ENERGY PROPORTIONAL MEMORY SYSTEMS

A DISSERTATION
SUBMITTED TO THE DEPARTMENT OF DEPARTMENT OF
ELECTRICAL ENGINEERING
AND THE COMMITTEE ON GRADUATE STUDIES
OF STANFORD UNIVERSITY
IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR THE DEGREE OF
DOCTOR OF PHILOSOPHY

Krishna Teja Malladi
August 2013

I certify that I have read this dissertation and that, in my opinion, it is fully adequate in scope and quality as a dissertation for the degree of Doctor of Philosophy.

Mark Horowitz, Primary Adviser

I certify that I have read this dissertation and that, in my opinion, it is fully adequate in scope and quality as a dissertation for the degree of Doctor of Philosophy.

Christoforos Kozyrakis

I certify that I have read this dissertation and that, in my opinion, it is fully adequate in scope and quality as a dissertation for the degree of Doctor of Philosophy.

Benjamin Lee

Approved for the Stanford University Committee on Graduate Studies.

Patricia J. Gumport, Vice Provost for Graduate Education

Abstract

The horizon of computer systems is changing. Moore's law is power constrained, making energy efficiency the primary goal of computer systems. In particular, memory energy has emerged as the key efficiency bottleneck, owing to extensive research in low power CPU architectures. On the other hand, the popularity of modern day internet services has lead to a vast expansion of datacenters, magnifying the importance of efficient computer system design. With datacenter energy provisioning and running costs running into millions of dollars, energy is an even more important criterion in datacenters.

To increase computer system energy efficiency, we need memory systems that keep pace with processor efficiency gains. Currently, servers use DDR3 memory, which is designed for high bandwidth but not for energy proportionality. A system using 20% of the peak DDR3 bandwidth consumes 2.3× the energy per bit compared to the energy consumed by a system with fully utilized memory bandwidth. Nevertheless, many datacenter applications stress memory capacity and latency but not memory bandwidth. In response, we architect server memory systems using mobile DRAM devices i.e. LPDDR2, trading peak bandwidth for lower energy consumption per bit and more efficient idle modes. We demonstrate 3-5× lower memory power, better proportionality, and negligible performance penalties for datacenter workloads.

Noting that LPDDR2 has performance impact for high bandwidth applications, we explore low-power, high-speed interfaces that can be applied to a broad variety of applications. We re-think DRAM power modes by modeling and characterizing inter-arrival times for memory requests to determine the properties an ideal power mode should have. This analysis indicates that even the most responsive of today's power modes are rarely used. As a result, up to 88% of memory is spent idling in an active mode. This analysis indicates that power modes must have much shorter exit latencies than they have today. Wake-up latencies less than 100ns are ideal.

To address these challenges, we present MemBlaze, an architecture with DRAMs and links that are capable of fast powerup, which provides more opportunities to powerdown memories. By eliminating DRAM chip timing circuitry, a key contributor to powerup latency, and by shifting timing responsibility to the controller, MemBlaze permits data transfers immediately after wake-up and reduces energy per transfer by 50% with no performance impact.

Alternatively, in scenarios where DRAM timing circuitry must remain, we explore mechanisms to accommodate DRAMs that powerup with less than perfect interface timing. We present MemCorrect which detects timing errors while MemDrowsy lowers transfer rates and widens sampling margins to accommodate timing uncertainty in situations where the interface circuitry must recalibrate after exit from powerdown state. Combined, MemCorrect and MemDrowsy still reduce energy per transfer by 50% but incur modest (e.g., 10%) performance penalties.

Finally, we demonstrate that we need to re-evaluate our design choices for last level caches whose static power is beginning to compete with the dynamic energy of new and efficient memory systems like LPDDR2. We propose a novel metric called "Average Memory Access Energy" that quantifies energy costs of the memory hierarchy and helps us make efficient design choices. We demonstrate how the low energy memory hierarchy helps us in lowering system operating costs.

Acknowledgements

My journey at Stanford had been academically, intellectually and personally rewarding and I would like to acknowledge all the help, guidance and support I received over the course of time, that made it so.

First of all, I would like to wholeheartedly thank my advisor, Prof. Mark Horowitz, who made this thesis possible. I used to look upto him as an undergraduate and getting my admission letter, signed by Mark, who was then the chair, was one of the most surreal experiences I had. I thank him for accepting me into his group and recommending me to the new research project on energy efficient datacenter systems. Mark is undoubtedly one of the most intellectual person I ever worked with, I was able to learn a lot from his sea of knowledge and experience. He fundamentally changed how I think about a problem and motivated me to adapt the strategy of simple, elegant research that solves difficult challenges, while being relevant to both academic and industrial community. It is also remarkable to see his encouragement and patience with students while helping out to the fine-grain details. I still remember working out circuits and math equations with him on a weekend evening in his office. I would also like to thank Mark for taking us out for wonderful group trips from where I will always hold great memories.

I thank Prof. Christos Kozyrakis who is my co-advisor and has also greatly shaped my research at Stanford. I am particularly grateful to him for organizing our research group, Energy Proportionality in Cloud (EPIC) and encouraging us to make individual contributions, early in the course of our research work. I am also thankful to him for bringing us to the same platform with researchers from Google. His architectural classes are some of the best class experiences and working with Mark and Christos helped me to blend crosscutting ideas from technology and architecture. Discussions with Christos were some of the most fruitful ones I had and I really appreciate how they are always positive, energetic and forward-looking.

I cannot also thank enough Prof. Benjamin Lee, with whom I had the good fortune of working throughout my time at Stanford, starting when he was a post-doctoral student here. He has undoubtedly played a great role in shaping me as a researcher and helping me with my expression of ideas and research. I am grateful to him for helping me through my research, and in particular at paper submission times and conferences. It is just amazing to see and learn how he looks at the same data in an insightful way and I hope to be as fine a researcher he is, one day.

I also thank Benchmark Capital for supporting me generously with Stanford Graduate Fellowship and helping me focus on my research. I would also like to thank all my colleagues and mentors at Google who helped me understand datacenter computer systems better, during my internship at Google. I'm grateful to Amer Diwan, Robert Hundt, Wolf-Dietrich Weber, Chris Sadler, Robert Kennedy who helped me at Google. I would also like to thank all my mentors at Rambus who helped me understand memory systems better, during my internship at Rambus. I'm grateful to Ian Shaeffer, James Tringali, Scott Best, Wayne Richardson, Jared Zerbe who helped me at Rambus.

I also thank all my colleagues in both Mark and Christos' research groups; Andrew, Jacob, David, Christina, Ofer, Steve, Mario. I thank Teresa Lynn and MaryJane Swenson a lot for always administering our groups and helping us a lot throughout our time at Stanford, from day one.

Finally, acknowledgements cannot be sufficient to express my feelings towards my parents, sister Deepika, brother-in-law Hari and my little nephew Aditya who made this thesis possible. They have always been behind me with constant love and encouragement during all times, keeping me in high spirits.

I dedicate this thesis to my wife, Tejaswini who I am lucky to have in my life. It is her love, encouragement, critique and support that have always kept me going forward.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

Moore's Law has fueled an exponential growth of computing industry for the past half century. CMOS technology advances resulted in doubling of transistor count on an integrated circuit every 18 months. Computer architects leveraged this growth to commensurately advance chip performance exponentially by investing the growing transistor count to increase single chip performance. Along with transistor scaling, operating voltages had also scaled enabling the increasing transistors within the same power-density budget. Called as Dennard scaling, this interplay between voltage and feature size made efficient transistor scaling possible.

Unfortunately, operating voltage has stopped scaling recently due to leakage problems. Scaling operating voltage also needs threshold voltage to be lowered to maintain transistor performance but this increases leakage exponentially. As a result, power constrains scaling and performance as shown in Figure 1.1. Clearly, we need to build more energy efficient computer systems, because in the power-limited era, improving the energy per operation enables us to operate more transistors within an energy budget, thereby improving performance. This focus on energy efficiency is especially important for computer systems used in datacenters.

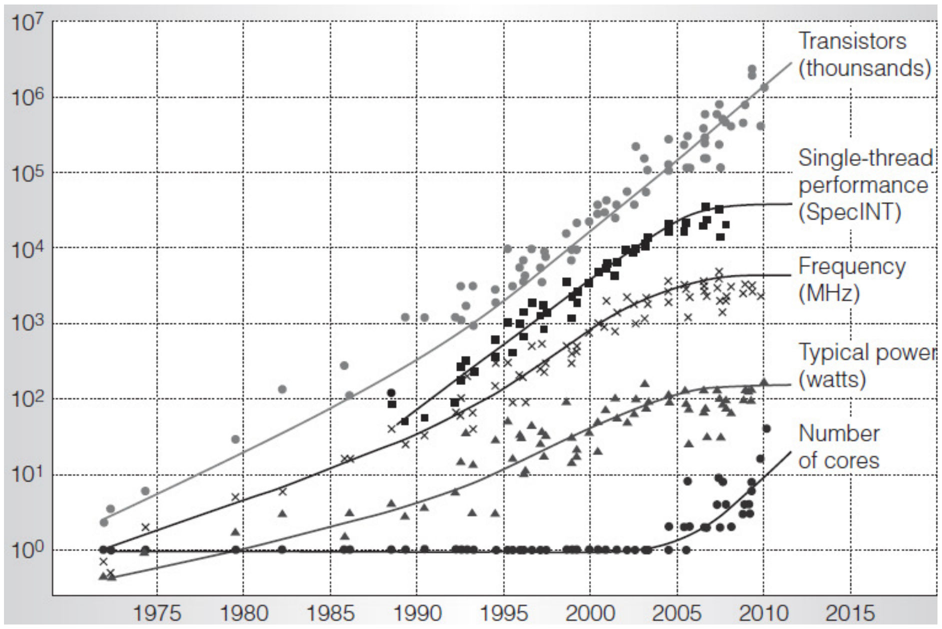


Figure 1.1: Transistor, frequency, performance and power scaling of computer systems. (Data originally collected and plotted by M. Horowitz, F. Labonte, O. Shacham, K. Olukotun, L. Hammond and C. Batten.)

1.1 Datacenters and Energy Proportionality

The popularity of modern day internet services has lead to the tremendous growth of datacenters often referred to as Warehouse Scale Computers (WSC). Consisting of an ensemble of thousands of servers, WSCs power the major webservices that companies like Google, Facebook, Microsoft, Twitter, Amazon offer.

Energy efficiency is critical to these systems, because they are provisioned for a large given power budget which needs power delivery and cooling infrastructure, whose establishment and operating costs could run into millions of dollars. Since WSCs must be provisioned to handle peak loads, they generally run at low (<30%) utilization [5]. As a result, in addition to energy efficiency, WSCs have placed high importance on "Energy proportionality". Energy proportionality signifies energy consumption being proportional to the amount of computation. While energy efficiency measures the energy consumption at the peak operating case, proportionality measures effective energy usage at other utilizations including the common case scenario. Fortunately, recent advances have eliminated most inefficiencies in power delivery and cooling, so new datacenters have a Power Usage Effectiveness (PUE) around 1.10. As a result, energy proportionality of the datacenter is directly related to the proportionality of servers. Unfortunately, datacenters servers today are very energy disproportional. It is very common for servers to spend 60-70% of their peak power even when their utilization is close to 20% [5].

1.2 Memory Systems

Improving energy efficiency of processors reveals that the biggest energy bottleneck in modern chips is in the memory systems. Memory accesses cause energy expensive data communication because memories represent global storage whose physical size generally grows as technology scales. For example at 40nm technology, the energy of 64 bit floating point multiply-add is 50pJ while the cost of reading the operand from register file is 14pJ [33]. In contrast, accessing large cache costs about 1000pJ while accessing DRAMs cost about 7000pJ as seen from Table 1.1. Not only is memory energy atleast two orders of magnitude larger than computation, it also could scale slowly compared to processor. Clearly, unless the energy inefficiency of memory systems is addressed, advances in processor will soon be rendered ineffective.

At the datacenter level, processor energy efficiency and proportionality have improved significantly over the years, benefiting from lowpower circuits, dynamic voltage-frequency scaling, and power gating for unused cores. The use of simpler cores, heterogeneous cores and specialized accelerators have the potential to further improve efficiency [51, 62]. In direct contrast, efficiency and proportionality of DRAM memory systems have improved at a much slower pace. Memory accounts for 25-40% of

Process Technology	**45nm**	**11nm**
16-bit integer multiply	2 pJ	0.4 pJ
64-bit floating-point multiply-add	50 pJ	8 pJ
64-bit access to 8-Kbyte SRAM	14 pJ	2pJ
256-bit access to 1-Mbyte SRAM	566 pJ	94 pJ
256-bit 10mm wire	310 pJ	174 pJ
256-bit DRAM interface	5,120 pJ	512 pJ
256-bit DRAM access	2,048 pJ	640 pJ

Table 1.1: The energy of basic compute and communication operations for 45nm and 11nm (16nm for DRAM) CMOS technology [74, 38, 33]. For memories, we list dynamic energy per operation, but omit the power associated with leakage currents (idle power).

datacenter energy [25, 46, 49] and this percentage will increase as applications demand larger memory capacities for virtualized multi-cores and memory-based caching and storage [64, 57]. In addition, DRAM energy is not proportional to workload, so at low utilizations DRAM power constitutes a large fraction of the server power.

1.3 Thesis Contributions

In this thesis, we address the challenge of making DRAM memory systems energy proportional without affecting the performance or the Quality of Service(QoS) of applications that directly translates to revenue in datacenters. In particular, we make the following contributions:

- **DRAM Energy Analysis for Emerging Applications:** We first characterizable emerging large datacenter workloads like web search, social media, analytics and observe that these applications need high memory capacity but under-utilize bandwidth (Chapter 2). For such emerging workloads, we find that DDR3 standby-power and interface power lead to a highly energy-disproportional memory system, measured in energy per bit transferred (Chapter 3). In addition, since emerging applications always underutilize memory bandwidth, this problem is exacerbated.

- **Energy Proportionality with existing DRAMs:** To address the DDR3 limitations, we turn to mobile-class memory, which addresses the DDR3 interface and significantly reduces idle power. We present a new memory architecture using existing commodity mobile memory parts. By using LPDDR2 which is DDR memory optimized for the mobile domain, we present high-capacity, scalable, new package and module designs. This LPDDR2 architecture provides DDR3-competitive capacity and good signal integrity despite the lack of on-die termination and delay-locked loops. We show that we can reduce main memory power by 3-5× without significant performance penalties for datacenter applications (Chapter 4).

- **Energy Proportionality with future DRAMs:** We also present architectures that future DRAM devices can adapt to be better tailored to energy proportional datacenters. The new interface designs achieve the same 3-5× energy savings as LPDDR2 and support similar bandwidths. Thus, they show no performance degradation for a more general class of workloads apart from datacenter workloads, which can stress memory bandwidth. Since the architectures modify the DDR interface while leaving the DRAM core array unchanged, they maintain the high DRAM manufacturing yield (Chapter 5).

- **Vertically Integrated Evaluation:** In addition to evaluating memory efficiency, we examine the implications for processor cache systems once main memory is made energy proportional. To understand and quantify the effects, we introduce a new metric called "Average memory access energy (AMAE)".

This metric exposes the energy inefficiency of large, shared caches. We also assess implications for datacenter capacity by analyzing Total cost of ownership (TCO) and demonstrate that using energy-efficient main memory increases datacenter capability at the same TCO (Chapter 6).

Building an efficient memory first needs us to make a choice among various contemporary memory technologies (ex: PCM, Flash, STTRAM etc). While many alternatives to DRAM are being proposed [43, 44, 60, 16, 20, 82, 76], we focus on DRAMs in the thesis for two reasons. First, for exploring circuit issues, it is critical to have a concrete technology to make informed tradeoffs and then verify our results. Without these constraints it is easy to miss critical tradeoffs, like the fact that for cost reasons DRAMs only have 3 layers of metal and one is high-resistance tungsten. Second, the energy in large memory arrays is mostly in the communication, which is unchanged with storage technology so the choice of specific memory cell is not critical. Hence, many of the insights from DRAMs will be equally applicable to other memory technologies. Furthermore, the alternative storage technologies seem to have higher read and write energies than DRAM, so their advantage over DRAM from an energy perspective is not clear at this point.

1.4 Thesis Organization

To motivate the need for energy proportional memory systems, Chapter 2, introduces how memory systems for datacenter computers are built, and the type of applications they run. This will lead to a discussion about DRAMs, since they are a major component in the memory system.

In Chapter 3, we quantify the problem of existing DRAM interfaces built using DDR3 systems by doing an "Energy per bit" analysis. This data shows DDR3 to be very inefficient at low memory utilization, which is very common on platforms running WSC applications.

To address this problem, in Chapter 4, we show how a different DRAM, commodity LPDDR2 DRAM provides much lower energy/bit, especially at low utilizations. Unfortunately, these parts were developed for small mobile systems, so we study the architectural implications of employing LPDDR2 in WSCs. Then, we propose a high capacity architecture using LPDDR2 and the energy and performance effects of the resulting memory system are evaluated.

Finding that LPDDR2 architecture has performance implications for bandwidth sensitive applications, we explore methodologies to mitigate the effect in future energy proportional parts in Chapter 5. We first characterize the time series behaviour of WSC workloads to understand the required transition time between powermodes. We then present three novel approaches : MemBlaze, MemCorrect, MemDrowsy, all of which enable deep powermodes for both DDR3 and LPDDR2. The resulting energy and performance look good on a wide variety of workloads, including bandwidth intensive applications.

Of course, the total memory energy needs to include the on-chip caches as well as the DRAM. So, in Chapter 6, we motivate the need for energy efficient memory hierarchy including caches, in conjunction with main memory systems. After proposing a novel AMAE metric, we discuss the tradeoffs of sizing last level caches, followed by a discussion of processor challenges. Then we evaluate the system integration benefits of the proposed technologies for datacenters, using TCO analysis.

We conclude by reporting the findings and the thesis contributions in Chapter 7.

Chapter 2

Background and Motivation

As we pursue energy proportionality in memory systems, we first need to understand power dissipation in memory systems. To do so, we need to first look at DRAMs, the principle components used in a memory system, and their power and performance characteristics. Since these components have a power even when idle, their energy efficiency depends on their average activity, so we need to look at the applications that run on these class of computers. Unfortunately, we find that the current DRAMs are not well matched to current applications, at least from an energy point of view, and we focus on addressing this problem in the rest of the thesis.

2.1 Memory System Organization

Modern processors use sophisticated scheduling and prefetching to help reduce the effective memory latency. These operations are generally performed in the memory controller that connects to the last level of the processor cache system and the memory devices. While the memory controller used to be on a separate chip, it now part of the processor die. The connection between the controller and the memory is called a *memory channel*, and in today's systems this channel consists of a 64 bit wide data bus, and a separate set of wires for address and control information. To achieve higher bandwidth, a processor can increase the number of memory channels it supports. Modern chips support from 1-4 channels. Since each channel contains it own address bus, a processor with 2 channels can fetch different addresses on the two channels simultaneously. DRAMs sit on the other end of these memory channels, generally mounted on small socketed PCBs called *Dual-Inline-Memory Modules (DIMM).*

Each DRAM chip has I/O width of either 4, 8, or 16 pins. As the channel is 64 bits, multiple chips (16, 8, 4 respectively) on the same DIMM are activated together and used in parallel to form the 64 bit wide interface. Such a set of chips forms the smallest granularity of accessing DRAM data and is referred to as a *rank*. Depending on the operating speed of the channel, electrical operating conditions (referred to as signal integrity), limit the total number of ranks on a given channel. All the ranks on the channel share the data bus and command/address bus. The number of channels and the interface's data rate determine memory system's peak bandwidth. Figure 2.1 illustrates an example memory system architecture with four ranks interfaced to a controller integrated to the processor.

The peak theoretical bandwidth BW and capacity C of the resulting memory system are given as

$$\begin{aligned}\texttt{BW} &= (\texttt{DRAM Pin Transfer Rate}) \times (\texttt{Channel Width}) \times (\texttt{Channel Count}) \\ \texttt{C} &= (\texttt{Chip Density}) \times (\texttt{Chips/Rank}) \times (\texttt{Ranks/Channel}) \times (\texttt{Channel Count})\end{aligned}$$

As applications demand more computation and data, modern servers provision platforms which use multiple processor sockets, each with integrated memory controllers and channels. For example, Intel-Nehalem and AMD-Barcelona processors have tri- and dual-channel DDR3 controllers per socket respectively. Cores issue

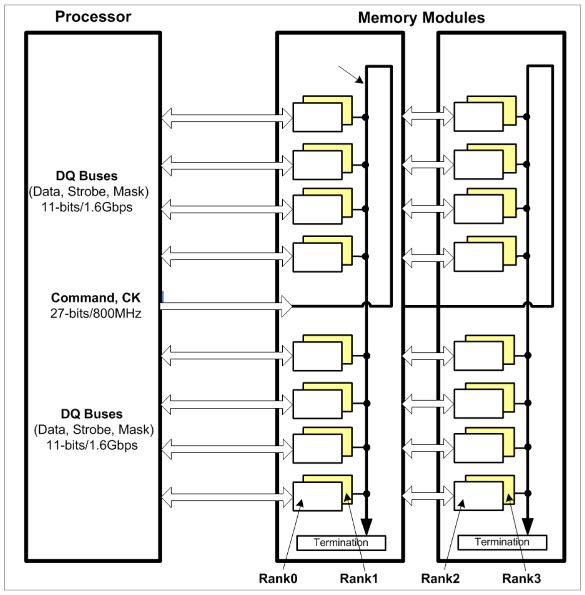
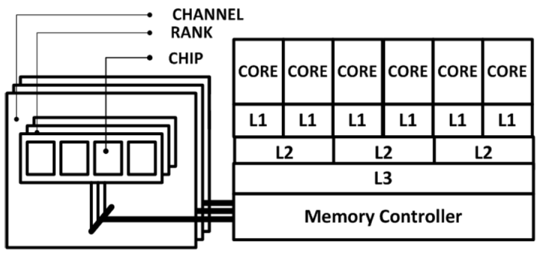


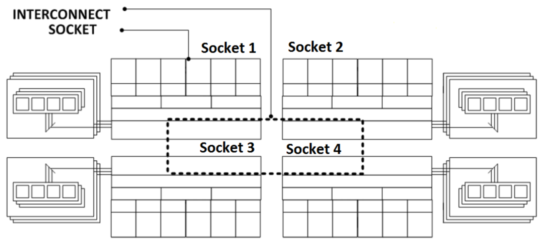
Figure 2.1: A DRAM system organization with four ranks on DIMMs interfaced to the processor. The interface has a separate data (DQ), Address and Command bus (CK).

requests to any controller via “QuickPath” (Intel) or “HyperTransport” (AMD) interconnects which are interchip links that connect sockets so that processors can communicate with other processors and their memory channels. These platforms provision a requisite amount of memory capacity by populating DIMM slots on all channels of each processor socket. An example server memory system organization is shown in Figure 2.2.

A typical dual-channel four-socket system, using Barcelona processors from AMD and filling two DIMM slots per channel each with two ranks comprising of 1Gb, DDR3-1600, x8 width DRAM chips will give 32 GB total capacity and 102 GB/s peak bandwidth.



(a)



(b)

Figure 2.2: (a) Organization of a Socket. Multiple cores, each with private caches share a last level cache and a memory controller. The controller is interfaced with 3-4 DRAM channels. Each channel has multiple ranks, with ranks constituted by multiple chips. (b) Multi Socket server systems with hierarchical DRAM organization. The sockets communicate through an interconnection link making processors and memory channels accessible to all other sockets.

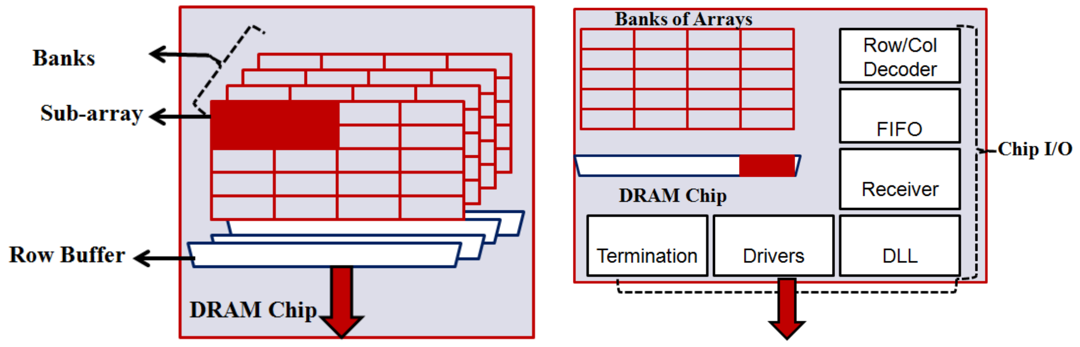
2.2 DRAM Chip Organization

The previous section described the architectural organization of DRAM memory systems in modern datacenter servers. The performance and energy of these systems depend on the characteristics of the DRAM, which we describe next.

A DRAM device provides volatile storage by logically storing data in two-dimensional arrays of memory cells. Internally, the chip is organized with multiple *banks* where data in different banks can be looked up concurrently. Each bank is again organized as a section of small blocks called *subarrays* which have cells of data arranged as *rows* and *columns*. The organization is illustrated in Figure 2.3 [55, 34].

Memory data is placed into independent channels each of which has multiple ranks that timeshare the same bus. Logically, each rank has multiple banks, each of which has row and columns. A word in memory is accessed by identifying the location of the address and this process is called as “Address mapping” [34, 47].

When a cache line request arrives at the controller, the controller issues a row access strobe (RAS) which uses a subset of address bits and identifies the appropriate bank and the row within that bank. This row is then read into a storage structure



(a) (b)

Figure 2.3: (a) Schematic of a DRAM array showing internal banks. Each bank has many subarrays which is each constituted by rows and columns of cells. (b) Schematic of DRAM chip showing storage array, decoders, FIFO, and I/O components like receivers, DLL and drivers which are all shared by all banks to communicate data to the processor.

called *row-buffer* or *sense amplifier*. The row is said to be *activated* and considered *open* [55, 34].

Further, the address is used to determine a selection of columns that correspond to the cache line within the activated row. A column access strobe (CAS) command is issued to select the columns and the data is transferred to the I/O interface of the chip. As the access of granularity is rank, all the chips within a rank operate together to read the cache line data. For example, consider an interface width of 8 per DRAM chip. 8 chips operate together to read 64 bits of data at a clock edge. To read the full typical cache line of 64 bytes, DRAMs employ a technique called *bursting* which reads other words of the line, by transferring data at 8 consecutive rising and falling clock edges. The above process helps in minimizing the number of address pins used by DRAMs [55]. In addition, by *prefetching* all the 8 bursts of data concurrently into a buffer using a single CAS command and serializing them over the interface, the DRAM core can run at a lower speed than the controller-device interface.

Since each bank has a row buffer, multiple rows can be open concurrently. If consecutive cache line requests are to the same row, the accesses can directly be read from the row buffer holding an open row without activating the row again. Such an access is considered to be a *row-buffer hit*. However, if the access is to a different row, the open row first needs to be closed or *precharged* before opening the new row. This policy of retaining an accessed row in the buffer is called *open-page policy*. Alternatively, in a *closed-page policy*, after every access, the row is closed after reading the columns to eliminate page-closing from the critical path of the next request. Modern workloads running on multi-cores generally have small row-buffer hit rates with little locality and systems prefer close-page policy to improve performance. In addition, address mapping policy also determines the amount of sustainable parallelism across banks and also the energy efficiency of memory accesses.

All the banks on a DRAM chip share a common interface to the controller. Consequently, even though the bank lookups can be concurrent, the transmission still needs to be serialized. To do the transmission, each chip's interface consists of many high-speed elements like Delay Locked Loop (DLL), On-Die Termination (ODT), Read and Write FIFOs, drivers and receivers, decode logic as shown in Figure 2.3 which help in clocking DDR3 at speeds as high as 800MHz and higher. [55, 12].

It is forecasted that future interfaces will double the data rate per DRAM pin, following the trend from previous generations. However, the maximum DRAM frequency will flatten and the datarate scaling is by increasing the DRAM prefetching [74].

On the other hand, DRAM energy is spent in activating the rows, reading and writing to the arrays, in cell leakage, refreshing the core in addition to the energy in the high speed interface. Flattening of DRAM voltage is also predicted with future generations thereby resulting in slowing down of power scaling of both DRAM core and interface [74].

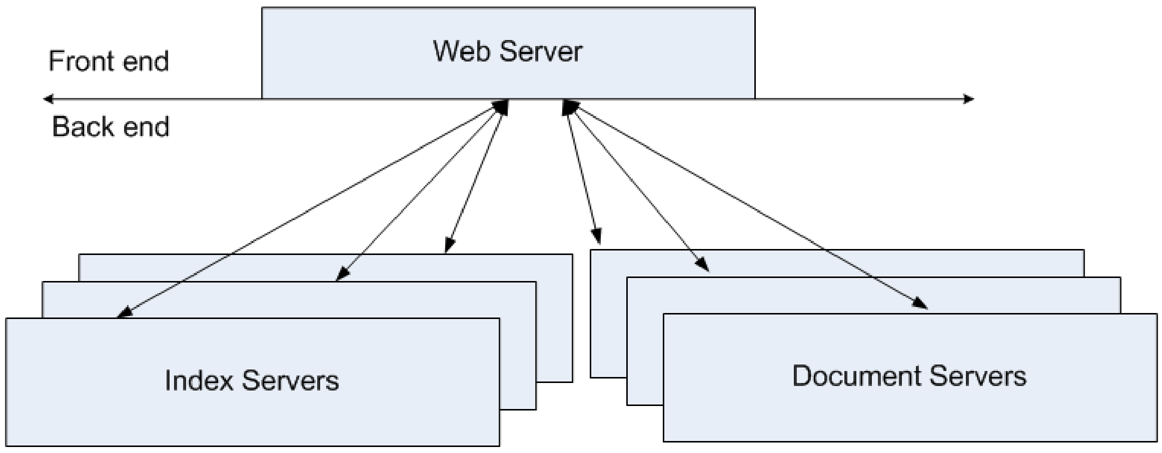


Figure 2.4: Search cluster architecture with a front end webserver that interfaces with a set of backend index and document servers. The backend servers store search data shards that are returned and accumulated at the front end.

2.3 Datacenter Applications

As we study energy proportionality for datacenters, we first describe three major applications that represent the services that are deployed in datacenters.

2.3.1 Websearch

Websearch is a large-scale webservice that is popular in many modern datacenters. Search is also representative of many other workloads that perform complex and distributed accesses. In general, a number of clusters of servers are dedicated to run a single application in datacenters. The clusters consist of a collection of servers that run the front end applications to service users and also back end systems that act as *index and document* servers behind the front end as shown in Figure 2.4.

Search consists two important execution phases. In the first, the front end processes the incoming search query to break it into smaller composing terms and is distributed across many index servers. Each index server processes a section of the huge resident index that holds the list of terabytes of webpages and data. The index performs a reverse lookup to retrieve webpages that contain the query. The section on index server is referred to as a *shard* and helps in parallelizing the massive lookup. Load balancers fan out the load to achieve high throughput and availability by assigning shard lookup across a pool of servers.

Each server performs a document lookup which returns a document list that matches the query term. These lists are aggregated by using intersection on the return lists of the smaller terms while data returned from across different shards is also accumulated. For each such returned document, a relevance score is also calculated to assist ranking the list of documents after returning the results to the user. In the second phase of the search, the returned list of documents is sent to another set of servers called as document servers. The lookup returns a posting list for all the documents in the aggregated order, to produce webpage snippets finally rendered to the user.

Most of the computation and hence most of the servers are in leaf node processing in the backend. Thus, they determine application characteristics like latency, availability and energy. Latency is an important criterion for search-like online services and influences user experience, directly impacting revenue. To minimize lookup latency, most of the index data completely resides on DRAMs. In addition, the servers are also provisioned to handle a high request rate of tasks. These tasks are generally CPU bound and exercise many subsystems like FPU, branches and instruction caches. In addition, the highly parallel architecture places an additional importance on availability. Such CPU activity coupled with distributed nature of processing makes it difficult to find periods of time when a leaf server is fully idle.

2.3.2 Memcached

Memcached is another important application that many WSCs deploy to improve the performance of database driven webservices. Each memcached server provides caching completely in DRAM to alleviate disk lookups/updates on a database. Many companies like Facebook, Twitter rely on memcached to provide object caching to significantly improve response time.

Memcached manages data using a hash-scheme that maps keys to values. Keys and values are generally small and can be cached in multiple, independent servers enabling performance and capacity scaling. The interface provides basic operations of a hash i.e. *UPDATE*, *INSERT*, *LOOKUP* in addition to some complex operations specific to memcached.

For example, *GET* operation hashes the query key and identifies the particular memcached server that caches the value. Upon a hit, the value is returned in a few hundred microseconds. Upon a miss, the query is immediately queued to the database, since the hash uniquely identifies a memcached server, which is independent from the rest of the servers. On the other hand, a *SET* request follows the same procedure as a get request and replaces the value with the new store. Upon a miss, an LRU candidate is identified to fetch the requested key. A *MULTIGET* request is also similar to a get request, but requests for more than one key-value pair.

Similar to search, a memcached based cluster has front end webservers that handle many clients using concurrent threads to get and set key/values. In addition, many servers running memcached are provisioned behind the front end, to handle the requests generated by the front end. The servers are organized as *pools* that divide the entire key space into logical partitions, based on the deployed applications. This minimizes contention and eviction from a subset of keys or coming from one specific workload. The server organization is similar to search architecture and exercises both DRAM and CPUs with an additional emphasis on network.

2.3.3 Map Reduce

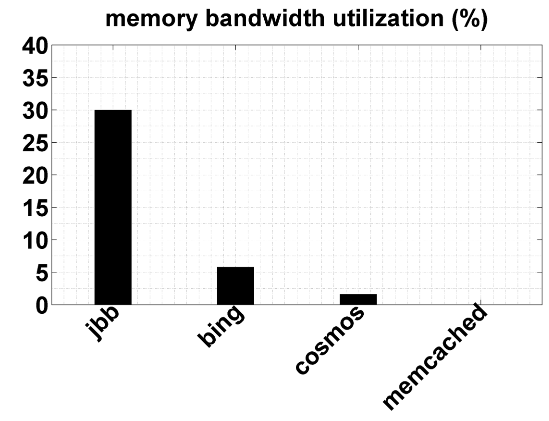
Map-reduce is a popular programming framework that allows abstraction of programming complexity, by using composable *map* and *reduce* tasks. While the map task operates on input key and value pairs to produce intermediate pairs, the reduce merges these values after grouping by a key. The nature of map-reduce lends itself naturally to parallelization in map and reduce tasks, which are often independent of each other. It also allows for abstracting the complexity using Map-reduce APIs like Hadoop and Dryad which allow the programmer to define the map and reduce operations. Map-reduce is naturally applicable to the computation on massive data in datacenters and forms another important class of workloads.

2.4 Datacenter Memory Requirements

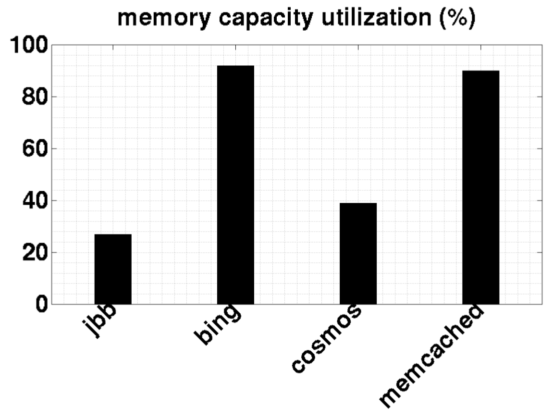
As we have seen in the previous section, WSC applications use the distributed memory of the cluster to store large databases to greatly decrease the latency of querying this data. As this data continues to grow, there will be a need for these machines to provide very large memory systems. With current memory technology, which allows a limited amount of DRAM/channel, this leads to systems with a large number of memory channels, and high peak bandwidth systems.

While some transactional applications such as TPC-C, TPC-H, and SAP perform few operations per data item, requiring up to 75GB/s of memory bandwidth [80], need the high bandwidth of DDR3 DRAM, most emerging datacenter applications exhibit different capacity and bandwidth demands. These applications stress memory capacity and latency but not bandwidth. Such applications include all the applications discussed in Section 2.3, i.e. web search, mapreduce data analytics, and distributed memory caching for social media.

In web search, each server's web index is sized to fit in memory to achieve short



(a)



(b)

Figure 2.5: Surveyed and projected demands for memory (a) bandwidth and (b) capacity

query latencies. Microsoft Bing uses 92% of a server's memory capacity for web indices but utilizes a tiny fraction of memory bandwidth [62]. Search threads are bound by memory latency as their data transfers from the index are short and have no locality.

Microsoft Cosmos, a framework for large-scale data analytics that is similar to MapReduce and Hadoop, under-utilizes memory bandwidth since analyses are often either compute-bound or limited by network bandwidth in a distributed storage system. Under stress testing, Bing and Cosmos servers reach 67-97% processor utilization but only 2-6% memory bandwidth utilization [41]. Google Content Analyzer, BigTable, and web search similarly require substantial memory capacity but modest bandwidth. On a dual-socket Intel Clovertown server, these applications have last-level cache miss rates of less than 10K/msec, which translate into 0.6GB/s of memory bandwidth (less than 3% of peak) [70].

Another important datacenter workload with low bandwidth requirements is social networking (e.g., Facebook, Google+, Twitter, etc). Facebook is a memcached service that uses thousands of servers, an in-memory, distributed object storage and provides 28TB of capacity. This caches 75% of all non-media data and serves complex user requests in reasonable deadlines [64]. Projects like RAMCloud take this approach one step further, replacing the in-memory cache with a distributed, in-memory filesystem for uniformly fast data accesses [57]. Server memory for such frameworks needs high capacity but sustains low bandwidth as memory traffic is limited by network bandwidth. The peak bandwidth of a 10 Gbps Ethernet adapter and a DDR3-1600 memory system are two orders of magnitude apart.

Figure 2.5 summarizes the memory requirements of datacenter applications [41, 69, 57]. The key observation is that even though the applications stress DRAM capacity, they completely underutilize bandwidth.

Table 2.1 illustrates very small memory bandwidth requirement for all the major datacenter applications in production cluster environments. Such underutilization of memory bandwidth is expected, both due to low average server activity and very small emphasis on memory bandwidth even when the server is in full utilization region. Such low memory bandwidth requirements for emerging workloads defy the conventional "Memory wall" that placed high emphasis on high memory bandwidth [63, 80].

Application	**BW (% of Peak)**
Google	
Search	6%
Map Reduce Analytics	2%
Microsoft	
Search	3%
Bigtable	1%
Content Analyzer	2%
Facebook	
Memcached	1%

Table 2.1: Memory bandwidth utilization of datacenter applications collected from production clusters [41, 70, 23].

2.5 Datacenter Memory Energy

Given the light memory bandwidth requirements, it is somewhat surprising that memory energy is a significant component of the overall server power. In fact, historically, majority of server energy was spent in the CPUs. However, modern systems have very different energy characteristics as illustrated in Figure 2.6. For instance, Google WSC servers which used to spend about 60% power budget on CPUs dropped this percentage much below 50% in modern servers. The advent of power-efficient multicore architectures coupled with lowpower CPU circuits, dynamic voltage-frequency scaling (DVFS), power gating unused cores, use of simpler/heterogeneous cores and specialized accelerators tremendously improved CPU power efficiency.

The underlying reason for this problem can be seen more clearly in the power usage of different server subsystems in a Google WSC. Figure 2.7 shows the power spent by a machine as the load varies from 100% to the fully idle case [5]. Although the CPU contribution is about 50% at peak usage, this percentage drops to less than 30% at smaller activity factors. Its energy depends on the load of the server. In contrast, energy of of other subsystems including DRAM, disk and network are to a

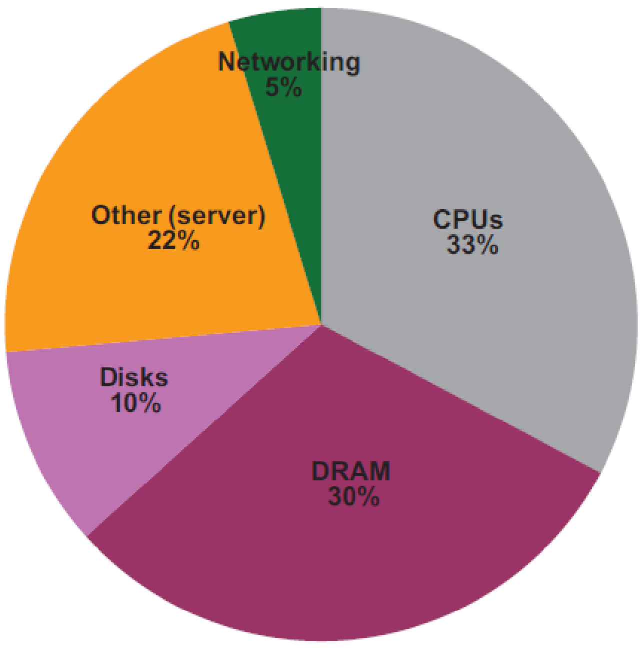


Figure 2.6: Breakdown of a Google datacenter power, demonstrating less than 50% power contribution from CPU. DRAMs also contribute a large fraction.

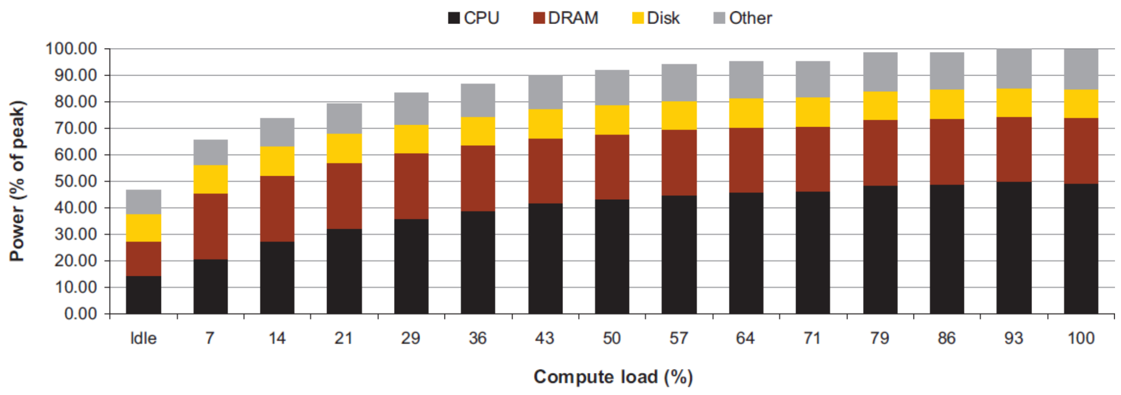


Figure 2.7: Power usage of a server's subsystems as the utilization of the machine is varied from 100% to fully idle.

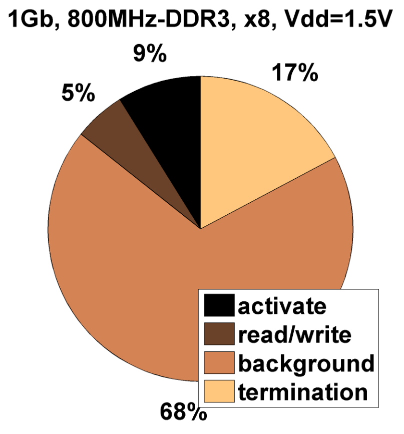


Figure 2.8: Breakdown of DDR3 DRAM power with a typical channel activity. Background and I/O termination power account for a major fraction of the total.

first order, independent of the load.

The fact that the memory energy is not proportional to load is doubly bad. As we have seen, it means that this energy does not decrease when the load on the server falls. It is also bad because it means for most WSC applications, which have low memory bandwidth, the memory energy is much higher that it intrinsically needs to be.

The root cause of this energy problem can be traced back to the high-speed interface that all DRAMs use. To support channel bandwidths of more than 10GB/s, high-speed DDR3 interfaces on every DRAM chip contain power hungry components such as clocked high-speed I/O and termination. These dissipate large amounts of static energy and this energy cost increases as the number of channels increase. This is particularly undesirable, since the number of channels is tied to capacity and as memory capacity increases, the energy problem will grow even worse. Figure 2.8 shows the breakdown of power in a typical DDR chip and we can observe that static power forms a large fraction of the total.

Clearly with DRAMs spending power that is static and also much larger than ideal at typical server utilizations, it is important to understand how energy usage of systems varies with load. "Energy proportionality" has been identified as the most effective figure of merit to characterize datacenter energy usage effectiveness. Formalizing, energy proportionality can be defined as the measure of energy per unit task a server performs, as the number of tasks is varied. In contrast to energy efficiency, proportionality characterizes the common case scenario.

`Energy Proportionality` $\rightarrow$

`Energy` $\propto$ `#Server tasks` (or)

`Energy/task` $=$ `constant` (or)

`Power` $\propto$ `Server utilization in %`

Datacenter applications are distributed in nature and have a large fanout of requests across the cluster. Such a behaviour often remarkably varies the number of tasks a server handles. In addition, datacenters today are provisioned across multiple geographic locations in order to improve the performance and provide better isolation of service requirements. Since each datacenter caters to only one such location, the usage can be highly dynamic across the times of the day. Due to the combination of these factors, datacenter servers could have utilizations varying from 20% to 75% of the provisioned performance. So, it is important to understand energy proportionality of various server subsystems as the server utilization varies.

If energy proportionality were to be considered a new figure of merit, we could consider two different approaches to improve it. One option is to leverage periods of idleness to consolidate applications in such a way that we utilize minimal number of server systems and subsystems while using sleep modes on the rest. However, systems need to be cognizant of the large performance and energy penalties in bringing them back to active states. Called as "idle powermodes", such methods were suitable for the mobile domain. However, these modes are relatively difficult to apply in the space of WSCs as we pay the powerup penalty more often, due to scant opportunities for full system idleness [5, 50]. While we could consider powerstates like C1E processor modes whose wakeup time is smaller, the energy savings are also much smaller and often not justified.

An alternative option is to improved efficiency across all the components in their active state to improve overall proportionality. For example, for DRAMs we need better adaptability to periods of low utilization; Disks might need methodologies to proportionality reduce their platter speed relative to utilization. Called as "active powermodes", these are more suitable to WSC servers. These modes tradeoff performance for better energy characteristics, while ensuring that the subsystem is still in an active-state. CPU DVFS is one such example. The frequency and operating voltage are dynamically adjusted to the activity and obtain better energy efficiency. Even if the transition time to the fully active state was high, these active powermodes are desirable because, the periods of low activity are relatively more common and much larger than high performance periods. In such scenarios, the transition time can fully be amortized which achieving the energy efficiency of the powermode.

Datacenter costs for provisioning, energy and cooling continue to increase and it is critical that we have energy proportionality. DRAMs are clearly the biggest contributors to this energy disproportionality in modern datacenters as evident from power numbers at low utilizations in Figure 2.7. Therefore, DRAM energy proportionality will directly propagate to the server and will lead to proportional datacenter computer systems with much lower operating energy costs.

2.6 Related Work

Energy has emerged as the key concern of datacenters. Barroso et al. demonstrate the importance of energy proportionality in datacenters that underlie the need to improve efficiency at low utilizations [5, 4]. Cluster level techniques consolidate applications by migrating them to a small subset of resources. However, it is challenging to apply them to emerging datacenter workloads as they need many servers to be powered on and operate together [5, 50]. In addition, migrating back to the original state takes time, making it difficult to handle sudden peaks. On the other hand, server level techniques that exploit full server idleness are also difficult to use because distributed applications have very few idle opportunities [50].

To improve CPU efficiency, prior research utilized lowpower circuits, dynamic voltage-frequency scaling, and core parking, use of simpler cores, heterogeneous cores and specialized accelerators [51, 62]. Coming to memory system studies, Lim et al. compare various grades and generations of DDR [45, 46] to improve the efficiency of server-class memory systems. These technologies provide modest performance and efficiency trade-offs. Prior research to memory energy proportionality have attempted to increase the dynamic range of these trade-offs by reconfiguring the voltage and frequency of memory systems [12, 14]. Such an approach allows users to match memory system capabilities to application demands and increase efficiency. However, voltage scaling applies only to the memory core and only in a narrow range defined by the device margins. Frequency scaling applies to the memory channel, reducing power, but often increasing the energy per bit because of the static power component.

Seeking DRAM proportionality, a large body of work manages data placement [17, 42, 71, 15, 59, 27] and batches memory requests [13, 29] to increase the length of idle periods, which are necessary to exploit power modes that have long exit latencies. Despite these techniques, DDR power modes often incur performance penalties due to batching and the latency of the critical word is always affected.

Other researchers work to narrow the access width of a memory operation. Instead of accessing a wide row that spans multiple KB, they access a smaller subset [1, 72, 77, 81] to make DRAM accesses more energy proportional. However, rank subsetting significantly increases the amount of peripheral circuitry in a chip, degrading density [74].

2.7 Summary

Memory system energy is a large fraction of system power, even more so for modern systems whose CPU power efficiency has significantly improved. In addition, even though prior research studies both datacenter power and memory systems, the key problem of DRAM static power has largely been unexplored. This power is difficult to amortize at low utilizations, leading to energy disproportionality.

In the current chapter, we made an observation that memory bandwidth is highly overprovisioned in datacenters. Modern DRAMs are built with DDR3, which spend a lot of static power in the interface, which is the key cause of the disproportional power. In the rest of the thesis, we will first quantify why existing DDR3 systems are not well designed for energy efficiency and proportionality. We also show how DDR's powermodes are not suitable and that we need better active powermodes. We then present ways to use bandwidth tradeoff in datacenters to architect high-capacity DRAM systems with good performance but have much larger efficiency than existing DDR3 systems.

Chapter 3

Inefficiencies of DDR3 Memory Systems

Given the importance of energy efficiency, and energy proportionality in current systems, this chapter examines these issues for DDR3 memory systems. For a memory, the unit of work is the number of bits read/written, so the natural metric for looking at energy proportionality is to look at the energy/bit transferred to/from the memory devices. This metric clearly shows the problem, since the high standby power causes the energy/bit to grow rapidly at low memory bandwidth. Since standby power is generally addressed by providing different power modes, Section 3.3 then investigates DRAM power modes and finds that the latency transitioning out of the low power mode causes a large performance issue, which limits it use.

3.1 DRAM Energy Proportionality

In Section 2.5, we defined energy proportionality in the context of datacenters and at the server level. In the current section, we extend this to understand DRAM proportionality.

At the DRAM context, energy proportionality refers to energy usage proportional to the total number of bits either read/written. Section 2.2 already illustrated the composition of a DRAM chip. In particular, DRAM core and interface are the major components. While the core holds high density volatile data, the interface enables accessing this data at high-speeds. Energy is spent in both reading and writing to the core and keeping the interface powered on to access it. The former includes activating a particular row, reading it the row-buffer and selecting a specific column and transferring these bits to the controller interface. This energy has a direct relation to the total number of accessed bits and is therefore proportional. However, the static power cost of keeping the interface on, is inherently disproportional. The energy of this block is proportional to the time it is on and not to the number of bits transmitted/received. If the memory channel utilization is low, this energy is amortized over a small number of bits, so the per bit cost is large, larger than the energy cost of the core.

Quantifying,

$\texttt{Energy Proportionality} \rightarrow$

$\texttt{Energy} \propto \# \texttt{ bits accessed}$ (or)

$\texttt{Power} \propto (\texttt{bitrate} = \texttt{Average Bandwidth})$ (or)

$\texttt{Energy/bit} = \texttt{constant}$

To illustrate this, we plot DDR3 energy per bit as a function of bandwidth utilization in Figure 3.1. DDR3-1600 has a peak bandwidth of 1.6 Gbps per pin. However, these pins use a lot of power. At high channel utilization, interface power is amortized over many transferred bits, reducing energy per bit. With 100% channel utilization, DDR3 requires 70 pJ/bit; 30% and 10% of this energy is background and I/O termination, respectively. These I/O overheads are incurred even when the chip is idle but

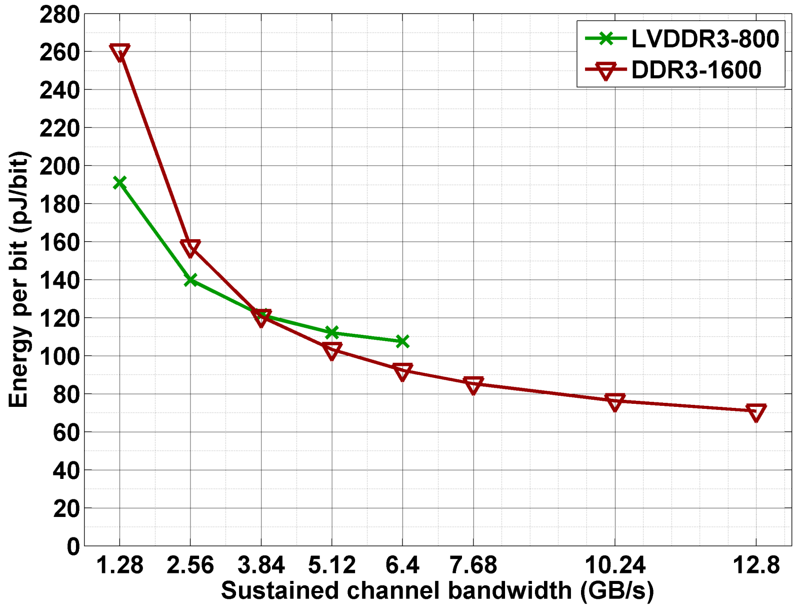


Figure 3.1: Energy per bit (mW/Gbps = pJ/bit) with varying channel utilization for DDR3 and LVDDR3. Assumes four ranks per channel and 3:1 read to write ratio. DDR3 is x8-1600 (Peak BW = 12.8GB/s).

in active power mode. Such "active-idle" power is particularly evident under more typical 20% utilization where background and termination energy is amortized over less work. At 10% channel bandwidth, 1.28GBps, energy per bit increases by 3.7× to 260 pJ/bit.

Also, we consider the energy profile of LVDDR3-800 which is a low-voltage version of DDR3 that tradesoff capacity and performance for slightly better energy. While LVDDR3-800s energy per bit is better than that of DDR3-1600 at these low utilizations by 1.4×, it is still high at 190 pJ/bit, since it halves bandwidth in exchange for modest power savings.

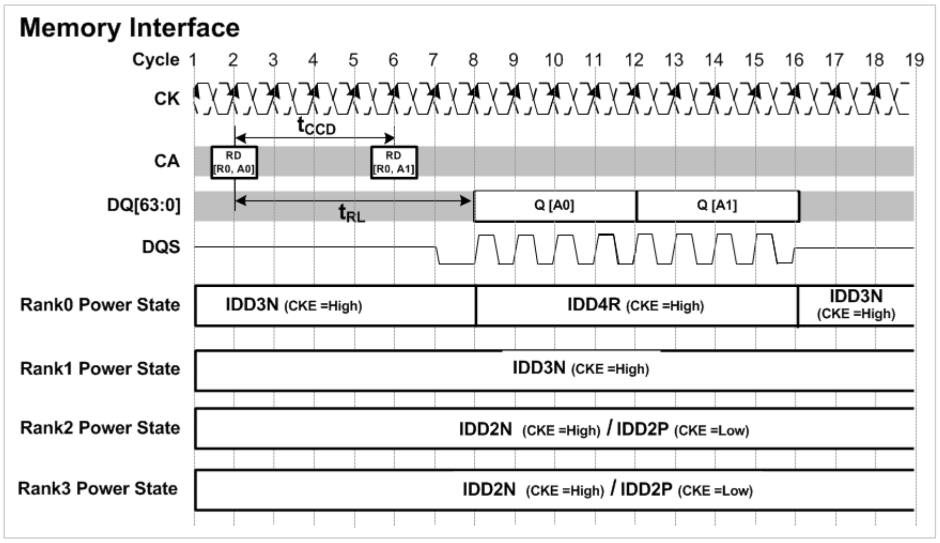


Figure 3.2: DDR3 DRAM Timing.

3.2 DRAM Interfaces and the Active Idle Problem

Clearly, DDR3 is energy disproportional and the key to navigating this challenge is in the interface. In particular, majority of the power is dissipated when the DRAM is in the active-idle state. In this section, we describe the two major components of the interface which dissipate this power : DRAM clocks and termination.

3.2.1 DRAM Clocking

We describe controller-DRAM communication to illustrate the functionality of clocking in DRAMs. The memory controller is connected to the devices via CA and DQ bus that propagate control and data signals. To synchronize signals, the controller generates and forwards a clock (CK) to the DRAMs. Controller circuitry aligns this clock with command and enable signals. Because these signals have lower bandwidth and experience the same loading conditions and discontinuities en route to DRAMs, skew is not an issue. Thus, commands and writes are synchronized.

However, synchronizing reads is more difficult. During a read, data signals are

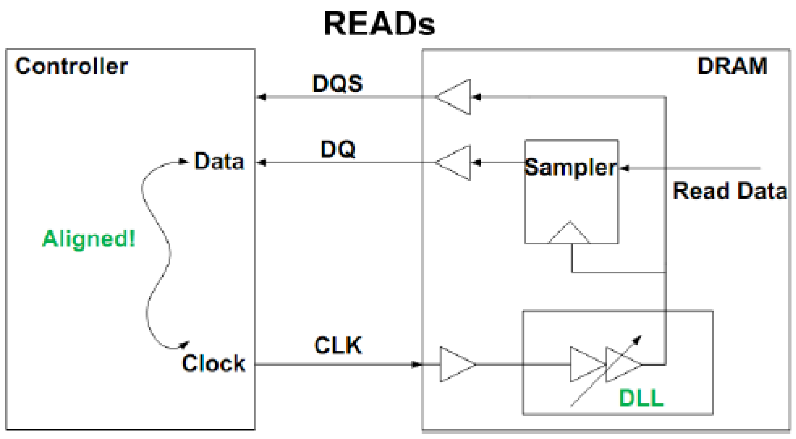


Figure 3.3: Controller-DRAM transmission for Read from DRAM chip needs on-chip DLL to align the strobe to a controller clock edge.

generated by DRAMs (DQ) while clock signals are generated by the controller (CK). Originating on different dies, these signals are subject to different loading conditions and variations in process, voltage, and temperature. Under these conditions, the controller has difficulty using CK edges to sample DQ for arriving read data, especially at high frequencies and narrow data windows.

To facilitate read synchronization, DRAMs explicitly communicate data timing to the controller with a data strobe signal (DQS) that is aligned with the clock (CK) and bus data (DQ). The controller samples DQ on DQS edges as illustrated in Figure 3.2. Data is available some latency after receiving a read command (RD on CA produces Q on DQ after tRL).

DQS edges and data windows must align with the controller-generated clock. DRAMs ensure alignment in two ways. First, during initialization, DQS and CK are calibrated to eliminate any skew due to wire length while the controller specifies worst-case tolerance for timing differences (tDQSCK). Second, during operation, a Delay Locked Loop (DLL), present on every DRAM chip dynamically adjusts the DRAM clock delays to compensate for voltage and temperature variations and keep the position of the DQS at the controller constant, to reduce timing uncertainty when

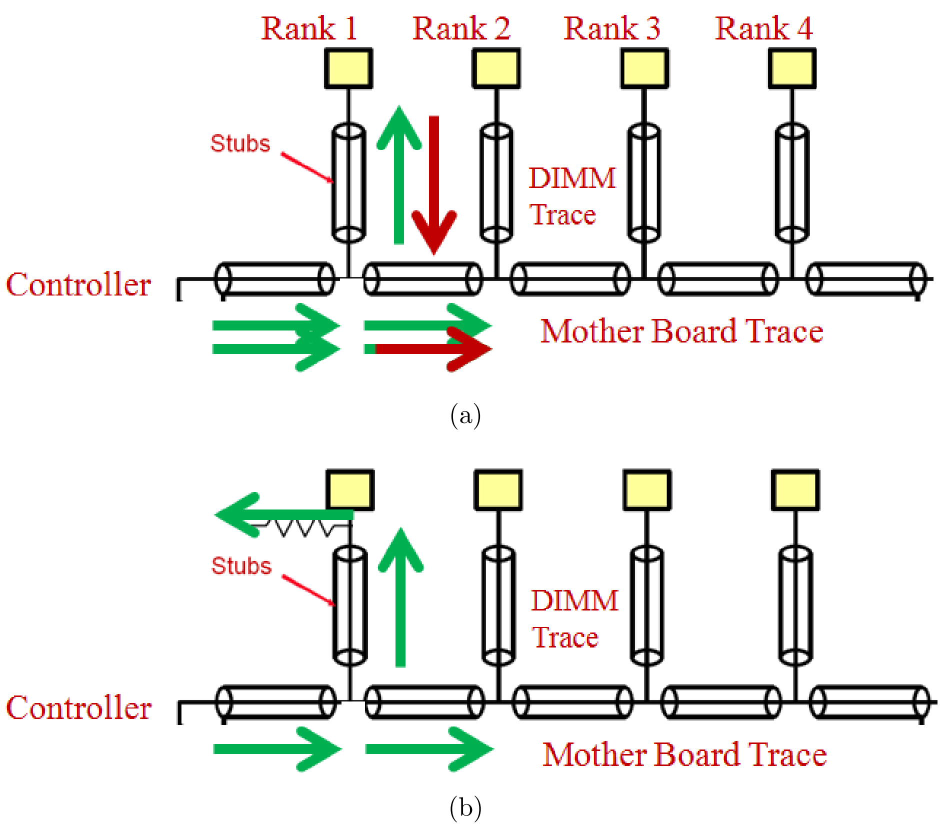


Figure 3.4: (a) Signal reflections on DRAM transmission lines, due to impedance discontinuity created by multiple ranks on the same channel, interfering with transmission. (b) ODT suppresses reflections and creates a clean signal path.

sampling data at high frequencies. The operation is illustrated in Figure 3.3

This way, DLLs help in reliable high speed operation. However, since the DLLs use high power analog components for feedback mechanism in order to adjust the delay, they burn static power contributing to disproportional DRAM power.

However, the larger fraction of non-proportional power comes from the DRAM clocks themselves. Power down mechanisms could stop DLL for a brief amount of time while also disabling the DRAM output buffers and transmitters. However, DRAMs must still continue to have the input clock circuitry (i.e. Clock and Clock Enable signals on the interface) ON, to read possible commands and wakeup. Receivers remain clocked on each cycle, with clocks operational and all of this dissipates dynamic power and contributing to the large interface power.

Simply turning off the clocks could potentially save this large power. However, disabling clocks needs the DLLs to be completely relocked, which takes a long time. In addition, receivers and the DLLs also have analog control signals that take a large time to settle after powering up. This clocking power on every DRAM chip is one of the major reasons for the large DRAM active idle power.

3.2.2 On Die Termination

Since electrical signals travel at the speed of light, the wires that carry very high-speed signals from memory controller to the DRAMs look like transmission lines. The signal is launched into the wire, and then moves at the speed of light to the other end. Since the signal does not know what is at the other end of the wire, the current that is required is set by the geometry of the wire, and is called its impedance. Any change in that impedance, which occurs where DRAMs connect to the bus, will cause a reflection to be generated, to adjust the current that is flowing down the wire. These reflections from the DRAM bus interface cause noise and lowers signal quality as shown in Figure 3.4(a)

The reflections settle after sometime, but can limit the data rate. To get high performance, we need to have very few loads on the bus. Unfortunately, server DRAM systems need high capacity, multi-rank memory architectures, that also operate at high speed. Consequently, multiple DRAM devices must share a bus, but reflections make this challenging. 1st generation DDR memory addressed signal quality by placing a resistor on the motherboard to suppress reflections. By absorbing the energy on the DRAM stub, reflection can be reduced. However, this was not adequate and it was also difficult to dynamically adjust the termination since the required value depends on the architecture and the number of ranks on the channel. Starting DDR2,

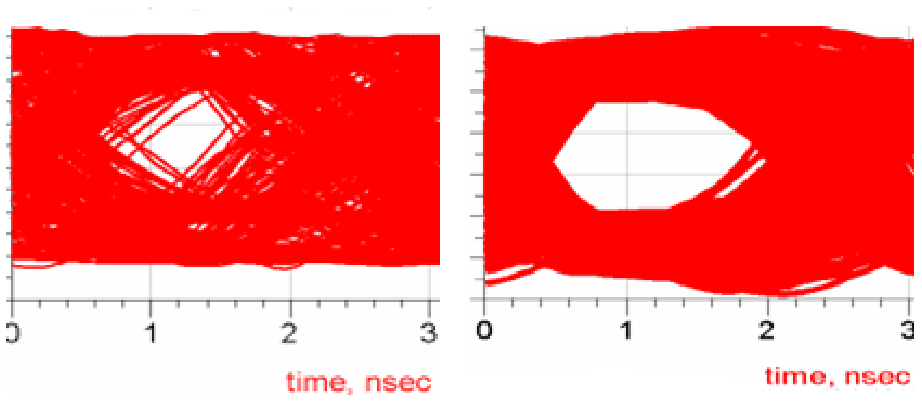


Figure 3.5: Eye Diagram before and after termination. ODT provides a much cleaner eye with sufficient voltage and timing margins.

the resistance was moved to individual DRAM chips and was referred to as On-Die Termination (ODT). This termination could be digitally adjusted for different configurations. Figure 3.4(b) illustrates the ODT operation while Figure 3.5 shows the resulting clean eye diagram of the memory link, with ODT on devices.

While ODT helps in creating a clean transmission path for DRAM signals, it connects signal lines through a resistor to Vdd or Gnd. This again dissipates static power and contribute to the large active idle power. These devices consume static power even when no data is on the bus. As a result, ODT is yet another major source of static power even when a rank is in idle state. Techniques such as memory frequency scaling that increase the active time will only worsen the average energy/bit at low utilizations.

3.3 DRAM Power Modes and the Power Down Problem

The normal way that active idle power is handled is by creating a lower power mode which the chip transitions to, when idle. DRAMs also have different powerstates under different operating mechanisms with some components powered off. Table 3.1 illustrates various powermodes of DDR3 memory systems. As the energy efficiency of the powerstate increases, the corresponding exit latency to return to the active state and start normal operation also increases. Modern memory controllers generally set an idle threshold timer for every DRAM rank. When a rank's idle time exceeds the threshold, the controller powers down the rank to a suitable powerstate. Clearly, deep powerdown modes are desirable in-view of their energy efficiency. However, as we will demonstrate in the next chapters, many emerging workloads in WSCs hardly exhibit idleness that matches the long wakeup DRAM latencies. Turning the DRAM clocks off can transition the device to the deep powerdown state but the necessary DLL reclocking, as seen in Section 3.2, causes a long wakeup penalty. Consequently, current powermodes are often limited to the fast exit state with much smaller energy efficiency compared to the deep powermodes.

Thus, we can conclude that the inefficient active-idle state coupled with powerdown states contribute to the observed DRAM energy disproportionality.

3.4 Workloads and Memory Bandwidth Demand

In Section 2.4, we outlined how datacenter workloads would underutilize memory bandwidth by qualitatively analyzing the workloads. In addition, we have also observed above that DDR3 is not well suited to this scenario. This section describes results from experiments to quantify memory power and performance under datacenter workloads. We begin by describing our methodology and the applications we will use.

Power Mode	**DIMM Idle Power (W)**	**Exit Latency (ns)**	**Mechanism**
Active idle	5.36	0	none
Precharge-idle	4.66	14	pages closed
Active powerdown	3.28	6	clock, I/O buffers, decode logic off
Fast exit powerdown	2.79	19.75	active powerdown + pages closed
Slow exit powerdown	1.60	24	fast exit powerdown + DLL frozen
Self Refresh	0.92	768	fast exit powerdown + DLL, CK off
Self Refresh + registers off	0.56	6700	self refresh + register PLLs off
Disabled	0	disk latency	DIMMs off

Table 3.1: Power Modes for a 4GB DDR3-x4-1333 RDIMM [12, 55]

3.4.1 Experimental Methodology

To understand the memory behavior of many different applications, we use an x86_64 execution-driven simulator based on a Pin front-end [48, 65]. We use 8 out-of-order (OOO) cores at 3 GHz matched with Intel's Nehalem microarchitecture. Each core has a private 8-way, 32-KB L1 data cache and a private 8-way 256KB L2 cache. All the cores share a 16-way associative 16MB L3 cache. Using the Nehalem model, the L1, L2, L3 latencies are set to 1, 7 and 27 cycles respectively. An integrated memory controller models multiple channels and standard DRAM devices. We use a closed-page policy, typical in multi-cores with low page locality[1]. Ranks use a fast-exit precharge power-down mode that uses the fast exit powerdown state (Table 3.1).

	Multi-Programmed	**Multi-Threaded**		**Datacenter**
DRAM B/W	**SPEC CPU 2006**	**SPEC OpenMP**	**PARSEC**	
Low	416.gamess, 447.dealll, 453.povray, 458.sjeng, 464.h264ref, 465.tonto, 481.wrf	ammp, equake	freqmine swaptions	Memcached, Websearch, SPECweb
Medium	400.perlbench, 401.bzip2, 403.gcc, 434.zeusmp 435.gromacs, 436.cactusADM, 445.gobmk, 454.calculix, 456.hmmer, 473.astar	apsi, fma3d wupwise	blackscholes, fluidanimate, streamcluster	SPECjbb, SPECPower
High	433.milc, 437.leslie3d, 450.soplex, 459.GemsFDTD, 462.libquantum, 470.lbm, 471.omnetpp, 482.sphinx3, 483.xalancbmk	applu, art mgrid, swim	canneal	

Table 3.2: Low, medium, high bandwidth applications, estimated from last-level cache miss rates. While applications with bandwidth demand greater than single channel DDR3 bandwidth are classified as high bandwidth, applications with bandwidth between 50% and 100% of single channel bandwidth are classified as medium and less than 50% are low bandwidth.

3.4.2 Applications

To study web search, we use Nutch, an open-source, Java-based web crawler and search engine. First, we index Wikipedia pages to produce a 30GB dataset, which is distributed across several servers' memories. We trace the search engine memory's activity as the 500 most common Wikipedia queries arrive at the server's maximum sustainable query throughput.

We evaluate distributed memory caching, by using Memcached, which is an open-source framework for distributed key-value stores in RAM. Hash functions distribute data and load across Memcached servers. Memory is broken into slabs of varying sizes and we consider 100B slabs (*Memcached.A*) and 10KB slabs (*Memcached.B*). To exercise the cache, we use a zipf distribution (parameter = 0.6) that models a long tail and reflects Facebook popularity distributions [67].

We also evaluate more diverse workloads that might run in virtualized, elastic clouds. We use *SPECjbb2005 (jbb)* with 12 warehouses each with 25MB of transaction data and *SPECpower_ssj2008 (power)* at the calibrated maximum sustainable transaction rate. *SPECweb2005 (web)* benchmarks a banking web server that uses Apache Tomcat. Finally, we consider 8-way multi-threaded SPEC OMP2001 and

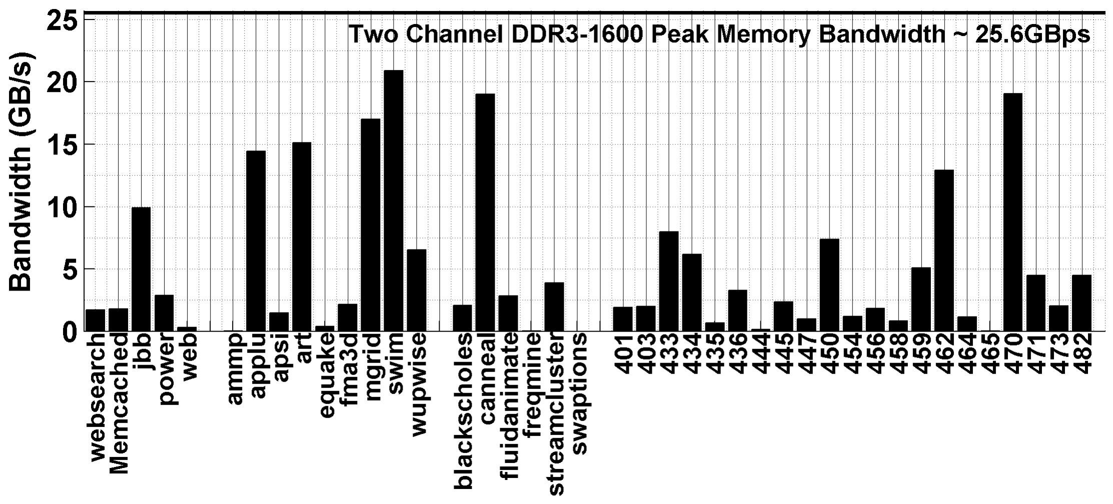


Figure 3.6: Application memory bandwidth demand simulated from last-level cache miss rates with a fixed memory latency. Sustained memory bandwidth is constrained by DRAM internal timings and applications' memory level parallelism (MLP).

PARSEC benchmarks as well as 8-way multi-programmed combinations of SPEC CPU2006 benchmarks with each core running one copy/thread. (Table 3.2).

We follow the methodology used in prior memory studies [1, 72, 39, 14, 35]. We match the number of application threads or processes to the number of cores. We fast-forward 10 to 20 billion instructions to skip warm-up and initialization stages and focus on memory behavior in steady state, which is consistent with prior architectural evaluations of enterprise workloads [80]. To model the distribution of activity across channels, ranks, and banks, we emulate virtual to physical address translation.

3.4.3 Workload Validation

We compare the bandwidth requirements of our workloads, as shown in Figure 3.6, against independently reported measurements from datacenters to validate our methodology and applications.

Nutch. Production search engines (e.g., Microsoft Bing) require less than 6% of peak memory bandwidth [41]. Commercial servers have tri-channel DDR3 with peak bandwidth of 32-39 GB/s. At 6% of peak, search would require 1.9-2.3 GB/s. We use VTune to characterize Nutch on indexed Wikipedia; it consumes 752 MB/s per thread on a Xeon X5670 system and 180 MB/s per thread on an Atom Diamondville. The former measurement suggests that our simulated 8-core system might scale Nutch bandwidth demand to 6.0 GB/s. But contention for shared multiprocessor resources reduces memory demand. Our Nutch simulations indicate 1.8 GB/s of utilized bandwidth, which is consistent with real system measurements for both Bing and Nutch.

SPECjbb, power, web. *Jbb* requires 6 GB/s on a server with four quad-core Intel Core-2 Duo processors [69]. In our environment, *jbb* requires about 10 GB/s. *Power* calibrates transaction rates to the platform's capabilities, which leads to differences in bandwidth demand on each system. On an IBM JS12 blade with a dual-core Power6, *power* uses 30% of peak bandwidth [22], which is 6 GB/s since Power6 has a dual-channel DDR3-1333 system with peak bandwidth of 21 GB/s. Our own VTune measurements for *power* on a four-core Intel Xeon E5507 show 1.2 GB/s. In our simulations, *power* requires 2.5 GB/s. Finally, *web* exhibits low L2 miss rates and does not exercise memory in Simics full-system simulation of a SPARC V9 system [6]. Our *web* simulations show a requirement of 0.3GB/s.

SPEC-OMP, SPEC-CPU, PARSEC. Our multi-threaded applications require up to 21GB/s and the multiprogrammed workloads require up to 24GB/s of main memory bandwidth. These numbers are consistent with similar workload deployments in prior studies [1, 39, 14, 72]. While there are some differences in the specific bandwidth numbers, the cross-validation indicates that we correctly identify applications with low, medium, and high bandwidth demand, which stress memory.

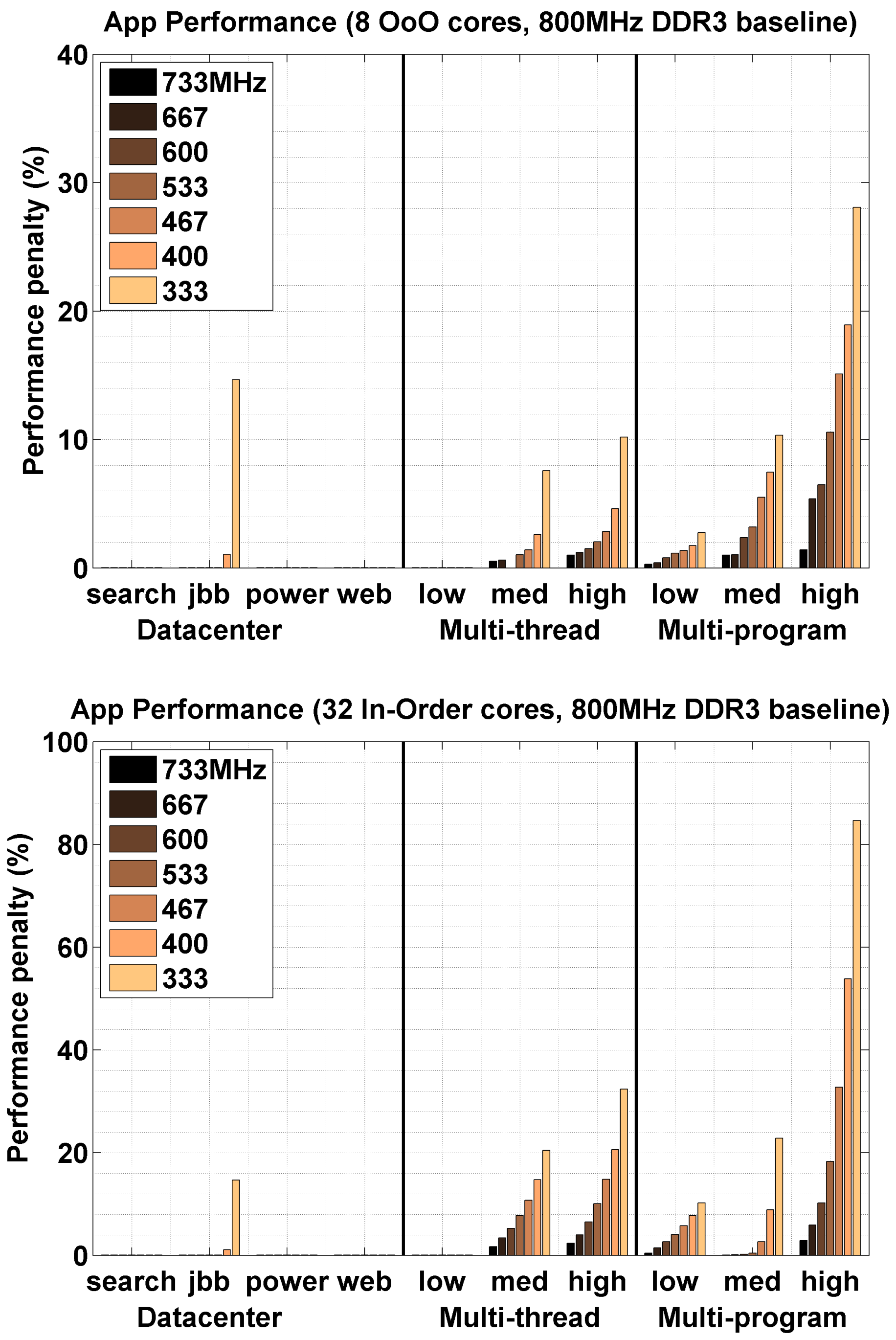


Figure 3.7: Application bandwidth sensitivity measured by performance penalties for various channel frequencies relative to DDR3-1600. Shown for Out-of-order/In-order cores with 2 channels. Groups refer to Table 3.2. Note change in Y-axis scale.

Server	DRAM Frequency	DRAM Pin Bandwidth (Gb/s)	Notation
Intel Core-2 Duo	533 MHz	1.066Gbps	DDR3-1066
Intel Xeon X5670	667 MHz	1.333Gbps	DDR3-1333
Intel Xeon E5507	667 MHz	1.333Gbps	DDR3-1333
IBM Power6	667 MHz	1.333Gbps	DDR3-1333
Sun SparcV9	667 MHz	1.333Gbps	DDR3-1333

Table 3.3: DRAM parameters for the bandwidth utilization experiments [69, 6].

3.4.4 Memory Bandwidth Demand

Architects choose the channel count to match either the memory capacity or the bandwidth demanded by applications of interest. We start with a dual-channel 16GB DDR3-1600 system using 2Gb parts ([55]). If higher capacity is needed, system architects may deploy more channels. However, adding channels will also increase a socket's memory bandwidth. For applications with low to medium bandwidth demand, adding channels costs us additional power, for the added bandwidth that we do not really need.

In Table 3.3, we notate all the server systems we mention in the analysis and experiments along with the DRAM parts they use, their bandwidth and their notations. To understand if we can reduce memory power by using a lower performance memory interface, we need to understand an application's sensitivity to bandwidth. For the dual-channel DDR3-1600 baseline, Figure 3.7 characterizes application performance penalties as channel frequency are scaled down for various processor scenarios. Confirming prior studies, most datacenter workloads do not fully exploit peak bandwidth [6, 22, 41]. Only SPECjbb incurs a 15% penalty and only after bandwidth has been throttled by 60%. Similar tradeoffs are observed for out-of-order and in-order core systems. Four channels further reduce already modest penalties, indicating that high-capacity, multi-channel systems over-provision bandwidth.

Clearly, our experiments concur with the empirical observations that datacenter applications do not emphasize memory bandwidth. Provisioning for high bandwidth makes DDR3 very energy disproportional. In the next chapters, we will describe how we can effectively tradeoff this extra bandwidth to build highly efficient memory systems for datacenters.

Chapter 4

Energy Proportionality with Existing DRAMs

Clearly, an ideal DRAM memory should not only be energy-proportional but also be available as commodity, to fit into the cost structure of datacenter servers. We first explore energy proportionality for memory systems using DRAM parts that are currently available. In this current chapter, mobile DRAM, memory designed for a high volume mobile platforms market is studied as an alternative for DDR3 memory in servers. In contrast to server DRAM, mobile DRAM has been optimized for low energy and hence is a viable candidate for energy efficiency. Examples include LPDDR2 and mobile XDR. In the following sections, the technology differences between DDR3 and LPDDR2 are first discussed. Next, suitable techniques are presented to build a viable server memory architecture using LPDDR2. Finally, the system implications of employing such memory in datacenters are presented.

4.1 Key Characteristics

Mobile DRAMs, like LPDDR2, share the same memory memory array as DDR3. However, the I/O interface is optimized to suit the needs of a low power system. This interface has much lower static power which is a very important goal in mobile platforms. LPDDR2 eliminates power-hungry delay-locked loops (DLLs) and on-die termination (ODT) from the DRAMs which are two major contributors to the active-standby power of DRAMs. This makes LPDDR2 characteristics particularly suitable to the datacenter benchmarks which have low memory bandwidth utilization and where most of the memory power is composed of active-standby static power.

LPDDR2 elimination of DLLs from the interface saves static power and also increases the opportunities to use powerdown mode. For timing, LPDDR2 uses strobe-based sampling of signals between memory controller and DRAM devices. The maximum distance of strobe from clock is bounded as tDQSCK(max). By using the worst case sampling delay, DRAM read timing is made deterministic. However, the absence of on chip DLLs challenges the maximum operable frequency of DRAM pins as we have to provision for the worst case variation in the absence of DLL feedback mechanism. Consequently, a large sampling window is needed to completely eliminate any timing uncertainty, directly affecting the operable datarate of LPDDR relative to DDR. As a result, LPDDR operates at roughly $\frac{1}{2}$ the DDR data rate.

As seen in 3.2.2, On Die Termination (ODT) is an important feature in modern DRAMs. The termination resistors cause static current to be drawn in many operating states of the device, especially when terminating to another active rank. Removing the termination from the DRAMs causes reflections on the bus that cause Inter-Signal-Interference (ISI). This lowers static power dramatically but makes high capacity memory systems challenging.

Technology Parameter	DDR2 [52]	DDR3 [53, 55]	LVDDR3 [53, 55]	LPDDR [36, 61]	LPDDR2 [58, 61]
Operating Voltage	1.8V	1.5V	1.35V	1.8V	1.2V
Operating Frequency	400MHz	800MHz	400MHz	200MHz	400MHz
Typical Device Width (pins)	4	8	8	16	16
Peak Channel Bandwidth	6.4GBps	12.8GBps	6.4GBps	3.2GBps	6.4GBps
Dynamic					
Timing (CAS, RAS, RC)	12, 40, 55ns	15, 38, 50ns	15, 38, 50ns	12, 40, 54ns	15, 42, 57ns
Active Power (read, write)	288, 288mW	270, 278mW	169, 175mW	234, 234mW	252, 210mW
Energy per bit (peak, typical)	111, 266pJ/b	70, 160 pJ/b	110, 190 pJ/b	110, 140 pJ/b	40, 50 pJ/b
Static					
Idle Power (power-down, standby)	90, 126mW	52, 67mW	30, 43mW	6, 36mW	2, 28mW
Min power-down period	84ns	90ns	90ns	20ns	20ns
Slow Powerdown Exit latency	20ns	24ns	24ns	7.5ns	7.5ns

Table 4.1: Memory technology comparison showing key latency and energy parameters for 2Gb parts. Power numbers are obtained by multiplying I_{dd} numbers from the datasheet with the corresponding V_{dd} values.

4.2 System Architecture Implications

With the absence of DLLs and ODT, LPDDR2 addresses the largest source of inefficiency in server memory: i.e. idle and termination power. However, without these elements, building high capacity memory systems with reliable signal integrity is complex. In addition, the absence of these high-speed components halves LPDDR2's maximum operable link rate or pin bandwidth. LPDDR2 chips have more I/O pins since individual memory chips, not modules, are deployed in mobile platforms, providing equal chip bandwidth as DDR3. However, for a fixed channel width to the processor (say 64 pins), the peak channel bandwidth is halved. Wider LPDDR2 can use fewer chips to supply the same number of bits for a channel, which may improve power efficiency [78, 13, 17, 81] but it complicates error correction which is discussed in Section 4.4.

4.2.1 Timing and Performance

Table 4.1 compares the device properties of DDR3 and LPDDR2 across two generations. Although the interfaces are different, mobile and server memories share the same memory core architecture and, thus, have similar timing parameters. Table 4.1 presents timing parameters for a column access (tCAS), for a row access (tRAS), and for the length of a read cycle (tRC) [36]. Since LPDDR2 operating voltage is lower that DDR3, the core latency is slightly increased but these differences are small. However, the maximum operable frequency of LPDDR2 is 400MHz compared to 800MHz of DDR3. This 2× reduction in pin bandwidth is primarily due to absence of on-chip DLLs as noted above.

The overall DRAM access latency to the processor is determined by
i) DRAM core timing which is independent of channel frequency and
ii) Bus latency which depends on operating rate but is much smaller than core timing.

In scenarios where bandwidth demand is moderate or low, core timing dominates the overall access time because it is much larger than the bus latency. For example at 800MHz transfer rate, Core timing is 38ns (tRAS) + 15ns (tCAS) while the bus time to get the critical word is only 1.25ns. In such scenarios, LPDDR2 performance is comparable to DDR3.

4.2.2 Energy

DRAM devices have 3 important powerstates: activate-read/write, active-standby, powerdown. While activate-read/write is the time period spent reading and writing to a DRAM, active-standby is that period spent with the interface in the active state but not reading/writing. Powerdown time is the rest of the time where the DRAM is powered down to a lower power state and takes some finite time (tXP) to return to the active state. In general, the powerdown state leaves the on-chip DLLs on and is referred as the Fast powerdown state as we already discussed in Section 3.3. Although fast powerdown state is the most commonly used powerdown state, DRAMs also provide a Slow powerdown state where the DLLs are frozen at a set phase, effectively halting the phase tracking capability of DLL. So, it is undesirable to use them for long periods of time. However, the incoming clocks are still sampled at every edge in the receivers. Even deeper powerdown states turn off the DLLs and clocking circuitry also, which saves lot more power but need the DLL to fully relock.

The total DRAM energy is determined by power in each mode multiplied by the time the devices spend in each powermode. Table 4.1 indicates that while LPDDR2 has lower power compared to the latest DDR3 revision, its active standby and powerdown are significantly lower (20 $\times$) than those of DDR3. This will result in much better standby states, which was the major inefficiency in using DDR3 for platforms running datacenter applications. This difference in standby power is the reason that the energy/bit as a function of bandwidth utilization shown in Figure 4.1 is much better for LPDDR2. Even at the peak utilization, Table 4.1 indicates that energy per bit is much smaller for LPDDR2, even though the active power numbers for DDR3 and LPDDR2 are comparable and larger than LVDDR3. DDR3-1600 amortizes bits better than LVDDR3-800 at peak rates, resulting in better energy/bit. On the other hand, the larger device width of LPDDR2 results in fewer chips being activated and read from a rank. In addition, LPDDR2 does not have any I/O termination power.

In contrast to DDR3 and LVDDR, mobile memory is nearly energy proportional; energy per bit is almost flat as utilization varies. At peak and typical utilization, LPDDR2 consumes 40 and 50pJ/bit, respectively. Compared to DDR3 and LVDDR3 at low utilization (e.g., 20%), LPDDR2 sees a 4-5$\times$ energy reduction in exchange for 2$\times$ lower peak bandwidth. For applications with modest bandwidth demands, this is an excellent trade-off.

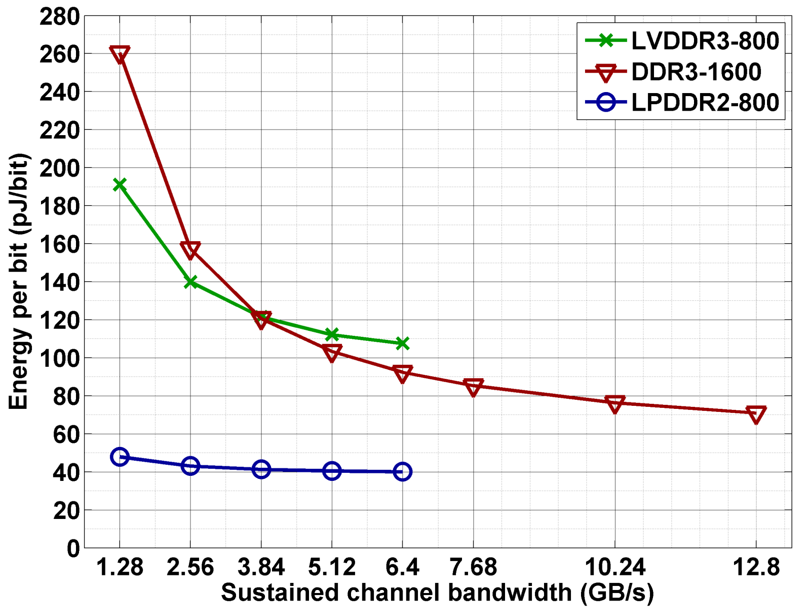


Figure 4.1: Energy per bit (mW/Gbps = pJ/bit) with varying channel utilization. Assumes four ranks per channel and 3:1 read to write ratio. DDR3 is x8-1600 (Peak BW = 12.8GB/s). LPDDR2 is x16-800 (Peak BW = 6.4GB/s).

4.3 Architecting Mobile DRAMs

While Section 4.1 shows the potential efficiency of LPDDR2 in servers, we must address several significant challenges posed by its power-efficient interfaces. First, non-terminated LPDDR2 memory chips increase vulnerability to inter-symbol interference, which complicates the design of high-capacity memory systems [66]. Second, wide LPDDR2 chip interfaces may increase error correction costs.

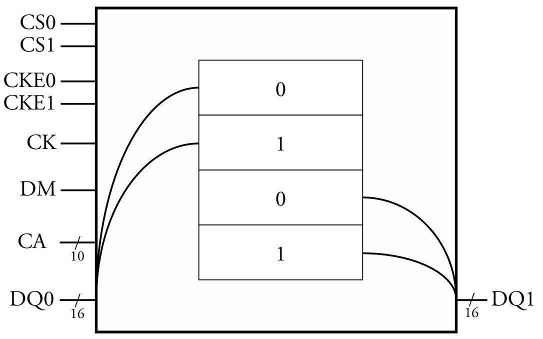
To address these issues, this section presents a new channel architecture using commodity LPDDR2 devices. A board design is combined with stacked dies to obtain DDR3-competitive capacities with acceptable signal integrity. To further increase capacity, we propose a new module architecture for LPDDR2, which draws lessons from registered/ buffered DDR3 and allows us to further scale channel and socket counts [3, 19]. This section ends with a proposal for handling error correction with wide memory parts.

4.3.1 Channel Architecture

LPDDR2 devices are edge-bonded, making them suitable for capacity stacking. Mass manufacturing of four-stacked LPDDR2 dies is already available for mobile systems [54]. Figure 4.2(a) shows a Micron LPDDR2 dual-rank package with four LPDDR2 dies, the basic block of our proposed architecture. If we consider 2Gb (128M $\times$ 16) LPDDR2 devices, a 64-bit channel would have 4 devices and be limited to 1GB/channel. This is not sufficient for high-capacity server memory that is necessary in datacenters.

To increase capacity, we can architect a given capacity with some combination of channels and packages per channel. Fewer channels with more packages per channel makes the system becomes less expensive but performance and signal integrity suffer.

To partially get around this tradeoff, we propose a novel architecture called Dual-Line Packages (DLPs). As shown in Figure 4.2(b), DLPs have closely spaced LPDDR2 packages on both sides of a board. The key observation is that by placing the devices close together, it is possible to reduce transmission line reflections. The key problem in multi rank channels was that there were multiple capacitive loads on the bus which caused reflections at more than one junction. Thus, since reflections travel in the opposite direction from the signal, it takes two reflections to interfere with the forward signal. However, if devices present a single point loading as shown in Figure 4.3, the reflections would greatly be limited as the reflected waves are absorbed by the termination in the controller and not re-reflected. Following this design methodology, four ranks are striped across multiple packages with two chips from each package sharing an output pin to controller. It is ensured that devices multiplexed on the same pins have short traces apart from each other. As LPDDR2 devices are already stacked close apart inside packages, this is feasible.



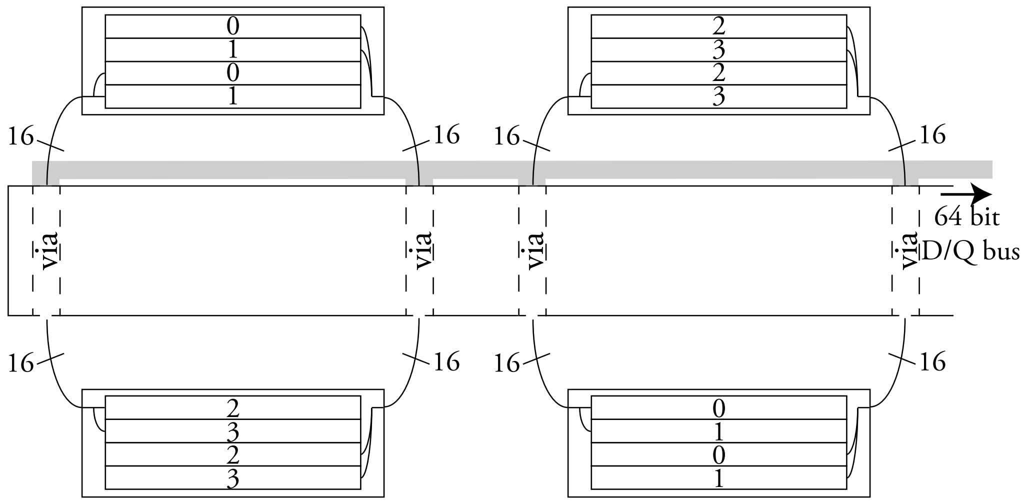


Figure 4.2: (a) LPDDR2 package with four x16 2Gb devices. (b) Dual Line Package (DLP) architecture with four packages per channel. Each chip select (CS) signal is shared by two devices. Each of the two sets of 16 pins are multiplexed by two devices. Constituent chips of the same rank are indicated by the same rank number (e.g., 0 to 3).

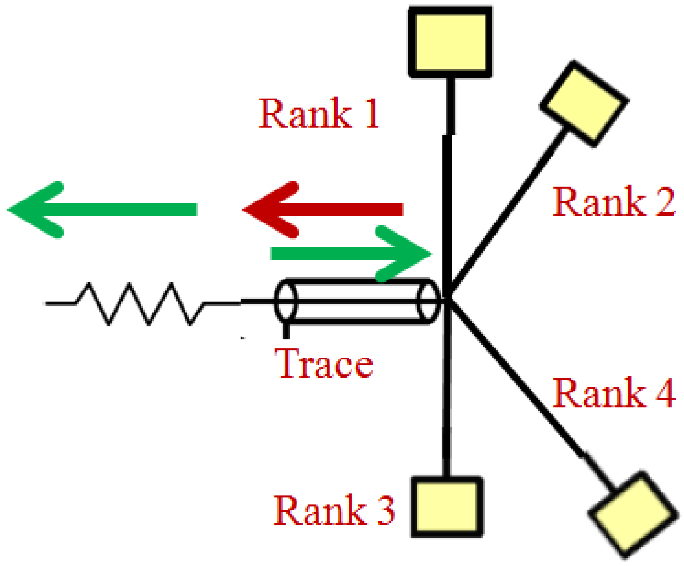


Figure 4.3: Design methodology of LPDDR2 architecture showing devices placed close together, multiplexing the same pins creating a single load. As the transmission stubs are short, they do not create individual reflections. So, the reflections travel back to the source, where they are absorbed

Since two devices share a Chip Select (CS) internally, they are placed in the same rank as they are activated together. The two sets of x16 bond-wires from a package are multiplexed with the mirroring wires from the package on the opposite side of the board through an on-board via. Since each edge on the via is x16, four such traces form the 64 bit output DQ bus as shown in Figure 4.2(b).

As each package has only two devices from the same rank, only two dies can have active column operations at a given time in the package. Other active-idle dies dissipate very little power and ensure thermal constraints are satisfied. With four ranks, each with 2Gb x16 devices, a total capacity of 4GB on channel is obtained, which is DDR3-competitive.

4.3.2 Signal Integrity

Memory Controller and DRAMs communicate through high speed links and these links are sensitive to the signal loading conditions. Since LPDDR2 eliminates some of the interface components that enable precise operation of the links at high speed, we need to analyze our proposed LPDDR2 architecture for feasibility at designated speeds.

To demonstrate this, we first study signal integrity of the link in our LPDDR2 architecture. Lumping DRAM dies together into a single package by placing them close apart moves the impedance discontinuity problem to the end of the link. Each DRAM I/O pin presents a load on the bus, causing a small impedance discontinuity and a reflection. Lumping these loads on the end of the link makes the discontinuities and reflections larger. As load increases, we will reach a point where the bus no longer functions. To study this, SPICE models of the PCB transmission lines and the bond wire inductance and the ESD/pad/driver capacitance associated with the DRAM package/pin (Figure 4.4) are simulated. The demonstrated simulations place the on-board via to place all the four devices together and show the design methodology. However, we have also simulated a link that separates two devices each on either side of the via and the results are similar. Few energy losses are modeled which is generally a worst-case situation for reflections. Eye-diagrams are used to understand link functionality, by exciting the line with a random stream of bits at the input and measuring if the output waveform presents discernible 1's and 0's or an "open-eye".

In Figure 4.4, the data eye diagrams for write data (controller to DRAM) and read data (DRAM to controller), clearly show open eyes in both directions, which means this type of communication is possible. Write data is much cleaner than read data which is expected. Consider link interfaces at the transmitter (i.e., memory controller) and receiver (i.e., DRAM). During writes, the link is terminated at the transmitter end, so the quality of the impedance near the receiver is not that critical.

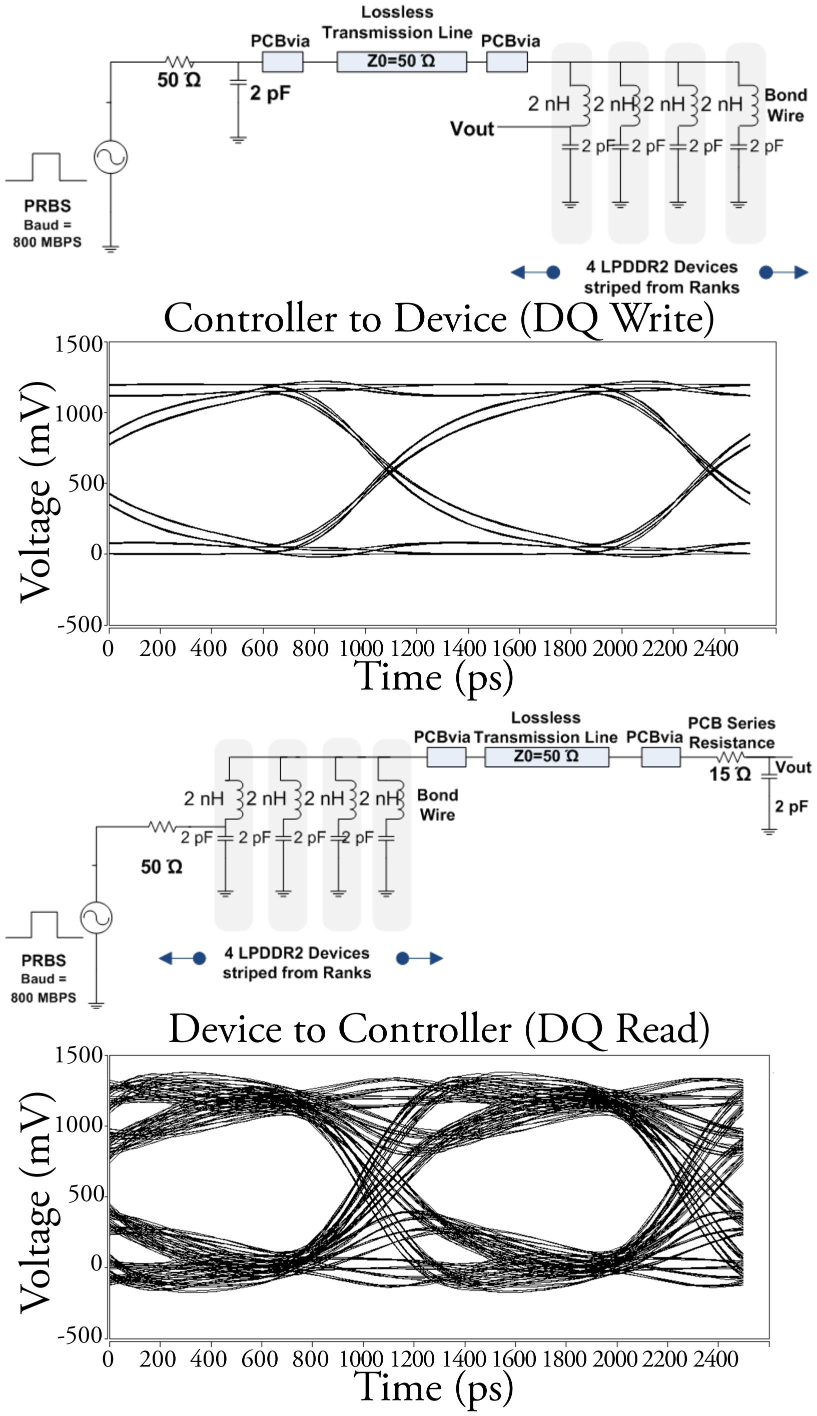


Figure 4.4: LPDDR2 signal integrity SPICE simulations with four packages per channel, demonstrating open eyes for controller-memory and memory-controller links. Signal integrity demonstrates feasibility of building the proposed capacity.

Discontinuities at the receiver will cause reflections, but these reflections travel back to the terminated transmitter and are absorbed: none of these new reflections will be able to reflect again and make it back to the receiver. Thus, even though the large load affects impedance at the receiver, the received signal for write data is good.

Read data looks worse since now the termination is on the transmitting DRAM, and the reflected data at the cleaner controller side is re-reflected back to the DRAM where it gets reflected back to the controller. Fortunately by placing the DRAMs close together, their reflections still leave a signal with good noise margins. The CA bus operates at DQ speeds but is more heavily loaded. However, CA communication is only in one direction (controller to DRAM), somewhat alleviating signal integrity challenges. Thus, the simulations indicate the feasibility of grouping four chips into a package and placing four packages onto a shared channel, despite using non-terminated LPDDR2 dies.

4.3.3 Enhancing LPDDR2 Capacity with Buffers

The signal integrity analysis shows that stacked modules support 4GB per channel using 2Gb x16 LPDDR2 chips. To further increase system capacity, we could add more channels. While some applications would benefit from the parallelism of more channels (Figure 3.7), many emerging applications do not. Additional channels also introduce complexity in controllers [3] and we cannot tune capacity/bandwidth ratios since adding a channel simultaneously increases both capacity and bandwidth. This bandwidth over-provisioning is less expensive for energy-proportional LPDDR2 than for DDR3 because LPDDR2 has small static power across memory utilizations which prohibits memory power to grow with the over-provisioning. But it still requires pins and chip area and so scaling channels is difficult. To solve this problem, we increase LPDDR2 single-channel capacity by following the example of Load-Reduced (LR) DDR3 DIMMs and create buffered LPDDR2 DIMMs, which buffer and re-time both

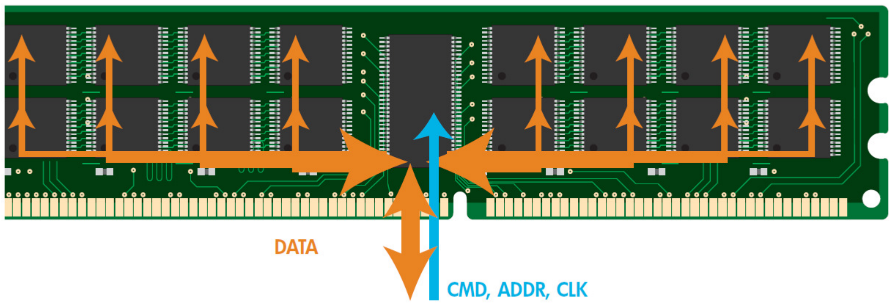


Figure 4.5: Schematic of an LRDIMM module.

DQ and CA buses to ensure signal integrity.

Figure 4.5 demonstrates a schematic of an LRDIMM memory module which has a memory buffer on the front end of DIMM with multiple DRAM ranks mounted on a module [56]. Conceptually, LRDIMM is similar to Registered DIMM (RDIMM). RDIMM retimes the command, clock and the address signals going from the controller to the DRAMs. This will help reduce the load, enabling one to increase the number of memory ranks, but increases the latency by a cycle. In addition to command and clock line buffering, LRDIMM fully buffers all the data pins enroute the memory controller, to reduce the load on the data pins also, further increasing DIMM capacity. This isolates the electrical load of the DIMM with multiple ranks from the bus interface to the controller. The buffer duplicates both data and command lines to provides point-point links to ranks on the same module. This enables the link to operate at a higher speed for a given capacity. Alternatively, we could support more capacity for a given operating speed, thus decoupling the tradeoff between DRAM capacity and operating speed.

LRDIMM also supports a feature called Rank Multiplication. This allows multiple physical ranks to appear as single, larger, logical rank to the memory controller. This is made possible by using the extra row address bits during the activate command as sub-rank select bits. However, read and write commands do not need these additional

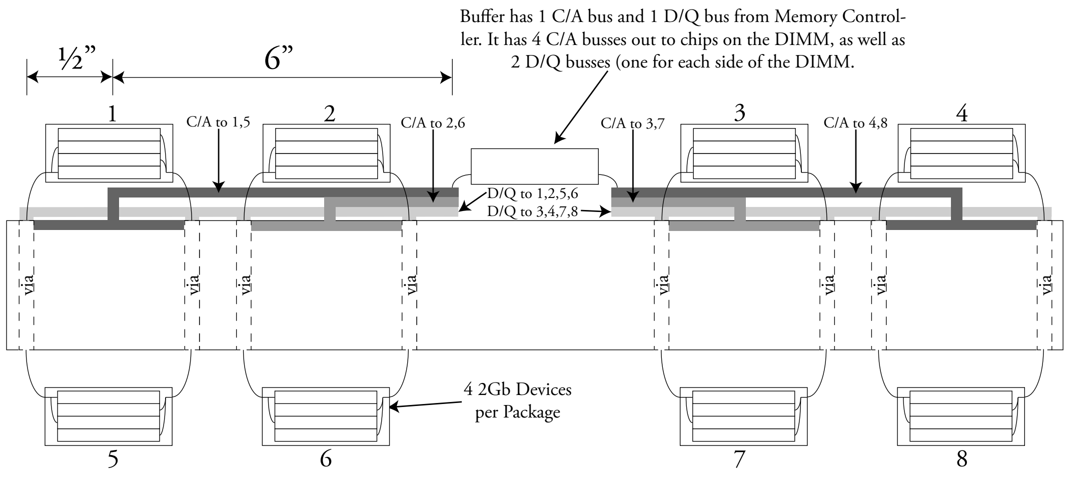
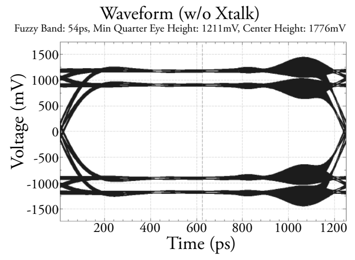

Buffer has 1 C/A bus and 1 D/Q bus from Memory Controller. It has 4 C/A busses out to chips on the DIMM, as well as 2 D/Q busses (one for each side of the DIMM.
½”
6”
1
2
3
4
C/A to 1,5
C/A to 2,6
C/A to 3,7
C/A to 4,8
D/Q to 1,2,5,6
D/Q to 3,4,7,8
via
4 2Gb Devices per Package
5
6
7
8

Figure 4.6: Channel architecture with load-reduced buffer that allow multiple LPDDR2 packages. Double DQ bus to provide point-point buffer-to-device connections. First line communicates with packages 1, 2, 5, 6. Each via has 16 bits to DQ that are multiplexed by two devices from the same package. Quadruple CA bus, reducing load to two packages per CA line.

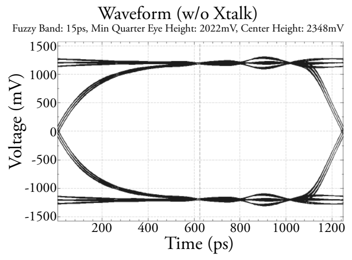
sub-rank bits as this information is already stored during the activate in the buffer. Rank multiplication supports a maximum of 8 physical ranks on an LRDIMM.

Following the above principles, a buffer is positioned between the channel and DRAM chips for an LPDDR2 LRDIMM. LPDDR2 chips and buffers communicate via point-to-point links to ensure reliable communication even as capacity increases, reducing channel load and permitting a larger number of stacked chips. Such buffers can double LPDDR2 capacity per channel at modest latency and pin cost.

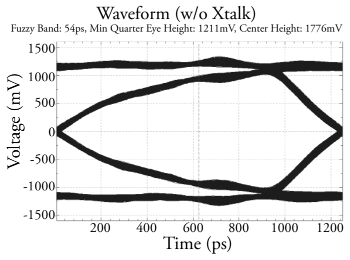
Figure 4.6 illustrates the new LPDDR2 module architecture with Load Reduction. The buffer has 64 DQ and 14 CA pins on the input side. These pins need to be duplicated to provide point-point links to the multiple LPDDR2 packages on the other side of the board. Experimental simulations show that doubling channel capacity from 4 to 8GB without compromising signal integrity, needs the output DQ bus be



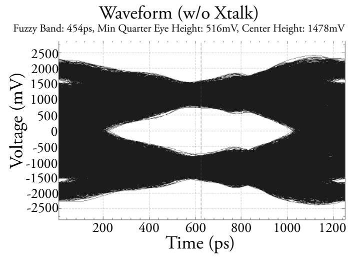
(a) Controller to On-DIMM Buffer



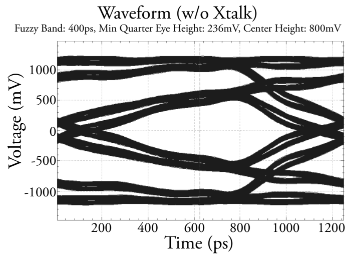
(b) Buffer to Controller



(c) On-DIMM Buffer to Device DQ



(d) Device DQ to On-DIMM Buffer



(e) On-DIMM Buffer to Device CA

Figure 4.7: Signal Integrity analysis for Load-Reduce Buffers to increase channel capacity using LPDDR2 Package Modules. Figure 4.7(a) to Figure 4.7(e) show open eyes (for reliable communication) using On board buffer.

replicated 2× while the CA bus needs to be replicated 4× due to its higher load. The replicated pins on the device side are time-multiplexed onto the controller bus via the buffer.

These new DLPs proposed are used with packages on both sides of the board. Figure 4.7 shows the Signal Integrity (SI) simulations of the proposed 8GB architecture with 8 packages per channel. The signal and timing margins are all sufficiently high for the controller-buffer and buffer-chip lines that allow for reliable communication in the relevant directions. Eye diagrams for both data and command lines are depicted and all of the diagrams have sufficient timing and voltage margins to ensure reliable operation.

The proposed load-reduced buffers static power overhead is small since termination is not required for LPDDR2. However, PLLs for clock re-timing incur a small active power cost. Registered DDR3 modules are very common in servers and LRDIMMs are increasingly becoming popular. It is envisioned that this architecture for buffered LPDDR2 DLPs will be just as attractive, but with excellent energy-proportionality.

4.4 Error Correction for LPDDR2

Modern DRAM systems are susceptible to errors, both hard and soft. To make computer systems reliable, memory system designers have used Error Correcting Codes (ECC) with Single Error Correction Double Error Detection (SECDED) [72, 78] to detect and correct memory bit errors that commonly arise from DRAM cell faults and stuck-at faults. These techniques were originally developed for DRAMs with single bit outputs, where SECDED would protect from chip failures arising from bits as well detecting double bit errors. This led to using 8 parity bits to protect a 64 bit data word, and the development of a 72 bit interface which has become a standard. Thus, much work has been done on protecting wider parts within this form factor. As the width of the DRAM increased, Single Symbol Correct, Double Symbol Detect(SSCDSD) codes were used, that could correct a multi-bit symbol, rather than a single bit [7].

To employ DRAMs in servers, we often need a high degree of reliability that can protect the system from the failure of an entire chip. Referred to as “Chip-kill”, the scheme is an SSCDSD scheme and corrects the failure of an entire DRAM chip while detecting the failure of two chips. There are multiple ways of implementing chip-kill, either by interleaving error correcting codes in multiple ranks [30] (ex: IBM x series) or by using stronger codewords [2] (ex: AMD Opteron). The former ensures that, for a given codeword, not more than one bit is used from any chip. For instance, 4 ranks can be used to distribute 4 bits from an x4 chip. While the solution is simple, it can lead to energy inefficiency. For example, to recover from a chip error, 72 bits need to be interleaved across 288 bits which make the ECC words. This results in activating 72 DRAM chips for every transaction while also making it even more difficult for x8 and x16 as they require 8 and 16 ranks respectively, to be activated.

Alternatively, using the B-adjacency algorithm, b bits can be protected through the use of two b-bit wide symbols [34]. While this code uses more parity bits, it can be implemented in the same overhead by protecting larger blocks of memory; percentage overhead decreases as block size increases. These types of algorithms have been used to protect DDR3 x4 devices using a block size of 128 bits, and have been extended to protect x8 devices.

The wide x16 LPDDR2 interface causes two challenges to apply ECC for LPDDR2 chips. The first is due to quantization: 72 bits is not divisible by 16 and we cannot naturally provide data using the standard memory interface. The second is the larger number of bits needed to protect against a loss of 16 bits. These challenges are similar to those in architectures that access fewer chips for energy-proportionality [1, 72, 81], and we leverage some of their techniques to address these issues. There are four ways to deal with the quantization issue: make x18 LPDDR2 DRAM, increase the overhead and create an 80/64 module, create a 144/128 bit channel, or embed the ECC data into the memory space. While the first solution seems simple, it would generate a different DRAM part type, which is expensive.

Using x16 parts, we can either use one to protect four or eight other DRAMs. In the former, we double the parity overhead, which increases the cost of the memory by 12.5% but makes protecting from chip-kill much easier. In the latter, we merge two channels into a single channel with twice the bandwidth and larger block transfers. The last option is to maintain a 64 bit interface and embed the ECC into the memory space instead of sitting in disparate devices [81, 78]. The memory controller performs two accesses, one for data and one for ECC. Embedded ECC does not require dedicated chips for parity and is energy-efficient. However, memory system capacity falls as ECC is embedded in the data space and memory controller complexity increases as it must map ECC words.

This basic idea is quite flexible, and can support multi-tiered error correction, [78] where the OS can determine which pages in memory only need error detection (i.e., clean pages) and which need protection and correction. Furthermore, since errors are infrequent, the correction data will be accessed infrequently, reducing overheads.

In particular, for SSCDSD correction schemes, we need 2b bits for correction of b consecutive bits in an ECC word and an additional b bits to detect an additional b-bit error. Thus, we need 3b total bits for SSCDSD. Tier-based error correction schemes further help decouple error detection and correction and need only 2b bits for the detection of double symbol errors [78]. Infrequent ECC correction bits are rarely accessed and not even allocated by the controller for clean pages reducing parity by b bits along with smaller energy overheads. These reduced 2b parity bits can now be embedded in the data space [81] making chip-kill/SSDDSD viable for x16 LPDDR2 devices.

Thus embedding ECC words provides efficient correction and detection schemes for LPDDR2. In addition, if a 160/128 x16 LPDDR interface were to be available in the future, we can not only provide SECDED but also SCDDCD on x16 devices using the tier-approach with low parity of 2b bits and energy costs. For example, in AMD Opteron with a 144/128 memory interface, 128 bit ECC words (2 ranks = 32 x4 DDR devices) are protected using 16 parity bits (4 x4 ECC chips) [1]. Similarly for LPDDR2, we can protect 128 bit data (2 ranks = 8 x16 LPDDR devices) with 32 parity bits (2 x16 ECC chips). So even though x16 parts lead to a modest increase in the number of parity bits, ECC is still feasible.

The above approaches are important for modern web applications that map large, slowly changing datasets in their memories. Thus, while LPDDR2 device width causes some challenges for ECC, some form of multi-tiered error correction can be implemented in these systems to support chip-kill with modest overhead.

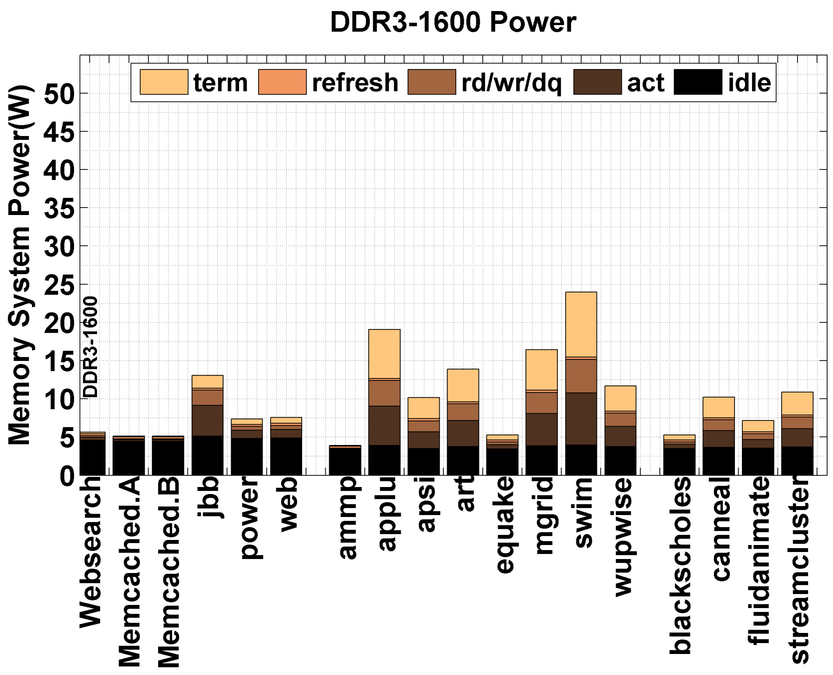
4.5 Evaluation of LPDDR2 systems

Section 4.3.3 describes a methodology that can build a 16GB LPDDR2 system using Load Reduced DLPs, with two channels per processor socket. Compared to DDR3 of the same capacity, LPDDR2 reduces memory system power by 3-5$\times$ across the applications (Figure 4.8(a), Figure 4.9(a) versus Figure 4.8(b), Figure 4.9(b)).

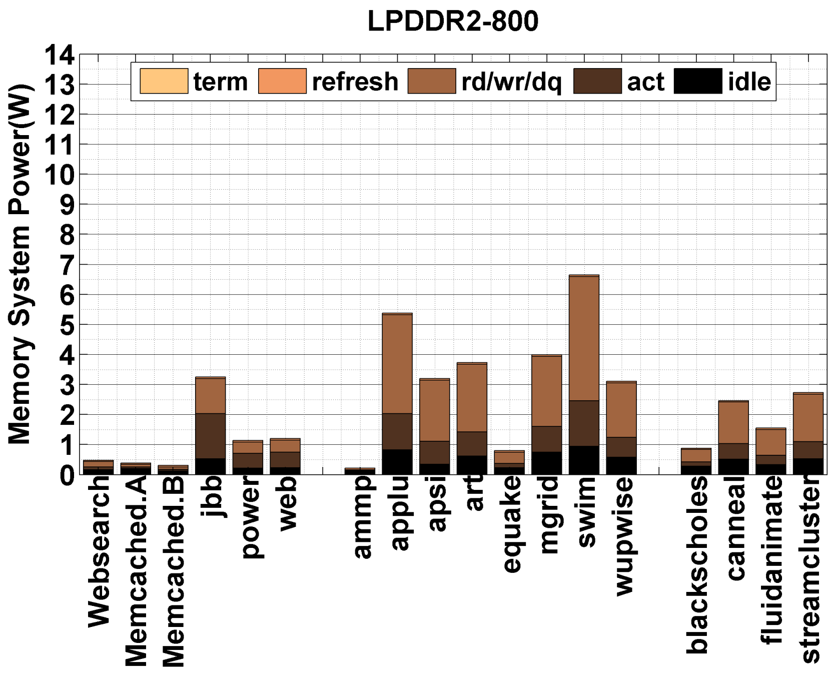
Strikingly, LPDDR2 total power is often less than DDR3 idle power. In particular, the idle power of LPDDR2 is much smaller due to better idle currents, while the lack of termination eliminates idle I/O power. LPDDR2 also benefits from the fast powerdown modes that further reduce background power and that, unlike DDR3, do not require slow DLL re-calibration, opening more opportunities to use the powerdown mode.

In addition to these lower idle overheads, LPDDR2 dynamic power also falls. The dynamic power of an LPDDR2 chip is also lower than DDR3 as the number of LPDDR2 chips accessed per a transaction is smaller due to its wider device width. In addition, due to lower activate current and voltage, the cost of activates, reads, and writes relative to DDR3 is also reduced. Datacenter benchmarks, such as websearch and memcached, show significant power reductions due to their low bandwidth requirements. Power falls from approximately 5W to well below 1W, a 5-6$\times$ reduction. Other applications, with diverse memory behavior, also save memory power, depending on memory bandwidth requirement.

Multiprogrammed benchmarks show a power reduction in the range of 3.3$\times$ for 482.sphinx3 to 16.8$\times$ for 465.tonto. The savings are proportional to bandwidth, with an average power reduction of 4.4$\times$. Significant contributors to this reduction are very

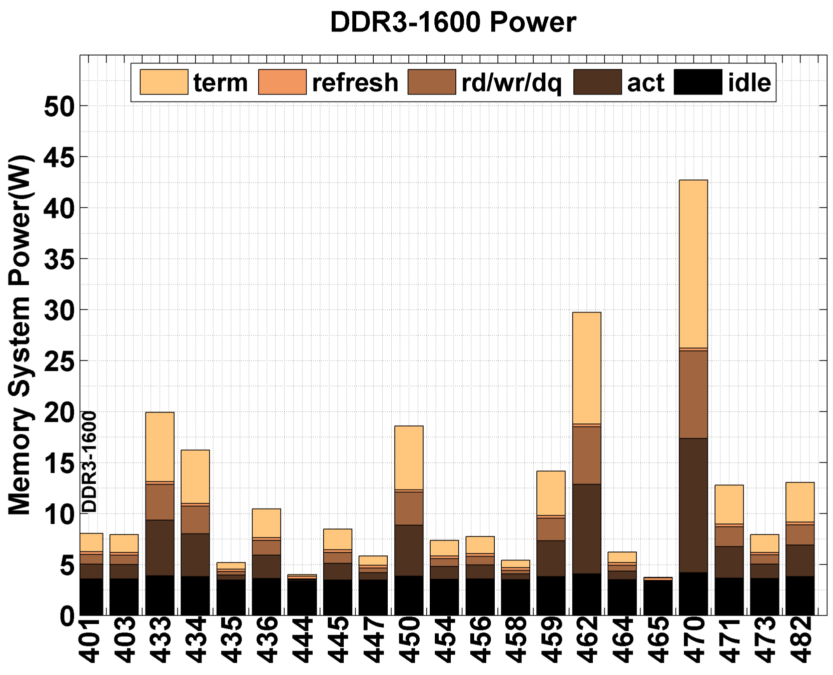


(a) DDR3-1600 power.

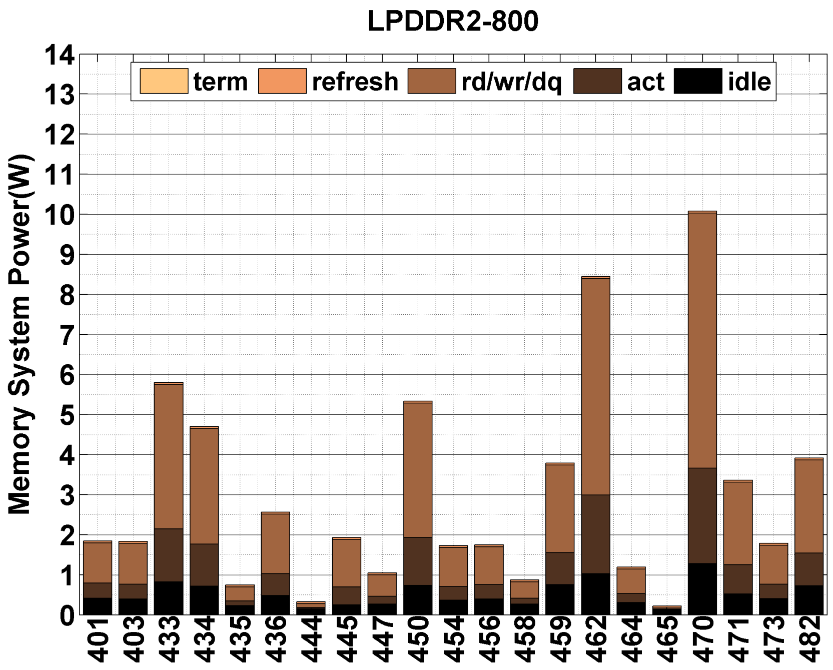


(b) LPDDR2-800 power.

Figure 4.8: Datacenter, Multithreaded workloads with 16GB. Idle, termination power are significant in DDR3. See the 4× change in y-axis scale for LPDDR2 results.

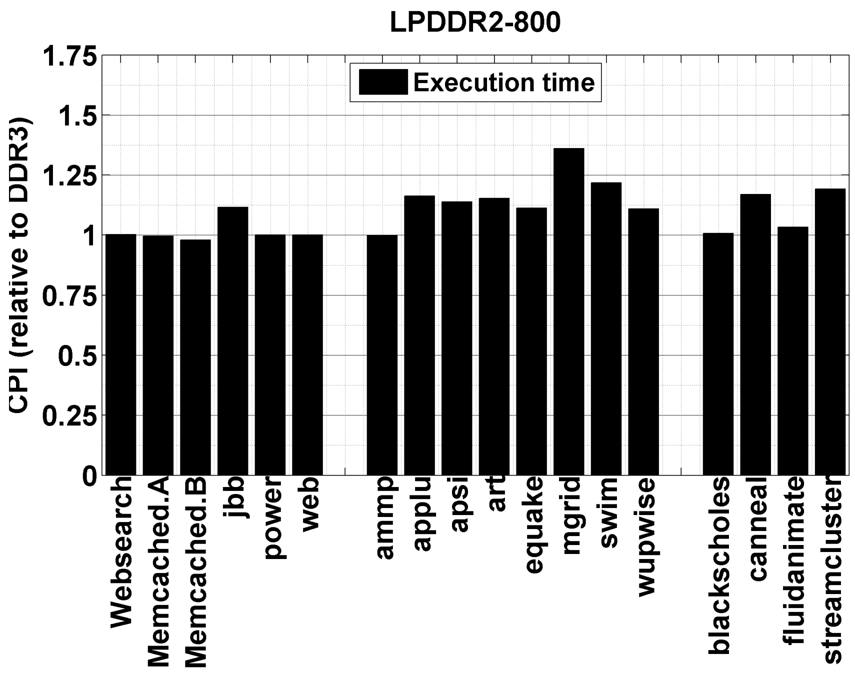


(a) DDR3-1600 power.

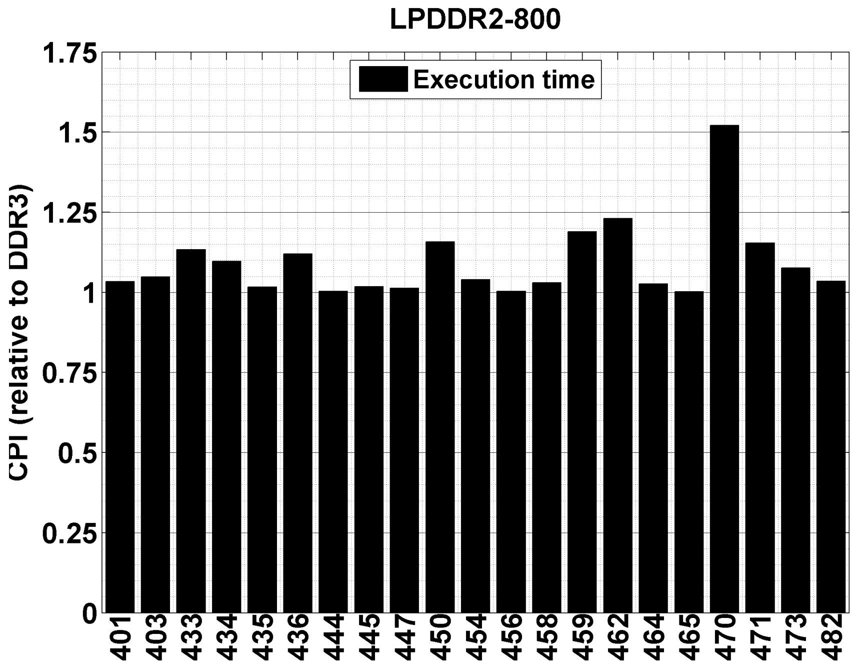


(b) LPDDR2-800 power.

Figure 4.9: Multiprogrammed workloads with 16GB. x-axis label numbers correspond to Table 3.2. See the 4× change in y-axis scale for LPDDR2 results.



(a) LPDDR2-800 performance for Multithreaded, datacenter applications.



(b) LPDDR2-800 performance for Multiprogrammed applications.

Figure 4.10: Performance results for LPDDR2-800. The execution time of the applications is demonstrated, relative to DDR3-1600 baseline. Multiprogrammed benchmarks' x-axis label numbers correspond to Table 3.2.

low active-idle states and a much better power-down state and low dynamic power. Similarly, multithreaded benchmarks show an average power reduction of 4.3× in a range of 3.3× for art to 18.8× for ammp.

Most applications realize significant power savings with modest performance penalties. Figure 4.10(a), Figure 4.10(b) show application time on an LPDDR2 memory system normalized to that on a DDR3 memory system. For instance, the performance of datacenter workloads like websearch, memcached and SPECJbb, SPECpower, and SPECweb is barely affected when DDR3 is replaced by LPDDR2. For conventional workloads (SPEC-CPU) multiprogrammed mixes, and PARSEC, SPEC-OMP, the performance impact varies from nearly zero for a majority of workloads to a worst case of 1.55× for high bandwidth applications.

Even when an application is stalled waiting for memory, it may not have sufficient memory-level parallelism (MLP) to benefit from additional sequential memory bandwidth. In such cases also, LPDDR2 will be beneficial. In addition, many emerging applications in datacenters often have complex dependencies between load-stores which limit outstanding memory misses even on an aggressive Out-of-Order processor. Note that any performance penalties arise entirely from reduced bandwidth and not latency, which is unchanged when adopting LPDDR2 over DDR3. The applications affected by the lower LPDDR2 bandwidth generally have bursty cache misses that introduce channel contention.

Thus, the current chapter demonstrates methods to build highly energy-proportional and high capacity main memory systems using LPDDR2. The performance and energy analysis show large wins for platforms running datacenter applications. However, for other conventional applications that require high memory bandwidth, LPDDR2 systems could degrade performance. To make its applicability more general, we would need to make LPDDR2 sustainable at high datarates while ensuring that we retain its large 5× energy savings.

Chapter 5

Energy Proportionality with Future DRAMs

In the previous chapter, we detailed methodologies to harness existing commodity DRAM parts like LPDDR2, to build high capacity memory systems that are both energy proportional and energy efficient. We demonstrated degrees of efficiency, while outlining the challenges of building such systems and approaches to overcome them. The large energy benefit has the potential to motivate datacenter operators to deploy LPDDR2 systems.

On the other hand, DRAM manufacturers will also need to adapt future DRAMs to address the growing disconnect between server needs for main memory and the operating characteristics of existing DRAMs. Energy proportionality for memory systems will emerge as an even more important issue and future DRAM interfaces could be better suited to improve proportionality. Such methodologies could also navigate the challenges of LPDDR2 systems at high bandwidth requirements and make them applicable for a wider class of workloads.

In the current chapter, we examine strategies to build such future DRAM parts more tailored for datacenters and study how they can be applied to LPDDR2. We seek to fundamentally rearchitect the memory interface for future DRAM generations.

5.1 Existing Challenges

Clearly, LPDDR2 presents the best starting point for future DRAM parts for datacenter applications. Its principal limitation is the large performance penalty for bandwidth sensitive applications. In order to make it more general and suitable to various other workloads, we need a higher bandwidth interface.

This section outlines challenges in building a high speed interface that is also energy efficient. In particular, Chapter 3 identified that the biggest bottleneck to DRAM energy proportionality is interface power. Attacking this problem requires us to build both a low energy high speed interface, reducing active idle power and creating an interface that can be powered on and off quickly to minimize the active idle time.

Section 3.2 and Section 3.3 have already demonstrated how DLLs, clocking and ODT compose a significant fraction of active-idle power. For example, DDR3 active-idle current is 2× that of LPDDR2 and much of this difference is attributed to the interface.

5.1.1 DDR Powermodes

In addition, DLLs affect efficiency in a second scenario, when the DRAMs are fully idle. In this mode, the DRAMs are in a powerdown mode. More efficient modes have higher powerup latencies (e.g., self-refresh in Table 3.1). While this state seems energy-efficient and is desirable for energy-proportionality, the next reference pays the cost as the DRAM spends tDLLK=512 active memory cycles, powering up the interface. This is a lot of energy. In addition, applications slow down, as indicated

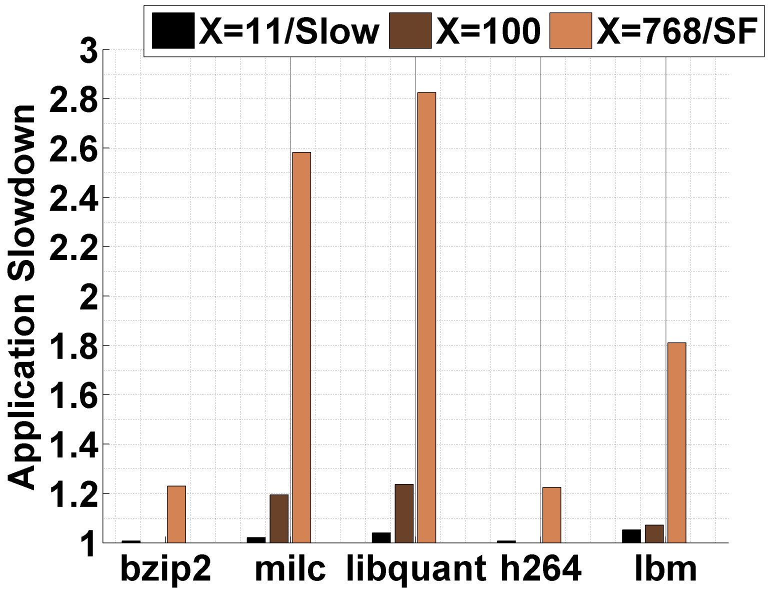


Figure 5.1: Performance sensitivity to dynamic power down modes at different exit latencies, denoted by X. "Slow" refers to slow powerdown and "SF' to Self-Refresh.

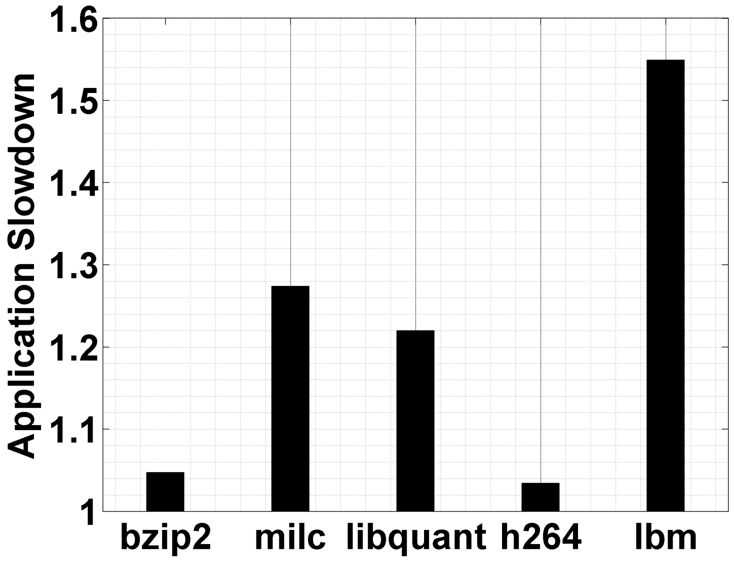


Figure 5.2: Performance sensitivity to static BIOS programming for disabling DLLs.

in Figure 5.1. Thus, existing DRAM interfaces impose unattractive performance and power tradeoffs.

Static mechanisms to reduce interface power fare no better. We can configure the memory mode registers (MR) in the BIOS [55], eliminating DLLs but this imposes performance penalties. First, the peak data rate is halved as channel frequency must be lowered to ensure signal integrity. Furthermore, without DLLs, timing is less certain and controllers must assume worst-case margins (i.e., tDQSCK=10ns [55]). Conservative timing increases critical word latency, affecting application performance as shown in (Figure 5.2).

Due to these punishing trade-offs, memory controllers invoke power modes conservatively. Modern controllers recommend a powerdown threshold no lower than 15 idle memory cycles [32]. Figure 5.3 shows the percent of time the DRAMs stay in each power state for this aggressive threshold (A), a moderate (M) threshold 10$\times$ larger, and a conservative (C) threshold 100 $\times$ larger. With such thresholds, up to 88% of memory time is in active-idle. Figure 5.4 shows the potentially large energy savings if the powermode limitations were addressed even for DDR3 parts, enabling the aggressive use of deep powerdown modes. Such efficiency gains would be even larger when applied for LPDDR2 like parts.

5.1.2 Future DRAMs

As noted above, the ideal future DRAMs would have LPDDR2 like interface which is highly optimized for low static power. However, to regain the high speed operation, we would also need the DLL operation on the interface. But simply putting the DLLs will harm the energy efficiency of LPDDR2. However, if it can be ensured that the DLLs stay ON only during transmission, the energy efficiency can be retained. In addition, the DRAM bus transmission time is very small even for bandwidth sensitive workloads since much of the is spent on the DRAM array lookup. This is even smaller

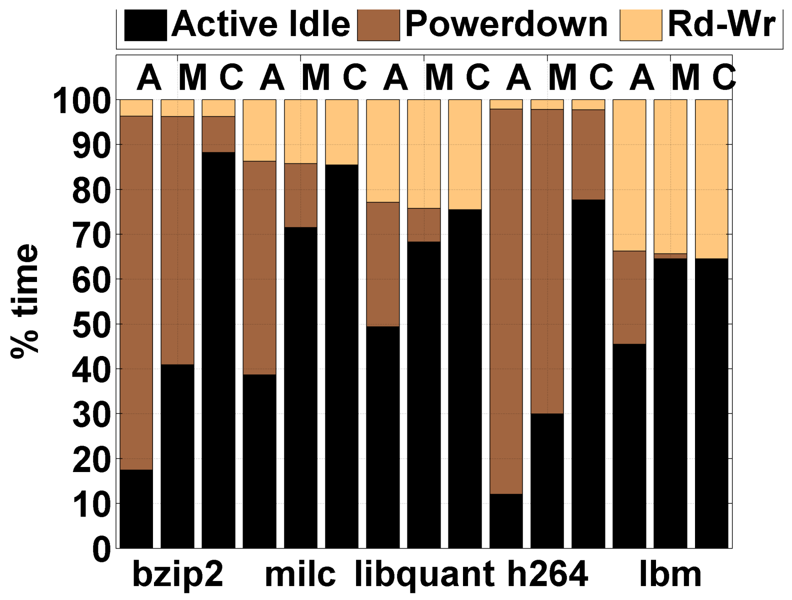


Figure 5.3: Memory time breakdown with aggressive (A), moderate (M), and conservative (C) thresholds

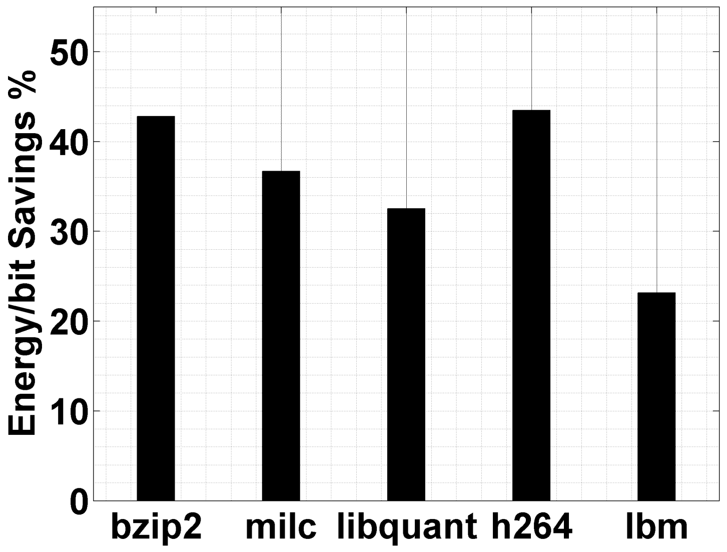


Figure 5.4: Potential efficiency from new power modes.

in multi rank systems as the active time of a rank decreases linearly with number of ranks. Consequently, we need short powerup DLL powermodes to retain LPDDR2 like energy efficiency while enabling high-speed operation. However, as analyzed above, existing DLL powermodes are very unattractive with large wakeup times from the DLL-off powermodes.

Clearly we need much smaller wakeups if we were to exploit the energy efficient powerdown modes. To evaluate the cost of different wakeup latencies, we need a more quantitative understanding of the nature of memory activity. which needs us to model the interarrival times of the memory references.

A memory bus which has bursts of activity followed by a fully idle bus and one which is uniformly lightly-utilized have the same average bus utilization but very different wakeup overheads. Clearly, to effectively exploit fine grain power down modes, we need methodologies to distinguish such differences.

5.2 Understanding Memory Activity

Since the precise benefits of fast exit power modes depend on the interaction between memory activity and the exit latency, we study it in two ways. First, we capture memory request inter-arrival times from emerging big data applications to study the memory requests' interarrival behaviour. Then, we probabilistically model memory requests in order to understand fundamental energy-delay trends and use the statistics to evaluate performance overhead.

5.2.1 Characterizing Emerging Applications

The nature of computing has changed significantly in the last decade [18] and many emerging datacenter applications have memory behavior that is not well understood. While a prior study quantifies memory idleness in websearch [50], it does so for coarse, 100ms, time periods. At this granularity, which is many orders of magnitude larger than device access times (e.g., 100ns), understanding application requirements for power modes is difficult. To address this, we first setup experiments to quantitatively characterize the memory behaviour of applications.

To study memory behavior at fine granularity, we use a custom simulation infrastructure with x86 instrumentation and a built-in scheduler [65] to benchmark a spectrum of real applications. From the spectrum of emerging data driven workloads, we characterize three representative workloads: memcached for distributed memory caching, Yahoo! Cloud Serving Benchmark (YCSB) for OLTP and data serving, and SPECjbb2005 for conventional enterprise computing.

Applications.

Memcached is a popular open source distributed key-value store that caches data in DRAM for fast retrieval [64]. As the cache fills, evictions occur according to an LRU policy. Memcached hashes keys to distribute load and data. Memcached activity is a function of data popularity and query rate. We model popularity with a zipf distribution and use a large α parameter to create a long tail. We model query inter-arrival times with an exponential distribution. Such models are consistent with observed memcached queries in datacenters [68].

Yahoo! Cloud Serving Benchmark (YCSB) is a benchmark for online transaction processing that interfaces to cloud databases, such as Cassandra, BigTable, and HBase. To characterize YCSB, we first populate a 6.2GB database. Next, we use the YCSB client model to generate a zipf distribution with operations that have a 95:5 read to write ratio, which is representative of modern, read-heavy applications [11].

SPEC Java Server Benchmark emulates a three-tier client/server system with an emphasis on the middle tier. SPECjbb performs work that is representative of business logic and object manipulation to simulate the middle tier. It exercises the Java Virtual Machine, Just-In-Time compiler, garbage collection, threads and some

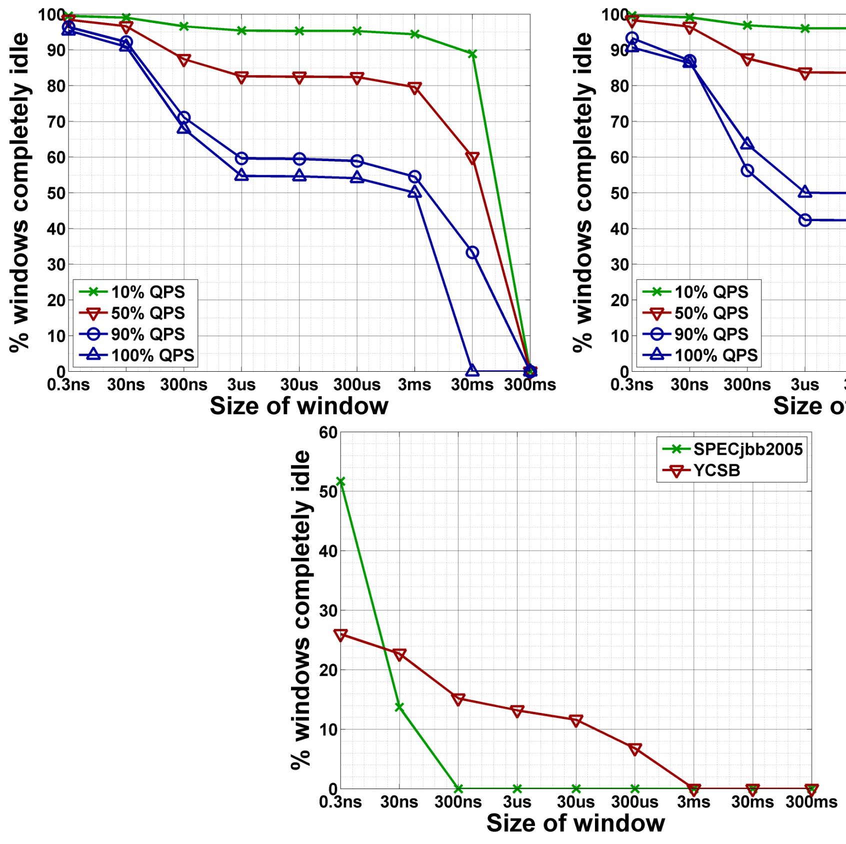


Figure 5.5: Activity graphs for (a-b) memcached at value sizes of 100B and 10KB. (c) SPECjbb2005 and YCSB.

aspects of the OS [69] .

Memory Activity.

To help understand how applications will interact with low power modes, we use rank-level activity graphs to visualize memory behavior [50]. These graphs characterize bus activity using windows that define a period of time. We sweep a window over the timeline of application execution and count the number of completely idle windows for varying different window sizes. If applicable, this measurement is also taken across various application loads, which is measured in queries per second (QPS) relative to the system's peak load (denoted as %QPS).

At small value sizes (100B), memcached is CPU-bound as the CPU must cope with many small, incoming packets. At large value sizes (10KB), memcached saturates the network connection. With limited network bandwidth (e.g., 10Gb/s), memcached rarely stresses memory bandwidth (e.g., 80Gb/s).

Although memcached does not saturate memory bandwidth, we must determine whether the memory channel has uniformly low utilization or has bursty traffic with many periods of idleness. The former would make power mode design difficult but the latter would benefit from many existing DRAM power modes, and even full-system power modes.

Figure 5.5(a-b) shows rank-level activity graphs for memcached configured at 100B and 10KB. A large percentage of short windows (e.g., 100ns) are idle. At typical loads between 10-50%, 95% of the windows encounter completely idle memory. Moreover, these idle periods are long. Even as we increase window size towards microsecond granularities, 80-90% of these windows encounter idle memory. However, idleness is difficult to find as application load increases to 90-100% or when windows widen to ms.

While memcached exhibits idleness, conventional datacenter workloads in data serving and online-transaction processing have fewer opportunities to exploit existing power-modes when run at 100% QPS. Figure 5.5(c) illustrates few idle windows for SPECjbb and YCSB even at small windows. At lower utilizations that are typical in datacenters [50], the idle fractions could be higher but the opportunities are scarce beyond 1μs.

5.2.2 Probabilistic Energy-Delay Analysis

Clearly, as idle opportunities for applications are small beyond 1μs, we need to understand memory behaviour and energy-delay tradeoffs at these granularities in more detail to design powermodes. We build a probabilistic model to quantitatively understand the tradeoffs. First, we take some representative applications and study

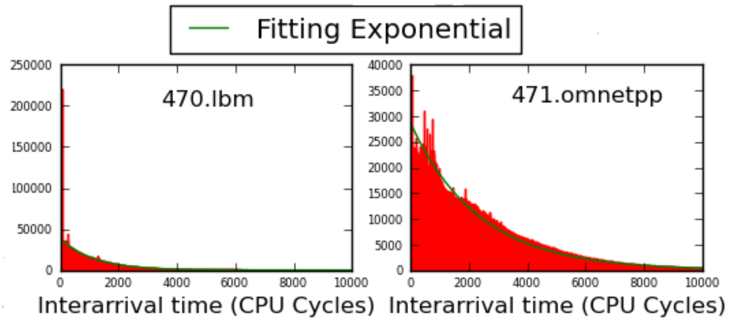


Figure 5.6: Exponential curve fitting for memory interarrival times of applications. The fit shows that for many applications, interarrival times follow an exponential statistical distribution that is identical across the interarrival sample values.

the distribution of the interarrival times to establish assumptions for the rest of the analysis.

The interarrival access times follow an exponential distribution, as indicated from the curve fitting illustrated in Figure 5.6. Although the exponential distribution establishes identicalness, we cannot assume independence between interarrival samples. So, we study the autocorrelation value (R) between interarrival samples of one of the memory banks as shown in Figure 5.7. R is studied as a function of Tau, which is the separation between the samples. We observe that across applications, R quickly decays to 0, even for small values of Tau and is the highest at Tau = 0. This clearly indicating statistical independence between samples. Each plot is also repeated by taking bigger window sizes and averaging the interarrival values over that window. The autocorrelation values still decay with increasing Tau, establishing independence.

Thus, by studying the distribution statistics, we can establish that the interarrival time samples are identical and independently distributed (i.i.d) with an exponential function. This simplifies our probabilistic analysis for understanding fundamental energy-delay trends to evaluate powermodes.

Accordingly, we model a stream of memory requests as a Poisson process. This analysis assumes that the time between requests follow an exponential distribution

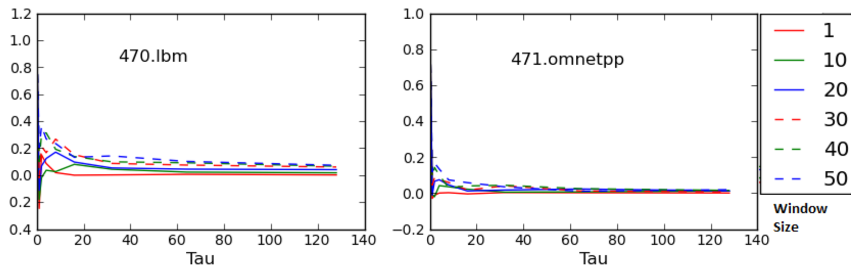


Figure 5.7: Autocorrelation of interarrival times of one of the banks in a memory system. The autocorrelation quickly decays to 0, even for small values of Tau and is the highest at 0, clearly showing that the interarrivals are statistically independent of each other. In each plot, the autocorrelations are repeated by taking bigger window sizes (indicated in the legend) and averaging the interarrival values over the window. The autocorrelation values still decay with increasing Tau.

and these inter-arrival times are statistically independent, which roughly match our data. As we described, histograms for memory inter-arrival times resemble exponential probability densities and the autocorrelation between inter-arrival times is nearly zero.

Let T_i be an exponentially distributed random variable for the idle time between two memory requests. The exponential distribution is parameterized by $1/T_a$ where T_a is the average inter-arrival time. Let P_d and P_u denote power dissipated in powerdown and powerup modes. The memory powers-down if idleness exceeds a threshold T_t. And it incurs a latency T_u when powering-up again.

Power-down is invoked with probability $f = P(T_i > T_t) = e^{-T_t/T_a}$. In this scenario, DRAM dissipates P_d for $T_i - T_t$ time while powered-down and dissipates P_u for $(T_t + T_u)$ time while powered-up. T_i is the only random variable; $\mathbb{E}[T_i] = T_a$.

$$\begin{aligned}
\mathbb{E}[E] &= f\times\mathbb{E}\left[P_d(T_i - T_t) + P_uT_t + P_uT_u\right] + (1-f)\times\mathbb{E}\left[P_uT_i\right] \\
&= f\times\left[P_d(T_a - T_t) + P_uT_t + P_uT_u\right] + (1-f)\times\left[P_uT_a\right] \\
&= P_d\left[f(T_a - T_t)\right] + P_u\left[f(T_t + T_u) + (1-f)T_a\right]
\end{aligned}$$

With this probabilistic formulation, the expectation for memory energy is given by $\mathbb{E}[E]$. And the expected impact on delay is $\mathbb{E}[\Delta D] = fT_u$, which conservatively assumes that powerup latency is exposed on the critical path.

Clearly, we would prefer to frequently use an efficient powerdown mode (i.e., large f and $P_d << P_u$). Energy falls as inter-arrival time increases beyond the threshold. Conversely, energy increases if powerup latency is large.

Energy-Delay Trade-offs. The relationship between $\mathbb{E}[E]$ and $\mathbb{E}[\Delta D]$ depends on average inter-arrival times (T_a), powerdown threshold (T_t), and powerup latency (T_u).

First, consider various inter-arrival times and powerdown thresholds. Each curve in Figure 5.8(a) plots trade-offs for a particular inter-arrival time T_a at $T_u = 1000$ns and points along a curve represent varying thresholds T_t. Short inter-arrival times ($T_a = 1000$ns) mean that the added energy costs to power back up are more expensive than the savings by invoking powerdown. Thus both the energy and delay increase with the powerdown fraction f. As inter-arrival times increase ($T_a \rightarrow 2000$ns), the energy saving in powerdown offsets the overhead of wakeups.

Since long wakeups will never be very energy efficient, we explore the effect of wakeup latency in Figure 5.8(b) Each curve in the figure plots trade-offs for a particular powerup latency and clearly shows the cost of slow wakeups. At one end of the spectrum, zero latency powerup reduces energy with no delay penalty ($T_u = 0$ns). Waiting to go to powerdown only costs more energy, as the energies are higher for low values of f. In contrast, today's approach to disabling DLLs and clocks is expensive, producing a horizontal trend line ($T_u = 1000$ns).

Clearly, if we want powermodes to be useful for applications, we need the wakeup latency to be much shorter. Approaches at the software and at the memory controller to hide the adverse impact of the long wakeups have limitations and are impractical for datacenter workloads. To close this gap between the ideal and practice, we re-think

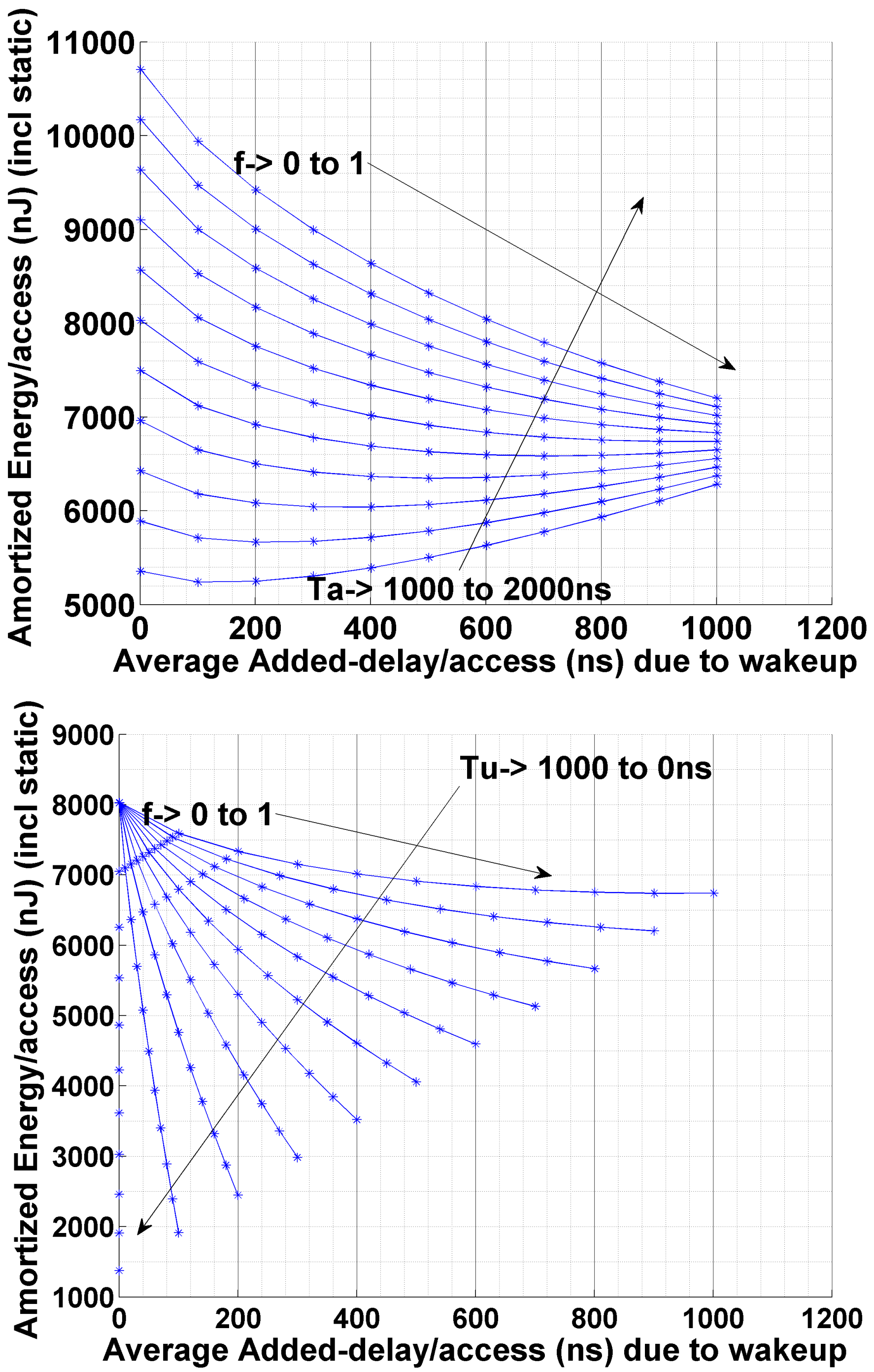


Figure 5.8: Probabilistic energy-delay trade-offs when powerup latency is exposed. E-D plots with different lines for different (a) memory request inter-arrival time, T_a varying from 1000 to 2000 ns with wakeuptime T_u=1000ns and (b) varying powerup time, T_u for a fixed T_a=1500ns. Each line's points sweep fraction of time, powerdown is invoked, f. Assumed P_d=5.36W and P_d=0.92W from a 4GB Intel R-DIMM.

memory architecture in the next section, to reduce powerup latency and accommodate practical inter-arrival statistics in real applications.

5.3 Architecting Effective Power Modes

Probabilistic analysis highlight the importance of fast wake-up for memory efficiency. After reviewing high-speed interfaces to explain today's long wake-up times, we propose several architectures with much shorter idle to active transitions.

A reliable, high-speed interface performs two critical tasks. First, the interface must drive the signal from the transmitter to the receiver. It converts a sequence of bits into a voltage waveform. Then, it drives that waveform on a wire with enough margin so that the receiver can distinguish between the voltages that represent ones and zeros. For high data rates, we engineer the wires as transmission lines and use termination to avoid reflections. Even so, loss in the wires and process variations cause high and low voltage levels to become degraded and mix the values of many bits together when they arrive at the receiver. Equalization helps to cleans up the waveforms.

But getting the signal to the receiver is only half the battle. The other half is knowing when to sample the signal to get the correct value of the bit. At a data rate of 1.6Gb/s, the time window for each bit is only 625ps, and this time includes transitions from the previous bit and to the next bit. To build a reliable link, the interface needs to sample the bit in the middle of the stable region, which required a very accurate sample clock.

Analog circuits drive and receive the bits, and align the clock so that bits are sampled at the right time. These circuits use voltage and current references for their operation, and often use feedback to learn the right corrections (e.g. sample time or equalization) that optimize link operation.

Because they are turned off during powerdown, these circuits must re-learn their connection settings before the link can be operated again. Worse, this re-learning cannot begin until voltage and current references stabilize after powerup. Because analog circuits demand precision and have lower bandwidth than digital ones, μs settling latencies are typical.

One of the critical circuits in a high-speed link is a delay locked loop (DLL), which uses feedback to align the phase (timing) of a clock. In links, DLLs align the sample clock to data, or align data and strobes to the system clock. DLLs compensate for changes in timing that would otherwise occur from variations in process, voltage, and temperature (PVT). Since voltage and temperature are dynamic, DLLs continue to run after initial calibration to track and remove their effect [9, 40].

To rapidly transition from idle to active mode, an interface must apply several strategies, such as digitally storing "analog" feedback state, using simpler analog circuits that power off quickly, and designing bias networks that power off and on quickly.

5.3.1 Fast Wake-up DRAMs

Existing link interfaces are generally symmetric: circuitry on both sides of the link need to be the same. But, symmetry is not optimal in a memory system that has a large number of DRAMs but a small number of controllers. Furthermore, because DRAM process technology is optimized for density, the speed of its transistors is much worse than that of transistors in a comparable logic process. Thus, we would rather shift link circuitry from DRAMs to the controller.

MemBlaze DRAMs. A post-DDR4 DRAM architecture is presented here with an asymmetric link interface that removes clock delay circuitry from DRAMs and places them on the memory controller. Because such circuitry determines wake-up latency in today's DRAMs, the system is capable of much faster power mode

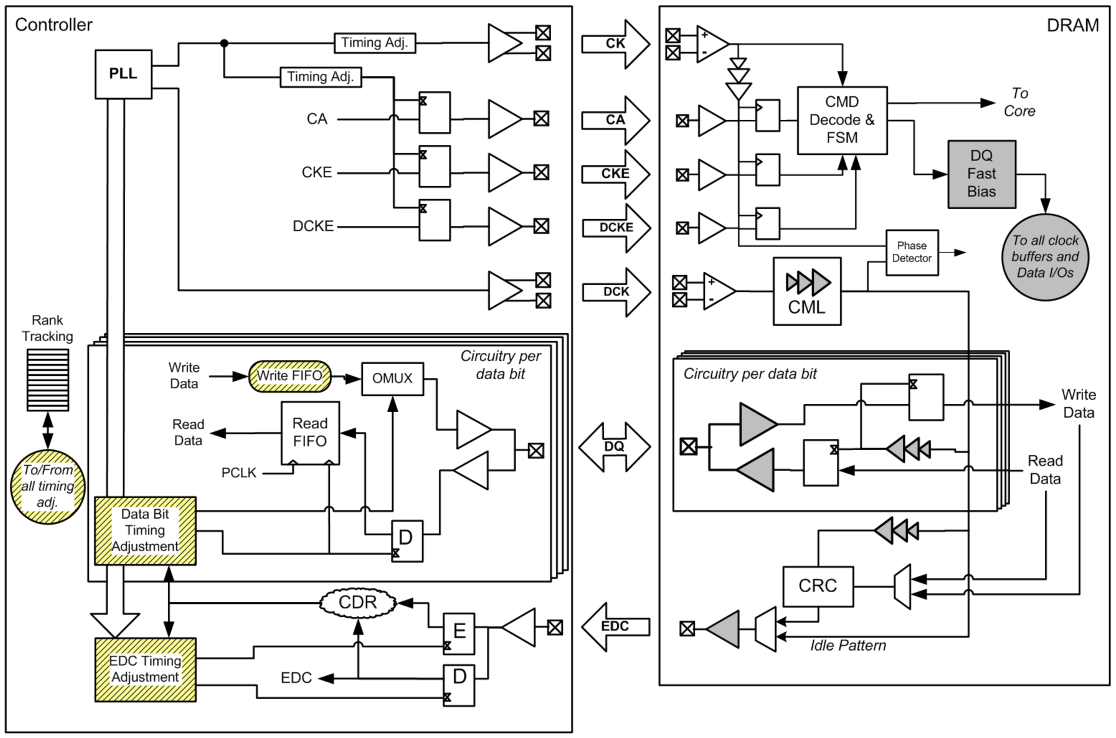


Figure 5.9: Proposed architecture introduces clock data recovery (CDR) circuitry to the controller, which uses the timing reference signal (TRS) transmitted across error detection and correction (EDC) pins.

transitions. The controller and memory interfaces have been implemented at Rambus and the silicon results are shown in Figure 5.9.

Synchronization. In this architecture, DRAMs no longer have DLLs for timing adjustment. For arriving commands and writes, DRAMs simply sample inputs at the rising edge of link clocks, CK and DCK received from the controller. But synchronizing reads (i.e., data from DRAM to controller) requires special treatment. DRAMs no longer send data strobes along with data, which raises two new issues. The first is how the controller can learn the correct timing, and the second is that this timing may be different for each DRAM.

To address these challenges, the controller uses a clock and data recovery (CDR) circuit to update its clock, and thus update its sample points for data reads, based on a timing reference signal (TRS) received from DRAMs. The DRAMs time-multiplex the TRS on a pin used for error detection and correction (EDC). For every read and write, DRAMs calculate and transmit an 8-bit EDC to the controller. The remainder of the 32-bit EDC burst transmits DRAM clock information.

Thus, during normal rank operation, the EDC pin transmits correction codes interleaved with a toggling pattern that guarantees some minimum edge density. The controller tracks timing variations for each DRAM in a rank as long as that rank sees activity and communicates edges across the EDC pin. Activity on one rank provides no timing for other ranks.

Accommodating Idle Ranks. With regular accesses to a rank, the controller tracks rank timing. But gaps in activity produce gaps in phase updates. Because our interfaces rely on these updates, DRAMs specify the maximum amount between rank accesses. Ranks with longer idle periods incur a recalibration latency before further data transfers.

Alternatively, data-less pings can maintain timing when data is not needed from the memory core but toggling patterns are needed on the EDC pin for phase updates. The ping furnishes a toggling pattern without page activation or column access strobe. In this scenario, the system uses EDC signals for timing and ignores DQ signals.

Memory systems rarely have long idle periods even when server utilization is low [50]. If idleness is rare, normal rank activity updates the clock phase and ping/recalibration sequences are unnecessary. When pings do occur, they coincide with periods of low channel utilization and thus do no interfere with normal traffic.

Fast Wake-up Protocol. Because MemBlaze DRAMs do not have DLLs, the critical latency during wake-up shifts from clock delay circuitry to the datapath. MemBlaze defines an extra control pin (DCKE) to enable the data clock domain, quickly powering the datapath, data clock buffering, and data I/O circuitry (shaded blocks in Figure 5.9).

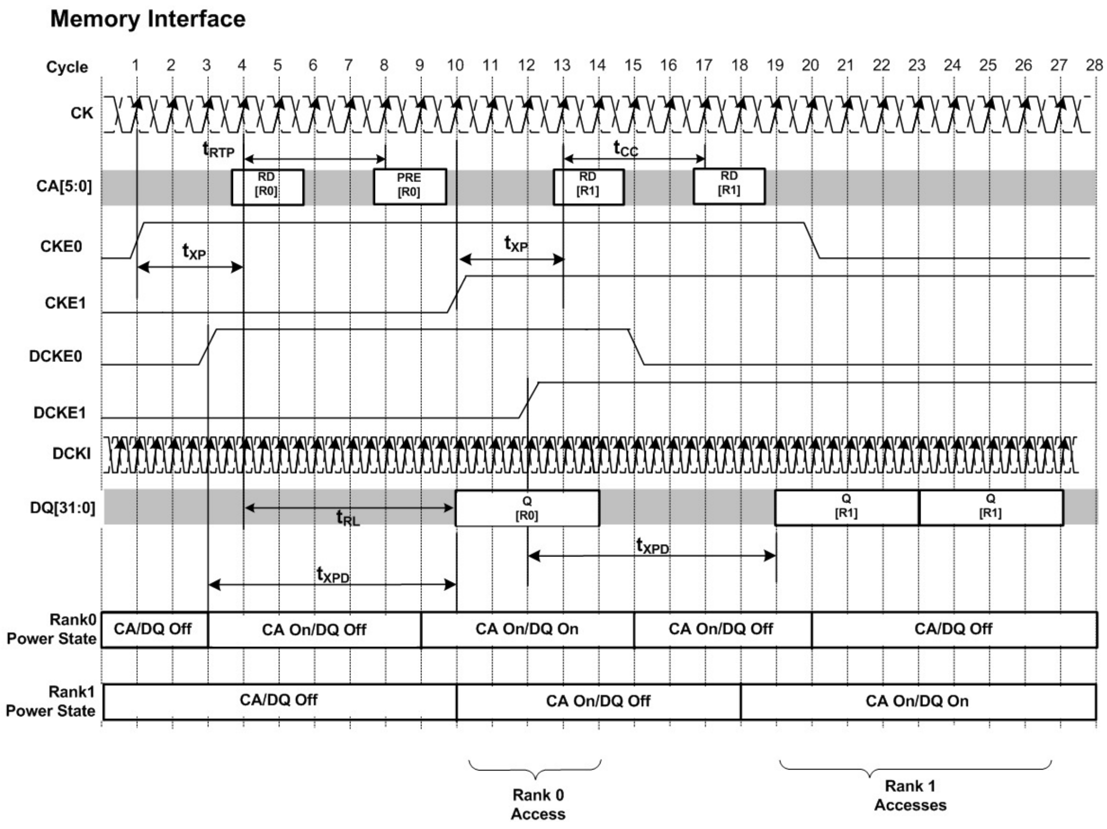


Figure 5.10: Timing diagram illustrates separate power management for command (CA) and data (DQ) paths.

Figure 5.10 illustrates operation in a two-rank MemBlaze system. Initially, data and clock enables (DCKE, CKE) for both ranks are de-asserted, which powers-down command and data blocks. At cycle 1, CKE0 for rank 0 is asserted and the command block powers-up; the data block remains powered-down. Command receivers (CA) are awake in time to receive a read for rank 0. At cycle 3, DCKE0 is asserted and the data block powers-up to transmit read data. Exit latency for command and data blocks are tXP and tXPD.

Similarly, the second rank exits command standby at cycle 10 and exits data standby at cycle 12. Reads arrive at cycles 13 and 17, satisfying constraints on consecutive reads, which is denoted by tCC. Power-up does not affect read latency as long as DCKE is asserted early enough (i.e., tXPD before first read data). By separating command and data block enable signals, the DQ interface circuitry can remain powered-down even as precharge and refresh commands arrive.

Hiding Datapath Wake-up Latency. By default, datapath wake-up requires approximately 10 ns. MemBlaze defines the DCKE pin to enable the data block early enough to avoid affecting latency and completely hide it under column access (CAS). Thus, we can quickly power up the datapath only when needed and separate the powerups for command and data blocks. For fast DRAM interface wake-up, MemBlaze exploits a number of circuit innovations including common-mode logic (CML) clock trees and fast-bias circuitry to powerup links quickly, developed at Rambus. Further, if we leverage more insights from a recently implemented serial link interface that transitions from idle to active well under 10 ns [79], we could simply use read or write commands to trigger datapath powerup eliminating the need for DCKE.

Timing, Datarate and Power. Both the MemBlaze Memory Controller and DRAM PHYs were taped out in a 28nm process and the chip was rigorously tested for functionality, correctness and the proposed fast wakeup speed in an industry-strength, serial-links laboratory[37]. In lieu of a DRAM core, the test chip pairs the new PHY blocks with test pattern generation and checking logic for emulating memory read and write transactions.

The transmit eye diagram at the DQ pins had clear eyes with sufficient timing and voltage margins at 6.4Gbps. The architecture also reduces power in both active-standby and precharge-standby power modes. Compared to today's power modes, this provides the performance of fast-exit powerdown with the DLL-off efficiency.

Specifically, this matches deep power down mode's power at a reduced exit latency of 10ns making it useful for many different applications including emerging ones that have short idle periods. The power difference between the active idle mode and the

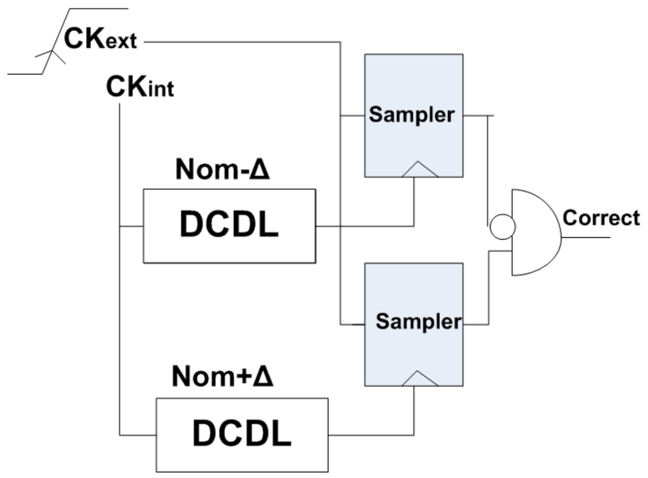


Figure 5.11: MemCorrect Error Detection with Digitally Controlled Delay Lines (DCDL) to sample CK_{ext} using delayed versions of nominal CK_{int}.

most efficient powerdown mode is attributed to DLLs, clock tree, and command pins. By powering down these components, MemBlaze lowers power in all other states except during active reads/writes.

Although standby efficiency improves, dynamic burst power is largely unchanged. During bursts, DLL power savings are offset by new power costs in current-mode clock circuitry and injection locked-oscillator power costs.

5.3.2 Imperfect Wake-up Timing

MemBlaze provides and requires perfect timing upon power-mode wake-up. As a result, great precision is required in its circuit design. We propose two new mechanisms in which DRAMs relax the perfect timing and power up constraint. This might allow more aggressive I/O power state changes with a less radical change to the interface circuits.

Reactive Memory Interfaces (MemCorrect). For interfaces that track timing less precisely, we introduce speculative DRAM data transfers immediately after waking up from deep powerdown modes. The key idea is that even before the long

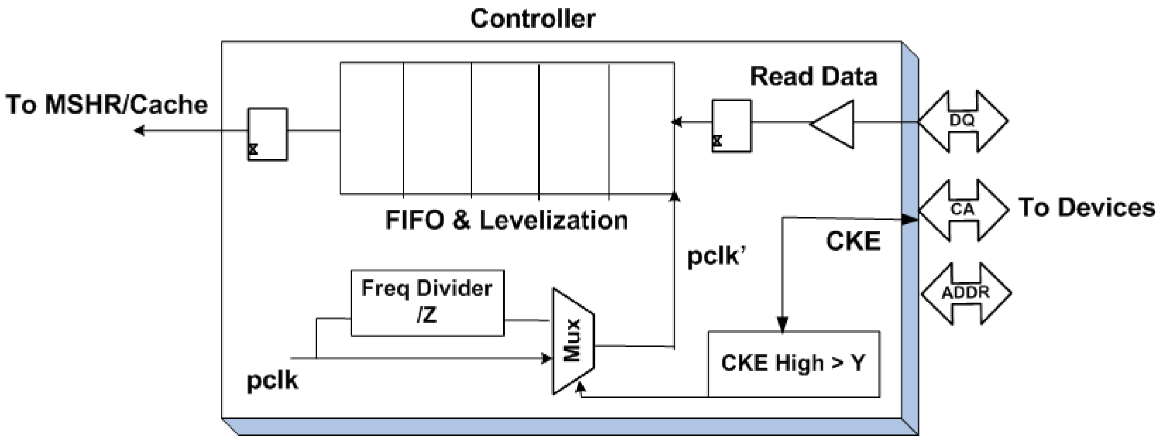


Figure 5.12: MemDrowsy Architecture.

DLL recalibration is complete, we speculatively allow the DRAM to start transferring read data. To guarantee that each memory transaction completes correctly, we architect error detectors. Our fast wake-up memory interface implements this error check on each transaction (Figure 5.11), which ensures that the clock transition is within a window ($\pm\Delta$) of its nominal location. This check handles cases when variations and drift during powerdown affect link operation. The error is communicated to the controller through a dedicated pin, *Correct*. If an error occurs, it is because the controller has issued a command too soon after powerup. The controller could simply wait a longer period of time before re-trying the command or could send a timing calibrate command to expedite wake-up.

Drowsy Memory Interfaces (MemDrowsy). Another option that removes DLL recalibration from the critical path on wakeup is that which begins transfers immediately but mitigates timing errors by halving the data rate for a certain period of time (Y) after wake-up. During this time, the DLL recalibrates while allowing read transfers to the controller. This slower rate more than doubles the timing margin of the link, greatly improving tolerance to small timing errors induced by voltage and temperature variations on the data strobe (DQS).

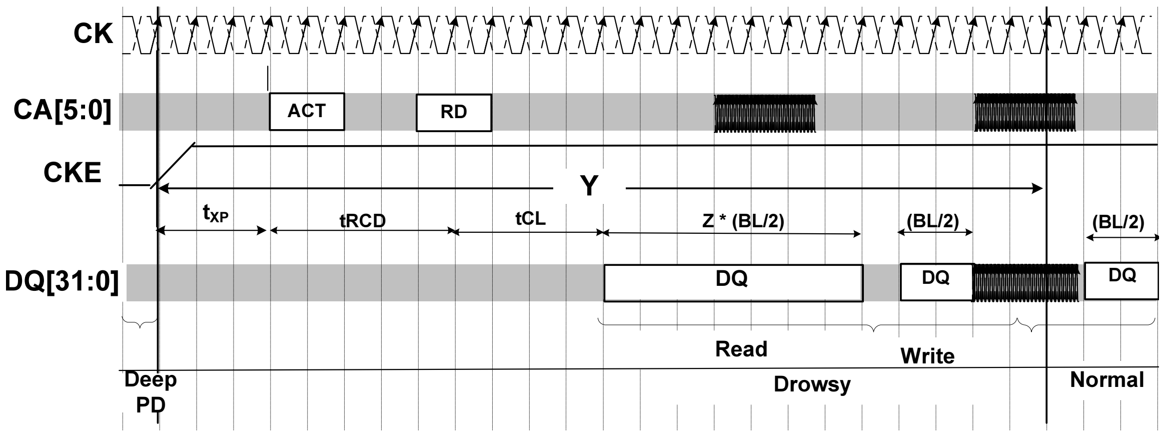


Figure 5.13: MemDrowsy Timing Diagram.

Thus, MemDrowsy reduces the effective data rate and relaxes timing precision after wake-up, for reads that need a locked DLL. The clock speed is still maintained at the full rate but the link effectively transmits each bit twice. This enables transmitting data while recalibrating and results in lengthening the valid data window. Of course, the controller must also shift the point at which data is sampled. After recalibration, the link operates at nominal data rates.

Figure 5.12 illustrates extensions to the memory controller. A rank is powered-down simply by disabling the clock (CKE low). Upon powerup, the clock is enabled (CKE high) and a timer starts. The clock operates at the nominal frequency f, providing a sufficient density of clock edges needed to facilitate timing feedback and recovery.

However, given timing uncertainty after wake-up, we use a frequency divider to reduce the rate at which we sample the incoming data; the drowsy rate is f/Z. A multiplexer chooses between sampling at the nominal clock rate or at a divided rate. During drowsy mode, the valid read window is lengthened by a factor of Z as illustrated in Figure 5.13.

Since the drowsy sampling period is an integral multiple of nominal sampling period, the controller clock is unchanged; it is simply sampled every Z-th cycle.[1] Sampling returns to the nominal frequency only after timing recalibration.

5.4 Evaluation

We evaluate system implications from two types of memory architectures. The first type, MemBlaze, provides perfect timing and synchronization after wake-up by eliminating expensive interface circuitry from the DRAMs. The second type, exemplified by MemCorrect and MemDrowsy, provides imperfect timing and requires corrective mechanisms. We quantify the performance loss these corrective actions cause.

The evaluation is first done on DDR3 parts. When applied to DDR, the proposed techniques hugely improve existing DDR powermodes. We then apply these interfaces to LPDDR2. Here, we regain high speed operation with DLL like functionality during active period. In addition, the high energy efficiency of LPDDR2, due to its superior standby power is also maintained since all the above methods provide quick mechanisms to power off the DLL.

5.4.1 Experimental Methodology

Simulators. We use an x86_64 execution-driven processor simulator based on a Pin front-end [48, 65]. We use eight out-of-order (OOO) cores at 3GHz matched with Intel's Nehalem microarchitecture and cache latencies as shown in Table 5.1.

The memory system is comprised of multiple channels and DRAM devices with parameters tuned to reflect the proposed architectures. We use 4GB R-DIMMs with 2Gb x4 parts clocked at 1333MT/s to populate 8 single-ranked DIMM slots distributed across 4 channels. Total system capacity is 32GB. The memory controller

[1]MemDrowsy clock rates are unchanged, differentiating it from work in channel frequency scaling [12].

Processor	Eight 3GHz x86 Out-of-Order cores
L1 cache	private, 8-way 32KB, cycle time = 1, 64B cache-lines
L2 cache	private, 8-way 256KB, cycle time = 7, 64B cache-lines
L3 cache	shared, 16-way 16MB, cycle time = 27, 64B cache-lines
Memory controller	Fast powerdown with threshold timer = 15 mem-cycles [32] Closed-page, FCFS scheduling

Table 5.1: Baseline System Simulation Parameters.

Main memory	32GB capacity, 2Gb x4 1333MT/s parts, single ranked 4GB-RDIMMs, four channels, 2DIMMs/channel [12, 31]
Active idle power	5.36W per 4GB DIMM
Precharge-idle power	4.66W per 4GB DIMM
Fast exit powerdown power	2.79W per 4GB DIMM
Slow exit powerdown power	1.60W per 4GB DIMM
Self Refresh power	0.92W per 4GB DIMM

Table 5.2: Memory System Simulation Parameters.

implements a close-page policy and First Come First Serve (FCFS) scheduling.

The memory simulator is extended to model three architectures: MemBlaze, MemCorrect, and MemDrowsy. MemBlaze is implemented in silicon and chip measurements are used to configure the simulator. For the other memory architectures, the simulator draws timing estimates from JEDEC specifications and energy estimates from Intel's analyses [12, 31]. In general, DDR3 systems dissipate about 1-1.5W/GB on average [55] and about 2.5W/GB at peak [24]. We validate that our experiments produce numbers in this range. Other memory simulator parameters are described in Table 5.2.

Workloads. We evaluate memory activity and the proposed architectures on datacenter workloads like memcached. [2] During evaluation, we fast-forward initialization phases and perform accurate simulations during the measurement phase by running for fixed number of instructions.

In addition, we evaluate a variety of multi-programmed (MP) SPEC CPU2006 as well as multi-threaded (MT) SPEC OMP2001 and PARSEC benchmarks, following prior memory studies [1, 72, 39, 14, 35]. Each core runs a copy of the program/thread

[2]100b value denoted by a and 10KB by b

Classification	**Multi-Programmed (MP) Benchmarks**
High B/W (MP-HB)	433.milc, 436.cactusADM, 450.soplex, 459.GemsFDTD, 462.libquantum, 470.lbm, 471.omnetpp, 482.sphinx3
Med. B/W (MP-MB)	401.bzip2, 403.gcc, 434.zeusmp, 454.calculix, 464.h264ref 473.astar
Low B/W (MP-LB)	435.gromacs, 444.namd, 445.gobmk, 447.dealll, 456.hmmer, 458.sjeng, 465.tonto
Classification	**Multi-Threaded (MT) Benchmarks**
High B/W (MT-HB)	applu, art, canneal, streamcluster, swim, mgrid
Med. B/W (MT-MB)	apsi, blackscholes, equake
Low B/W (MT-LB)	ammp, fluidanimate, wupwise

Table 5.3: Benchmark Classification.

depending on the benchmark and the number of application threads or processes are matched to the cores. We fast-forward 10 to 20 billion instructions to skip warm-up and initialization stages and focus on memory behavior in steady state for weighted IPC calculations. We classify MP and MT applications into 3 groups (HB, MB, LB) as shown in Table 5.3.

Metrics. For each memory architecture, we plot efficiency and performance. Efficiency is measured in energy per bit (mW/Gbps). In this metric, static and background power are amortized over useful data transfers. Performance penalties measure the impact on cycles per instruction (CPI). In each workload group, worst case performance penalty and best case energy savings are plotted on the top of each bar.

Energy savings are measured relative to baseline DDR3 DRAM that aggressively exploits fast-powerdown whenever encountering 15 idle memory cycles, which is taken from Intel SandyBridge architecture [32]. This low threshold also gives an optimistic baseline. Realistic, high-performance systems would set the threshold an order of magnitude higher, which would only magnify both active-idle energy costs and our architectures' advantages.

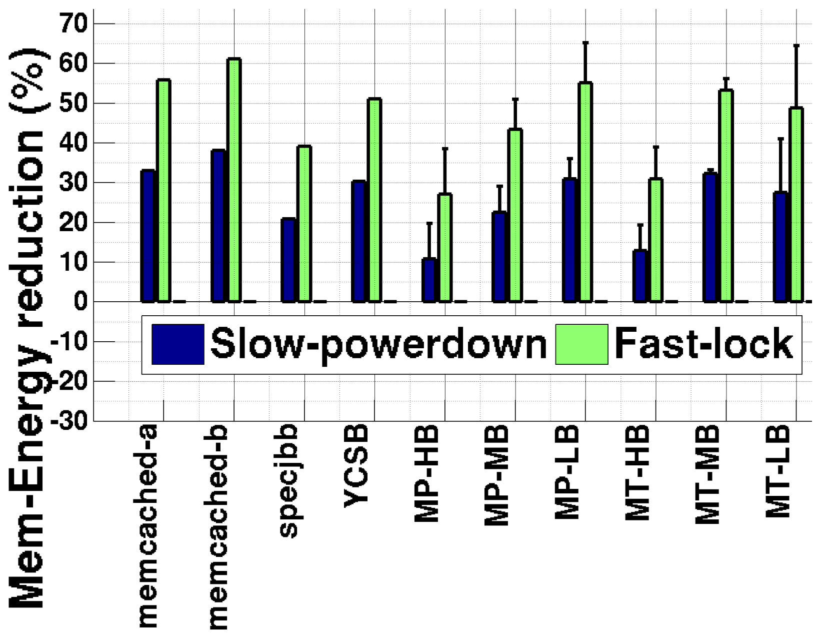


Figure 5.14: MemBlaze (fast-lock) energy savings relative to DDR3 DRAM baseline (fast-powerdown) and compared against DDR3 DRAM baseline (slow-powerdown).

5.4.2 MemBlaze

MemBlaze efficiency arises from two key features. First, it eliminates DRAMs' DLLs and clocks, thus eliminating long-latency DLL recalibration, which is on the critical path for today's mode exits. Capable of fast exits, MemBlaze can spend more time in powerdown and less time in active-idle.

Second, for any remaining time spent in active idle (i.e., neither bursting data nor in powerdown), MemBlaze consumes very little energy. With MemBlaze links that are capable of fast wake-up, DRAMs' data blocks are powered-on by DCKE precisely when they are needed and no earlier. Only the command blocks remain active, consuming a small fraction of the original active-idle power.

Given these advantages, MemBlaze energy savings are substantial. Even though

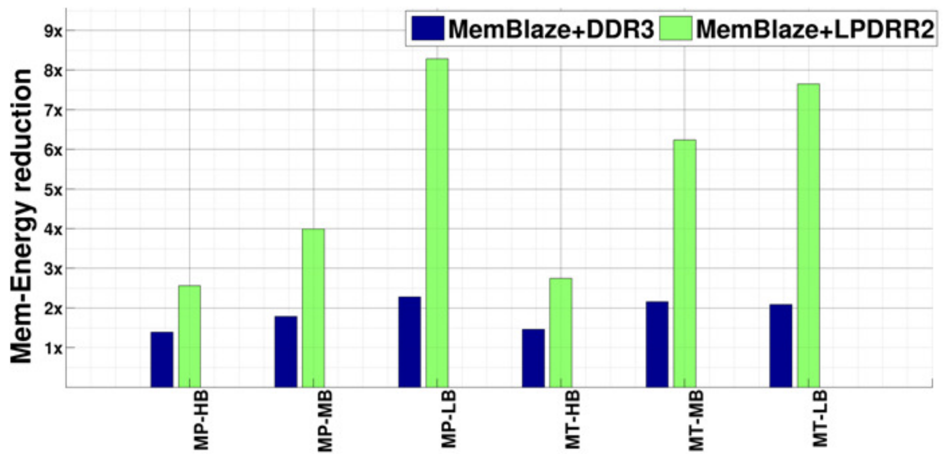


Figure 5.15: Comparision of energy savings by using MemBlaze on DDR3 and LPDDR2. LPDDR2 benefits from the combination of low power interface and fast powerup DLL like mechanism of MemBlaze.

silicon results indicated feasibility at much larger datarates, we use DDR3-1333 transfer rate in our simulations to make the comparison conservative. Figure 5.14 compares savings from MemBlaze (fast-lock) and the most efficient power-mode in today's DRAMs (slow-powerdown). When DLLs in today's DRAMs are kept in a quiescent state, slow-power down improves efficiency by 22%. But this efficiency requires a performance trade-off. Exit latency is 24ns, which affects the critical word latency.

DDR3 and LPDDR2

MemBlaze fast-lock improves efficiency by 43%. Even memory-intensive applications, like *433.milc* and *471.omnetpp*, dissipate 25-36% less energy. Applications that demand little bandwidth (LB) like *444.namd* consume 63% less energy. With compared against a baseline that uses power-modes more conservatively, these savings would increase by 2×.

MemBlaze is equally applicable on LPDDR2 memory parts. When applied, such a DRAM will have DLL locking mechanism on chip, in addition to the low power characteristics of LPDDR2. While the DLL helps in operating the link at high speeds, MemBlaze enables powering down the DLL quickly to go to the deep powerdown states. The energy savings of LPDDR2 with are illustrated in Figure 5.15. We observe that MemBlaze helps in preserving the same 3-5× energy savings of LPDDR2 parts. This is mainly due to the 20× better powerdown state in addition to zero termination power and lower active power. In addition, the performance degradation is negligible even for bandwidth sensitive benchmarks due to the presence of high speed DLL on chip. Clearly, this is an excellent DRAM memory part that is applicable to all kinds of benchmarks and will dramatically reduce DRAM power in datacenters.

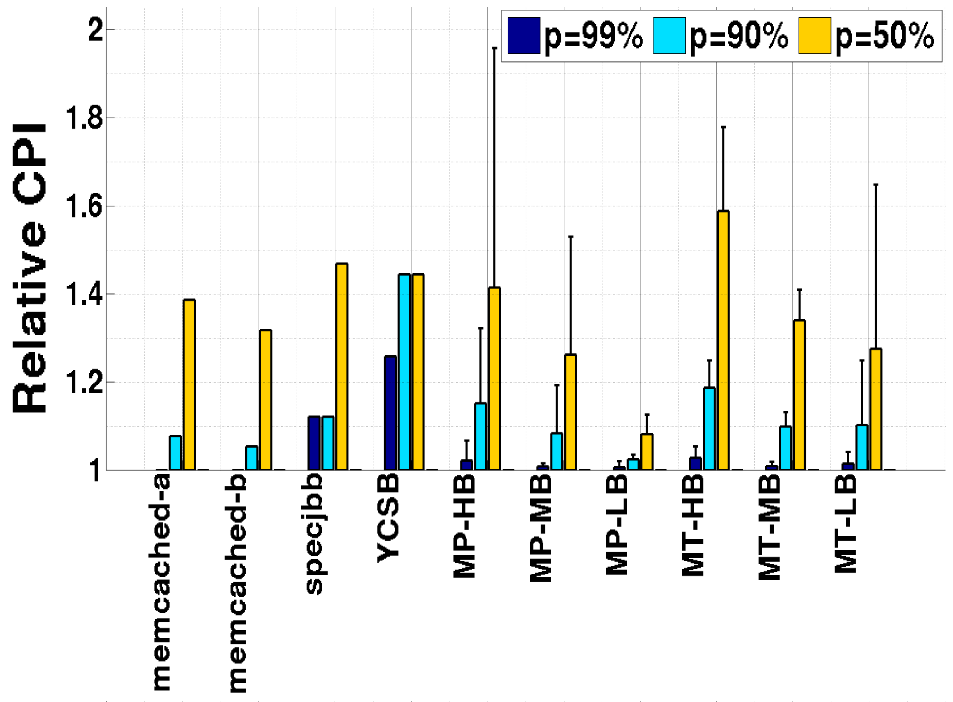
Moreover, efficiency comes with better performance than the baseline since MemBlaze power mode exit latency is comparable to that of fast-powerdown in today's DRAMs; neither incur DLL-related wake-up latencies. Fast links powerup the datapath in 10ns. Because this latency is hidden by the command access, we reduce energy with no performance impact.

With attractive energy savings and no delay trade-off, MemBlaze is an order of magnitude better than approaches that aggressively power-off DLLs at run-time or modify the BIOS to disable DLLs at boot-time. These mechanisms all require large performance trade-offs since today's high-performance DRAMs rely on DLLs for timing.

5.4.3 MemCorrect

While MemBlaze provides perfect timing, other interfaces (including today's DRAMs with DLLs) may be susceptible to timing errors when aggressively exploiting power modes. MemCorrect provides circuitry to detect timing errors, allowing the system to speculate that the timing was correct. We assess performance and energy relative to the DDR3 baseline that goes to fast powerdown state after encountering 15 cycles of idleness. The results are demonstrated in Figure 5.16

We evaluate MemCorrect based on the probability p of correct timing. In the best-case, p=100% and timing is never affected when using power-modes. And p=0% is



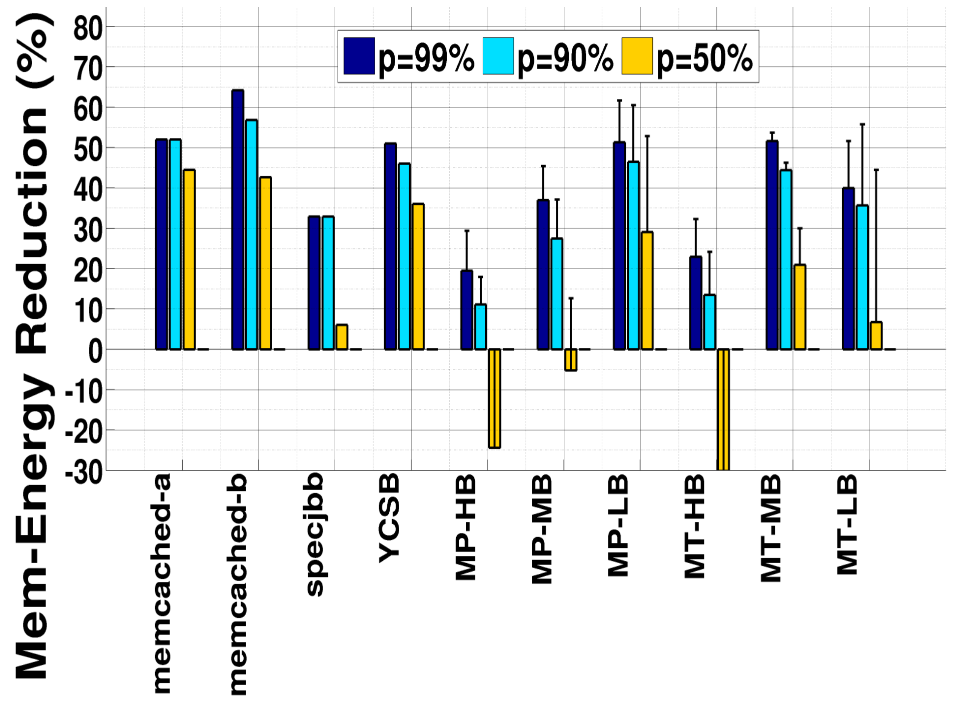


Figure 5.16: MemCorrect (a) performance as measured in Cycles per Instruction (CPI) and (b) energy relative to DDR3 DRAM baseline. The probability p of correct timing for transfer immediately after wakeup is varied. Plotted for MP, MT, datacenter benchmarks. Error bars represent ranges over mean value in the group.

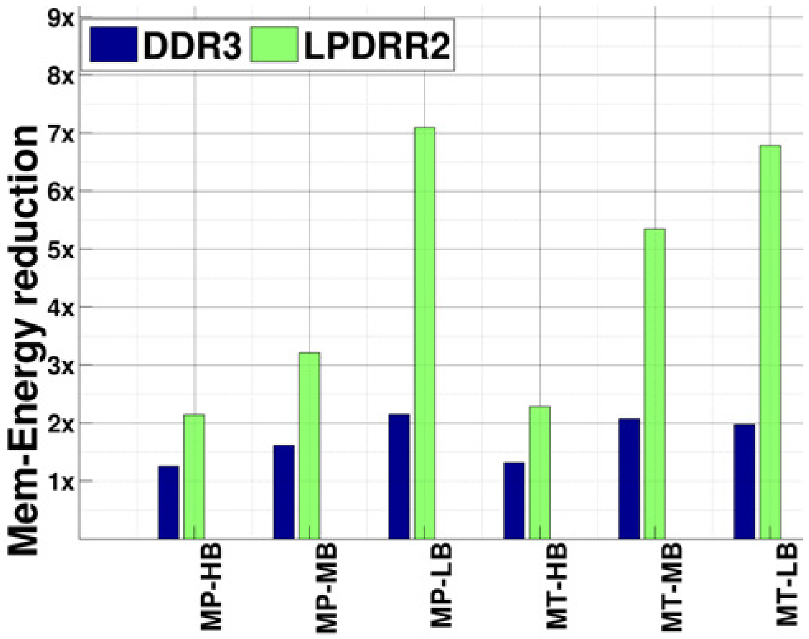


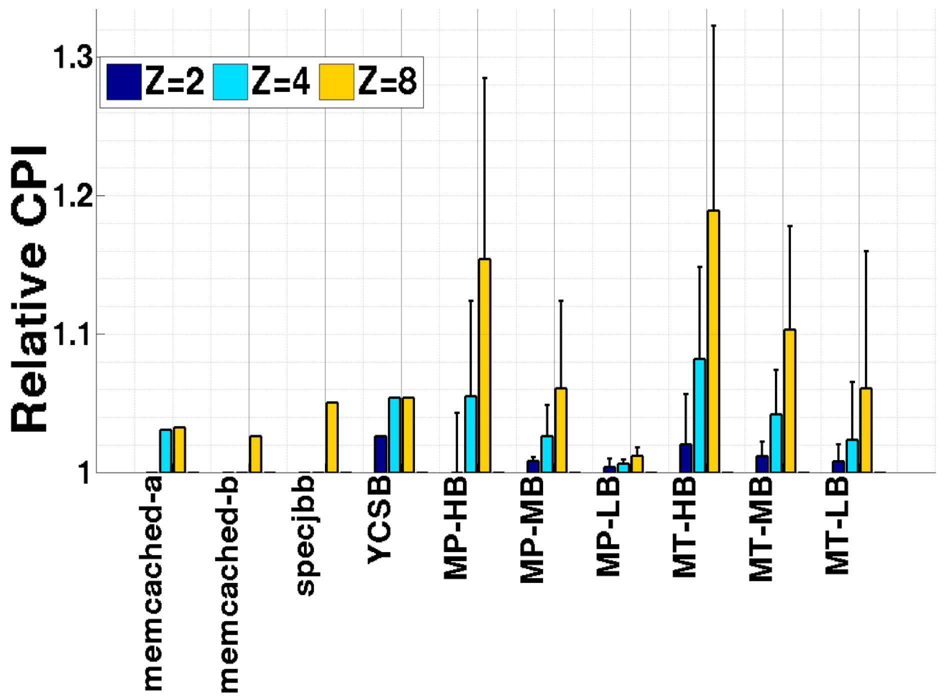
Figure 5.17: Comparision of energy savings by using MemCorrect on DDR3 and LPDDR2. LPDDR2 retains large energy savings with comparable performance to DDR3.

the worst-case in which every wake-up requires a long-latency recalibration. Smaller values for p degrade performance. When p=50%, performance degrades by as much as 2× compared to the DDR3 baseline.

DDR3 and LPDDR2

In exchange for the occasional delay, MemCorrect can exploit power-modes more aggressively. DRAMs with DLLs might bypass recalibration, start transfers immediately after wake-up, and detect errors as they occur. In such a system, MemCorrect energy savings are 38% and 30% when timing is correct for p =99% and p =90% of the transfers. This conservatively assumes that DRAM parts have termination power and an optimistic baseline. However, if errors are too common, workloads encounter large penalties and low, or even negative, energy savings.

MemCorrect when applied on LPDDR2 helps in preserving their 3-5× energy savings across a variety of benchmarks as seen in Figure 5.17.



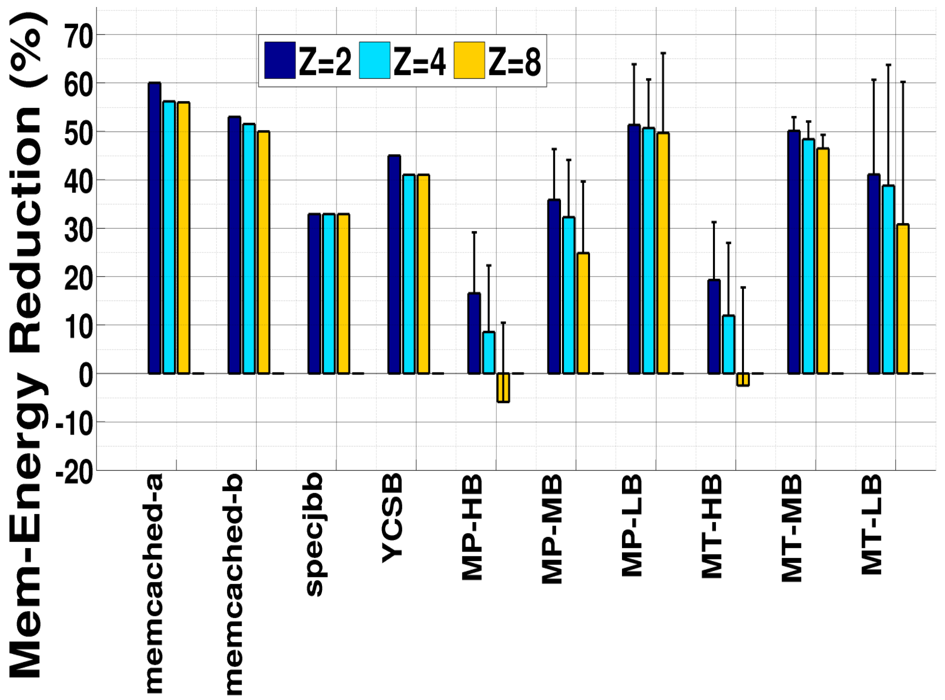


Figure 5.18: MemDrowsy (a) performance as measured in Cycles per Instruction (CPI) and (b) energy relative to DDR3 DRAM baseline. The drowsy rate reduction factor Z for transfer is varied by using Y=512 memory clock cycles. Plotted for MP, MT, datacenter benchmarks. Error bars represent ranges over mean value in the group.

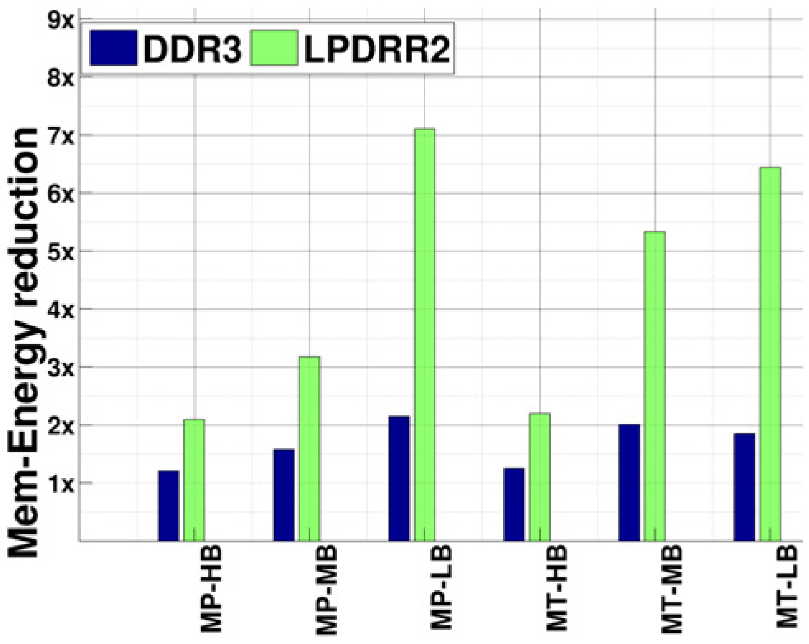


Figure 5.19: Comparision of energy savings by using MemDrowsy on DDR3 and LPDDR2. LPDDR2 retains large energy savings with comparable performance to DDR3.

5.4.4 MemDrowsy

If correct timing cannot be ensured at nominal data sampling rates, the system could operate in drowsy mode and reduce its sampling rate by a factor of Z for Y=512 memory clock cycles (tDLLk from DDR3 datasheets). In practice Z depends on timing margins at the DRAM interface. $Z = 2$ is realistic because existing LPDDR2 systems eliminate DLLs and transfer at half the data rate to ensure timing. We also assess sensitivity to more conservative margins ($Z = 4$, $Z = 8$).

Reducing the data sample rate more aggressively produces larger penalties in Figure 5.18(a); latency-sensitive *streamcluster* sees a 32% penalty when $Z = 8$. Less drowsy transfers have far more modest penalties, ranging from 1-4%.

MemDrowsy is also parameterized by how long the DRAM must operate in drowsy mode. In practice, this parameter is defined by the nominal wake-up latency. In other words, transfers are drowsy until the interface can ensure correct timing (e.g., current DRAM DLLs require 512 DRAM clock cycles). Performance is insensitive to the duration of drowsy operation as only the first few transfers after wake-up are slowed.

DDR3 and LPDDR2

For these modest penalties, MemDrowsy achieves significant energy savings in Figure 5.18(b). Drowsy transfers allow applications to enter power modes more often with fewer penalties. Clearly, applications that demand memory bandwidth (HB) are more sensitive to drowsy operation. Indeed, average energy per transfer might increase due to larger termination energy from higher bus utilization and also the idle power during the extra cycles.

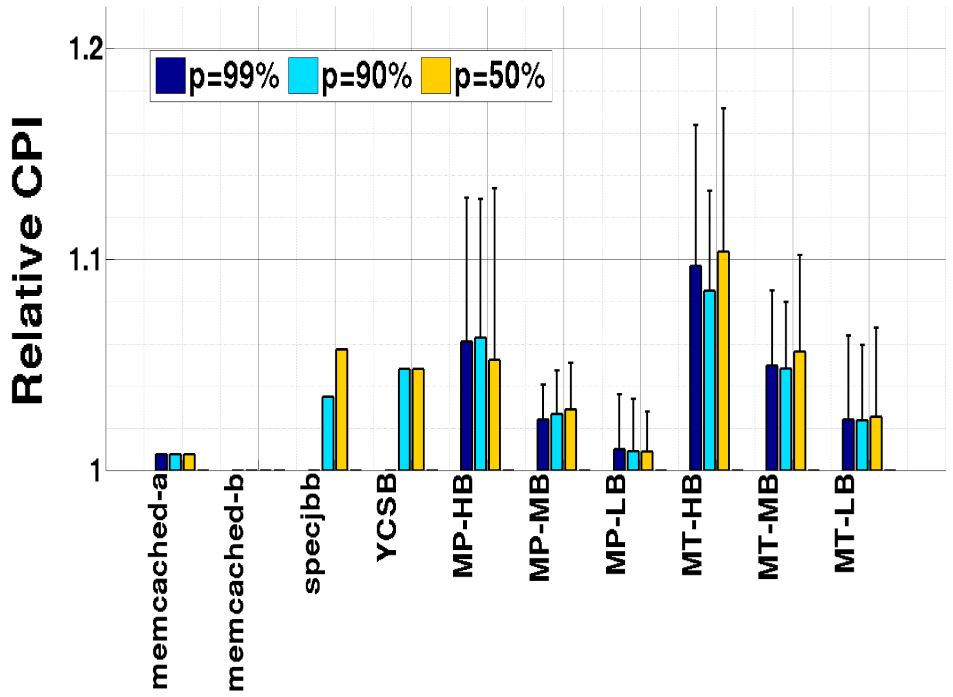
MemDrowsy when applied on LPDDR2 helps in preserving their 3-5$\times$ energy savings across a variety of benchmarks as seen in Figure 5.19.

5.4.5 MemCorrect and MemDrowsy

Suppose MemCorrect detects a timing error for a transfer immediately following a wake-up. Instead of delaying the transfer for the nominal wake-up latency, the system invokes MemDrowsy and begins the transfer immediately at a slower rate. Clearly, performance and efficiency in MemCorrect+MemDrowsy will be better than either approach applied individually. Immediately after wake-up, transfers begin immediately either at the nominal or reduced data rate.

With MemCorrect+MemDrowsy, exploiting power modes and transferring data immediately after wake-up has performance penalties between 10-20%, as shown in Figure 5.20(a) and is <10% for most applications. In exchange, power modes are more often exploited and energy savings are more consistent. Figure 5.20(b) shows the memory energy savings of MemCorrect+MemDrowsy, when applied to DDR3 that has termination.

MemBlaze promises large energy savings with an architecture that provides perfect



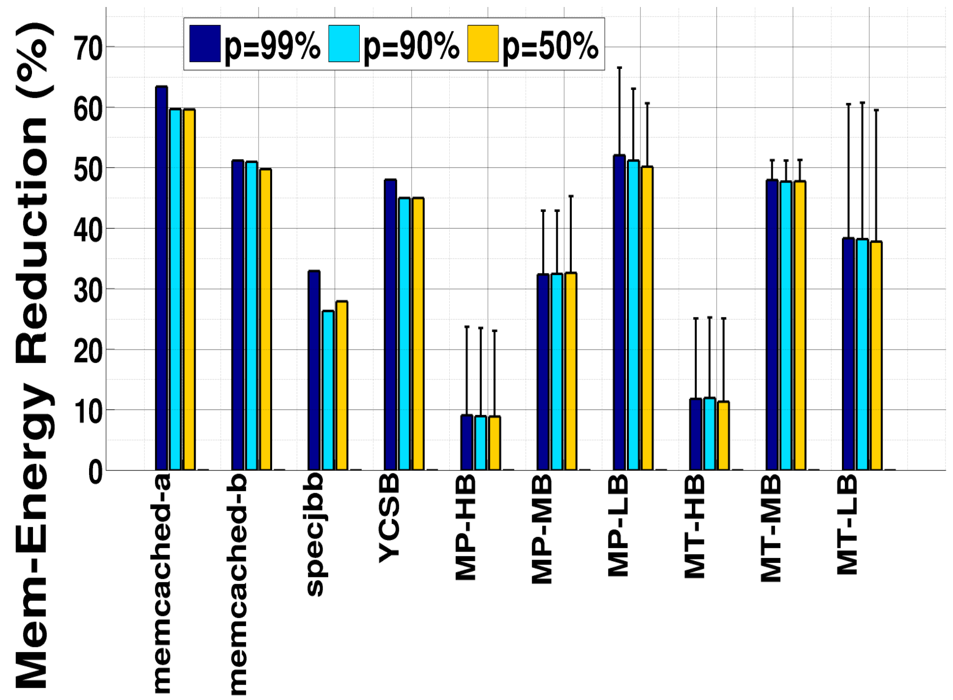


Figure 5.20: MemCorrect+MemDrowsy (a) performance as measured in Cycles per Instruction (CPI) and (b) energy relative to DDR3 DRAM baseline. The probability p of correct timing for transfer immediately after wakeup with $Z = 4$ is varied. Plotted for MP, MT, datacenter benchmarks. Error bars represent ranges over mean value in the group.

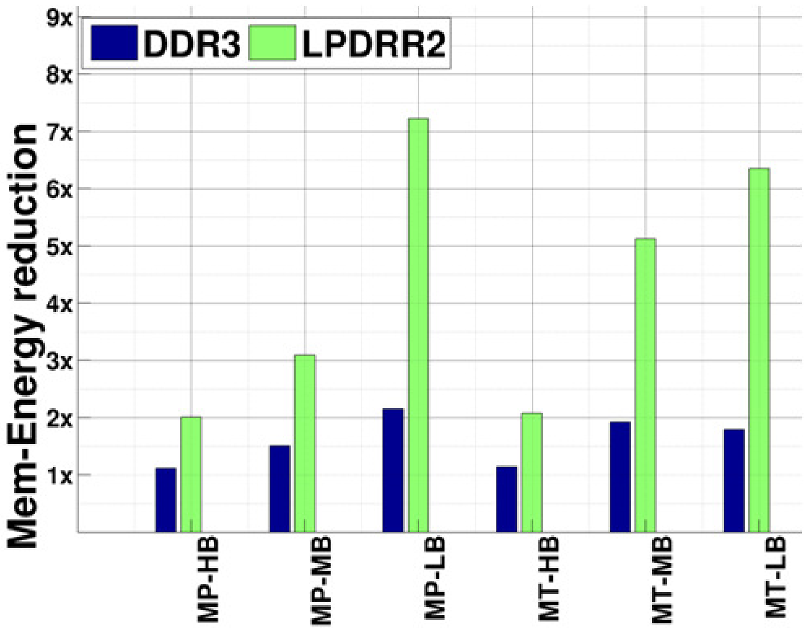


Figure 5.21: Comparision of energy savings by using MemCorrect+MemDrowsy on DDR3 and LPDDR2. LPDDR2 retains large energy savings with comparable performance to DDR3. In particular, MemCorrect+MemDrowsy mitigates LPDDR2's performance degradation at high bandwidths without trading off the low energy of LPDDR2.

timing information. Without such timing guarantees, however, MemCorrect+MemDrowsy provide the next best thing: comparable efficiency and modest (<10%) performance degradation for many applications.

MemCorrect+MemDrowsy when applied on LPDDR2 helps in preserving their 3-5× energy savings across a variety of benchmarks as seen in Figure 5.21, consistent with results from Chapter 4. On the other hand, the performance results are similar to DDR3. As such, the techniques make for a compelling case for deploying them in datacenters to dramatically improve memory power. On the other hand, they also overcome the limitations of LPDDR2 memory for memory bandwidth sensitive benchmarks by putting back the DLL on chip while providing a mechanism to keep them powered off for the majority of the time.

In the current chapter, we have described building low power systems that can also provide high bandwidth. In Chapter 4, we demonstrated low power high capacity memory systems for datacenter systems, only by using readily available LPDDR2 parts. In this chapter, we rethink how future interfaces could be built by taking LPDDR2 systems as a starting point. By proposing composable techniques such as MemBlaze, MemCorrect and MemDrowsy, we demonstrate their applicability to LPDDR2 making it viable at high bandwidth utilizations. Thus, we describe how the challenges of building energy proportional memory can be overcome.

Chapter 6

System Integration

Energy proportional and efficient memory with LPDDR2, enhanced with high speed link techniques like MemBlaze have high potential to offset datacenter operating power and costs. However, we need to reexamine on-chip systems, particularly the cache hierarchy to ensure overall system efficiency. In the current chapter, we quantify the energy interaction between the processor cache and the DRAM systems. In addition, we assess the economic benefit of deploying new memory systems by analyzing Total Cost of Ownership (TCO).

6.1 Caches

Memory performance and power efficiency impact processor cache architecture. In particular, modern applications exercise memory systems with large working set along with complex access patterns. The previous sections have demonstrated ways to make main memory systems energy efficient. With energy efficient main memory in place, the next bottleneck for system efficiency is in the memory hierarchy that includes the caches. In particular, the last level caches (LLC) play a crucial role in modern systems performance and energy efficiency. In the era of multi cores, LLC design is even more important as cores regularly contend for LLC capacity to minimize off-die communication.

While DRAMs are built for high density, LLCs are built with SRAMs, that are optimized for integration compatibility with processor technology with low access time. However SRAMs also dissipate large static power that makes memory energy efficiency challenging.

With accesses to main memory being expensive from both performance and energy perspective, prior work had emphasized high capacity LLC to filter accesses going to main memory [28, 10]. However energy efficient main memory like LPDDR2 could expose the static energy costs of LPDDR2. In addition, recent work had also considered STTRAM and eDRAM as potential replacement candidates for SRAM in LLC [8]. Clearly, the choice of design and technology of various levels of memory hierarchy impact the overall efficiency.

To quantify and understand these effects, we introduce a new metric called *Average Memory Access Energy (AMAE)*. This is analogous to the conventional Average Memory Access Time (AMAT). AMAE provides a way to evaluate new memory technologies by combining the effects of dynamic and static energy in both processor caches and main memory. Quantifying the average number of Joules per memory instruction,

$\mathtt{AMAE}_{\mathtt{L(i)}} = \mathtt{Ed}_{\mathtt{L(i)}} + \mathtt{Es}_{\mathtt{L(i)}} + \mathtt{MR}_{\mathtt{L(i)}} \times \mathtt{AMAE}_{\mathtt{L(i+1)}}$.

Dynamic energy for accessing level $\mathtt{i}$ in the memory hierarchy is denoted by $\mathtt{Ed}_{\mathtt{L(i)}}$, the miss rate is denoted by $\mathtt{MR}_{\mathtt{L(i)}}$. $\mathtt{MR}_{\mathtt{L(i)}}$ is the local fraction of misses of the cache level going to the next level of cache (ranging from 0.0 to 1.0). $\mathtt{Es}_{\mathtt{L(i)}}$ is the total static energy consumed during an application's execution amortized over the number of memory accesses. This is computed by dividing the static power of a memory level by the rate of load/store issue in that level of the cache. Alternatively, this can also be calculated as

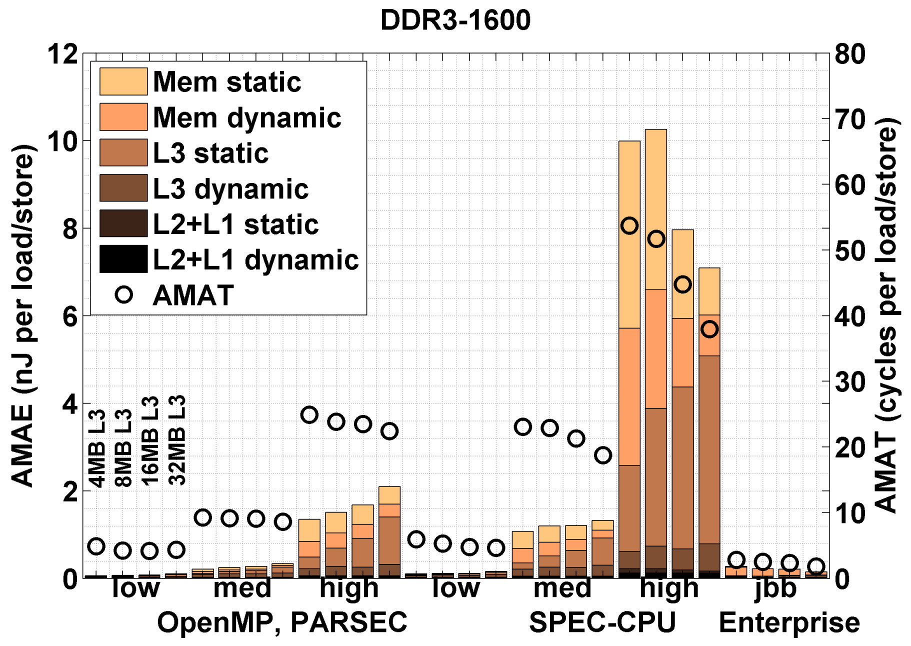
$$\frac{\sum_i (\texttt{Pd}_{\texttt{L(i)}} + \texttt{Ps}_{\texttt{L(i)}})}{\texttt{Rate of CPU Loads/Stores}}$$

where $\texttt{Pd}_{\texttt{L(i)}}$, $\texttt{Ps}_{\texttt{L(i)}}$ are the dynamic and static power of cache level $\texttt{i}$. We use CACTI[26] to estimate cache energy, DRAMSim[75] to estimate activate, read-write energy, and Micron power calculators [53] to estimate DQ energy.

Conventional wisdom is that the high latency and energy of DRAM, means that larger processor caches will improve average memory access time and energy. By reducing memory activity, caches also increase opportunities for DRAM low-power modes while reducing the likelihood of memory contention. However, larger caches dissipate more static power and are less effective for emerging applications (e.g., web search, memcached) whose working memory set is often a few GigaBytes and have LLC missrates approaching 100%. In such cases, large caches of tens of MBs are hardly justified.

Figure 6.1 illustrates AMAE for a variety of L3 cache sizes. The trade-offs between dynamic and static energy vary across applications, which are placed along the x-axis in order of decreasing memory intensity. Web applications like search and memcached have memory behaviour that matches that of low memory bandwidth applications. Accessing DDR3 is expensive and larger caches mitigate its cost. AMAT falls with cache size, especially for bandwidth-intensive applications. In contrast, the net change in AMAE from larger caches is modest. The increased energy of larger caches cancels, in large part, reductions in DRAM dynamic energy (due to fewer accesses) and DRAM static energy (shorter execution time).

Consuming less energy, LPDDR2 reduces AMAE when compared to DDR3. Lower LPDDR2 energy also magnifies the impact of static energy as L3 cache sizes increase. For many workloads, increasing the cache size leads to flat or increasing AMAE. Although larger caches reduce execution time, LPDDR2 is energy-proportional and



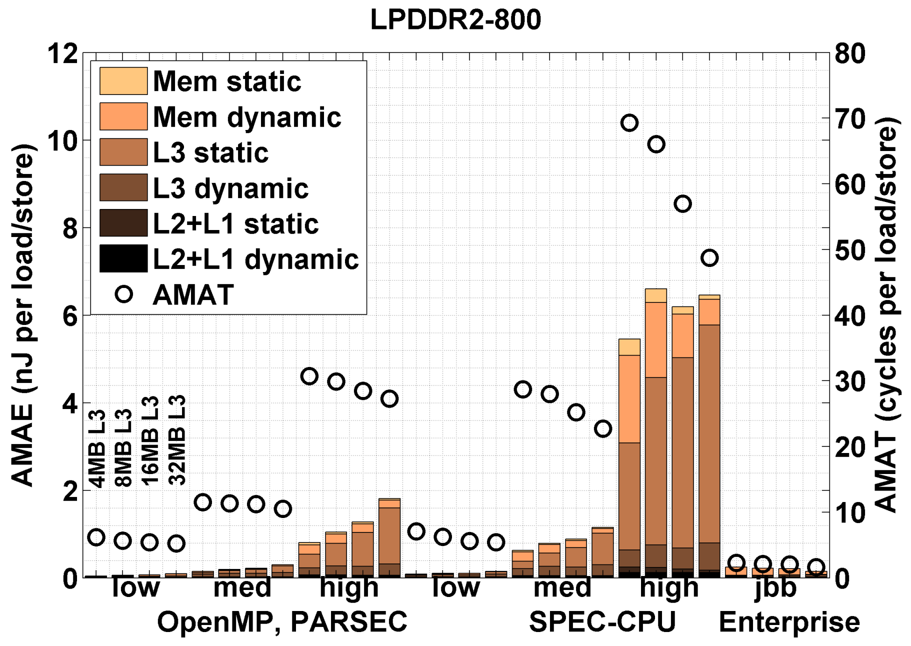


Figure 6.1: DDR3 and LPDDR2 average memory access energy (AMAE) in nJ per memory instruction and average memory access time (AMAT) in cycles per memory instruction. Shown for 4, 8, 6, and 32MB L3 cache sizes.

opportunities to further reduce memory static energy are small. On the other hand, larger caches introduce a new problem in cache static energy. For AMAE, it is preferable to pay the high dynamic energy accessing DRAM rather than rather than continuously consume high static energy for a large L3. Most importantly, perhaps, we illustrate an analysis framework and an AMAE metric that allows architects to reason about emerging memory technologies and their interactions with other layers in the cache and memory hierarchy.

Apart from caches, processors might not always be well matched to datacenters' emerging needs. While big cores have stopped scaling frequently running into thermal limits, throughput cores are expensive and are difficult to commoditize and also lack some important resources for datacenter applications. Small cores have difficulty in yielding parallelism to offset for energy efficient design that sacrifices single thread performance [73]. To rethink processor design choices, we can extend our AMAE metric to effectively guide design CPUs for better energy efficiency and performance.

6.2 Matching Processors and Memory

This section looks at the economic impact of deploying energy efficient LPDDR2 memory systems in datacenters. As servers adopt mobile hardware for efficiency, more of each dollar is spent on computing and less is spent on overheads. On the other hand, at least initially, capital costs for mobile hardware may be higher. Table 6.1 analyzes these trade-offs, accounting for capital costs in datacenter construction and IT equipment and operating costs from power [21, 62]. The model assumes \$0.07/kWh, \$200M facility cost, and a 15MW budget. Facility and IT capital costs are amortized over 15 and 3 years, respectively.

TCO-neutral prices. In the early stages of a new technology (e.g., LPDDR2-based servers) when costs are evolving, end-users might more tractably reason about

	Xeon+DDR3 8-cores		**Atom+DDR3** 16-cores		**Xeon+LPDDR2** 8-cores		**Atom+LPDDR2** 16-cores	
	Cost ($)	Power (W)	Cost ($)	Power (W)	Cost ($)	Power (W)	Cost ($)	Power (W)
Processor (2-socket)	760	125	360	25	760	125	360	25
Motherboard	200	30	1340	3	200	30	1340	3
Network Interface	0	5	0	5	0	5	0	5
Memory (32GB/2-sockets)	600	40	600	40	775	10	775	10
Storage (HDD)	100	10	100	10	100	10	100	10
Total	**1660**	**210**	**2400**	**83**	**1835**	**180**	**2575**	**53**
No. Servers ($\times 10^3$, in 15MW)	70		180		83		283	
TCO ($ per server per month)	$86.4		$86.4		$86.4		$86.4	
Capability	1.0		2.5		1.2		4.0	

Table 6.1: Define TCO-neutral price with Xeon+DDR3 baseline ($86). Atoms use TCO-neutral price for custom board ($1340). Atom+LPDDR2 shows TCO-neutral price for mobile memory ($775 for 32GB). Capability quantifies data center throughput normalized against Xeon+DDR3.

the price they would willingly pay for expected benefits. We define a TCO-neutral price for a component as the price that produces a TCO matching some baseline. Our baseline is Xeon+DDR3 (TCO=$86).

Until now, advances in processor efficiency have out-paced those for memory. Moreover, simpler OOO cores or in-order cores, such as mobile Atoms are being considered for servers. Conventional motherboards are over-provisioned for such cores motivating systems like the one from SeaMicro, which eliminates 90% of motherboard components [51]. By adopting Atoms or other power efficient processors, server power falls from 190 to 63W, a 3× reduction. To realize such efficiency, datacenters might willingly pay for custom boards as long as TCO does not change. By sweeping board prices, we calculate that TCO can be maintained at $86, if such a customization costs less than $1340.

As processor efficiency improves, memory becomes an efficiency bottleneck. DDR3 dissipates 4-5× more power than LPDDR2 for applications with moderate memory activity, such as web search. Because mobile memory reduces power costs, datacenters might willingly pay a premium for LPDDR2. By sweeping LPDDR2 prices to find a break even price, we find that Atom+LPDDR2 is justified if mobile memory

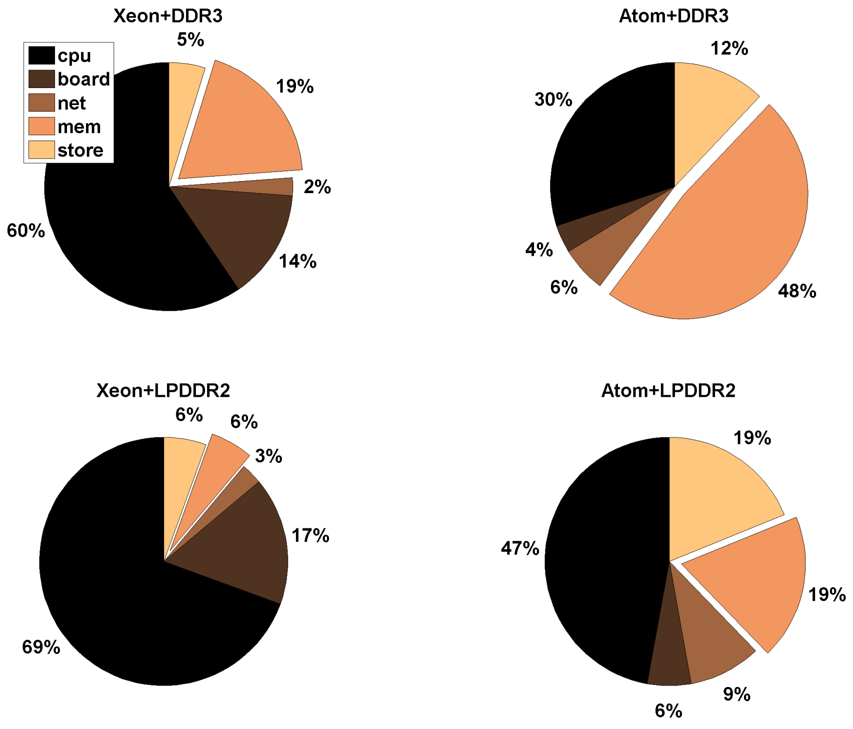


Figure 6.2: Power breakdown across server components.

prices are less than \$775 per 32GB, a 30% premium over DDR3 prices. This analysis is conservative because it precludes TCO increases, which might be justified by additional datacenter capacity enabled by mobile hardware.

Capacity. Mobile processors and memory shift power and TCO breakdowns (Figure 6.2, Figure 6.3). Of each dollar spent, 89% goes to server costs and not infrastructure overheads. In contrast, with Xeon+DDR3 servers, only 58% of costs go to servers. Table 6.1 presents datacenter capacity normalized to Xeon+DDR3 for web search based on published measurements [62]. Within a 15MW critical load, we can deploy 2.5× more 16-core Atom servers than 8-core Xeon servers, leading to a commensurate capacity increase even when taking into account that an Atom core sustains 0.5× the query throughput of a Xeon core. Atom+LPDDR2 power is even lower and allows a further 1.6× increase in the number of servers. Capacity increases

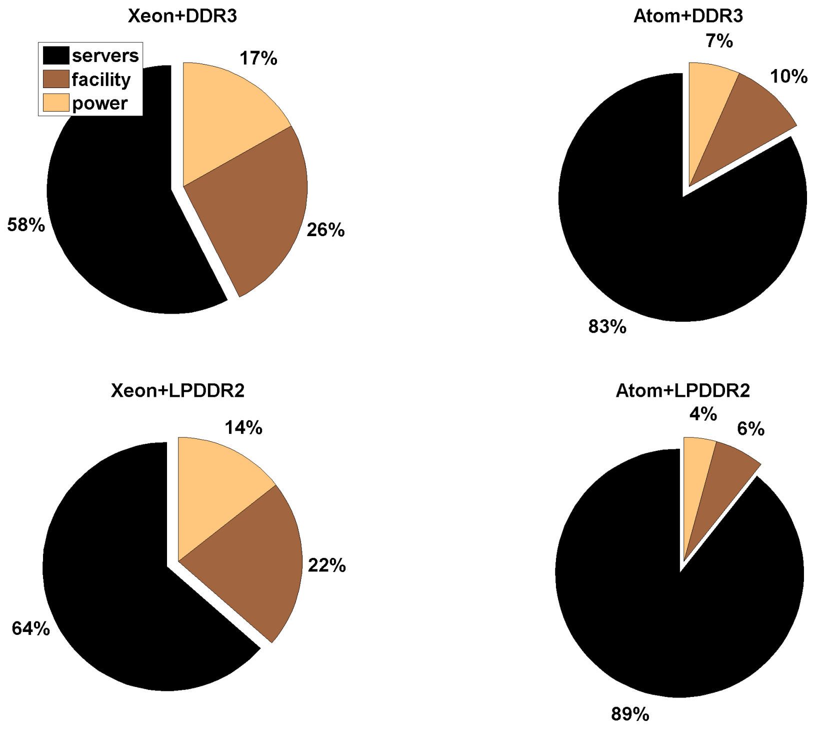


Figure 6.3: Total cost of ownership breakdown across servers and infrastructure.

by a cumulative total of 4.0× over Xeon+DDR3.

This analysis assumes no performance penalty from mobile memory. While true for search, which uses less than 10% of DDR3 peak bandwidth [41], other applications may see performance penalties that degrade the 1.6× gain from LPDDR2. For example, if application performance falls by 20% when using mobile memory, the 1.6× increase in servers is offset, in part, by the 0.8× impact on per server capacity.

In the previous chapters, we described how to build an energy proportional main memory for a variety of applications. In the current chapter, we extend the efficiency to processor caches. We developed a framework to help us make energy efficient choices for memory hierarchy. In addition, we have also described a TCO-driven methodology to make economic choices for large scale datacenters using efficient memory systems.

Chapter 7

Conclusion

This thesis addressed the challenge of making DRAM memory systems energy proportional and energy efficient. This is both an important and a challenging problem in the context of datacenters, where energy directly affects cost of operation. In particular, the thesis focused on datacenter servers which have a large memory footprint and spend 30-40% of total power in the memory systems.

We began by characterizing the DRAM behaviour of emerging large datacenter workloads like web search, social media, analytics and observe that these applications need high memory capacity but under-utilize memory bandwidth. This was a surprising observation, particularly because studies have conventionally placed importance on high bandwidth DRAMs. However, we learned that DRAM in datacenters is used as a fast storage device where latency and capacity are important. In addition, as datacenters regularly overprovision servers by a large factor, systems generally run at 30% utilization, running their subsystems including DRAMs at even more low utilizations. As a result, energy proportionality is particularly important from an overall energy perspective.

The next interesting observation we make is that DDR3 memory systems are particularly inefficient in this low utilization region. DDR3 was optimized for high bandwidth with a high speed interface. Unfortunately, this interface also dissipates lot of static power and in the region of low utilization, this active-idle power is amortized over less work, making the energy/bit cost very expensive. DDR powerdown mechanisms do not help either, because the wakeup time of energy efficient powermodes is very large and datacenter workloads do not justify using such modes. This problem was particularly unique because on one hand, the average memory bandwidth is low and yet, the bus utilization is light with very little fully-idle opportunities to powerdown DRAMs.

These facts motivated using LPDDR2 memory in datacenters, though they were originally optimized for the mobile market. We make a surprising observation that commodity LPDDR2 DRAM chips are very well suited to exactly this scenario of low memory utilization. LPDDR2 systems have power efficient interfaces that significantly reduce interface power by eliminating components like DLLs and ODT. In the absence of these static energy overheads, they have a constant energy/bit which is also very low, making it an excellent choice at low memory utilizations.

Even though the energy/bit compels us to use LPDDR2 in datacenter servers, the power efficient interface presents many challenges that need to be addressed. We first studied the system implications of using LPDDR2 systems. LPDDR2 memory systems tradeoff underutilized peak bandwidth, resulting in significant gains in energy efficiency and proportionality. However, they make high capacity memory system design challenging. To mitigate this problem for datacenters, where memory capacity is particularly important, we describe a novel architecture design to build scalable and large capacity LPDDR2 systems. Using our proposed architecture, we show large 3-5$\times$ power savings with negligible performance degradation on datacenter workloads.

While LPDDR2 systems are both energy proportional and efficient for datacenter workloads, we see performance degradation on memory bandwidth intensive applications. To address this and make LPDDR2 systems applicable to this scenario also, we then explore building low-power high-speed interfaces.

We proposed three different methodologies: MemBlaze, MemCorrect and MemDrowsy which modify the memory interface to different degrees and enable the DRAMs to rapidly transition to deep powerdown states. By eliminating or mitigating long-latency DLL wake-ups, these systems aggressively uses efficient powerdown states during short idle periods with negligible performance penalty. Equally applicable to both DDR3 and LPDDR2, they yield the same 3-5$\times$ energy savings with LPDDR2 with small or no performance impact, even under the high bandwidth scenario.

Having described how to build an energy proportional main memory for a variety of applications, we make a surprising observation that the static power of large last level caches (Ex: 30MB in Intel Xeon) has begun to compete with DRAM dynamic power. We then developed a framework using a new metric called Average Memory Access Energy to help us make energy efficient choices for memory hierarchy. Using this metric, we show that there are energy optimal cache sizes, larger is no longer always better, which is different from conventional studies. Finally, we learn how a TCO-driven methodology can help us to quantitatively make viable economic choices for datacenters deploying the efficient memory systems.

Energy efficiency has emerged as the single most important criterion in computer system design. With Moore's law running into its limits, systems have become power-limited, making it difficult to scale performance without addressing power. In addition, datacenters are growing tremendously in size, where provisioning energy and cooling infrastructure costs millions of dollars. In response, this thesis directly addresses the energy efficiency of memory systems, that are the biggest contributors to energy in systems. With significant 5-6$\times$ energy savings without any performance degradation, the thesis contributions can positively impact future computer systems.

Bibliography

[1] J.H. Ahn et al. Future scaling of processor-memory interfaces. In *SC*, 2009.

[2] AMD. Amd, bios and kernel developers guide for amd npt family 0fh processors. Technical report, 2007.

[3] M. Awasthi et al. Handling the problems and opportunities by multiple on-chip memory controllers. In *PACT*, 2011.

[4] L. Barroso et al. The case for energy proportional computing. *IEEE Computer*, 2007.

[5] Luiz Andre Barroso and Urs Holzle. *The Datacenter as a Computer.* Morgan and Claypool, 2009.

[6] A. Bosque et al. Characterization of Apache web server with Specweb2005. In *MEDEA*, 2007.

[7] Bossen et al. b-adjacent error correction. *IBM Journal of Research and Development*, 1970.

[8] Mu-Tien Chang, Paul Rosenfeld, Shih-Lien Lu, and Bruce Jacob. Technology comparison for large last-level caches (l3cs): Low-leakage sram, low write-energy stt-ram, and refresh-optimized edram. In *High Performance Computer Architecture (HPCA2013), 2013 IEEE 19th International Symposium on*, 2013.

[9] Chih-Kong and Ken Yang. Delay-locked loops - an overview. In *Phase Locking in High-Performance Systems. IEEE Press*, 2003.

[10] P. Conway, N. Kalyanasundharam, G. Donley, K. Lepak, and B. Hughes. Cache hierarchy and memory subsystem of the amd opteron processor. *Micro, IEEE*, 2010.

[11] Brian F. Cooper, Adam Silberstein, Erwin Tam, Raghu Ramakrishnan, and Russell Sears. Benchmarking cloud serving systems with YCSB. In *ACM Symposium on Cloud Computing*, 2010.

[12] H. David, O. Mutlu, et al. Memory power management via dynamic voltage/frequency scaling. In *ICAC*, 2011.

[13] V. Delaluz et al. DRAM energy management using software & hardware directed power mode control. In *HPCA*, 2001.

[14] Qingyuan Deng et al. Memscale: Active low-power modes for main memory. In *ASPLOS*, 2011.

[15] Bruno Diniz, Dorgival Guedes, and Ricardo Bianchini. Limiting the power consumption of main memory. In *ISCA*, 2007.

[16] Xiangyu Dong, Xiaoxia Wu, Guangyu Sun, Yuan Xie, Helen Li, and Yiran Chen. Circuit and microarchitecture evaluation of 3d stacking magnetic ram (mram) as a universal memory replacement. In *Proceedings of the 45th annual Design Automation Conference*, pages 554–559, 2008.

[17] X. Fan, C.S. Ellis, and A.R. Lebeck. Memory controller policies for DRAM power management. In *ISLPED*, 2001.

[18] Michael Ferdman, Almutaz Adileh, Onur Kocberber, Stavros Volos, Mohammad Alisafaee, Djordje Jevdjic, Cansu Kaynak, Adrian Daniel Popescu, Anastasia Ailamaki, and Babak Falsafi. Clearing the clouds: a study of emerging scale-out workloads on modern hardware. In *International Conference on Architectural Support for Programming Languages and Operating Systems*, 2012.

[19] B. Ganesh et al. FB DIMM memory architectures: Understanding mechanisms, overheads, & scaling. In *HPCA*, 2007.

[20] Xiaochen Guo, Engin Ipek, and Tolga Soyata. Resistive computation: avoiding the power wall with low-leakage, stt-mram based computing. In *Proceedings of the 37th annual international symposium on Computer architecture*, pages 371–382, 2010.

[21] J. Hamilton. Cost of power in large-scale data centers. `http://perspectives.mvdirona.com`.

[22] H. Hanson et al. What computer architects need to know about memory throttling. In *WEED*, 2010.

[23] Steven Hart, Eitan Frachtenberg, and Mateusz Berezecki. Predicting memcached throughput using simulation and modeling. In *Proceedings of the 2012 Symposium on Theory of Modeling and Simulation - DEVS Integrative M&S Symposium*, 2012.

[24] Hewlett-Packard. DDR3 memory technology. Technology brief TC100202TB, Hewlett-Packard, 2010.

[25] U. Hoelzle and L.A. Barroso. *The Datacenter as a Computer*. Morgan and Claypool, 2009.

[26] HP. CACTI 5.1. `http://www.hpl.hp.com/techreports/2008/HPL-2008-20.html`.

[27] H. Huang et al. Improving energy efficiency by making DRAM less randomly accessed. In *ISLPED*, 2005.

[28] Min Huang, M. Mehalel, R. Arvapalli, and Songnian He. An energy efficient 32nm 20 mb l3 cache for intel x00ae; xeon x00ae; processor e5 family. In *Custom Integrated Circuits Conference (CICC), 2012 IEEE*, 2012.

[29] I. Hur and C. Lin. A comprehensive approach to DRAM power management. In *HPCA*, 2008.

[30] IBM Microelectronics Division. A white paper on the benefits of chipkill-correct ecc for pc server main memory. Technical report, IBM, 1997.

[31] Intel. Intel memory 3-sigma power analysis methodology. Data sheet, Intel.

[32] Intel. Intel xeon processor e3-1200 family datasheet. Data sheet, Intel, 2011.

[33] Process integration, devices and structures. Technical report, International Roadmap for Semiconductors, 2011.

[34] B. Jacob et al. *Memory Systems: Cache, DRAM, Disk.* Morgan Kaufmann, 2007.

[35] A. Jaleel et al. High performance cache replacement using re-reference interval prediction (RRRIP). In *ISCA*, 2010.

[36] JEDEC. JEDEC standard for LP-DDR2. Standard JESD209-2B, JEDEC, 2010.

[37] K. Kaviani et al. A 6.4-Gb/s near-ground single-ended transceiver for dual-rank dimm memory interface systems. In *International Solid-State Circuits Conference*, February 2013.

[38] Steve Keckler. Life after dennard and how i learned to love the picojoule. Keynote at the he 44th Intl. Symposium on Microarchitecture, December 2011.

[39] Kim et al. Thread cluster memory scheduling: Exploiting differences in memory access behavior. In *MICRO*, 2010.

[40] Jaeha Kim, Mark Horowitz, and Gu-Yeon Wei. Design of cmos adaptive-bandwidth plls/dlls: A general approach. In *IEEE Transactions on Circuits and Systems-II: Analog and Digital Signal Processing*, 2003.

[41] C Kozyrakis, S Sankar, et al. Server Engineering Insights for Large-Scale Online Services. *IEEE Micro*, 2010.

[42] A. Lebeck, X. Fan, H. Zeng, , and C. Ellis. Power aware page allocation. In *ASPLOS*, 2000.

[43] B.C. Lee, E. Ipek, O. Mutlu, and D. Burger. Architecting phase change memory as a scalable DRAM alternative. In *Proc. 36th IEEE/ACM International Symposium on Computer Architecture (ISCA)*, June 2009.

[44] B.C. Lee, P. Zhou, J. Yang, Y. Zhang, B. Zhao, E. Ipek, O. Mutlu, and D. Burger. Phase change technology and the future of main memory. *IEEE MICRO*, 2010.

[45] K. Lim et al. Understanding and designing new server architectures for emerging warehouse-computing environments. In *ISCA*, 2008.

[46] K. Lim et al. Disaggregated memory for expansion and sharing in blade servers. In *ISCA*, 2009.

[47] Wi-fen Lin. Reducing dram latencies with an integrated memory hierarchy design. In *Proceedings of the 7th International Symposium on High-Performance Computer Architecture*, 2001.

[48] C. Luk et al. Pin: Building customized program analysis tools with dynamic instrumentation. In *PLDI*, 2005.

[49] D. Meisner, B. Gold, and T. Wensich. PowerNap: Eliminating server idle power. In *ISCA*, 2009.

[50] David Meisner, Christopher M. Sadler, Luiz Andre Barroso, Wolf-Dietrich Weber, and Thomas F. Wenisch. Power management of online data-intensive services. In *International Symposium on Computer Architecture*, 2011.

[51] R. Merritt. Startup SeaMicro packs 512 Intel Atoms in server. *EE Times*, 2010.

[52] Micron. Calculating memory system power for DDR2. Technical Note tn4704, Micron, 2005.

[53] Micron. Calculating memory system power for DDR3. Technical Note TN-41-01, Micron, 2007.

[54] Micron. 152-ball x32 mobile lpddr pop (ti-omap). Data Sheet MT46HxxxMxxLxCG, Micron, 2008.

[55] Micron. Micron 2Gb: x4, x8, x16 DDR3 SDRAM. Data Sheet MT41J128M16HA-125, Micron, 2010.

[56] Micron. Micron 16GB: x72, DDR3 LRDIMM. Data Sheet MT36KSZF2G72LDZ-1G6, Micron, 2011.

[57] J. Ousterhout et al. The case for RAMClouds:scalable high-performance storage entirely in DRAM. *SIGOPS*, 2010.

[58] R. Palmer et al. A 4.3GB/s mobile memory interface with power-efficient bandwidth scaling. In *VLSI*, 2009.

[59] V. Pandey, W. Jiang, Y. Zhou, and R. Bianchini. DMA-aware memory energy management. In *HPCA*, 2006.

[60] M. Qureshi, V. Srinivasan, and J. Rivers. Scalable high performance main memory system using phase-change memory technology. In *Proc. 36th IEEE/ACM International Symposium on Computer Architecture (ISCA)*, June 2009.

[61] Rambus. Mobile XDR memory versus LPDDR2. `http://www.rambus.com/us/technology/solutions/mobile_xdr/mxdr_vs_lpddr2.html`.

[62] V.J. Reddi et al. Web search using mobile cores: Quantifying and mitigating the price of efficiency. In *ISCA*, 2010.

[63] B.M. Rogers, A. Krishna, G.B. Bell, K. Vu, X. Jiang, and Y. Solihin. Scaling the bandwidth wall: Challenges in and avenues for CMP scaling. In *ISCA*, 2009.

[64] P. Saab. Scaling memcached at Facebook. *Facebook Engineering Note*, 2008.

[65] D. Sanchez et al. The ZCache: Decoupling ways and associativity. In *MICRO*, 2011.

[66] R. Schmitt et al. Signal and power integrity limitations for mobile memory in 3D packaging. *EE Times*, 2010.

[67] N. Sharma, S. Barker, D. Irwin, and P. Shenoy. Blink: Managing server clusters on intermittent power. In *ASPLOS*, 2011.

[68] Navin Sharma, Sean Barker, David Irwin, and Prashant Shenoy. Blink: Managing server clusters on intermittent power. In *International Conference on Architectural Support for Programming Languages and Operating Systems*, 2011.

[69] K. Shiv et al. SPECjvm2008 performance characterization. In *SPEC Benchmark Workshop on Computer Performance Evaluation and Benchmarking*, 2009.

[70] L. Tang et al. The impact of memory subsystem resource sharing on datacenter applications. In *ISCA*, 2011.

[71] M. Tolentino et al. Memory MISER: Improving main memory energy efficiency in servers. *IEEE Trans*, 2009.

[72] Aniruddha N. Udipi, Naveen Muralimanohar, Niladrish Chatterjee, Rajeev Balasubramonian, Al Davis, and Norman P. Jouppi. Rethinking DRAM design and organization for energy-constrained multi-cores. In *International Symposium on Computer Architecture*, 2010.

[73] Urs Holzle. Brawny cores still beat wimpy cores, most of the time. Technical note, Google, 2010.

[74] T. Vogelsang. Understanding the energy consumption of dynamic random access memories. In *MICRO*, 2010.

[75] David Wang et al. DRAMSim2: A cycle accurate memory system simulator. *IEEE Comput. Archit. Letters*, 2011.

[76] Xiaobin Wang and Yiran Chen. Spintronic memristor devices and application. In *Proceedings of the Conference on Design, Automation and Test in Europe*, 2010.

[77] F.A. Ware and C. Hampel. Improving power and data efficiency with threaded memory modules. In *ICCD*, 2006.

[78] Doe Hyun Yoon and Mattan Erez. Virtualized and flexible ecc for main memory. In *International Conference on Architectural Support for Programming Languages and Operating Systems*, 2010.

[79] Jared Zerbe, Barry Daly, Lei Luo, Bill Stonecypher, Wayne D. Dettloff, John C. Eble, Teva Stone, Jihong Ren, Brian S. Leibowitz, Michael Bucher, Patrick Satarzadeh, Qi Lin, Yue Lu, and Ravi Kollipara. A 5gb/s link with matched source synchronous and common-mode clocking techniques. In *IEEE Journal of Solid-State Circuits*, 2011.

[80] L Zhao et al. Exploring Large-Scale CMP Architctures using Manysim. *IEEE Micro*, 2007.

[81] Hongzhong Zheng, Jiang Lin, Zhao Zhang, Eugene Gorbatov, Howard David, and Zhichun Zhu. Mini-rank: Adaptive DRAM architecture for improving memory power efficiency. In *International Symposium on Microarchitecture*, 2008.

[82] Ping Zhou, Bo Zhao, Jun Yang, and Youtao Zhang. Energy reduction for stt-ram using early write termination. In *Proceedings of the 2009 International Conference on Computer-Aided Design*, pages 264–268, 2009.