

# Leveraging Heterogeneity in DRAM Main Memories to Accelerate Critical Word Access \*

Niladrish Chatterjee<sup>‡</sup> Manjunath Shevgoor<sup>‡</sup> Rajeev Balasubramonian<sup>‡</sup> Al Davis<sup>‡</sup>  
Zhen Fang<sup>§†</sup> Ramesh Illikkal<sup>∞</sup> Ravi Iyer<sup>∞</sup>

<sup>‡</sup>University of Utah {nil,shevgoor,rajeev,ald}@cs.utah.edu  
<sup>§</sup>Nvidia Corporation zfang@nvidia.com  
<sup>∞</sup>Intel Labs {ramesh.g.illikkal,ravishankar.iyer}@intel.com

## Abstract

*The DRAM main memory system in modern servers is largely homogeneous. In recent years, DRAM manufacturers have produced chips with vastly differing latency and energy characteristics. This provides the opportunity to build a heterogeneous main memory system where different parts of the address space can yield different latencies and energy per access. The limited prior work in this area has explored smart placement of pages with high activities. In this paper, we propose a novel alternative to exploit DRAM heterogeneity. We observe that the critical word in a cache line can be easily recognized beforehand and placed in a low-latency region of the main memory. Other non-critical words of the cache line can be placed in a low-energy region. We design an architecture that has low complexity and that can accelerate the transfer of the critical word by tens of cycles. For our benchmark suite, we show an average performance improvement of 12.9% and an accompanying memory energy reduction of 15%.*

## 1. Introduction

The main memory system is a significant performance and energy bottleneck in modern high-performance platforms. Several trends threaten to amplify the memory bottleneck. It is increasingly evident that overall performance/cost and performance/watt are improved by using simpler cores [10]. This shifts the energy bottleneck from processor to memory. Simpler cores offer increased core count per die, placing greater pressure on memory bandwidth. Simpler cores also tend to be less effective in hiding memory latency. There is also a demand for higher memory capacities because of the data-intensive workloads that execute on many datacenter platforms. For example, the RAMCloud project argues that disks are incapable of fulfilling high query rates in future datacenters and disks must be replaced by large DRAM memories accessed via a network [31]. There is also a push towards in-memory databases within the database community. This trend will also increase the contribution of the memory system to overall system energy and performance.

To date, the main memory system has been largely homogeneous; the DIMMs that are employed in a server have uniform timing parameters and only exhibit minor variations. But it is well known that all data elements do not have a uniform impact on overall performance. Some data elements are more *critical* to performance than others. This argues for having a heterogeneous main memory design where critical words or cache lines are placed in low-latency portions of memory and non-critical elements are placed in low-energy portions. Little prior work has explored the use of different types

of DRAM chips to build and exploit a heterogeneous memory system. This paper takes an important step in uncovering the potential of such a heterogeneous DRAM memory system.

The DRAM industry already produces chips with varying properties. Micron offers a Reduced Latency DRAM (RLDRAM) product that offers lower latency and lower capacity, and is targeted at high performance routers, switches, and network processing [7]. Micron also offers a Low Power DRAM (LPDRAM) product that offers lower energy and longer latencies and that has typically been employed in the mobile market segment [6]. Our work explores innovations that can exploit a main memory that includes regular DDR chips as well as RLDRAM and LPDRAM chips. To reduce complexity, we do not integrate different types of DRAM chips on the same DIMM or even the same channel. We assume that each DRAM chip type has its own DIMM and can be accessed via its own channel and memory controller. We propose a *critical-word-first (CWF)* optimization that organizes a single cache line across multiple channels. We observe that the first word in a cache line is typically most critical and benefits from being placed in a low-latency channel/DIMM. Other non-critical words in the cache line can be placed in a low-energy channel/DIMM, thus improving performance *and* reducing energy. The phenomenon of critical word regularity was first reported by Gieske [17] and was exploited in the same work to optimize accesses to the on-chip SRAM caches. Our results show that the CWF optimization yields an average 12.9% performance improvement and a 15% memory energy reduction, while incurring low implementation overheads.

Modern memory systems already incorporate a CWF optimization where the critical word is returned before the other words when servicing a cache line request (p. 372 [19], [13]). However, this only prioritizes the critical word by a few CPU cycles, while not yielding energy efficiency for the non-critical words. We show that our CWF optimization prioritizes the critical word by 70 CPU cycles on average because the critical word has its own independent channel with lower queuing delays, *and* it allows non-critical words to be placed in low-energy DRAM chips. It is therefore a novel and significant advancement over the state-of-the-art.

## 2. Background

In a modern server, the main memory system is comprised of multiple channels, each having one or more Dual Inline Memory Modules (DIMMs). Each DIMM consists of several DRAM chips which are grouped into ranks. All DRAM chips in a rank work in unison to service a cache line request. Each rank is itself partitioned into many banks and all these banks may be in operation at the same time, each servicing a different request, subject to various timing constraints. Ranks and banks on a single channel enable memory-level parallelism, but the accesses are finally serialized on the channel because

\* This work was supported in parts by NSF grants CCF-0811249, CCF-0916436, NSF CAREER award CCF-0545959, and the University of Utah.

† Work done while at Intel Labs.

the data transfer has to utilize this shared resource. Each bank in a DRAM chip consists of 2D arrays of DRAM cells and a row buffer which stores the most recently accessed row of the bank. An activate operation fetches data from an entire row (on the order of 8 KB) into the row buffer. A column-read command transfers a cache line from the row buffer to the processor. If a subsequent access is to the same row, it yields a row buffer hit and can be completed at a lower delay and energy cost. Before activating a new row, a precharge operation must be performed. A close-page policy precharges a bank after every column-read command to prepare the bank for access to a new row. An open-page policy keeps the row open to service row buffer hits and precharges the bank only when a new row is encountered. We next examine the specific properties of different flavors of DRAM chips that can be used to construct memory systems.

### 2.1. DDR3

The most widely used DRAM variant in mid- to high-end computing is the Dual-Data-Rate (DDR3) module. DDR3 DRAM chips attempt to strike a balance between performance and density and are produced in very high volumes for desktop, laptop, and server market segments. We model a standard x8 (data output width of 8 bits), 800 MHz DDR3 part that has a pin data bandwidth of 1600MBps [28] and a capacity of 2Gb. Each DRAM chip has an 18 bit address and a 5 bit command bus. The address is provided in two cycles (row and column). Two primary determinants of the possible throughput of a DRAM chip are its bank count and bank-turnaround time. While a larger number of banks promotes better concurrency, the bank-turnaround time (tRC) determines the minimum time that needs to elapse before a new row in the bank can be activated. To maximize density, DDR3 chips are seldom designed with more than 8 banks, and DDR3 bank-turnaround is about 50 ns.

### 2.2. LPDRAM

Low Power DRAM or mobile DRAM chips are used as an alternative to DDR3 chips in mobile devices. Designed for low-power usage, LPDDR2 [29] has lower per pin bandwidth due to the lower operating frequency (400 MHz). The core densities and bank counts remain the same across LPDDR2 and DDR3. Due to lower operating voltages, the core latencies are increased by a small fraction leading to a bank-turnaround time of 60 ns. The lower voltages allow low active power consumption; the output drivers are also designed to consume less power. In addition to regular low-power modes, LPDDR2 supports many additional low-power modes like Temperature Compensated Self-Refresh (TCSR), Partial Array Self-Refresh (PASR), and Deep Power Down (DPD) modes, all suited for use in a mobile environment. While current LPDDR2 modules employ a 10-bit address/command bus that is dual-pumped, future low-power DRAM chips could employ DDR3-style 23-bit address/command buses.

### 2.3. RLD RAM

Reduced Latency DRAM (RLDRAM3 [30]) was designed as a deterministic latency DRAM module for use in high-speed applications such as network controllers [40]. While the pin-bandwidth of RLDRAM3 is comparable to DDR3, its core latencies are extremely small, due to the use of many small arrays. This sacrifice in density is justified by a bank-turnaround time (tRC) of 10-15 ns compared to 50 ns for DDR3, and a higher bank count (16 as opposed to 8 for DDR3). The maximum capacity offered by an RLDRAM3 chip

currently (576 Mb) is several times less than the capacity of a DDR3 chip - however Micron roadmaps [1] show plans for 1Gb and 2Gb parts in the near future. In DDR3 devices, to limit the current draw, a timing window tFAW is defined, during which only 4 bank activations can be issued. RLDRAM does not have any such restrictions, favoring low latency over peak power guarantees. RLDRAM uses SRAM-style addressing - the entire address is provided with a single READ or WRITE command, instead of separate RAS and CAS commands (although, this could be modified in future RLDRAM chips). After a read or a write, the bank gets precharged automatically. Thus, effectively, an RLDRAM chip can only operate with a close-page policy.

## 3. Motivational Data

Most high-end servers employ DDR3 parts because they provide the best combination of memory capacity and high performance. If capacity constraints are removed, the use of RLDRAM3 chip latencies can lead to a dramatic improvement in performance. Likewise, if DDR3 parts are replaced with LPDDR2 parts, performance is worsened, but memory energy is reduced. The graph in Figure 1.(a) shows the sensitivity of applications to these different DRAM properties, while still presenting a homogeneous main memory system. We show results for an 8-core system running programs from SPEC2k6 and NAS Parallel Benchmark (NPB) suites. We show results for homogeneous memory systems where the chips are all either RLDRAM3, DDR3, or LPDDR2. More methodology details are in Section 5.

RLDRAM3 outperforms DDR3 by 31% across the simulated workloads (Figure 1a) while LPDDR2 suffers a 13% reduction in throughput. The average main memory access time of RLDRAM3 is about 43% lower than that of DDR3. The breakdown of the latencies in Figure 1b shows that each memory read request spends substantially less time in the queue in a RLDRAM3 system (*Queue Latency*) compared to DDR3, and also requires less time to be serviced from the memory array (*Core Latency*). The low bank-turnaround time of RLDRAM3 (10ns, compared to 50ns for DDR3) leads to the low queuing delay. The lack of write-to-read turnaround delays, as well as the higher bank counts of RLDRAM3, also allow it to provide 24% higher sustained bandwidth compared to DDR3, even though the pin bandwidth of the RLDRAM3 system is the same as that of the DDR3 baseline. On the other hand, the LPDDR2 latency is 41% higher than DDR3. The higher latency of LPDDR2 is due to the higher bank-turnaround time, the slower arrays (due to lower voltages) as well as the lower data bus frequency.

However, RLDRAM3 offers much lower capacity. It is well-known that many applications require large amounts of main memory space [21, 31, 36, 42] and frequent page faults can lead to severe drop in performance -for example, Qureshi et al. quantify the page fault rates seen in several representative benchmarks as a function of DRAM capacity [36]. Thus, an RLDRAM3 system that has the same silicon area as a DDR3 system will have much lower capacity and potentially lower performance for applications with large memory footprints.

While RLDRAM3 is a clear winner in terms of access latencies, it incurs a power overhead. Figure 2 shows the power consumption of the 3 DRAM variants for different bus utilization values. Section 5 outlines the power calculation methodology. We see that at low utilization, the RLDRAM3 power consumption is much higher (due to the high background power), while at higher activity scenarios,



width. The memory controller complexity is also increased by striping some pages across 3 DIMMs (for high bandwidth [35]) and some pages across a single DIMM. Therefore, in our work, we employ a high-performance DRAM module that is built with RLDRAM devices and that offers low latency and high bandwidth. We employ a second type of DRAM module that is built with LPDRAM devices and that has lower bandwidth, higher latency, and lower power. A third type of DRAM module can also be built out of regular DDR3 devices that represents an intermediate design point in terms of performance and power. To reduce cost and complexity, we implement DIMMs that are themselves homogeneous, i.e., a DIMM only has chips of one type.

**Homogeneous DIMMs on a Channel.** In the proposed system, each DIMM type has its own dedicated channel. The memory controller would have high complexity if a single channel controlled DIMMs of different types. In all of our discussions, for simplicity, we assume that only 1 or 2 DIMMs of the same type are attached to each channel. This is consistent with projections that future DDR generations will only support 1-2 DIMMs per high-speed channel. However, our proposals are equally applicable to configurations that can support more DIMMs or ranks per channel. Modern high-performance processors implement multiple memory channels. Our heterogeneous memory system is built by designating a subset of these channels as performance-optimized channels and a subset as power-optimized channels. In our designs, our performance-optimized DIMM/channel is representative of state-of-the-art RLDRAM technology and operates at high frequency. Our power-optimized DIMM/channel uses LPDRAM chips at lower frequency, an open-row policy (that is known to minimize energy), and an aggressive sleep-transition policy.

**Adapting LPDRAM and RLDRAM for Server Memories.** In our work, we rely on existing DRAM chips and do not introduce extensive chip optimizations of our own. We therefore restrict ourselves to modeling DRAM chips that closely resemble regular DDR3, LPDRAM, and RLDRAM properties. We allow a few modifications that can be easily integrated into future generations of these chips. For example, we assume that x8 and x9 RLDRAM chips can be supported (only x18 and x36 are available today) and that LPDRAM chips support ODT and DLL.

As described in Section 2, RLDRAM parts are designed for low latency response as well as high sustained bandwidth. In such environments, RLDRAM chips are soldered onto the motherboard, communicating one-on-one with the processor. LPDRAM modules, used in the embedded domain, are also soldered onto the system PCB. This is in contrast to a regular server/desktop environment where individual DDR chips form a DIMM, which is then plugged onto the DRAM channel on the motherboard. While using LPDRAM parts in a DIMM form factor, it is important that additional features be implemented to ensure signal timing and integrity, particularly when high-speed signalling is employed. RLDRAM parts already employ these features.

**ODT :** An important feature present in modern DDR memory is on-die-termination (ODT) for signals. The purpose of ODT is to match the terminating impedance on the transmission die to the impedance of the transmission line. A mismatch in the terminating impedance leads to signal reflections which act as a source of noise on the memory channel. RLDRAM devices, by virtue of their high-speed operating environments, already incorporate ODT for signal integrity. LP-

DRAM devices, typically operated in power-sensitive systems, use lower frequencies, and hence, LPDDR2 is not equipped with ODT. However, LPDDR3 chips will incorporate ODT resistors to increase signal integrity in high-frequency environments [32]. ODT resistors can be incorporated in the LPDDR2 chips with very little area overhead as they reside near the data pins [43]. The main overhead of introducing the termination resistors is the increase in static power which we model in the same way as in a DDR3 chip.

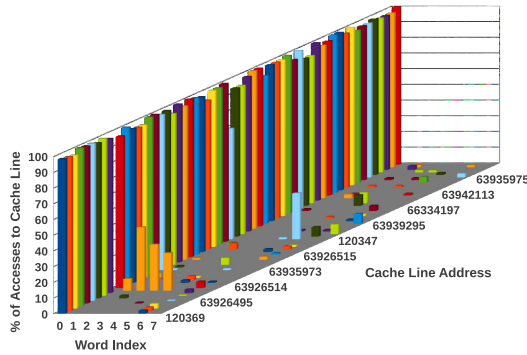
**DLL :** Modern DDRx SDRAM devices utilize an on-chip DLL (delay-locked loop) to synchronize the data signals and the source-synchronous clocking reference signal DQS with the global clock used by the memory controller. The data sent out on the data bus by a DIMM is synchronous with the DQS signal which in turn is used by the memory controller to sample the data. To make sure that the skew between the global clock fed to the DRAM chip and the DQS signal seen by the memory controller is not catastrophic, the on-chip DLL delays the DQS signal (and the data signals) enough so as to put them in phase with the global clock [19]. DDRx devices as well as RLDRAM 3 devices already use DLLs for this purpose while LPDRAM chips, due to their power constraints, do not have any such feature. We assume that DLLs can be integrated into the I/O circuitry of the LPDRAM chips at a power cost. In Section 5 we describe how we account for this increased power cost in our simulations.

**Silicon Cost:** It is well known that changes to a DRAM chip must be very sensitive to area and cost. A recent paper [43] pointed out that changes made to a DRAM chip are most disruptive to chip density when they are instituted in the bit-line sense-amplifier stripe, followed by the local word-line driver stripe, then in the column logic, and finally in the center stripe. To implement the DLLs and the ODTs, we need to introduce and modify circuits close to the I/O pads near the center stripe - we can therefore localize the changes to the least sensitive portion of the DRAM chip. The inclusion of ODTs in future generation LPDDRs (LPDDR3 [32]) demonstrates that adapting LPDRAM for higher frequency and long channel environments is indeed commercially viable.

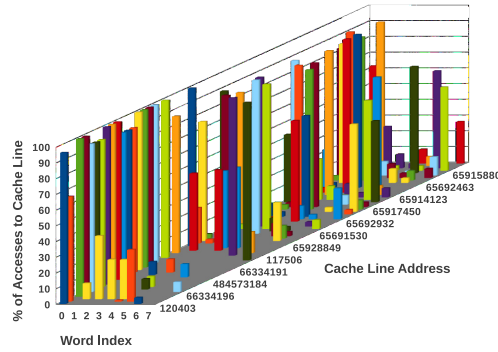
We have thus instituted a low-complexity heterogeneous main memory by introducing the following features: (1) the use of only two types of DIMMs (performance-optimized vs. power-optimized) in one design, (2) the use of only one type of DIMM per channel, (3) the integration of DLL and ODT in LPDRAM chips to make them server-class. It should also be noted that the use of heterogeneity need not be exposed to application developers and does not pose programmer complexity. Similar to the non-uniformity introduced by row buffer hits/misses, memory latencies will differ based on whether heterogeneity is successfully exploited or not.

## 4.2. Accelerating Critical Words

**4.2.1. Motivation.** The few prior efforts [35, 37] that have considered non-uniform or heterogeneous memories have attempted to classify data at the granularity of cache lines or OS pages. Instead, we focus on another data criticality opportunity that has not been previously leveraged by heterogeneous memories. A cache line is fetched from memory when a given CPU word request results in on-chip cache misses. It is well known that the word requested by the CPU is more critical than the other words in that cache line. When transferring the cache line, it is re-ordered so that the critical word is sent before the other words, thus reducing access latency by a few cycles [19, 18]. We argue that the transfer of the critical word can



(a) leslie3d



(b) mcf

Figure 3: Distribution of critical words in highly accessed cache-lines

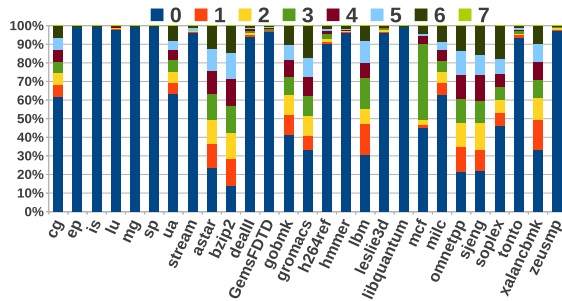


Figure 4: Distribution of Critical Words

be improved by tens of cycles if it is placed on a separate DIMM with lower device latencies and lower queuing delays. Since the other non-critical words in the cache line are not required urgently, they can be placed in low-power DRAM modules without incurring a performance penalty.

For such an optimization it is necessary that the critical word in a cache line remain fairly constant over a long period of time. The graphs in Figure 3 demonstrate such critical word regularity. We monitor DRAM accesses over a billion cycle period to identify the criticality of words in the cache. In Figure 3, we plot the percentage of accesses to different words, of the most accessed cache lines for two applications. The graphs show that for most cache lines, some words are more critical than others. For leslie3d (Figure 3.a), the most accessed critical word is word 0. On the other hand, in mcf (Figure 3.b), other words (i.e., 1 thru 7) are often the most frequently accessed critical words in the cache line. But clearly, in both cases, we see that within a cache line, there is a well-defined bias towards one or two words. We see the same pattern for all the simulated benchmarks.

A critical word first (CWF) optimization is facilitated if the critical word for a cache line remains unchanged over the execution of a program. If every cache line has a different critical word, more book-keeping would be required to track the identity of the critical word. The implementation is simplified if the same word is always the critical word for all cache lines and is placed in a low-latency DIMM.

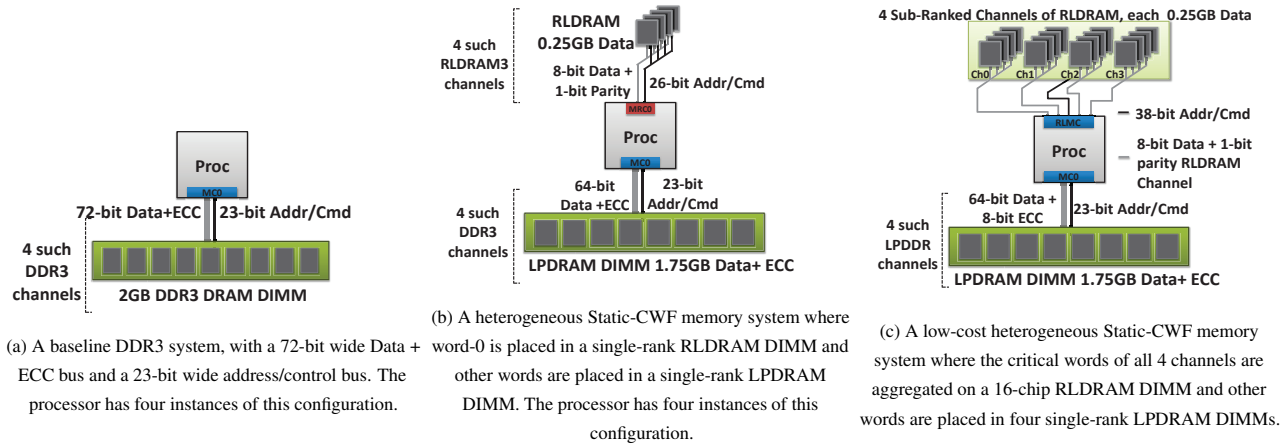
Figure 4 shows the distribution of critical words for our benchmark suite. For 21 of 27 programs, of the 8 words in a cache line, word-0 is critical in more than 50% of all cache line fetches. In 6 programs, there is no well-defined bias and all words have roughly an equal probability of being the critical word. The Appendix includes

a description explaining these biases for a variety of programs. In summary, if the program is performing a sequential scan across an array of data, early words in a cache line have a higher probability of being critical than later words in the cache line, especially if the stride is small and the word alignment is favorable. Pointer chasing codes tend to exhibit less bias.

**4.2.2. Data Placement.** On average, across the entire suite, word-0 is critical for 67% of all cache line fetches. To keep the design simple, we therefore assume a static architecture where word-0 is always placed in a low-latency DIMM and words 1-7 are placed in a low-power DIMM. We later also describe a more complex and flexible data organization.

In our baseline system (Figure 5a), an entire cache line plus SECDED ECC is placed on a single DDR3 DIMM and striped across 9 chips, where each chip is x8 wide. The proposed design for a single channel is shown in Figure 5b. The 9-chip 72-wide DDR3 rank is replaced by an 8-chip 64-wide LPDRAM rank. The 8 chips store words 1-7 and the ECC codes for each cache line. Word-0 for each cache line is now stored in a separate low-latency DIMM (rank) made of RLDRAM chips, and controlled by its own independent channel and memory controller. It is important for the RLDRAM DIMM to have its own independent controller. Since the RLDRAM DIMM has lower queuing delays, the request for word-0 from the RLDRAM DIMM can be issued many cycles before the request for words 1-7 from the LPDRAM DIMM gets issued. This allows the critical word to arrive tens of cycles before the rest of the cache line, having an impact much higher than other CWF optimizations that have been considered by processors [19, 18]. Since RLDRAM has lower density, we assume that 4 RLDRAM chips are required to support the same capacity as one DDR3 or LPDRAM chip. In order to remain processor-pin-neutral, the RLDRAM channel is assumed to only be 8 bits wide (plus a parity bit, explained shortly). We have thus replaced a single conventional 72-bit channel/DIMM with a 64-bit low-energy channel/DIMM and 9-bit low-latency channel/DIMM. The 9 DDR chips that made up a rank have now been replaced by 8 low-power LPDDR chips and 4 low-latency RLDRAM chips. The low-power and the low-latency channels require their own independent 23-bit address/command bus and independent memory controllers on the processor. We will shortly propose optimizations to reduce these overheads.

In our CWF design, whenever there is an LLC miss, an MSHR entry is created and two separate memory requests are created, one in the LPDRAM memory controller and one in the RLDRAM mem-



**Figure 5: A basic Static-CWF heterogeneous memory system and an optimized low-cost version of it.**

ory controller. Both channels are completely independent and will access their respective DIMMs and return data many cycles later. In the common case, word-0 arrives first and is buffered at the MSHR. If it is the critical word, it is returned to the CPU. When the second part of the cache line arrives, the access is deemed complete. The caches are then populated, and the MSHR entry is freed. The added complexity is the support for buffering two parts of the cache line in the MSHR and the additional memory controller for the RDRAM channel.

**4.2.3. Handling ECC Checks.** Memory systems are often required to provide ECC support that can detect and recover from a single bit error [38]. The ECC check can only happen after the entire cache line arrives. However, our CWF design wakes up a waiting instruction before confirming that the data is free of errors. Typically, the waiting instruction is at the head of the reorder buffer and will be retired soon after the critical word arrives and likely before the ECC code arrives. If the ECC-check flags an error, roll-back is often not possible because the instruction that consumed the critical word has already retired. We solve this problem as follows. A parity bit is attached to every byte stored in the RDRAM DIMM. RDRAM chips are already available in x18 and x36 configurations, i.e., they already are provisioned for additional bits that can be used for fault tolerance. When word-0 arrives at the processor, it is forwarded to the waiting instruction if there is no parity error. If there is a parity error, the data is forwarded only after the ECC code arrives and the error has been corrected. Parity checks can lead to silent data corruption if there are multi-bit errors. However, this is already beyond the scope of the baseline fault tolerance support which typically can only guarantee recovery from a single bit failure per 64-bit word. As in the baseline, our design will detect a 2-bit error on the 64-bit word when the ECC finally arrives. This general approach of lightweight error detection within RDRAM and full-fledged error correction support within LPDRAM can also be extended to handle other fault tolerance solutions such as chipkill [9]. To summarize, the parity solution does not replace the SECDED scheme; it augments SECDED. So the error coverage is exactly the same as in the baseline SECDED. The only deviation is that a multi-bit error (not detected by parity) will cause an erroneous result to commit; this will be detected by the regular SECDED a few cycles later when the full cache line is received. The new model will also fail-stop and the point of failure will be precisely known.

**4.2.4. Reducing Overheads.** We propose two optimizations to improve upon the basic design described so far. First, we recognize that energy overhead is increased by striping word-0 across 4 chips on the RDRAM DIMM. Instead, we place each word-0 and its parity bit on a single x9 RDRAM chip. Hence, the RDRAM DIMM is implemented as 4 ranks, each with one x9 chip. The address/command bus is now 26-bits wide to accommodate chip-select signals (see Figure 5b).

By having more narrow ranks, we not only reduce activation energy, we also increase rank and bank level parallelism and reduce queuing delays. While narrow ranks lead to a narrow row, that is not a problem because RDRAM employs a close-page policy and does not yield row buffer hits anyway.

The second optimization targets the overhead of having an extra 26-bit wide address/command bus and memory controller for each critical-word RDRAM channel. Our baseline has four 72-bit DDR3 channels. If our CWF proposal is applied to each channel, it produces four separate 9-bit RDRAM channels, each with its own memory controller and 26-bit address/command bus. To reduce these overheads, we aggregate the four RDRAM channels into a single memory controller and channel (see Figure 5c). The aggregated RDRAM channel will now have a 36-bit data bus and a 38-bit address/command bus (note the extra signals for chip-select). This organization is similar to Rank-Subsetting [8] where one address/cmd bus is used to control many skinny data channels. The only change from our previous scheme is that the four 9-bit wide channels will all share the same address/command bus. It is safe to share the address/command bus among four data channels without creating contention. This is because the transfer of word-0 from a RDRAM chip keeps the data bus busy for eight clock edges, but keeps the address bus busy for only two clock edges<sup>1</sup>. We can thus afford to increase the utilization of the address/command bus by a factor of four. The address/command bus now drives 16 chips on the RDRAM DIMM, an already common scenario in modern channels.

With these optimizations in place, some of the initial overheads are reduced by a factor of four. Now, every access activates the same number of DRAM chips (9) as the baseline, keeping activation

<sup>1</sup>RDRAM uses a close-page policy and does not require an explicit precharge on the address/cmd bus after every data access. Hence, the utilization of the data and address/cmd bus is in the ratio 4:1. We assume double-data-rate for the address/command bus, as is done for LPDRAM [29], and GDDR5 [5]

energy in check. We are introducing only one additional memory controller and only 42 (38 for the address/cmd bus and 4 for parity) new pins to control the RLDRAM channel. This is a small overhead given that modern processors have 1000+ pins.

**4.2.5. Adaptive Placement.** Design complexity is an important guiding metric in creating our static CWF architecture. We therefore assume that word-0 is always critical and place it in the RLDRAM DIMM/channel. This makes it easy to locate and assemble the two parts of a cache line. As a potential study, we also consider a more complex organization where every cache line is allowed to designate one of its eight words as the critical word. We assume that the critical word on the previous fetch will also be the critical word on the next fetch. Hence, when a dirty line is evicted from cache, the line is re-organized and placed in DRAM memory such that the predicted critical word is placed in the low-latency DIMM. With such prediction, 79% of all critical words will be found in the low-latency DIMM, compared to the 67% success rate of the static CWF architecture. The critical word would need a 3-bit tag in cache and DRAM to identify itself as one of the eight words in the cache line. We present results for this model to showcase the additional room for improvement with an adaptive CWF optimization, but acknowledge the relatively high design complexity required to realize this added improvement.

**4.2.6. Discussion.** To summarize, our innovations allow us to enhance system performance and lower the memory system energy. To achieve this, we add a memory controller, one extra address/command bus and four extra data pins. On the processor, some support from the MSHR is required to handle the fragmented transfer of a cache line. The total memory capacity and DRAM error coverage is the same as in the baseline. The use of RLDRAM will lead to a higher purchase cost.

## 5. Methodology

Processor	
ISA	UltraSPARC III ISA
CMP size and Core Freq.	8-core, 3.2 GHz
Re-Order-Buffer	64 entry
Fetch, Dispatch, Execute, and Retire	Maximum 4 per cycle
Cache Hierarchy	
L1 I-cache	32KB/2-way, private, 1-cycle
L1 D-cache	32KB/2-way, private, 1-cycle
L2 Cache	4MB/64B/8-way, shared, 10-cycle
Coherence Protocol	Snooping MESI
DRAM Parameters	
DDR3	MT41J256M4 DDR3-1600 [28],
RLDRAM3	Micron MT44K32M18 [30]
LPDDR-2	Micron MT42L128M16D1 [29] (400MHz)
Baseline DRAM Configuration	4 72-bit Channels 1 DIMM/Channel (unbuffered, ECC) 1 Rank/DIMM, 9 devices/Rank
Total DRAM Capacity	8 GB
DRAM Bus Frequency	800MHz
DRAM Read Queue	48 entries per channel
DRAM Write Queue Size	48 entries per channel
High/Low Watermarks	32/16

Table 1: Simulator parameters.

**Simulator Details.** We use the Wind River Simics [25, 4] simulation platform for our study. Table 1 details the salient features of the simulated processor and memory hierarchy. We model an

Parameter	DRAM	RLDRAM3	LPDDR2
tRC	50ns	12ns	60ns
tRCD	13.5ns	-	18ns
tRL	13.5ns	10ns	18ns
tRP	13.5ns	-	18ns
tRAS	37ns	-	42ns
tRTRS	2 Bus Cycles	2 Bus Cycles	2 Bus Cycles
tFAW	40 ns	-	50ns
tWTR	7.5ns	0	7.5ns
tWL	6.5ns	11.25ns	6.5ns

Table 2: Timing Parameters [30, 29, 28, 23]

out-of-order processor using Simics’ *ooo-micro-arch* module and use a heavily modified *trans-staller* module for the DRAM simulator. The DRAM simulator is adapted from the USIMM simulation framework [12]. We also model a stride prefetcher. At the memory controller, demand requests are prioritized over prefetch requests in general, unless the prefetch requests exceed some age threshold, at which point they are promoted over demand requests.

**DRAM Simulator.** The memory controller models a First-Ready-First-Come-First-Served (FR-FCFS) scheduling policy (for DDR3 and LPDRAM2) and models all DRAM, RLDRAM3 and LPDDR2 commands and timing parameters. We describe the most pertinent ones in Table 2. The DRAM device model and timing parameters were derived from Micron datasheets [28, 19, 29, 30]. DRAM address mapping parameters for our platform (i.e., number of rows/columns/banks) were adopted from Micron data sheets [28, 29, 30]. The open row address mapping policy from [19] is used in the baseline and LPDDR2. We use this address mapping scheme because this results in the best performing baseline on average when compared to other commonly used address interleaving schemes [44, 19]. For the RLDRAM modules, we use a close-page policy. Note that the queuing delays that we see in the baseline in our simulations (Figure 1.b) are lower than those observed in prior work such as [41] because we simulate more channels and larger last level cache capacities which reduce the traffic in each channel.

**Workloads.** Our techniques are evaluated with full system simulation of a wide array of memory-intensive benchmarks. We use multi-threaded workloads (each core running 1 thread) from the OpenMP NAS Parallel Benchmark [11] (cg, is, ep, lu, mg, sp) suite and the STREAM [3] benchmark. We also run multiprogrammed workloads from the SPEC CPU 2006 suite (astar, bzip2, dealII, gromacs, gobmk, hmmmer, h264ref, lbm, leslie3d, libquantum, mcf, milc, omnetpp, soplex, sjeng, tonto, xalancbmk and zeusmp). Each of these single threaded workloads are run on a single core - so essentially each workload is comprised of 8 copies of the benchmark running on 8 cores. For multi-threaded applications, we start simulations at the beginning of the parallel-region/region-of-interest of the application. For the multiprogrammed SPEC benchmarks, we fast forward the simulation by 2 billion instructions on each core before taking measurements. We run the simulations for a total of 2 million DRAM read accesses after warming up each core for 5 million cycles. Two million DRAM read accesses correspond to roughly 540 million program instructions on average. For comparing the effectiveness of the proposed schemes, we use the total system throughput defined as  $\sum_i (IPC_{shared}^i / IPC_{alone}^i)$  where  $IPC_{shared}^i$  is the IPC of program  $i$  in a multi-core setting.  $IPC_{alone}^i$  is the IPC of program  $i$  on a stand-alone single-core system with the same memory system.

**Power Modeling.** We use the Micron DRAM power calculators [2] to model the power consumption of each DRAM chip. While the

DDR3 and RDRAM3 calculators can directly be used to model the power of each of these variants of DRAM chips, we modify the DDR3 power calculator to serve as the power model for LPDDR2. We change the current values from those of DDR3 to the corresponding LPDDR2 values for all except the background currents (Idd3P and Idd3PS). This is to ensure that we do not artificially inflate the LPDDR2 power savings. As mentioned in Section 4.1 we assume that an LPDDR2 chip used on a DIMM will have a DLL. This DLL consumes power in idle modes. To account for this component of the idle power, we assume that an LPDDR2 chip consumes the same amount of current that a DDR3 chip does in idle state. We also assume that LPDRAM requires ODT and calculate the static power contribution of such termination resistors.

## 6. Evaluation

### 6.1. Critical Word Optimization

**6.1.1. Performance Analysis.** We evaluate the proposed critical word optimizations for three memory configurations. In the first two of these, an eighth of the DRAM capacity (meant for storing critical words) is constructed out of RDRAM3 while the rest is built with DDR3 and LPDDR2 respectively. In the third configuration, DDR3 is used to store the critical word and the rest of the word is stored in LPDDR2. The three configurations are as follows.

- **RD** : 1GB RDRAM3 and 7GB DDR3 (plus 1GB ECC)
- **RL** : 1GB RDRAM3 and 7GB LPDDR2 (plus 1GB ECC)
- **DL** : 1GB DDR3 and 7GB LPDDR2 (plus 1GB ECC)

In Figure 6, we present the throughput of each of these systems normalized to an 8GB DDR3 baseline, while Figure 7 shows the average DRAM latency of the requested critical word. Each system represents a different point on the power-performance space, depending on the type of DRAM chip in use. We see in Figure 6 that RD has an average throughput improvement of 21% over DDR3 while RL has an average improvement of 12.9% over the baseline. The source of these improvements are the 30% and 22% reductions in critical word latency for RD and RL respectively. The overall performance degradation with the power optimized DL scheme is 9%.

Maximum improvements with the RDRAM based configurations are observed for those benchmarks that have a large fraction of critical words placed in RDRAM. Figure 8 shows the fraction of total critical word requests served by the faster RDRAM module. Applications like *cg*, *lu*, *mg*, *sp*, *GemsFDTD*, *leslie3d*, and *libquantum* show significant improvements with RL and RD because most of the requested critical words for these benchmarks are to word 0 and hence are serviced with RDRAM3 latency. In fact, for these benchmarks, RD and RL perform very similarly because the latency of the slower DRAM module rarely affects the system performance. For the same reason, DL is able to hide the latency of LPDDR2 to a large extent in these cases and suffers minimal performance degradation.

On the other hand, applications like *lbm*, *mcf*, *milc*, and *omnetpp* have a large fraction of critical words that are not the first word in the cache line and are hence serviced from the slower DRAM module (Figure 8). For these applications, RL performs nominally better than the baseline. In fact, with RL, *bzip2* performs about 4% worse than the baseline because of the increased critical word latency.

To further understand the performance gains from the RL scheme, we collected statistics on the time gap between the first request for a cache-line and the subsequent access to the same cache-line, but to a different word. This is to determine if accesses to words 1 through

7 in the cache-line occur sufficiently late in the instruction window to tolerate the increased LPDDR latency. We compare the average gap between the first two accesses to a cache-line to the average LPDRAM access latency for all the benchmarks which have word-0 as the most frequently accessed critical word. We see that in the majority of the applications that benefit from our proposed architecture, this gap is greater than or very close to the LPDRAM latency for more than 82% of all accesses. This means that there is very little additional slowdown compared to an all DDR baseline for these applications. However, some applications such as *tonto* and *dealII*, which see reduced critical word latencies with both RD and RL, due to the large percentage of critical word 0 accesses, still experience marginal overall slowdown compared to the baseline. This is because in these applications, most of the second accesses to cache-lines (more than 75%) occur before the whole line is returned from the LPDDR.

All the results above are with a system that employs a stream prefetcher. In the absence of the prefetcher the performance gain with the RL system is 17.3%, simply because there is more opportunity for latency hiding with CWF.

We also perform an experiment where the critical words are assumed to be randomly mapped between the RDRAM and LPDRAM systems (with the critical word being 7 times more likely to be found in the LPDRAM system compared to the RDRAM system). This experiment is done to ensure that the intelligent data mapping contributes to the performance gains observed above. We see that with a random mapping, the average performance improvement is only 2.1% for RL (compared to the DDR3 baseline) with many applications showing severe performance degradation because a large percentage of their critical words are fetched from LPDRAM.

**6.1.2. Analyzing RDRAM+LPDDR performance.** Since the RL scheme has comparable performance to the RD scheme and (as shown later) has significant energy advantages over RD, we choose it as our flagship configuration. We apply the adaptive critical word placement on it and present the results in the following graph, Figure 9.

The bar **RL OR** in Figure 9 represents system throughput with an oracular scheme, where we assume that every critical word is fetched from RDRAM3. The performance obtained with RL-OR is the highest (28% improvement) that can be achieved by the RL configuration. Note that the performance thus obtained will be lower than an all-RDRAM3 system represented by the last bar, RDRAM3, in Figure 9. Note that in RL OR, only the critical word comes from the RDRAM3 part, compared to the whole cache line in RDRAM3. The other factor restricting the performance of the RL OR system is the limited address/command bus bandwidth. Even though the RDRAM chips are organized into 4 independent data channels, the single address and command bus can become a bottleneck, especially for applications with high memory pressure such as *mcf*, *milc*, *lbm*.

The RL AD bar represents the performance of the adaptive critical word placement scheme. The RL AD scheme re-organizes the layout of the cache word to place the last used critical word in the RDRAM3 part. This scheme performs better (15.7% improvement over baseline) than the RL scheme (12.9% improvement over the baseline) because it lowers the critical word latency of more requests compared to the RL scheme. *Mcf* is one of the applications that benefits from adaptive placement (a 20% improvement over baseline compared to 12% for RL). This is because in *mcf*, word 0 and word



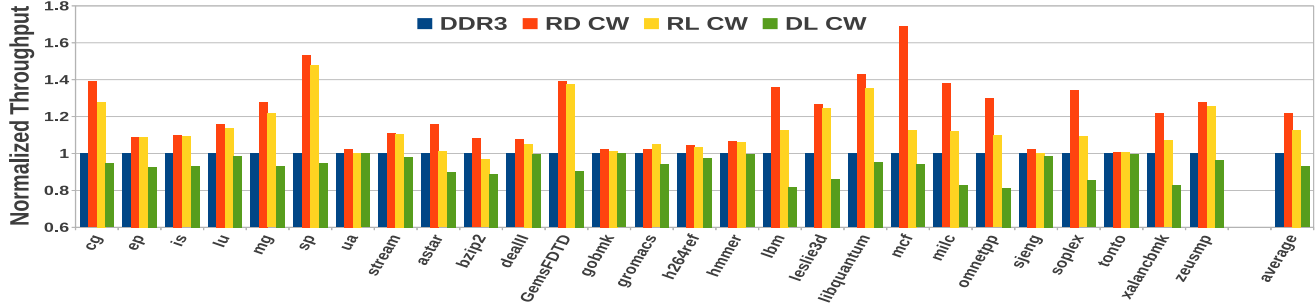


Figure 6: Performance of CW optimization

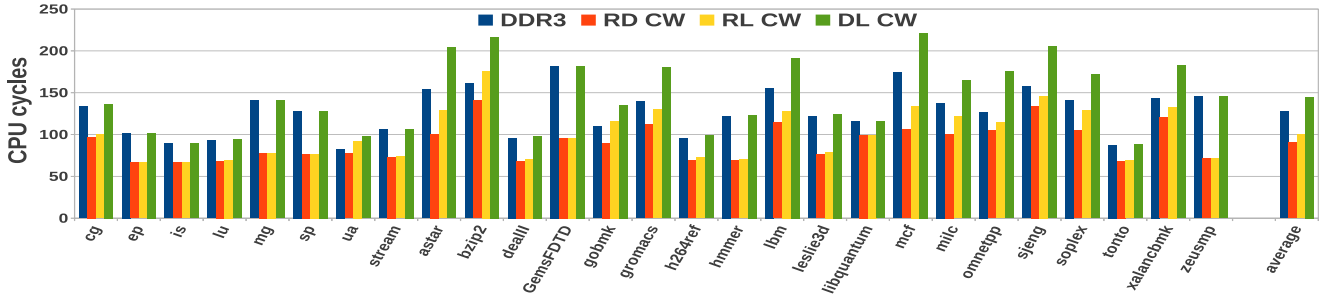


Figure 7: Critical Word Latency

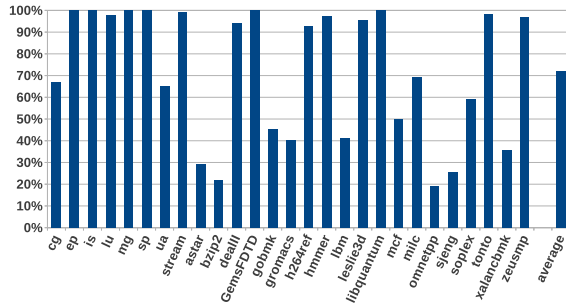


Figure 8: Percentage of Critical Word Accesses Served By RLDram3

3 are the most accessed critical words (see Figure 4). With adaptive placement, during a write to the lines which have word 3 as the critical word, the layout of the cache line is altered and subsequent reads to these lines have lower critical word latencies. The performance of the RL AD scheme is dependent on the write-traffic - i.e., unless a word is written to, its organization in the main memory is not altered. Therefore, not all applications that have words 1 through 7 as critical words can be accelerated with the RL AD scheme.

**6.1.3. Energy Analysis.** In this section, we analyze the energy consumption of the ensemble. The system energy consumption for the different configurations is shown in Figure 10.

**Methodology:** To calculate system energy, we assume the power consumption of the DRAM system in the baseline to be 25% of the entire system [27, 22]. We assume that one-third of the CPU power is constant (leakage + clock), while the rest scales linearly with CPU activity. We calculate the power consumption of DDR3, RLDram3 and LPDDR2 chips using activity factors from the simulator, which are then fed to the Micron DRAM power calculators [2].

As seen in Figure 2, the power consumption of a single RLDram3 chip is higher than that of a DDR3 chip at the bus utilization values that we see in our simulations (between 5% and 40%). In an RL system, we have 16 RLDram3 chips and 32 LPDDR2

chips compared to 36 DDR3 chips in the baseline. In the baseline, every cache line request requires the activation of 9 DDR3 chips, while in RL, 1 RLDram3 chip and 8 LPDDR2 chips are activated for each request. The high power consumption of RLDram3 is alleviated by the lower power consumption of the LPDDR2 chips. In our experiments we find that by only fetching one cache-word from the RLDram3 chip, its active power consumption is kept in check. Also, since each RLDram3 chip sees one-fourth of the activity seen by each DDR3 chip (because of the sub-ranking optimization), the I/O power of the RLDram3 chips is kept lower. The LPDDR2 channels, due to their lower frequency and lower operating voltage consume about 20% less power on average compared to the DDR3 system. The faster power-down entry and exit modes allow the LPDDR2 ranks to be put to sleep more frequently compared to the baseline. The overall memory power decreases by 1.9% and memory energy reduces by 15%.

We see that the overall system energy consumption drops by about 6% with RL while the DL scheme consumes about 13% lower energy compared to a DDR3 baseline. The RL scheme exhibits maximum energy savings for high-bandwidth applications with high critical word 0 accesses - e.g., mg, sp, GemsFDTD, leslie3d, and libquantum. High DRAM utilization bridges the gap between DDR3's power consumption and that of RLDram3 (Figure 2) to a certain extent. When such high-bandwidth applications can benefit from the critical word optimization, i.e., have a high number of critical word 0 accesses, the overall system energy is reduced. On the other hand, high bandwidth applications such as lbm, mcf, and milc, which show modest performance improvements with our schemes do not show a marked reduction in system energy consumption.

Applications like bzip2, deall and gobmk, have low bandwidth requirements. As a result, the power consumption of the RLDram3 component is high enough that the DRAM power in the RL configuration exceeds the baseline DDR3 power consumption. Coupled with this are the marginal execution time reductions (increase in bzip2) for these applications with RL, resulting in overall system

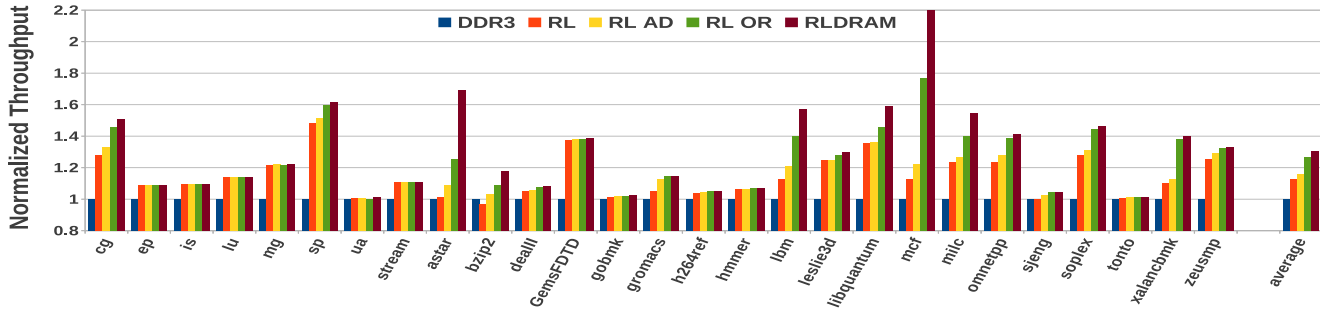


Figure 9: Performance of various RLD3+LPDDR2 configurations

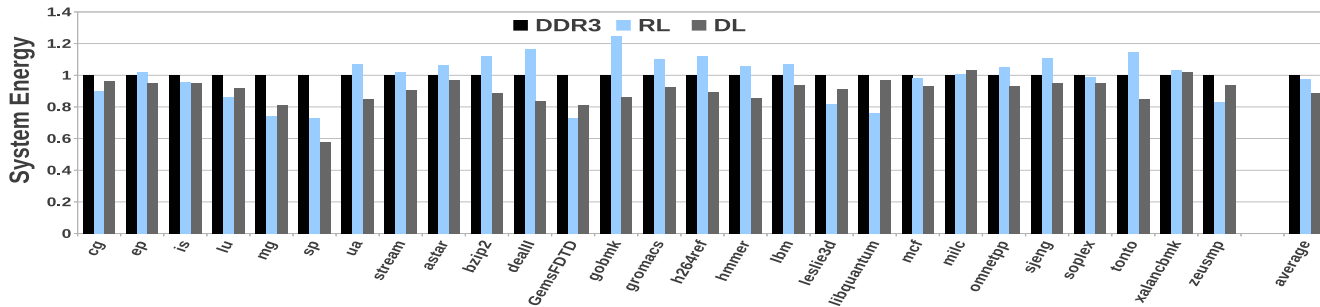


Figure 10: System Energy Normalized to DDR3 Baseline

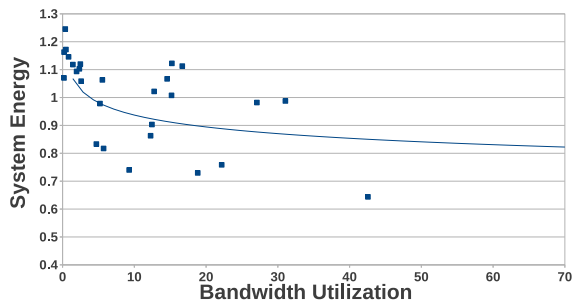


Figure 11: Bandwidth Utilization vs Energy Savings

energy consumption increases.

Figure 11 shows the energy savings obtained for the RL scheme as a function of bandwidth utilization (each point in the figure represents a different workload). As explained above, the energy savings are a function of two parameters - the critical word regularity and the bandwidth utilization. We notice in Figure 11 that in general, with increasing bandwidth utilization, the system energy savings are progressively greater. This is because the energy gap between RLD3 and DDR3 shrinks at high utilization. This implies that in future systems constrained by bandwidth, our innovations can be expected to provide greater energy savings and performance improvements.

## 7. Alternate Heterogeneous Memory Design Options

### 7.1. Comparison to Page Placement Proposals

Most heterogeneous main memory proposals, including that of Phadke et al., place a group of cache-lines (usually belonging to the same physical page) in a single DRAM variant on a single channel. To compare the merits of such a strategy against our proposal we evaluate a heterogeneous memory system that consists of RLD3 and LPDDR2. An application is statically profiled and a fraction of its pages are placed in RLD3 and the rest in LP-

DRAM2. In this experiment, to ensure that the memory configuration is practical, we assume that the system consists of four 72 bit wide channels. Three contain 2GB LPDDR2 DIMMs each while the fourth channel has only .5GB of RLD3 memory. This design allows the RLD3 system to enjoy the pin bandwidth of a conventional system and curbs the energy consumption of the RLD3 system by restricting the number of chips. Hence the baseline configuration and the one described above are iso-pin-count and iso-chip-count. We do not consider the performance effects of the reduced RLD3 capacity. The applications are profiled offline and the top 7.6% (.5GB/6.5GB) of the accessed pages were placed in RLD3. We see that the performance varies widely, from a 9.3% loss to a 11.2% improvement, yielding an average improvement of about 8%. Those applications, most of whose working set fits in the RLD3 memory, benefit from this organization showing higher benefits compared to our scheme. However, the loss in performance is due to many accesses to the LPDRAM - since the top 7.6% of pages only account for a maximum of 30% of all accesses for any of the programs. In addition, [39] demonstrates that for most applications, few cache-lines in a page make up the majority of all accesses to that page. Thus, allocating space on RLD3 at a page granularity does not yield much benefit. It is also necessary to note that the improvements reported here will likely be less once page-fault effects due to the smaller overall capacity of the memory system are taken into consideration (our simulations are not long enough to consume even the reduced memory capacity). The power consumption of this system is decidedly better than our scheme, because

- there is less RLD3 in the system compared to the proposed scheme, thus less background power.
- pages in LPDRAM have low access frequencies, thus the LPDRAM can stay in power down modes for longer, thereby reducing power consumption further.

### 7.2. Alternate LPDRAM Design

As noted in Section 9, Malladi et al. have looked at adapting LPDRAM for use in server memories [26]. Their analog simulations

show that even in the absence of On-Die-Termination, the "eye" of the read and write signals is wide enough to allow using unmodified LPDRAM chips (grouped into packages of four) to construct the memory channel. We simulate a similar system in our experiments and find that the LPDRAM power is further reduced in this case with very little loss in performance (due to deeper sleep modes) and thus the energy savings are boosted to 26.1%.

## 8. Cost Analysis

The total cost of ownership of a server system is an important design metric. The acquisition cost of a product is intrinsically tied to the volume of production. LPDRAM is already being used in high-volume markets (portable compute devices) which will help reduce its price. The higher cost-per-bit of the RLD3 devices is kept in check in our design by using RLD3 for only 1/8th of the total memory space. The CapEx involved will reduce if a case can be made for large-scale adoption of such devices. In the recent future, we expect non-volatile memories to relieve the DRAM system of its high density requirements. It can therefore be expected that DRAM products, their target markets, and their volumes will evolve accordingly in the future. Energy-optimized memory systems, like the one we propose in this work, will drive down the OpEx by reducing overall system-energy. We expect this to be a big motivation for the adoption of such a design in the data-center space and this will likely help drive up the volume of these specialty parts.

## 9. Related Work

**Critical Word Regularity :** Regularity in the critical word accesses was reported by Gieske et al. [17]. In this work the repeatable patterns in critical words to optimize the L2 cache's power and performance characteristics via a "critical-word-cache".

**Heterogeneous Memory :** A recent paper [35] proposes one possible application of heterogeneous DRAM based memories. The authors propose an offline analysis of the LLC miss-rate and average wait-time of an instruction at the head of the queue to determine if an application is latency or bandwidth sensitive. Section 3 already provides a qualitative comparison. Quantitatively, we observed that for a similar pin count and chip count as the baseline, this approach only yields an 8% improvement.

Dong et al. [15] discuss the possibility of having a two-tiered DRAM-based hierarchy where an on-chip dram is used in conjunction with traditional off-chip DDR DIMMs. However, in contrast to convention [24, 47, 20], the on-chip dram is not used as a large LLC, but instead considered a part of the main memory. The authors describe hardware accounting techniques to identify candidate pages to migrate from the on-chip dram to the main DRAM.

**Memory design with LPDRAM :** Very recently, two techniques to construct server memory entirely out of mobile DRAM parts have been proposed [45, 26]. In these papers, the authors aim to exploit the low background-power characteristics of LPDRAM and focus on solutions to overcome the lower bandwidth, lower channel capacity and ECC implementation problems of memory systems built with LPDDR. Our proposals are complementary to their designs. Specifically, we demonstrate that the addition of RLD3 to such a LPDDR based design can provide significant performance benefits for most applications.

**Hybrid DRAM+NVM Memory :** Several papers have suggested integrating Phase Change Memory (PCM) technology in the main memory hierarchy [23, 48, 36, 46]. Considering the longer read

and write latencies of PCM, it seems likely that DRAM cannot be completely substituted by PCM - rather, a combination of the two technologies would be the better choice. Consequently, the management of such a hybrid memory space has been studied by some recent papers. Proposals for managing such a hybrid system consist of placement of performance critical and frequently written pages in DRAM and the rest in PCM [37], write-frequency counter guided page placement to manage wear-levelling [14] and migration of data pages between PCM and DRAM based on access counts to mitigate background power consumption [33].

## 10. Conclusions

We show that modern memory systems can be enhanced by promoting heterogeneity in their construction. Existing production DRAM parts can be utilized to construct an efficient heterogeneous main memory system with low complexity with changes to the memory controller and channel organization. We also identify a novel pattern in the repeatability of critical word accesses at the DRAM level for a variety of workloads and achieve 12.9% performance improvement and 6% system energy reduction by exploiting the critical word regularity with a heterogeneous memory. Trends indicate that our techniques can be used in future bandwidth-hungry systems to achieve higher performance and energy savings.

It is also likely that in the future, 3D-stacked DRAM technology will become mainstream. The recently announced Hybrid Memory Cube (HMC) from Micron [34] is one embodiment of such 3D technology. Each HMC device consists of 2D DRAM dies that are stacked on top of one another and connected via high-bandwidth TSVs to a logic layer at the bottom. The HMC is touted as a solution for the scaling problems with current DDR technology by virtue of its high capacity, high bandwidth and low power consumption. If we witness widespread use of HMCs in the future, then we can think of integrating some kind of heterogeneity in a HMC-based memory system. There are two ways to enable a critical-data-first architecture with HMCs. In one possible variant, one could include dies with different latency/energy properties and the critical data could be returned in an earlier high-priority packet. In another implementation, one could imagine having a mix of high-power, high-performance and low-power, low-frequency HMCs. The high-speed signalling employed in the baseline HMC makes it power-hungry. Thus a critical data bit could be obtained from a high-frequency HMC and the rest of the data from a low-power HMC.

## 11. Acknowledgments

We thank our reviewers (especially our shepherd Derek Chiou) and members of the Utah Arch group for their suggestions to improve this work.

## References

- [1] "Micron RLD3 Memory," [http://www.micron.com/~/media/Documents/Products/Product%20Flyer/rldram\\_flyer.pdf](http://www.micron.com/~/media/Documents/Products/Product%20Flyer/rldram_flyer.pdf).
- [2] "Micron System Power Calculator," <http://goo.gl/4dzK6>.
- [3] "STREAM - Sustainable Memory Bandwidth in High Performance Computers," <http://www.cs.virginia.edu/stream/>.
- [4] "Wind River Simics Full System Simulator," <http://www.windriver.com/products/simics/>.
- [5] "Quimonda GDDR5- White Paper," <http://www.hwstation.net>, 2007.
- [6] "All You Need to Know About Mobile LPDDR," [http://download.micron.com/pdf/flyers/mobile\\_lpddr\\_flyer.pdf](http://download.micron.com/pdf/flyers/mobile_lpddr_flyer.pdf), 2008.

- [7] “RLDRAM3 Press Release,” <http://www.issi.com>, 2011.
- [8] J. Ahn *et al.*, “Future Scaling of Processor-Memory Interfaces,” in *Proceedings of SC*, 2009.
- [9] AMD Inc., “BIOS and Kernel Developer’s Guide for AMD NPT Family 0Fh Processors.”
- [10] O. Azizi, A. Mahesri, B. Lee, S. Patel, and M. Horowitz, “Energy-Performance Tradeoffs in Processor Architecture and Circuit Design: A Marginal Cost Analysis,” in *Proceedings of ISCA*, 2010.
- [11] D. Bailey *et al.*, “The NAS Parallel Benchmarks,” *International Journal of Supercomputer Applications*, vol. 5, no. 3, pp. 63–73, Fall 1994.
- [12] N. Chatterjee, R. Balasubramonian, M. Shevgoor, S. Pugsley, A. Udipi, A. Shafiee, K. Sudan, M. Awasthi, and Z. Chishti, “USIMM: the Utah Simulated Memory Module,” University of Utah, Tech. Rep., 2012, uUCS-002-12.
- [13] V. Cuppu, B. Jacob, B. Davis, and T. Mudge, “A Performance Comparison of Contemporary DRAM Architectures,” in *Proceedings of ISCA*, 1999.
- [14] G. Dhiman, R. Ayoub, and T. Rosing, “PDRAM: A Hybrid PRAM and DRAM Main Memory System,” in *Proceedings of DAC*, 2009.
- [15] X. Dong, Y. Xie, N. Muralimanohar, and N. Jouppi, “Simple but Effective Heterogeneous Main Memory with On-Chip Memory Controller Support,” in *Proceedings of SC*, 2010.
- [16] K. Ganesan, J. Jo, and L. K. John, “Synthesizing Memory-Level Parallelism Aware Miniature Clones for SPEC CPU2006 and ImplantBench Workloads,” in *Proceedings of ISPASS*, 2010.
- [17] E. J. Gieske, “Critical Words Cache Memory : Exploiting Criticality Withing Primary Cache Miss Streams,” Ph.D. dissertation, 2008.
- [18] J. L. Hennessy and D. A. Patterson, *Computer Architecture: A Quantitative Approach*, 4th ed. Elsevier, 2007.
- [19] B. Jacob, S. W. Ng, and D. T. Wang, *Memory Systems - Cache, DRAM, Disk*. Elsevier, 2008.
- [20] X. Jiang, N. Madan, L. Zhao, M. Upton, R. Iyer, S. Makineni, D. Newell, Y. Solihin, and R. Balasubramonian, “CHOP: Integrating DRAM Caches for CMP Server Platforms,” *IEEE Micro (Top Picks)*, Jan/Feb 2011.
- [21] C. Kozyrakis, A. Kansal, S. Sankar, and K. Vaid, “Server Engineering Insights For Large-Scale Online Services,” in *IEEE Micro*, 2010.
- [22] J. Laudon, “UltraSPARC T1: A 32-Threaded CMP for Servers,” 2006, invited talk, URL: <http://www.cs.duke.edu>.
- [23] B. Lee, E. Ipek, O. Mutlu, and D. Burger, “Architecting Phase Change Memory as a Scalable DRAM Alternative,” in *Proceedings of ISCA*, 2009.
- [24] G. Loh and M. Hill, “Efficiently Enabling Conventional Block Sizes for Very Large Die-stacked DRAM Caches,” in *In Proceedings of MICRO*, 2011.
- [25] P. Magnusson, M. Christensson, J. Eskilson, D. Forsgren, G. Hallberg, J. Hogberg, F. Larsson, A. Moestedt, and B. Werner, “Simics: A Full System Simulation Platform,” *IEEE Computer*, vol. 35(2), pp. 50–58, February 2002.
- [26] K. T. Malladi, F. A. Nothaft, K. Periyathambi, B. C. Lee, C. Kozyrakis, and M. Horowitz, “Towards Energy-Proportional Datacenter Memory with Mobile DRAM,” in *Proceedings of ISCA*, 2012.
- [27] D. Meisner, B. Gold, and T. Wenisch, “PowerNap: Eliminating Server Idle Power,” in *Proceedings of ASPLOS*, 2009.
- [28] “Micron DDR3 SDRAM Part MT41J256M8,” Micron Technology Inc., 2006.
- [29] “Micron Mobile LPDDR2 Part MT42L128M16D1,” Micron Technology Inc., 2010.
- [30] “Micron RLDRAM 3 Part MT44K32M18,” Micron Technology Inc., 2011.
- [31] J. Ousterhout *et al.*, “The Case for RAMclouds: Scalable High-Performance Storage Entirely in DRAM,” *SIGOPS Operating Systems Review*, vol. 43(4), 2009.
- [32] J. P. Park, S. J. Rhee, S. B. Ko, Y. Jeong, K. S. Noh, Y. Son, J. Youn, Y. Chu, H. Cho, M. Kim, D. Yim, H. C. Kim, S. H. Jung, H. I. Choi, S. Yim, J. B. Kee, J. S. Choi, and K. Oh, “A 1.2V 30nm 1.6Gb/s/pin 4Gb LPDDR3 SDRAM with Input Skew Calibration and Enhanced Control Scheme,” in *Proceedings of ISSCC*, 2012.
- [33] Y. Park, D. Shin, S. Park, and K. Park, “Power-aware Memory Management For Hybrid Main Memory,” in *Proceedings of ICNIT*, 2011.
- [34] T. Pawlowski, “Hybrid Memory Cube (HMC),” in *HotChips*, 2011.
- [35] S. Phadke and S. Narayanasamy, “MLP-aware Heterogeneous Main Memory,” in *In Proceedings of DATE*, 2011.
- [36] M. Qureshi, V. Srinivasan, and J. Rivers, “Scalable High Performance Main Memory System Using Phase-Change Memory Technology,” in *Proceedings of ISCA*, 2009.
- [37] L. Ramos, E. Gorbato, and R. Bianchini, “Page Placement in Hybrid Memory Systems,” in *Proceedings of ICS*, 2011.
- [38] B. Schroeder *et al.*, “DRAM Errors in the Wild: A Large-Scale Field Study,” in *Proceedings of SIGMETRICS*, 2009.
- [39] K. Sudan, N. Chatterjee, D. Nellans, M. Awasthi, R. Balasubramonian, and A. Davis, “Micro-Pages: Increasing DRAM Efficiency with Locality-Aware Data Placement,” in *Proceedings of ASPLOS-XV*, 2010.
- [40] C. Toal, D. Burns, K. McLaughlin, S. Sezer, and S. O’Kane, “An rldram ii implementation of a 10gbps shared packet buffer for network processing,” in *Proceedings of the 2nd NASA/ESA Conference on Adaptive Hardware and Systems*, 2007.
- [41] A. N. Udipi, N. Muralimanohar, N. Chatterjee, R. Balasubramonian, A. Davis, and N. Jouppi, “Rethinking DRAM Design and Organization for Energy-Constrained Multi-Cores,” in *Proceedings of ISCA*, 2010.
- [42] VMWare Performance Team, “Ten Reasons Why Oracle Databases Run Best on VMware,” <http://blogs.vmware.com/performance/2007/11/ten-reasons-why.html>, 2007.
- [43] T. Vogelsang, “Understanding the Energy Consumption of Dynamic Random Access Memories,” in *Proceedings of MICRO*, 2010.
- [44] D. Wang *et al.*, “DRAMsim: A Memory-System Simulator,” in *SIGARCH Computer Architecture News*, September 2005.
- [45] D. H. Yoon, J. Chang, N. Muralimanohar, and P. Ranganathan, “BOOM: Enabling Mobile Memory Based Low-Power Server DIMMs,” in *Proceedings of ISCA*, 2012.
- [46] W. Zhang and T. Li, “Exploring Phase Change Memory and 3D Die-Stacking for Power/Thermal Friendly, Fast and Durable Memory Architectures,” in *Proceedings of PACT*, 2009.
- [47] Z. Zhang, Z. Zhu, and Z. Zhang, “Design and Optimization of Large Size and Low Overhead Off-chip Caches,” *IEEE Transactions on Computer*, July 2004.
- [48] P. Zhou, B. Zhao, J. Yang, and Y. Zhang, “A Durable and Energy Efficient Main Memory Using Phase Change Memory Technology,” in *Proceedings of ISCA*, 2009.

## A. Appendix: Explaining Critical Word Accesses in Programs

A profile of LLC filtered memory accesses of different applications shows that for almost all cache lines, one particular word in the cache line accounts for the vast majority of critical accesses to the cache line. The existence of such critical word regularity is not surprising, i.e., it is reasonable to expect that data in a region will be traversed in a similar order on multiple occasions. Such critical word regularity was reported and exploited by Gieske to optimize LLC performance [17].

Applications that stream through large data arrays will exhibit critical word accesses to words near the beginning of the cache line – most often to word 0 (at the DRAM level). Applications which have strided accesses with small stride lengths are most likely to have word 0 as the most critical word. For example, an application like *hmmr* has a dominant stride length of 0 for 90% of all accesses [16] – hence word 0 is the most popular critical word for *hmmr*. The *STREAM* benchmark has 4 parts, Copy, Scale, Sum and Triad. Each of these kernels uses unit strided accesses over large arrays, also results in word 0 being critical.

On the other hand, applications like *mcf* and *xalancbmk* spend most of their time in chasing pointers. For *xalancbmk*, 80% of misses are generated by two nested loops of pointer chasing, while in *mcf*, 70% of accesses are generated by random pointer chasing. These kinds of applications demonstrate a more uniform critical word distribution.