Copyright © 2007 IEEE. Reprinted from Proceedings of the 13th International Symposium on High Performance Computer Architecture (HPCA).

This material is posted here with permission of the IEEE. Such permission of the IEEE does not in any way imply IEEE endorsement of any of the University of Maryland's products or services. Internal or personal use of this material is permitted. However, permission to reprint/republish this material for advertising or promotional purposes or for creating new collective works for resale or redistribution must be obtained from the IEEE by writing to pubs-permissions@ieee.org.

By choosing to view this document, you agree to all provisions of the copyright laws protecting it.

Fully-Buffered DIMM Memory Architectures: Understanding Mechanisms, Overheads and Scaling

Brinda Ganesh†, Aamer Jaleel‡, David Wang†, and Bruce Jacob†

†University of Maryland, College Park, MD
‡VSSAD, Intel Corporation, Hudson, MA

{brinda, blj}@eng.umd.edu

Abstract

Performance gains in memory have traditionally been obtained by increasing memory bus widths and speeds. The diminishing returns of such techniques have led to the proposal of an alternate architecture, the Fully-Buffered DIMM. This new standard replaces the conventional memory bus with a narrow, high-speed interface between the memory controller and the DIMMs. This paper examines how traditional DDRx based memory controller policies for scheduling and row buffer management perform on a Fully-Buffered DIMM memory architecture. The split-bus architecture used by FBDIMM systems results in an average improvement of 7% in latency and 10% in bandwidth at higher utilizations. On the other hand, at lower utilizations, the increased cost of serialization resulted in a degradation in latency and bandwidth of 25% and 10% respectively. The split-bus architecture also makes the system performance sensitive to the ratio of read and write traffic in the workload. In larger configurations, we found that the FBDIMM system performance was more sensitive to usage of the FBDIMM links than to DRAM bank availability. In general, FBDIMM performance is similar to that of DDRx systems, and provides better performance characteristics at higher utilization, making it a relatively inexpensive mechanism for scaling capacity at higher bandwidth requirements. The mechanism is also largely insensitive to scheduling policies, provided certain ground rules are obeyed.

1. Introduction

The growing size of application working sets, the popularity of software-based multimedia and graphics workloads, and increased use of speculative techniques, have contributed to the rise in bandwidth and capacity requirements of computer memory sub-systems. Capacity needs have been met by building increasingly dense DRAM chips (the 2Gb DRAM chip is expected to hit the market in 2007) and bandwidth needs have been met by scaling front-side bus data rates and using wide, parallel buses. However, the traditional bus organization has reached a point where it fails to scale well into the future.

The electrical constraints of high-speed parallel buses complicate bus scaling in terms of loads, speeds, and widths [1]. Consequently the maximum number of DIMMs per channel in successive DRAM generations has been decreasing. SDRAM channels have supported up to 8 DIMMs of memory, some types of DDR channels support only 4 DIMMs, DDR2 channels have but 2 DIMMs, and DDR3 channels are expected to support only a single DIMM. In addition the serpentine routing required for electrical path-length matching of the data wires becomes challenging as bus widths increase. For the bus widths of today, the motherboard area occupied by a single channel is significant, complicating the task of adding capacity by increasing the number of channels. To address these scalability issues, an alternate DRAM technology, the Fully Buffered Dual Inline Memory Module (FBDIMM) [2] has been introduced.

The FBDIMM memory architecture replaces the shared parallel interface between the memory controller and DRAM chips with a point-to-point serial interface between the memory controller and an intermediate buffer, the Advanced Memory Buffer (AMB). The on-DIMM interface between the AMB and the DRAM modules is identical to that seen in DDR2 or DDR3 systems, which we shall refer to as DDRx systems. The serial interface is split into two uni-directional buses, one for read traffic (northbound channel) and another for write and command traffic (southbound channel), as shown in Fig 1. FBDIMMs adopts a packet-based protocol that bundles commands and data into frames that are transmitted on the channel and then converted to the DDRx protocol by the AMB.

The FBDIMMs' unique interface raises questions on whether existing memory controller policies will continue to be relevant to future memory architectures. In particular we ask the following questions:

- How do DDRx and FBDIMM systems compare with respect to latency and bandwidth?
- What is the most important factor that impacts the performance of FBDIMM systems?
- How do FBDIMM systems respond to choices made for parameters like row buffer management policy, scheduling policy and topology?

Our study indicates that an FBDIMM system provides significant capacity increases over a comparable DDRx system at relatively little cost. for systems with high bandwidth requirements. However, at lower utilization requirements, the FBDIMM protocol results in an average latency degradation of 25%. We found the extra cost of serialization is also apparent in

configurations with shallow channels i.e. 1-2 ranks per channel, but can be successfully hidden in systems with deeper channels by taking advantage of the available DIMM-level parallelism.

The sharing of write data and command bandwidth is a significant bottleneck in these systems. An average of 20-30% of the southbound channel bandwidth is wasted due to the inability of the memory controller to fully utilize the available command slots in a southbound frame. These factors together contribute to the queueing delays for a read transaction in the system.

More interestingly, we found that the general performance characteristics and trends seen for applications executing on the FBDIMM architecture were similar in many ways to that seen in DDRx systems. We found that many of the memory controller policies that were most effective in DDRx systems were also effective in an FBDIMM system.

This paper is organized as follows. Section 2 describes the FBDIMM memory architecture and protocol. Section 3 describes the related work. Section 4 describes the experimental framework and methodology. Section 5 has detailed results and analysis of the observed behavior. Section 6 summarizes the final conclusions of the paper.

2. Background

2.1 FBDIMM Overview

As DRAM technology has advanced, channel speed has improved at the expense of channel capacity; consequently channel capacity has dropped from 8 DIMMs to a single DIMM per channel. This is a significant limitation—for a server designer, it is critical, as servers typically depend on memory capacity for their performance. There is an obvious dilemma: future designers would like both increased channel capacity and increased data rates—but how can one provide both?

For every DRAM generation, graphics cards use the same DRAM technology as is found in commodity DIMMs, but operate their DRAMs at significantly higher speeds by avoiding multi-drop connections. The trend is clear: to improve bandwidth, one must reduce the number of impedance discontinuities on the bus.

This is achieved in FBDIMM system by replacing the multi-drop bus with point-to-point connections. Fig 1 shows the altered memory interface with a narrow, high-speed channel connecting the master memory controller to the DIMM-level memory controllers (called *Advanced Memory Buffers*, or *AMBs*). Since each DIMM-to-DIMM connection is a point-to-point connection, a channel becomes a *de facto* multi-hop store & forward network. The FBDIMM architecture limits the channel length to 8 DIMMs, and the narrower inter-module bus requires roughly one third as many pins as a traditional organization. As a result, an FBDIMM organization can handle roughly *24 times* the storage capacity of a single-DIMM DDR3-based system, without sacrificing any bandwidth and even leaving headroom for increased intra-module bandwidth.

The AMB acts like a pass-through switch, directly forwarding the requests it receives from the controller to successive DIMMs and forwarding frames from southerly DIMMs to northerly DIMMs or the memory controller. All frames are processed to determine whether the data and commands are for the local DIMM.

2.2 FBDIMM protocol

The FBDIMM system uses a serial packet-based protocol to communicate between the memory controller and the DIMMs. Frames may contain data and/or commands. Commands include DRAM commands such as row activate (RAS), column read (CAS), refresh (REF) and so on, as well as channel commands such as write to configuration registers, synchronization commands etc. Frame scheduling is performed exclusively by the memory controller. The AMB only converts the serial protocol to DDRx based commands without implementing any scheduling functionality.

The AMB is connected to the memory controller and/or adjacent DIMMs via serial uni-directional links: the southbound channel which transmits both data and commands and the northbound channel which transmits data and status information. The southbound and northbound data paths are respectively 10 bits and 14 bits wide respectively. The channel is dual-edge clocked i.e. data is transmitted on both clock edges. The FBDIMM channel clock operates at 6 times the speed of the DIMM clock i.e. the link speed is 4 Gbps for a 667 Mbps DDRx system. Frames on the north and south bound channel require 12 transfers (6 FBDIMM channel clock cycles) for transmission. This 6:1 ratio ensures that the FBDIMM frame rate matches the DRAM command clock rate.

Southbound frames comprise both data and commands and are 120 bits long; data-only northbound frames are 168 bits long. In addition to the data and command information, the frames also carry header information and a frame CRC checksum that is used to check for transmission errors.

The types of frames defined by the protocol [3] are illustrated in the figure 2. They are

Command Frame: comprises up to three commands (Fig 2(a)). Each command in the frame is sent to a separate

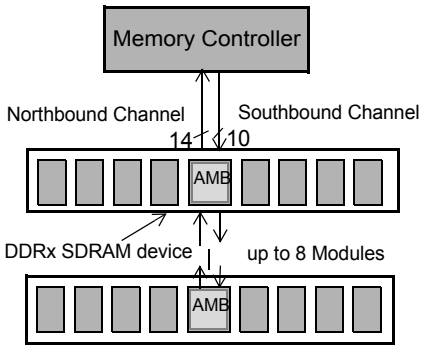


Figure 1. FBDIMM Memory System. TIn the FBDIMM organization, there are no multi-drop busses; DIMM-to-DIMM connections are point-to-point. The memory controller is connected to the nearest AMB, via two uni-directional links. The AMB is in turn connected to its southern neighbor via the same two links.

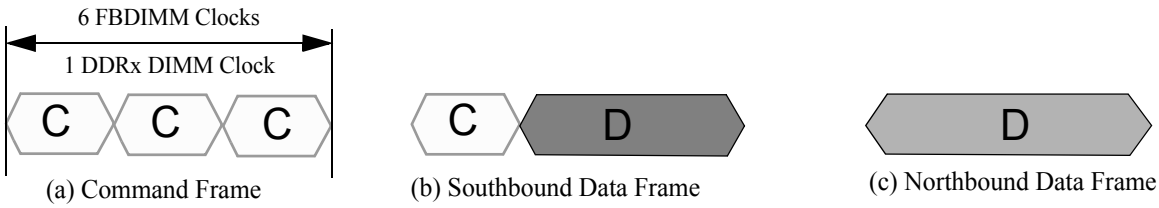


Figure 2. FBDIMM Frames. The figure illustrates the various types of FBDIMM frames used for data and command transmission.

DIMM on the southbound channel. In the absence of available commands to occupy a complete frame, no-ops are used to pad the frame.

Command + Write Data Frame: carries 72 bits of write data - 8 bytes of data and 1 byte of ECC - and one command on the southbound channel (Fig 2b). Multiple such frames are required to transmit an entire data block to the DIMM. The AMB of the addressed DIMM buffers the write data as required.

Data frame: comprises 144 bits of data or 16 bytes of data and 2 bytes of ECC information (Fig 2(c)). Each frame is filled by the data transferred from the DRAM in two beats or a single DIMM clock cycle. This is a northbound frame and is used to transfer read data.

A northbound data frame transports 18 bytes of data in 6 FBDIMM clocks or 1 DIMM clock. A DDRx system can burst back the same amount of data to the memory controller in two successive beats lasting an entire DRAM clock cycle. Thus, the read bandwidth of an FBDIMM system is the same as that of a single channel of DDRx system. A southbound frame on the other hand transports half the amount of data, 9 bytes as compared to 18 bytes, in the same duration. Thus, the FBDIMM southbound channel transports half the number of bytes that a DDRx channel can burst write in 1 DIMM clock, making the write bandwidth in FBDIMM systems one half that available in a DDRx system. This makes the total bandwidth available in an FBDIMM system 1.5 times that in a DDRx system.

The FBDIMM has two operational modes: the fixed latency mode and the variable latency mode. In fixed latency mode, the round-trip latency of frames is set to that of the furthest DIMM. Hence, systems with deeper channels have larger average latencies. In the variable latency mode, the round-trip latency from each DIMM is dependent on its distance from the memory controller.

Figure 3 shows the processing of a read transaction in an FBDIMM system. Initially a command frame is used to transmit a command that will perform row activation. The AMB translates the request and relays it to the DIMM. The memory controller schedules the CAS command in a following frame. The AMB relays the CAS command to the DRAM devices which burst the data back to the AMB. The AMB bundles two consecutive bursts of a data into a single northbound frame and transmits it to the memory controller. In this example, we assume a burst length of 4 corresponding to 2 FBDIMM data frames. Note that although the figures do not identify parameters like t_CAS, t_RCD and t_CWD [4, 5] the memory controller must ensure that these constraints are met.

Figure 4 shows the processing of a memory write transaction in an FBDIMM system. The write data requires twice the number of FBDIMM frames than the read data. All read and write data are buffered in the AMB before being relayed on, indicated by the staggered timing of data on the respective buses. The CAS command is transmitted in the same frame as the last chunk of data.

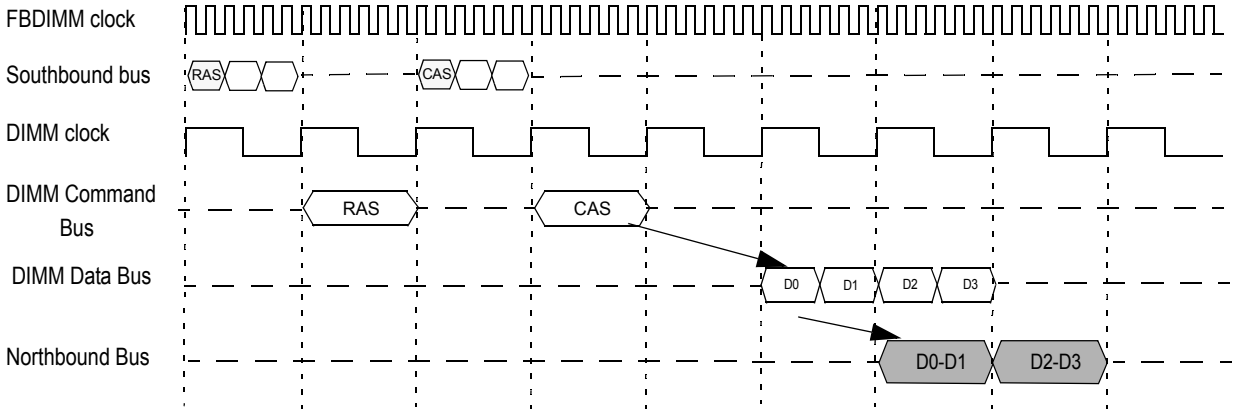


Figure 3. Read Transaction in FBDIMM system. The figure shows how a read transaction is performed in an FBDIMM system. The FBDIMM serial buses are clocked at 6 times the DIMM buses. Each FBDIMM frame on the southbound bus takes 6 FBDIMM clock periods to transmit. On the northbound bus a frame comprises of two DDRx data bursts.

3. Related Work

Burger et. al. [6] demonstrated that many high-performance mechanisms lower latency by increasing bandwidth demands. Cuppu et. al. [7] demonstrated that memory manufacturers have successfully scaled data-rates of DRAM chips by employing pipelining but have not reduced the latency of DRAM device operations. In their follow-on paper [8] they showed that system bus configuration choices significantly impact overall system performance.

There have been several studies of memory controller polices that examine how to lower latency while simultaneously improving bandwidth utilization. Rixner et. al. [9] studied how re-ordering accesses at the controller to increase row buffer hits can be used to lower latency and improve bandwidth for media processing streams. Rixner[10] studied how such re-ordering benefitted from the presence of SRAM caches on the DRAM for web servers. Natarajan et. al. [11] studied how memory controller policies for row buffer management and command scheduling impact latency and sustained bandwidth.

Lin et. al. [12] studied how memory controller based prefetching can lower the system latency. Zhu et. al. [13] studied how awareness of resource usage of threads in an SMT could be used to prioritize memory requests. Zhu et. al. [14] study how split-transaction support would lower the latency of individual operations.

Intelligent static address mapping techniques have been used by Lin et. al. [12] and Zhang et. al. [15] to lower memory latency by increasing row-buffer hits. The Impulse group at University of Utah [16] proposed adding an additional layer of address mapping in the memory controller to reduce memory latency by mapping non-adjacent data to the same cacheline and thus increasing cacheline sub-block usage.

This paper studies how various memory controller policies perform on the fully-buffered DIMM system. By doing a detailed analysis of the performance characteristics of DDRx specific scheduling policies and row buffer management policies on the FBDIMM memory system, the bottlenecks in these systems and their origin are identified. To the best of our knowledge, this is the first study to examine the FBDIMM in this detail.

4. Experimental Framework

This study uses DRAMsim [4], a stand-alone memory system simulator. DRAMsim provides a detailed execution-driven model of a fully-buffered DIMM memory system. The simulator also supports the variation of memory system parameters of interest including scheduling policies, memory configuration i.e. number of ranks and channels, address mapping policy etc.

We studied four different DRAM types: a conventional DDR2 system with a data-rate of 667Mbps, a DDR3 system with a data rate of 1333Mbps and their corresponding FBDIMM systems: an FBDIMM organization with DDR2 on the DIMM and another with DDR3 on the DIMM. FBDIMM modules are modelled using the parameters available for a 4GB module [3]; DDR2 device are modelled as 1Gb parts with 8 banks of memory [5]; DDR3 parameters were based on the proposed JEDEC standard[17]. A combination of multi-program and multi-threaded workloads are used. We used application memory traces as inputs.

- **SVM-RFE (Support Vector Machines Recursive Feature Elimination)** [18] is used to eliminate gene redundancy from a given input data set to provide compact gene subsets. This is a read intensive workload (reads comprise approximately 90% of the traffic). It is part of the BioParallel suite[24].
- **SPEC-MIXED** comprising of 16 independent SPEC [19] applications including AMMP, applu, art, gcc, lucas, mgrid, swim, vortex, wupwise, bzip2, galgel, gzip, mcf, parser, twolf and vpr. The combined workload has a read to write traffic ratio of approximately 2:1.
- **UA** is Unstructured Adaptive (UA) benchmark which is part of the NAS benchmark suite, NPB 3.1 OMP [20]. This workload has a read to write traffic ratio of approximately 3:2.
- **ART** the Open MP version of the SPEC 2000 ART benchmark which forms part of the SPEC OMP 2001 suite[21]. This has a read to write ratio of around 2:1.

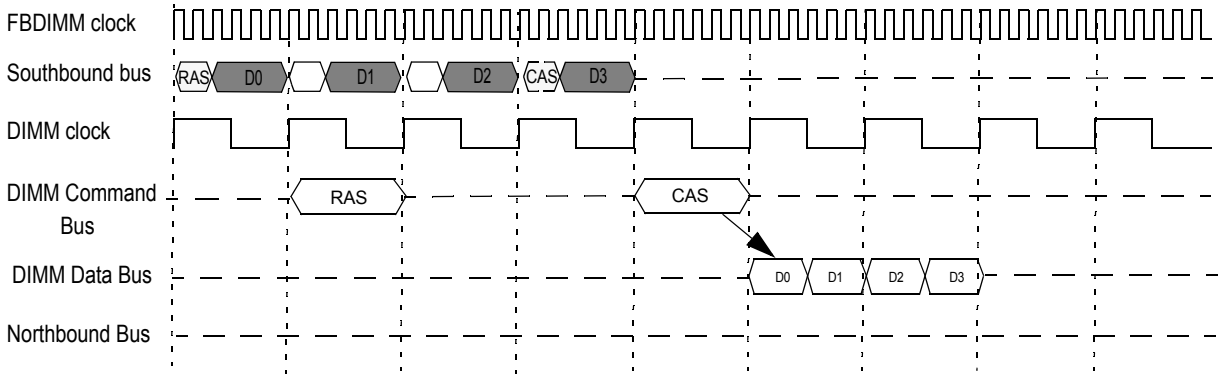


Figure 4. Write Transaction in FBDIMM system. The figure shows how a write transaction is performed in an FBDIMM system. The FBDIMM serial buses are clocked at 6 times the DIMM buses. Each FBDIMM frame on the southbound bus takes 6 FBDIMM clock periods to transmit. A 32 byte cacheline takes 8 frames to be transmitted to the DIMM.

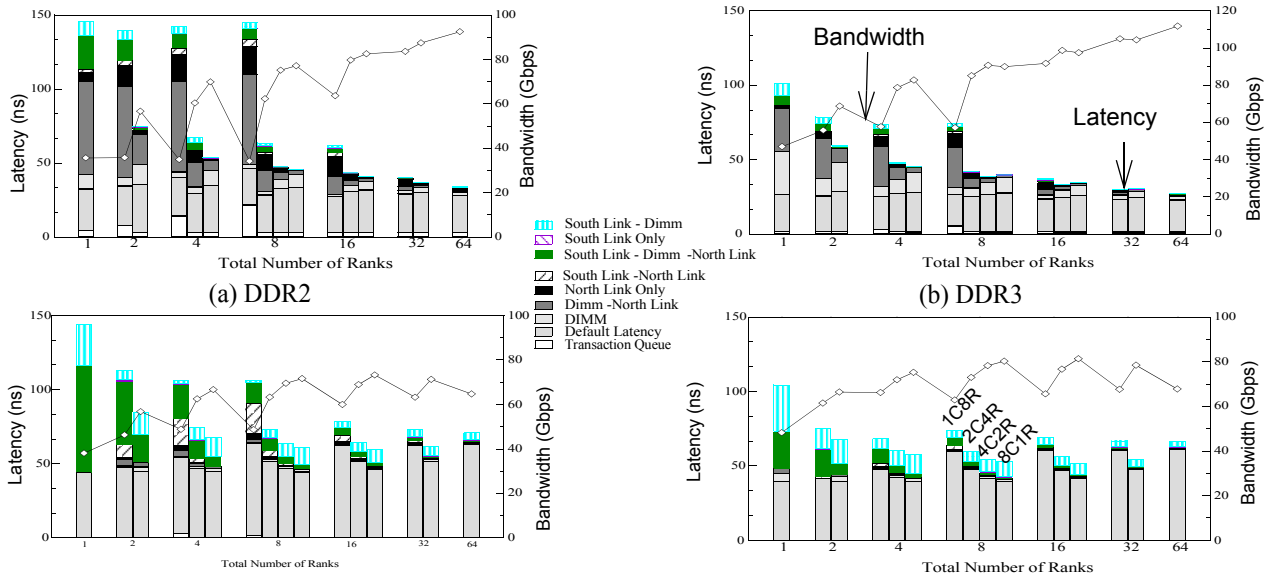


Figure 5. Latency and Bandwidth characteristics for different DRAM technologies. The figure shows the best latency and bandwidth characteristics for the SPEC-Mixed workload in an open page system. Systems with identical numbers of dimms are grouped together on the x-axis. Within a group, bars on the left represent topologies with lesser numbers of channels. The left y-axis depicts latency in nano-seconds, while the right y-axis plots bandwidth in Gbps. Note that the lines represent bandwidth and the bars represent latency.

All multi-threaded traces were gathered using CMP$im [22], a tool that models the cache hierarchy. Traces were gathered using 16-way CMP processor models with 8 MB of last-level cache. SPEC benchmark traces were gathered using Alpha 21264 simulator sim-alpha [23], with a 1 MB last-level cache and an in-order core.

Several configuration parameters that were varied in this study include:

Number of ranks and their configuration. FBDIMM systems support six channels of memory, each with up to eight DIMMs. This study expands that slightly, investigating a configuration space of 1-64 ranks, with ranks organized in 1-8 channels of 1-8 DIMMs each. Though many of these configurations are not relevant for DDRx systems, we study them the impact of serialization on performance.We study configurations with one rank per DIMM.

Row-buffer management policies . We study both open-page and closed-page systems. Note that the address mapping policy is selected so as to reduce bank conflicts in the system. The address mapping policies sdram_close_page_map and sdram_hiperf_map [4] provide the required mapping for closed page and open page systems respectively.

Scheduling policies . Several DDRx DRAM command scheduling policies were studied. These include

- **Greedy** The memory controller issues a command as soon as it is ready to go. Priority is given to DRAM commands associated with older transactions.
- **Open Bank First (OBF)** is used only for open-page systems. Here priority is given to all transactions which are issued to a currently open bank.
- **Most pending** is the "most pending" scheduling policy proposed by Rixner [9], where priority is given to DRAM commands to banks which have the maximum number of transactions.
- **Least pending:** is the "least pending" scheduling policy proposed by Rixner [9]. Commands to transactions to banks with the least number of outstanding transactions are given priority.
- **Read-Instruction-Fetch-First (RIFF):** prioritizes memory read transactions (data and instruction fetch) over memory write transactions.

5. Results

We study two critical aspects of the memory systems performance: the average latency of a memory read operation, and the sustained bandwidth.

5.1 The Bottom Line

Figure 5 compares the behavior of the different DRAM systems studied. The graphs show the variation in latency and bandwidth with system configuration for the SPEC-Mixed workload executing in an open page system. In the graphs systems with different topologies but identical numbers of ranks are grouped together on the x-axis. Within a group, the bars on the left represent topologies with more ranks/channel and as one moves from left to right, the same ranks are distributed across more channels. For example, for the group of configurations with a total of 8 ranks of memory, the

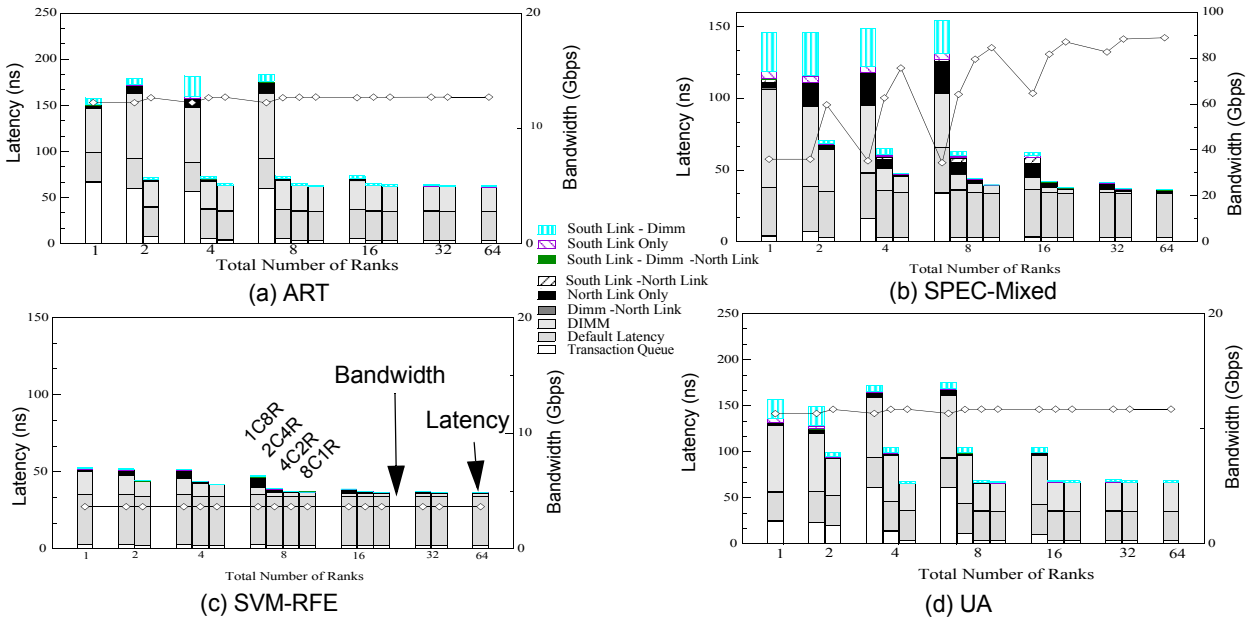


Figure 6. Latency and Bandwidth characteristics for different benchmarks. The figure shows the best latency and bandwidth characteristics for a DDR2 closed page system. Systems with identical numbers of dimms are grouped together on the x-axis. Within a group, the bars on the left represent topologies with lesser numbers of channels.The left y-axis plots latency in nano-seconds, while the right y-axis plots bandwidth in Gbps. Note that the lines represent bandwidth and the bars represent latency.

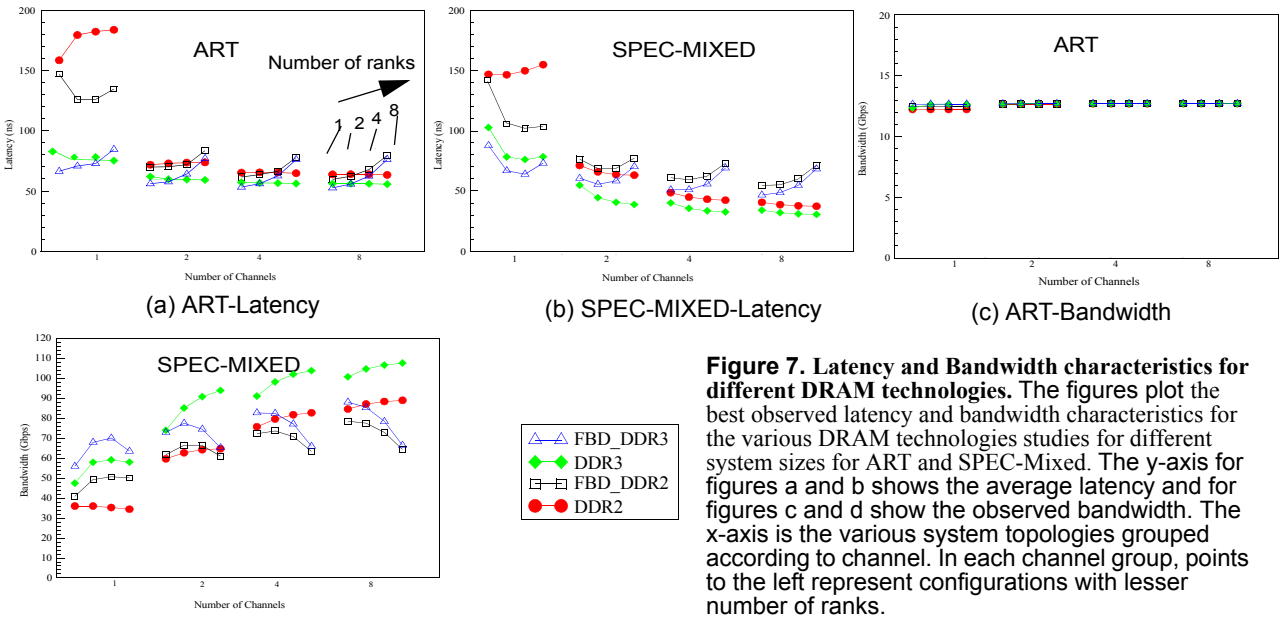
leftmost or first bar has the 8 ranks all on 1 channel, the next bar has 2 channels each with 4 ranks, the third bar has 4 channels with 2 ranks each and the rightmost in the group has the 8 ranks distributed across 8 channels i.e. 8 channels with 1 rank each. A more detailed explanation of the latency components will be given in later sections.

The variation in latency and bandwidth follow the same general trend regardless of DRAM technology. This observation is true for all the benchmarks studied. At high system utilizations, FBDIMM systems have average improvements in bandwidth and latency of 10% and 7% respectively, while at lower utilizations DDRx systems have average latency and bandwidth improvements of 25% and 10%. Overall though, FBDIMM systems have an overall average latency which is 15% higher than those of DDRx systems, but have an average of 3% better bandwidth. Moving from a DDR2 based system to DDR3 system improves latency by 25-50% and bandwidth by 20-90%.

Figure 6 compares the behavior of the different workloads studied. All data is plotted for a DDR2 closed page system (not open page system as in the earlier figure) but is fairly representative for other DRAM technologies as well. The variation in latency and bandwidth are different for each different workload. The performance of SPEC-Mixed improve until a configuration with 16 ranks, after which it flattens out. ART and UA see significant latency reductions and marginal bandwidth improvements as channel count is increased from 1 to 4. By contrast, SVM-RFE which has the lowest memory request rate compared to all the applications does not benefit by increasing the number of ranks.

Figure 7 shows the latency and bandwidth characteristics for the SPEC-MIXED and ART workload in a closed page system. The relative performance of a FBDIMM system with respect to its DDRx counterpart is a strong function on the system utilization. At low system utilizations the additional serialization cost of an FBDIMM system is exposed, leading to higher latency values. At high system utilizations, as in the single channel cases, an FBDIMM system has lower latency than a DDRx system due to the ability of the protocol to allow a larger number of in-flight transactions. Increasing the channel depth typically increases the latency for FBDIMM systems while DDRx systems experience a slight drop. For DDRx systems the increase in ranks essentially comes for free because the DDRx protocols do not provide for arbitrary-length turnaround times. For FBDIMM systems with fewer channels the latency of the system drops with increasing channel depths. By being able to simultaneously schedule transactions to different DIMMs and transmit data to and from the memory controller, FBDIMM systems are able to offset the increased cost of serialization. Further, moving the parallel bus off the motherboard and onto the DIMM helps hide the overhead of bus turn-around time. In the case of more lightly loaded systems like some of the multi-channel configurations, due to the lower number of in-flight transactions in a given channel, we find that FBDIMMs are unable to hide the increased transmission costs.

The bandwidth characteristics of an FBDIMM system are better than that of the corresponding DDRx system for the heterogeneous workload - SPEC-Mixed (Fig 7d). SPEC-Mixed have concurrent read and write traffic and are thereby able to better utilize the additional system bandwidth available in an



(a) ART-Latency

(b) SPEC-MIXED-Latency

(c) ART-Bandwidth

(d) SPEC-MIXED-Bandwidth

Figure 7. Latency and Bandwidth characteristics for different DRAM technologies. The figures plot the best observed latency and bandwidth characteristics for the various DRAM technologies studies for different system sizes for ART and SPEC-Mixed. The y-axis for figures a and b shows the average latency and for figures c and d show the observed bandwidth. The x-axis is the various system topologies grouped according to channel. In each channel group, points to the left represent configurations with lesser number of ranks.

FBDIMM system. The improvements in bandwidth are more in configurations which are more busy i.e. lesser number of channels. Busier workloads tend to use both FBD channels simultaneously, whereas the reads and writes have to take turns in a DDRx system. As channel count is increased and channel utilization gets lower, and the gains in bandwidth diminish till they vanish completely. This is especially true for DDR3 based systems due to the significantly higher data rates and consequently lower channel utilization. In the case of the homogenous workloads, e.g. ART (Fig 7c), which have lower request rates, varying the number of channels and ranks does not have signifcant impact.

The RIFF scheduling policy which prioritizes reads over writes has the lowest read latency value for both DDRx and FBDIMM systems. No single scheduling policy is seen to be most effective in improving the bandwidth characteristics of the system.

5.2 Latency

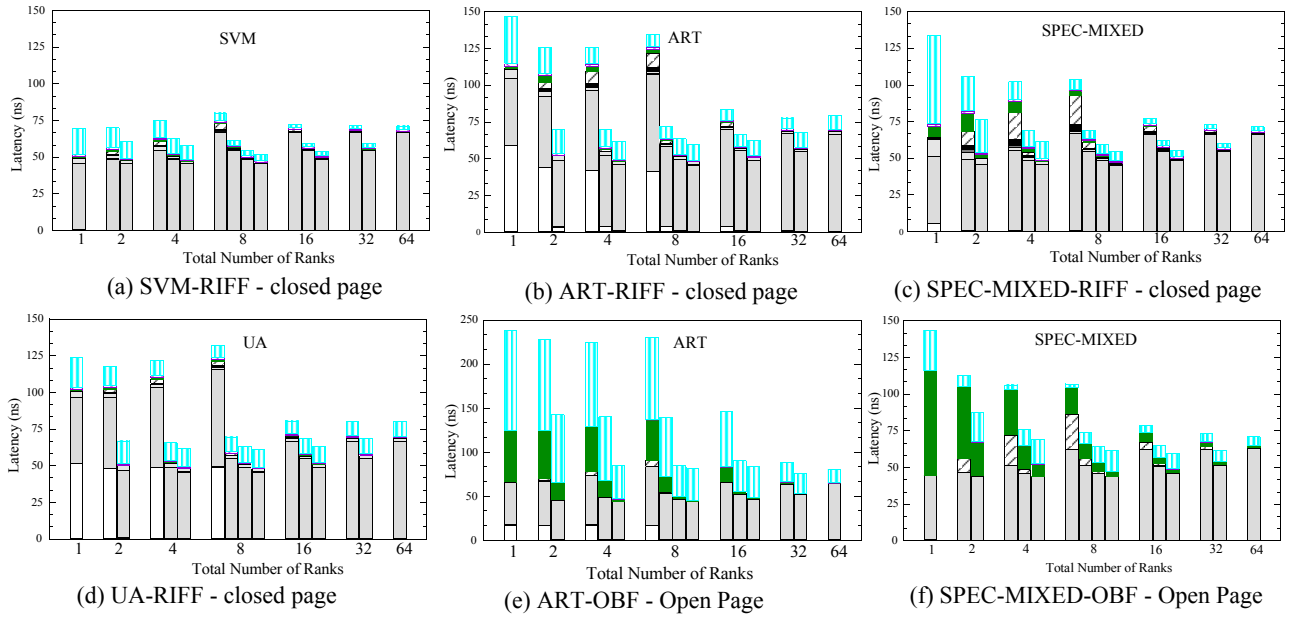
In this section, we look at the trends in latency, and the factors impacting them, in further detail. The overall behavior of the contributors to the read latency are more characteristic of the system and not particular to a given scheduling policy or row buffer management policy. Hence, we use a particular row buffer management policy, RIFF in the case of closed page systems and OBF in case of open page systems for the discussion.

Figure 8 shows the average read latency divided into queueing delay overhead and the transaction processing overhead. The queueing delay component refers to the duration the transaction waited in the queue for one or more resources to become available. The causes for queueing delay include memory controller request queue availability, south link availability, DIMM availability (including on-DIMM command and data buses and bank conflicts), and north link availability. Note that the overlap of queueing delay is monitored for all components except the memory controller request queue factor. The default latency cost is the cost associated with making a read request in an unloaded channel.

The queueing delay experienced by a read transaction is rarely due to any single factor, but usually due to a combination of factors. Changing the system configuration, be at by adding more ranks in the channel or increasing the number of channels in the system, result in a change in the availability of all the resources to a different degree. Thus, we see that the latency trends due to changes in a configuration are affected by how exactly these individual queueing delays change.

Typically, single-rank configurations have higher latencies due to insufficient number of memory banks to distribute requests to. In such systems the DIMM is the dominant system bottleneck and can contribute as much as 50% of the overall transaction queueing delay. Adding more ranks in these system helps reduce the DIMM-based queueing delays. The reductions are 20-60%, when going from a one rank to a two-rank channel. For 4-8 rank channel, the DIMM-based queueing delay is typically only 10% of that in the single-rank channel.

In an FBDIMM system, the sharing of the southbound bus by command and write data results in a significant queueing delay associated with southbound channel unavailability. Applications ART, SPEC-MIXED and UA, which have a fair proportion of write traffic experience fairly significant queueing delays due to the unavailability of the southbound channel (Fig 8 b, c, d). In nearly all cases, the southbound channel queueing delay is reduced by increasing the number of channels in the system. Some reductions in southbound channel unavailability are also achieved by adding more ranks in the system. In a multi-rank channel, the memory controller is more likely to be able to pack multiple commands in the same frame. Though the



(a) SVM-RIFF - closed page

(b) ART-RIFF - closed page

(c) SPEC-MIXED-RIFF - closed page

(d) UA-RIFF - closed page

(e) ART-OBF - Open Page

(f) SPEC-MIXED-OBF - Open Page

Figure 8. Read Latency for Various Configurations. Systems with identical numbers of dimms are grouped together. Within a group, the bars on the left represent topologies with fewer numbers of channels. The dram configuration was FBD-DDR2 using fixed latency mode. The y-axis shows latency in nano-seconds. The y-axis scales are different for graphs (e)

opportunities to do so are limited, this can result in a decrease in southbound channel unavailability by 10-20% when increasing the number of ranks in a channel from 1 to 2.

Increasing the channel depth raises the transmission costs for frames on the channel. This results in an increase in the read latency with increasing channel depth for SVM-RFE (Fig 8 a) and for the other workloads at larger channel widths when the utilization is low. Interestingly, the additional DRAM-level parallelism available in a deeper channel can counter the rise in frame transmission costs sufficiently to reduce overall latency (Fig 8 c- f). The gains in parallelism are especially apparent when going from a single-rank to a two-rank channel. These gains gradually taper off for further increases in channel depth.

With deeper channels, the increased number of in-flight transactions results in an increase in the queueing delay due to the link-unavailability. This is attributable to the heightened competition for the bus due to the additional number of in-flight transactions. This increase can combine with the increase in processing cost to offset the gains due to increased DIMM-level parallelism. Thus, we see that for UA and ART, Fig 8d and 8 b respectively, that the latency increases in a single channel system when the channel depth is increased from 4 to 8. The interaction of the increase in processing overhead, number of in-flight transactions using the channel, the type of these transactions and the lowering of DIMM-level conflicts result in the different latency trends observed across various benchmarks.

All the applications had higher latency values in an open page system than in a closed page system. Interestingly the differences were more pronounced for the homogenous workloads, such as UA etc., than for the SPEC-MIXED workload. Figure 8 e, f show the latency characteristics for ART and SPEC-MIXED respectively, in an open page system using the Open-Bank-First (OBF) scheduling policy. Although single-application workloads like ART see significant locality of accesses, the access pattern is such that requests to the same DRAM row are interspersed with accesses to different DRAM rows in the same bank. Consequently, the OBF memory scheduler delays handling read requests that arrived earlier in order to satisfy a later arriving a request that maps to a currently open row. This can result in an increase in queueing delay for the read transactions which in turn increases average read latency. The SPEC-MIXED workload experiences a 20-30% bank hit rate which is far lower than the 50-70% experienced by the homogenous workloads, but experiences a far lesser degradation in read latency in an open page system. The arrival and interaction of the various address streams at the memory controller results in the memory controller keeping pages open for a shorter duration. Hence, only requests to the same DRAM page that are a short distance apart are able to use the open page. More importantly, requests that are far apart and target the same page do not end up delaying the servicing of intermediate requests to different DRAM rows.

Figure 9 provides a pictorial representation of the scheduling policy having the lowest latency for different system topologies for the ART and SPEC-MIXED workloads. Similar behavior was exhibited by the other workloads as well. RIFF, which prioritizes read requests over write requests, performs the best for most of the cases. Although RIFF is effective in reducing read

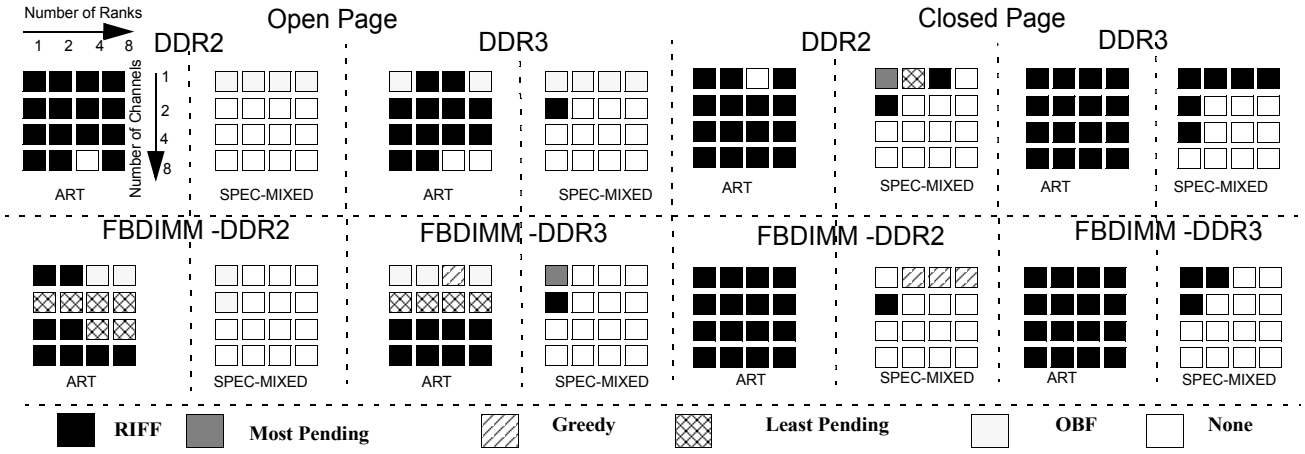


Figure 9. Best Scheduling Policy for Total Bandwidth The figure shows the scheduling policy having the best latency characteristics for each different configurations for the various applications studied.A scheduling policy emerges as the winner if the observed mean latency for the given configuration is 2.5% or more better than the worst performing scheduling policy for a given configuration point. Note that none implies that no clear winner emerged. All data for FBDIMM is in fixed latency mode.

latency it can increase the latency of a read that is stalled waiting for an earlier write to complete. In general, this phenomenon does not exhibit itself often enough for most of the applications. In an open page system the RIFF scheduling policy also does better than a scheduling policy like OBF, which attempts to reduce read latency by exploiting DRAM locality. The OBF scheduler can delay servicing a read request in order to handle an earlier issued write command which experiences a page hit. The RIFF scheduler on the other hand prioritizes the read traffic, thereby resulting in the RIFF scheduler performing better than the OBF scheduler.

Figure 10 shows the average percentage improvement in latency obtained by operating the system in variable latency mode. All numbers are given for a closed page system, but similar behavior was observed in an open page system as well. FBD-DDR2 performed similarly to FBD-DDR3. The variable latency mode was effective in lightly loaded systems, as in the case of systems with 2-8 channels of memory and all configurations for the SVM-RFE workload. As expected, the variable latency mode is more effective for systems with deeper channels.

On the other hand in a heavily loaded system, using variable latency mode increases the average read latency. In variable latency mode the read data latency for a transaction is dependent on the distance of the rank from the memory controller. Consequently using this mode lowers the effective utilization of the north link by introducing idle slots between read data returns from different ranks. As the channel depth increases, the duration of the idle gaps increases, making the latency characteristics worse (Figure 10 (b)).

5.3 Channel Utilization

5.3.1 Sustained Bandwidth

Unlike the case of read latency, the sustained bandwidth is less sensitive to the choice of scheduling policy. More than 80% of the cases are relatively insensitive to the scheduling policy and no clear winner emerges for the remaining 20% of the cases. Figure 11 shows the variation in bandwidth for the SPEC-Mixed and UA.

The sustained bandwidth achieved improves for the SPEC-Mixed (Fig 11 a, b) with increasing channel depth for both open page and closed page systems. Note that all the bandwidth curves are dome-shaped, with the general shape of the dome a function of the number of channels in the system. This shows that as DIMMs are added to a channel, bandwidth is increased via increased support for concurrency. However, as more ranks are added to a channel, you get increasing numbers of resource conflicts which lowers bandwidth. A closed page system con-

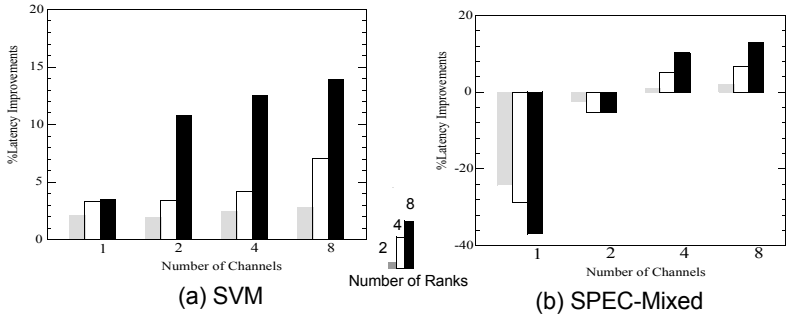
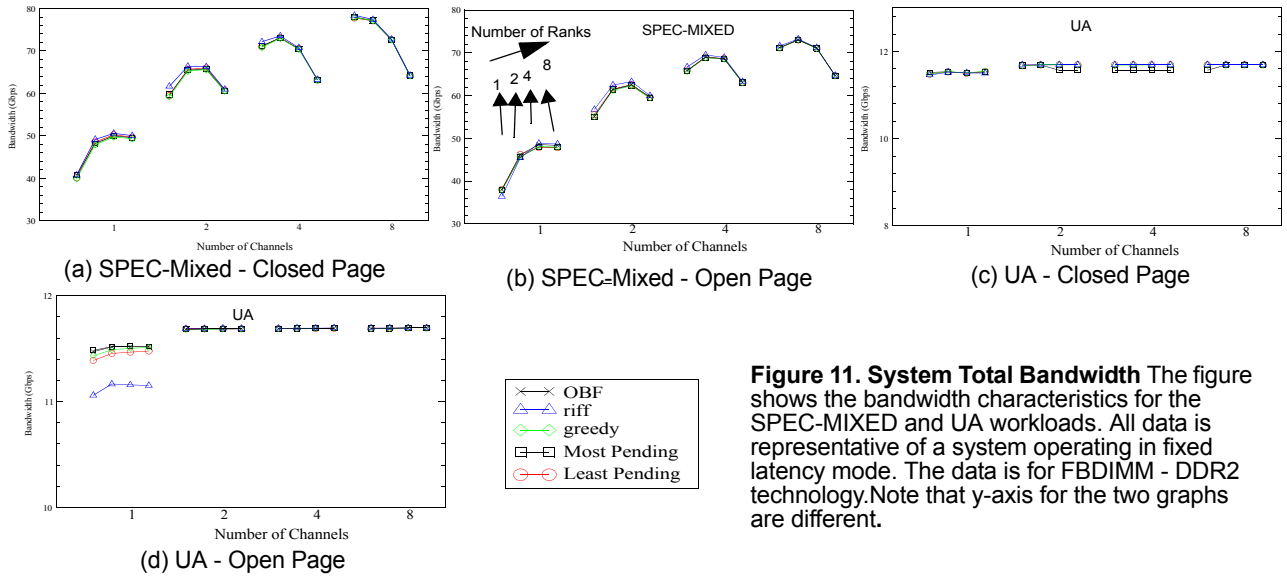


Figure 10. Latency Improvements using Variable Latency Mode. The data in these graphs are for a FBD-DDR2 system. Configurations with the same number of channels are grouped together on the x-axis. Within a group, the bars on the left represent fewer numbers of ranks. Values for single rank configurations are not displayed because the operating mode does not make a difference for these configurations.



(a) SPEC-Mixed - Closed Page

(b) SPEC-Mixed - Open Page

(c) UA - Closed Page

(d) UA - Open Page

Figure 11. System Total Bandwidth The figure shows the bandwidth characteristics for the SPEC-MIXED and UA workloads. All data is representative of a system operating in fixed latency mode. The data is for FBDIMM - DDR2 technology.Note that y-axis for the two graphs are different.

figuration see 2.5-10% improvement in bandwidth over an open page system.

In the case of UA (Fig 11 b, d), closed page configurations had slightly better bandwidth characteristics than open page configurations. In a single channel configuration, all the scheduling policies that re-ordered requests based on DRAM locality were more effective than the RIFF policy. This is because these approaches tended to satisfy transactions, irrespective of type faster than RIFF. By prioritizing read traffic, the RIFF policy would end up delaying writes to a currently open bank and thereby lower overall throughput.

5.3.2 South Link Utilization

The FBDIMM south link is shared by both commands and write data. The graphs in figure 12 provide the distribution of bandwidth utilization of the different types of FBDIMM frames. A command frame can carry up to three commands while a command-data frame carries 9 bytes of data (8-bytes of data and 1 ECC-byte) and one command. In the figure, command frames are further distinguished by the number of occupied command slots. The system does not send a data frame unless data is present thus there are no zero-data frames.

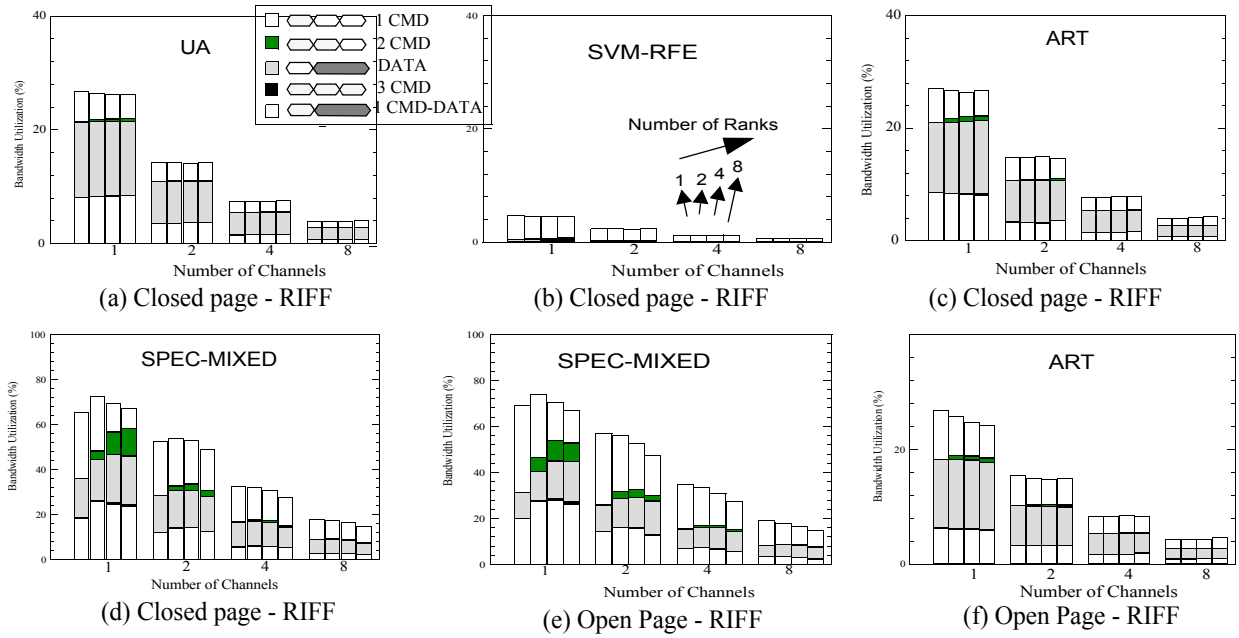
For workloads with a significant proportion of write traffic, the south link emerges as the bottleneck. The UA-B, ART and SPEC-MIXED workloads (Fig 12 (a, c, d) use 35-70% of the south link bandwidth. Nearly two-thirds of this utilized bandwidth is used to transmit write data. On the other hand for a read-dominated workload like SVM-RFE (Fig 12(b)), the southbound link is used to transmit only command frames.

Despite the ability to send multiple commands in a frame, the opportunities to do so are more limited. Across all benchmarks, we found that the memory controller is more likely to schedule a command by itself. In a read-intensive workload like SVM-RFE (Fig 12 b), with a lower proportion of write traffic the command frame is more likely to be sent with 2 NO-OPS. For workloads with write-traffic like UA-B, SPEC-Mixed and ART (Fig 12 (a, c, d)) a data payload is sent along with the command. With increases in channel depth, these workloads see a slight increase in the number of command frames that have 2 valid commands. Regardless of the channel depth, less than 5% of the frames sent have 3 commands in them and less than 10% of the frames have 2 or more commands.

The command bandwidth used by an open page system is a function of the bank hit rate. A transaction in an open page system uses only one DRAM command when there is a hit to an open row and at least three commands (RAS, CAS and PRECHARGE) when there is a bank conflict. A closed page system on the other hand uses two commands, RAS and CAS with implicit precharge for each operation. Any open page system that has less than 50% bank hit ratio will have a higher command utilization. In the case of the SPEC-Mixed workloads the bank hit ratio is in the range of 20-30% and hence the command bandwidth requirement is higher (as seen by comparing Fig 12(d) and Fig 12 (e)) but in the case of ART, the bank hit rate is around 50 - 70 % resulting in slightly lower command bandwidth requirements for an open page case (Figs 12(c) and (f)).

5.4 Summary of Results

Our study indicates that an FBDIMM system provides significant capacity increases over a DDRx system at comparable performance. Although an FBDIMM system experiences an overall average latency increase compared to the corresponding DDRx system, we found that by efficiently exploiting the extra DIMM-level parallelism in a busier system the latency costs can be significantly lowered and even eliminated. The split-bus architecture makes the performance of the system sensitive to the ratio of read to write traffic. For instance, it enables workloads with both read and write traffic to sustain higher bandwidths than in a DDRx system. Workloads that use the system heavily tend to record higher gains because they tend to use both channels simultaneously.



(a) Closed page - RIFF (b) Closed page - RIFF (c) Closed page - RIFF

(d) Closed page - RIFF (e) Open Page - RIFF (f) Open Page - RIFF

Figure 12. Southbound Channel Utilization Distribution. FBDIMM southbound frames are command frames or write data-command frames. The former carries 1-3 valid commands and the latter can carry 1 command and 9 bytes of data or only 9 bytes of data (8 bytes data + 1 byte ECC). The y-axis shows the distribution of bandwidth utilization for the southbound channel. The x-axis is the various system topologies grouped according to channel. In each channel group, bars to the left represent configurations with lesser number of ranks.

Even more interestingly the behavior of an application in the conventional memory architecture and the newer one are similar. The RIFF scheduling policy typically has the lowest latency in all cases, while no single scheduling policy had significantly higher bandwidth utilization. The performance of open page systems was sensitive to the distance between accesses to the same DRAM row and the types of the requests. Many of the scheduling policies, such as open bank first, most pending etc., attempt to exploit locality in the DRAM rows to lower latency. We found that in addition to all these factors, a controller policy that prioritizes reads over writes is more effective. A designer who implements such a policy should take care to give higher priority to outstanding write transactions that delay read transactions to the same bank and to periodically drain write requests in order to prevent the memory transaction queue from filling up.

The sharing of the write data and command bandwidth results in the southbound channel being a significant bottleneck. This competition for the bus impacts the latency of a memory read access as well. This problem is further aggravated because the memory controller is unable to fully utilize the command slots in a southbound frame.

While the variable latency mode is effective in lowering the latency of a read transaction in a system by 2.5-20% in lightly loaded systems, the exact opposite effect was observed at higher utilizations. This suggests that a controller that dynamically switches the mode of operation in response to changes in system utilization would see less performance degradation.

6. Conclusions

To the best of our knowledge, this is the first study of the performance characteristics of a fully-buffered DIMM memory system. This study examines how currently existing memory controller policies perform on this new memory architecture and how the FBDIMM architecture performs relative to a DDRx system. We found that the performance of an FBDIMM system is comparable to that of a conventional DDRx system and memory controller policies used on the latter are still relevant in the former, provided that reads and writes are prioritized appropriately. The latency and bandwidth characteristics of an FBDIMM system are better than a DDRx system for configurations where there is more DIMM-level parallelism and sufficient memory traffic. Like in DDRx based systems a scheduling policy that favours read traffic over write traffic has better latency characteristics.

FBDIMM systems are able to provide increased capacity at higher utilizations with no performance losses. However, these gains do not come for free. The AMB is expected to increase the per DIMM power consumption by the order of 4 W in an FBDIMM-DDR2 system [3]. In our future work we would like to study how memory controller policies can be used to improve the power envelope of these systems.

7. Acknowledgements

The authors would like to thank Sadagopan Srinivasan, Jessica Tseng, Nuengwong Tuaycharoen and Ankush Varma for their feedback and comments. The work of Brinda Ganesh, and Aamer Jaleel was supported in part by NSF

CAREER Award CCR-9983618, NSF grant EIA-9806645, and NSF grant EIA-0000439. The work of Dave Wang was supported in part by NSF CAREER Award CCR-9983618, NSF grant EIA-9806645, NSF grant EIA-0000439, and Cray Inc. The work of Bruce Jacob was supported in part by NSF CAREER Award CCR-9983618, NSF grant EIA- 9806645, NSF grant EIA-0000439, DOD MURI award AFOSR-F496200110374, the Laboratory of Physical Sciences in College Park MD, the National Institute of Standards and Technology, and Cray Inc.

8. References

[1] J. Haas and P. Vogt, "Fully-buffered dimm technology moves enterprise platforms to the next level," in *Technology@Intel Magazine*, March 2005.

[2] P. Vogt, "Fully buffered dimm (fb-dimm) server memory architecture: Capacity, performance, reliability, and longevity." Intel Developer Forum, February 2004.

[3] Micron, *240-Pin 512MB/1GB/2GB DDR2 SDRAM FBDIMM (DR FB x72) Features*, April 2005.

[4] D. Wang, B. Ganesh, N. Tuaycharoen, K. Baynes, A. Jaleel, and B. Jacob, "Dramsim: A memory-system simulator."," *SIGARCH Computer Architecture News*, vol. 33, 2006.

[5] Micron, *1Gb: x4,x8,x16 DDR SDRAM Features*, Jan 2006.

[6] D. Burger, J. R. Goodman, and A. Kagi, "Memory bandwidth limitations of future microprocessors," in *ISCA'96: Proceedings of the 23rd annual international symposium on Computer architecture*,

[7] V. Cuppu, B. Jacob, B. Davis, and T. Mudge, "A performance comparison of contemporary dram architectures," in *ISCA'99: Proceedings of the 26th annual international symposium on Computer architecture*, (Washington, DC, USA), pp. 222–233, IEEE Computer Society, 1999.

[8] V. Cuppu and B. Jacob, "Concurrency, latency, or system overhead: which has the largest impact on uniprocessor dram-system performance?," in *ISCA'01: Proceedings of the 28th annual international symposium on Computer architecture*, (New York, NY, USA), pp. 62–71, ACM Press, 2001.

[9] S. Rixner, W. J. Dally, U. J. Kapasi, P. R. Mattson, and J. D. Owens, "Memory access scheduling," in *ISCA*, pp. 128–138, 2000.

[10]S. Rixner, "Memory controller optimizations for web servers," in *MICRO 37: Proceedings of the 37th annual International Symposium on Microarchitecture*, 2004.

[11]F. A. B. Chitra Natarajan, Bruce Christenson, "A study of performance impact of memory controller features in multi-processor server environment.," in *Workshop on Memory performance issues*, 2004.

[12]W. fen Lin, S. K. Reinhardt, and D. Burger, "Reducing DRAM latencies with an integrated memory hierarchy design," in *HPCA*, 2001.

[13]Z. Zhu and Z. Zhang, "A performance comparison of dram system optimizations for smt processors," in *HPCA*, 2005.

[14]Z. Zhu, Z. Zhang, and X. Zhang, "Fine-grain priority scheduling on multi-channel memory systems," in *8th International Symposium on High Performance Computer Architecture, (HPCA-8)*, 2002.

[15]Z. Zhang, Z. Zhu, and X. Zhang, "A permutation-based page interleaving scheme to reduce row-buffer conflicts and exploit data locality," in *MICRO 33: Proceedings of the 33rd annual ACM/IEEE international symposium on Microarchitecture*, (New York, NY, USA), pp. 32–41, ACM Press, 2000.

[16]J. B. Carter, W. C. Hsieh, L. Stoller, M. R. Swanson, L. Zhang, E. Brunvand, A. Davis, C.-C. Kuo, R. Kuramkote, M. Parker, L. Schaelicke, T. T. Kuramkote, M. Parker, L. Schaelicke, and T. Tateyama, "Impulse: Building a smarter memory controller," in *HPCA*, 1999.

[17]"www.jedec.org."

[18]A. Jaleel, M. Mattina, and B. Jacob, "Last-level cache (llc) performance of data-mining workloads on a cmp–a case study of parallel bioinformatics workloads.," in *Proc. 12th International Symposium on High Performance Computer Architecture*, 2006.

[19]J. Henning, "Spec cpu2000: Measuring cpu performance in the new millenium," in *IEEE Computer*, July 2000.

[20]H. Jin, M. Frumkin, and J. Yan, "The openmp implementation of nas parallel benchmarks and its performance."

[21]V. Aslot, M. Domeika, R. Eigenmann, G. Gaertner, W. B. Jones, and B. Parady, "Specomp: a new benchmark suite for measuring parallel computer performance," in *Workshop on OpenMP Applications and Tools*, 2001.

[22]A. Jaleel, R. S. Cohn, C.Luk, and B. Jacob. CMP$im: A Binary Instrumentation Approach to Modeling Memory Behavior of Workloads on CMPs", Technical Report - UMD-SCA-2006-01

[23]R. Desikan, D. Burger, S. Keckler, and T. Austin, "Sim-alpha: a validated, execution-driven alpha 21264 simulator," Tech. Rep. TR-01-23, The University of Texas at Austin, Department of Computer Sciences, 2001.

[24]http://www.ece.umd.edu/BioBench