACM Sigplan Notices*, vol. 44, no. 3, pp. 205–216, 2009.

[47] J. Mukundan and J. F. Martinez, "MORSE: Multi-objective reconfigurable self-optimizing memory scheduler," *HPCA*, vol. 0, pp. 1–12, 2012.

[48] O. Ozturk, G. Chen, M. Kandemir, and M. Karakoy, "Cache miss clustering for banked memory systems," in *IEEE/ACM international conference on Computer-aided design*, ICCAD '06, pp. 244–250, ACM, 2006.

[49] O. Ozturk and M. Kandemir, "ILP-Based energy minimization techniques for banked memories," *ACM Trans. Des. Autom. Electron. Syst.*, vol. 13, pp. 50:1–50:40, July 2008.

[50] V. Pandey, W. Jiang, Y. Zhou, and R. Bianchini, "DMA-aware memory energy management," in *HPCA*, pp. 133 – 144, feb. 2006.

[51] J. Pisharath, A. Choudhary, and M. Kandemir, "Reducing energy consumption of queries in memory-resident database systems," in *Proceedings of the 2004 international conference on Compilers, architecture, and synthesis for embedded systems*, pp. 35–45, ACM, 2004.

[52] I. Rodero, S. Chandra, M. Parashar, R. Muralidhar, H. Seshadri, and S. Poole, "Investigating the potential of application-centric aggressive power management for hpc workloads," in *International Conference on High Performance Computing (HiPC), 2010*, pp. 1 –10, dec. 2010.

[53] K. Sudan, K. Rajamani, W. Huang, and J. Carter, "Tiered memory: An iso-power memory architecture to address the memory power wall," *IEEE Transactions on Computers*, 2012.

[54] A. Udipi, N. Muralimanohar, R. Balsubramonian, A. Davis, and N. Jouppi, "LOT-ECC: LOcalized and Tiered Reliability Mechanisms for Commodity Memory Systems," in *Proceedings of ISCA*, 2012.

[55] A. Udipi, N. Muralimanohar, N. Chatterjee, R. Balasubramonian, A. Davis, and N. Jouppi, "Rethinking DRAM design and organization for energy-constrained multi-cores," in *ACM SIGARCH Computer Architecture News*, vol. 38, pp. 175–186, ACM, 2010.

[56] Z. Wang and X. Hu, "Energy-aware variable partitioning and instruction scheduling for multibank memory architectures," *ACM Transactions on Design Automation of Electronic Systems (TODAES)*, vol. 10, no. 2, pp. 369–388, 2005.

[57] H. Zheng, J. Lin, Z. Zhang, E. Gorbatov, H. David, and Z. Zhu, "Mini-rank: Adaptive DRAM architecture for improving memory power efficiency," in *MICRO*, pp. 210–221, IEEE, 2008.

[58] H. Zheng and Z. Zhu, "Power and Performance Trade-Offs in Contemporary DRAM System Designs for Multicore Processors," *IEEE Transactions on Computers*, vol. 59, no. 8, pp. 1033–1046, 2010.

[59] P. Zhou, V. Pandey, J. Sundaresan, A. Raghuraman, Y. Zhou, and S. Kumar, "Dynamic tracking of page miss ratio curve for memory management," in *ACM SIGOPS Operating Systems Review*, vol. 38, pp. 177–188, ACM, 2004.

[60] D. Kaseridis, J. Stuecheli, and L. K. John, "Minimalist open-page: a DRAM page-mode scheduling policy for the many-core era," in *MICRO*, MICRO-44 '11, (New York, NY, USA), pp. 24–35, ACM, 2011.

[61] H. Kim and I.-C. Park, "High-performance and low-power memory-interface architecture for video processing applications," *IEEE Transactions on Circuits and Systems for Video Technology,*, vol. 11, pp. 1160 – 1170, nov 2001.

[62] Y. Li and T. Zhang, "Reducing DRAM Image Data Access Energy Consumption in Video Processing," *IEEE Transactions on Multimedia*, vol. 14, pp. 303 –313, april 2012.

[63] K. Sudan, N. Chatterjee, D. Nellans, M. Awasthi, R. Balasubramonian, and A. Davis, "Micro-pages: increasing DRAM efficiency with locality-aware data placement," in *ACM SIGARCH Computer Architecture News*, vol. 38, pp. 219–230, ACM, 2010.

[64] M. E. Tolentino, J. Turner, and K. W. Cameron, "Memory MISER: Improving Main Memory Energy Efficiency in Servers," *IEEE Trans. Comput.*, vol. 58, pp. 336–350, Mar. 2009.

[65] J. Ahn, N. Jouppi, C. Kozyrakis, J. Leverich, and R. Schreiber, "Future scaling of processor-memory interfaces," in *Proceedings of the Conference on High Performance Computing Networking, Storage and Analysis*, p. 42, ACM, 2009.

[66] K. Fang, H. Zheng, and Z. Zhu, "Heterogeneous mini-rank: Adaptive, power-efficient memory architecture," in *39th International Conference on Parallel Processing (ICPP), 2010*, pp. 21–29, IEEE, 2010.

[67] O. Seongil, S. Choo, and J. H. Ahn, "Exploring energy-efficient DRAM array organizations," in *54th International Midwest Symposium on Circuits and Systems (MWSCAS), 2011*, pp. 1 –4, aug. 2011.

[68] G. Zhang, H. Wang, X. Chen, S. Huang, and P. Li, "Heterogeneous multi-channel: fine-grained DRAM control for both system performance and power efficiency," in *Proceedings of the 49th Annual DAC*, pp. 876–881, ACM, 2012.

[69] H. Hanson and K. Rajamani, "What computer architects need to know about memory throttling," in *Computer Architecture*, pp. 233–242, Springer, 2012.

[70] J. Lin, H. Zheng, Z. Zhu, E. Gorbatov, H. David, and Z. Zhang, "Software thermal management of DRAM memory for multicore systems," in *ACM SIGMETRICS Performance Evaluation Review*, vol. 36, pp. 337–348, ACM, 2008.

[71] J. Lin, H. Zheng, Z. Zhu, Z. Zhang, and H. David, "DRAM-level prefetching for fully-buffered DIMM: Design, performance and power saving," in *ISPASS*, pp. 94–104, IEEE, 2007.

[72] S. Liu, S. Memik, Y. Zhang, and G. Memik, "A power and temperature aware DRAM architecture," in *DAC*, pp. 878–883, IEEE, 2008.

[73] J. Trajkovic, A. Veidenbaum, and A. Kejariwal, "Improving SDRAM access energy efficiency for low-power embedded systems," *ACM Transactions on Embedded Computing Systems (TECS)*, vol. 7, no. 3, p. 24, 2008.

[74] D. Yoon, M. Jeong, and M. Erez, "Adaptive granularity memory systems: a tradeoff between storage efficiency and throughput," in *ISCA*, pp. 295–306, ACM, 2011.

[75] D. H. Yoon, J. Chang, N. Muralimanohar, and P. Ranganathan, "BOOM: Enabling mobile memory based low-power server DIMMs," in *ISCA*, pp. 25 –36, june 2012.

[76] H. Zheng, J. Lin, Z. Zhang, and Z. Zhu, "Decoupled DIMM: building high-bandwidth memory system using low-speed DRAM devices," in *ISCA*, pp. 255–266, ACM, 2009.

[77] N. Aggarwal, J. Cantin, M. Lipasti, and J. Smith, "Power-efficient DRAM speculation," in *HPCA*, pp. 317–328, IEEE, 2008.

[78] C. Isen and L. John, "Eskimo-energy savings using semantic knowledge of inconsequential memory occupancy for DRAM subsystem," in *MICRO*, pp. 337–346, IEEE, 2009.

[79] S. Mazumdar, D. Tullsen, and J. Song, "Inter-socket victim cacheing for platform power reduction," in *Computer Design (ICCD), 2010 IEEE International Conference on*, pp. 509–514, IEEE, 2010.

[80] G. Chen, M. Kandemir, N. Vijaykrishnan, M. J. Irwin, and W. Wolf, "Energy savings through compression in embedded java environments," in *Proceedings of the tenth international symposium on Hardware/software codesign*, CODES '02, pp. 163–168, ACM, 2002.

[81] R. Tremaine, P. Franaszek, J. Robinson, C. Schulz, T. Smith, M. Wazlowski, and P. Bland, "IBM memory expansion technology (MXT)," *IBM Journal of Research and Development*, vol. 45, no. 2, pp. 271–285, 2001.

[82] L. Yang, R. P. Dick, H. Lekatsas, and S. Chakradhar, "Online memory compression for embedded systems," *ACM Trans. Embed. Comput. Syst.*, vol. 9, pp. 27:1–27:30, Mar. 2010.

[83] H. David, C. Fallin, E. Gorbatov, U. R. Hanebutte, and O. Mutlu, "Memory power management via dynamic voltage/frequency scaling," in *Proceedings of the 8th ACM international conference on Autonomic computing*, ICAC '11, (New York, NY, USA), pp. 31–40, ACM, 2011.

[84] M. Ghosh and H. Lee, "Smart refresh: An enhanced memory controller design for reducing energy in conventional and 3D Die-Stacked DRAMs," in *MICRO*, pp. 134–145, IEEE Computer Society, 2007.

[85] J. Liu, B. Jaiyen, R. Veras, and O. Mutlu, "RAIDR: Retention-aware intelligent DRAM refresh," in *ISCA*, pp. 1 –12, june 2012.

[86] T. Ohsawa, K. Kai, and K. Murakami, "Optimizing the DRAM refresh count for merged DRAM/logic LSIs," in *Proceedings of the 1998 international symposium on Low power electronics and design*, pp. 82–87, ACM, 1998.

[87] K. Patel, L. Benini, E. Macii, and M. Poncino, "Energy-efficient value-based selective refresh for embedded DRAMs," *Integrated Circuit and System Design. Power and Timing Modeling, Optimization and Simulation*, pp. 909–909, 2005.

[88] S. Phadke and S. Narayanasamy, "MLP aware heterogeneous memory system," in *Design, Automation & Test in Europe Conference & Exhibition (DATE), 2011*, pp. 1–6, IEEE, 2011.

[89] J. Stuecheli, D. Kaseridis, H. Hunter, and L. John, "Elastic refresh: Techniques to mitigate refresh penalties in high density memory," in *MICRO*, pp. 375–384, IEEE, 2010.

[90] C. Lin, C. Yang, and K. King, "PPT: joint performance/power/thermal management of DRAM memory for multi-core systems," in *Proceedings of the 14th ACM/IEEE international symposium on Low power electronics and design*, pp. 93–98, ACM, 2009.

[91] J. Lin, H. Zheng, Z. Zhu, H. David, and Z. Zhang, "Thermal modeling and management of DRAM memory systems," in *ISCA*, vol. 35, ACM, 2007.

[92] S. Liu *et al.*, "Hardware/software techniques for DRAM thermal management," in *HPCA*, pp. 515–525, 2011.

[93] S. Khaitan *et al.*, "Fast parallelized algorithms for on-line extended-term dynamic cascading analysis," in *IEEE/PES PSCE*, pp. 1–7, 2009.