Memory Access Pattern Analysis

Mary Brown* Roy M. Jenevein† Nasr Ullah
System Performance, Modeling, and Simulation, Motorola Inc.
6300 Bridgepoint Parkway
Austin, TX 78730
mdb@eecs.umich.edu, rmj@powerquest.com, nasr@ibmoto.com

Abstract

A methodology for analyzing memory behavior has been developed for the purpose of evaluating memory system design. MPAT, a memory pattern analysis tool, has been used to profile memory transactions of dynamic instruction traces. This paper will first describe the memory model and metrics gathered by MPAT. Then the metrics are evaluated in order to determine what hardware and software changes should be made to improve memory system performance.

1 Introduction

It is well known that as processor performance increases, the gap between memory access and processor execution rate widens. We must provide the methodology to better understand the patterns of memory access so that more optimum performance can be achieved from the memory subsystem. In order to develop a better understanding of memory usage, a methodology has been developed for the study of these memory access patterns. This study has allowed us to observe the impact of memory access patterns on the utilization of the memory system. The results of the study are used to help make specific architectural decisions for memory controller and memory system design. Without information about the impact of program behavior on the memory sub-system, it is difficult to make the necessary architectural design and sizing decisions for a memory controller. In addition, operating system designers should be aware of the detrimental impact of page coloring algorithms on memory system performance, which is revealed by this study.

The main parts of the memory system architecture that we are focusing on in this study are the necessity for open page retention, the open page replacement algorithms used by the memory controller, and the requirements for the memory transaction scheduler. It is possible to determine the cost-performance benefits of these features by profiling the physical memory accesses to measure the benefit of keeping DRAM pages open.

When a system makes the first memory reference to a memory subsystem, there are no memory pages open and the memory controller must issue a full memory address (both row and column addresses) to the selected DRAM bank. Subsequent references to the same page (with the same row address) can be accessed by simply providing the column address if the memory controller keeps the current row address active. This method allows lower latency accesses to the same page than would be possible if the controller had to provide a full address for each access. Current-generation memory controllers provide the necessary register space to keep DRAM pages open. This paper presents some novel techniques for visualizing the memory paging behavior of applications in order to evaluate the benefits of maintaining open page information.

2 Methodology

The use of open pages in the memory sub-system is governed not only by the memory activity of the application, but also by the refresh interval of the DRAMs used in the memory sub-system as well as the number of DRAM banks used to implement the memory structure. In order to properly investigate the performance impact of open pages, it is necessary to consider all these factors when modeling a memory sub-system. The more DRAM banks a memory subsystem has, the greater the number of possible open pages. DRAM refresh impacts open-page performance because all open pages are closed during a refresh cycle.

The key metrics of paging activity which must be measured are how many pages are open before each refresh, how many unique pages are accessed each refresh interval, and the page hit ratio for each interval.

A tool named MPAT (Memory Pattern Analysis Tool)

*The University of Michigan at Ann Arbor
†PowerQuest Corporation. On leave from The University of Texas at Austin

has been developed for profiling the memory transactions of programs. This tool accepts an instruction trace or memory access trace generated by software or a bus monitor as input. After all memory accesses of the trace are filtered through a cache model, the physical memory transactions are profiled through a memory model for evaluating the memory scheduler and open page demands.

2.1 Cache model

The cache models that were used depict the caching structure and dataflow of the PowerPC^{TM} microprocessor family[3]. The L1 instruction and data caches used by MPAT are both 32KB write-back caches with 32-byte lines, 8-way set-associativity, and an LRU way replacement algorithm. The L2 cache is a 2-way set-associative write-back cache with 32-byte lines. Simulations were run using both a 512KB and 1MB L2 cache. All cache accesses as well as cache misses are processed in the order that they occur in the instruction trace.

2.2 Memory model

The synchronous DRAM model used by MPAT is parameterizable with respect to page sizes, number of DRAM banks, open page retention, and refresh intervals. The page retention buffer is a collection of registers which assert the row address strobe for a specified bank. In the memory model used by MPAT, there may be as many registers in the page retention buffer as banks in the memory system. The buffer is fully associative but only one row address strobe may be asserted for any bank. Thus in the MPAT model there can be as many open pages as the the number of DRAM banks in the memory subsystem. Although some DRAM subsytems support more than one open page per bank, these will not be considered in this paper. MPAT considers any two addresses which are in the same bank and have the same row address strobe to be in the same page. Any page which is accessed will remain open until one of the following conditions occurs[2]:

- The new page to be accessed lies in the same bank as a currently open page. The currently open page is forced to close in order to precharge the new row address strobe.
- There is an access to an unopened page on a bank which does not have any open pages and the open page retention buffer is full. One of the open pages must be closed.
- A refresh occurs. All pages are closed when a refresh occurs.

2.3 Metrics of Interest

The following metrics are gathered within each refresh interval of the instruction trace:

- Number of open pages at the end of the refresh interval. This is how many pages were forced to close due to a page refresh. If the number of open pages is less than or equal to the maximum number of open pages, this metric indicates how many banks were used in this refresh interval.
- Number of times pages were opened within the window.
- Number of unique pages accessed within the window. This helps determine the working set size of the program.
- Open page miss ratio. This is calculated as the number of times a page had to be opened divided by the total number of DRAM accesses.
- Number of I/O transactions and ROM accesses.
- Number of accesses to each bank. This helps determine the spatial distribution of the working set. It is possible that some of the banks used within one refresh interval actually had very few accesses to them. If this happens, more space in the open page retention buffer will be utilized without adding significant benefits to performance. Because of this, measuring the open page miss ratios using a variable maximum number of open pages is a better way to determine the cost-performance optimal page retention buffer size.
- Number of transactions since the most recent access to the same page. It is not uncommon for the addresses accessed to alternate or ”ping-pong” between two pages on the same bank. In such a situation, if all transactions are processed in order, there would be a forced RAS precharge before every transaction. If a memory scheduler were used, many of the precharges could be avoided if the transactions were issued close together in time. In order to determine how much benefit arises from processing memory transactions out of order, the amount of ”ping-pong” activity must be measured. For every memory transaction, MPAT counts how many accesses have occurred since the most recent access to the same page. A number, which we will refer to as a ping-pong distance, is associated with each transaction. The ping-pong distance is defined as the number of memory transactions since the most recent transaction to the same page. If a memory transaction were to the same page as the previous transaction, it would have a ping-pong distance of one. The ping-pong distances for all transactions are measured throughout the simulations.

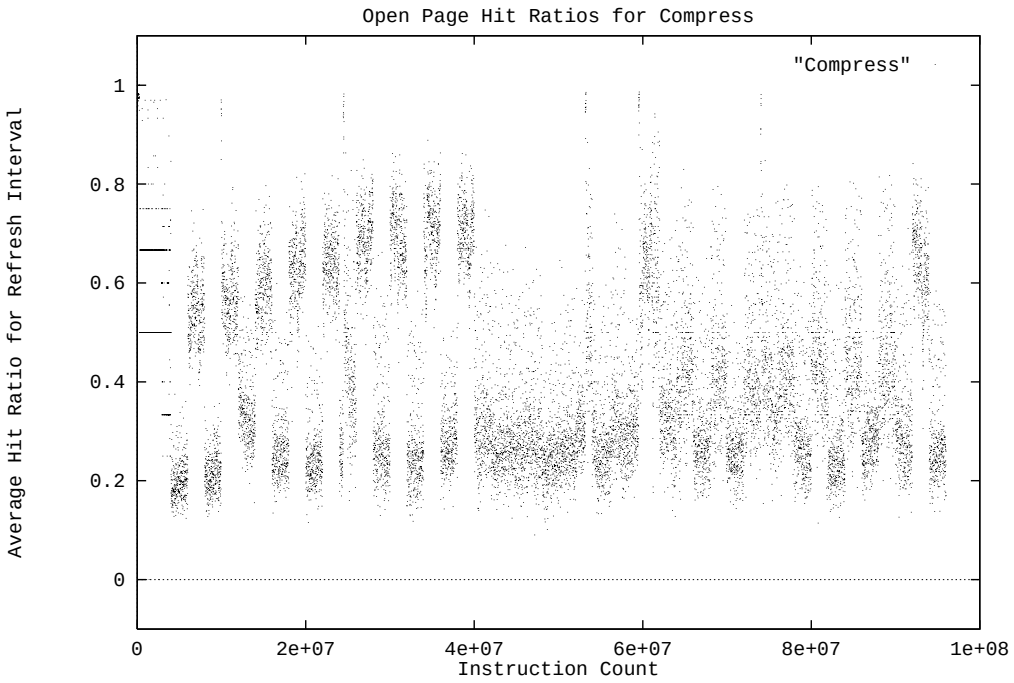

Figure 1. Open page hit ratios for the compress benchmark. This simulation used a 512KB L2 cache and a memory configuration with a maximum of 16 open DRAM pages.

2.4 MPAT Constraints

Because the instruction traces used by MPAT may not contain time stamps, an instruction count is used as an approximation of time. The number of instructions between DRAM refresh intervals is calculated using the processor cycle time, a standard refresh time interval of 15.6 microseconds[2], and the approximated CPI for the processor, benchmark, and memory configuration. This approximation affects not only the refresh interval time, but the time between transactions as well.

The order as well as time of memory transactions may not reflect the actual system behavior. However, since we estimate the time of memory transactions with instructions and assume all transactions occur in program order, we are still characterizing the behavior of the application without being constrained by reordering within the processor-system model. When an instruction trace is used as input, the instruction address, followed by the data address for load or store instructions, is processed in order for each instruction in the trace. This means any rescheduling effects due to the processor pipeline, caches, processor bus, or any modules between the processor bus and memory controller are not modeled. This does not present a problem because analyzing the measurements of ping-pong activity while assuming transactions occur in program order allows us to obtain an upper bound on the effects of processor-memory controller rescheduling.

2.5 Benchmarks

Simulations have been run on the multimedia applications Specular InfiniD, Macromedia FreeHand, Adobe Photoshop, MPEG, image rotation, gaussian blur, and Tartus-Rave3D, as well as the SPECint95 benchmarks gcc, go, li, perl, compress, vortex, ijpeg, m88ksim. The SPECint95 traces were generated on a PowerPC 604e^{TM} microprocessor running on AIX, and the multimedia programs were generated on a PowerPC 604e microprocessor running on MacOS. Simulations for all benchmarks were run using identical DRAM and cache configurations.

The ratio of the speed of main memory to the processor speed necessitates the use of extremely long instruction traces. All of the traces used in these simulations contained on the order of one hundred million instructions. Trends in the memory patterns of certain benchmark algorithms have been observed to last up to and possibly longer than several tens of millions of instructions.

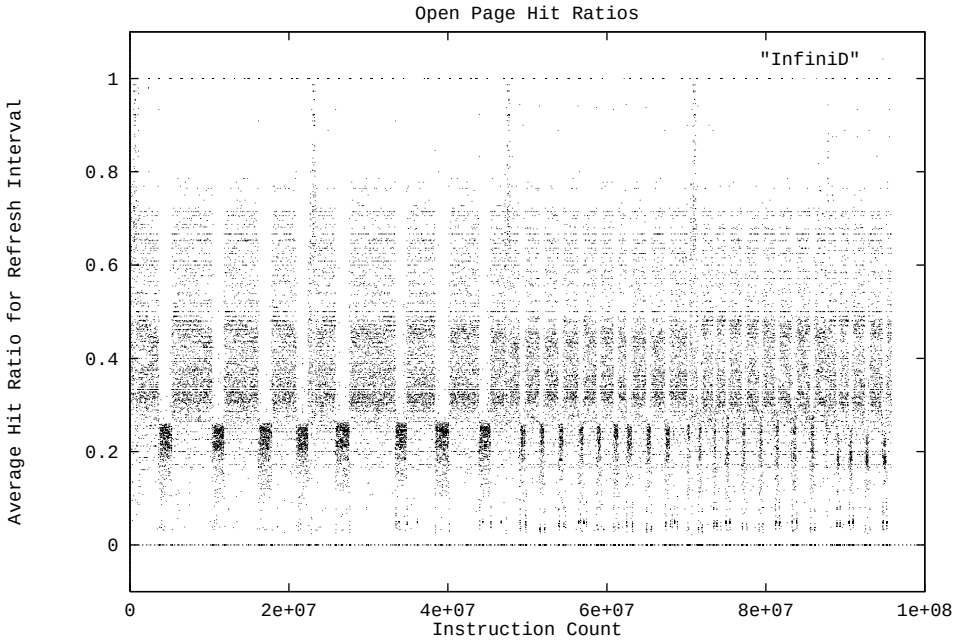

Figure 2. Open page hit ratios for the InfiniD benchmark. This simulation used a 512KB L2 cache and a memory configuration with a maximum of 16 open DRAM pages.

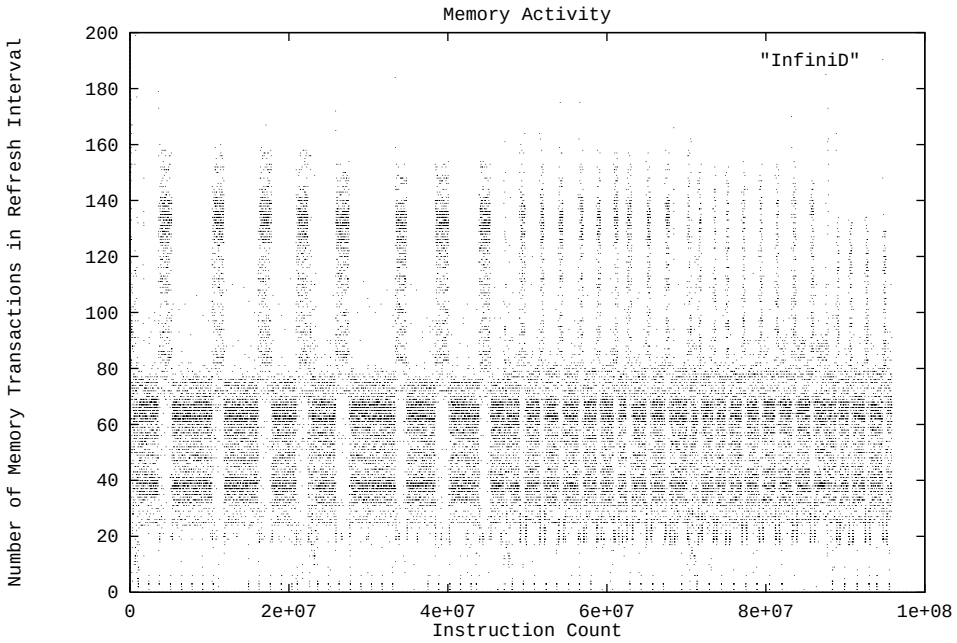

Figure 3. Memory Activity for the InfiniD benchmark. This simulation used a 512KB L2 cache and a memory configuration with a maximum of 16 open DRAM pages.

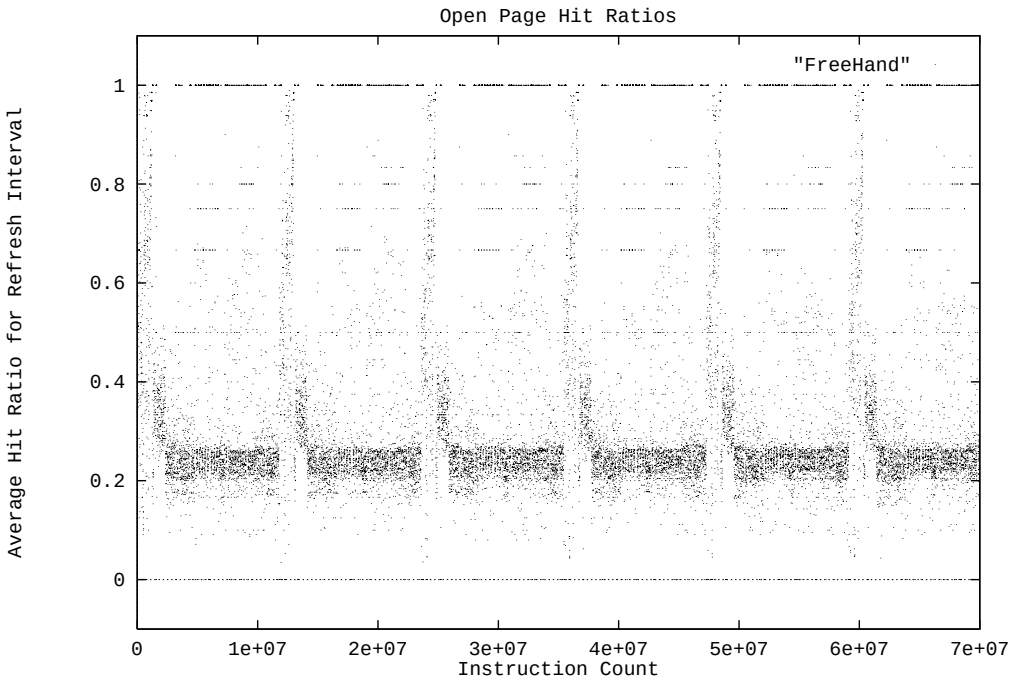

Figure 4. Open page hit ratios for the Adobe Freehand benchmark. This simulation used a 512KB L2 cache and a memory configuration with a maximum of 16 open DRAM pages.

3 Metrics Evaluation

In order to obtain a general characterization of the memory access patterns, we used several approaches for evaluating the metrics. First, overall statistics for each benchmark such as the averages and standard deviations of the metrics were determined. These statistics were too general and were biased by periods of time when the memory system was inactive and periods of time when the memory system activity was highly irregular.

Next, instead of looking at overall averages, we focused specifically on the metrics for each refresh interval. The number of pages open before each refresh as well as the miss ratios and number of unique pages accessed per refresh interval were plotted for each benchmark trace. Plots of open page hit ratios for the SPECint95 benchmark compress and the multimedia benchmarks InfiniD and FreeHand are shown in figures 1, 2, and 4. These plots show the changes in the open-page hit ratios over time which are characteristic of the program under execution. The x-axis of each graph is the instruction count of the trace. It provides a measure of progress through the trace (i.e. time). The y-axis represents the average open-page hit ratio for each refresh interval. Each marked pixel of the plot represents the average number of times that an open page is hit in one refresh interval. The graph appears dense because of the great number of refresh intervals during the execution of the trace. Figures 1,2, and 4 provides a fingerprint or pattern of the applications use of open-pages.

Figure 3 shows the density of memory accesses throughout the InfiniD trace. Each marked pixel of the trace indicates the number of memory accesses within one refresh interval. Comparisons of figures 2 and 3 illustrate the strong correlation between the amount of memory activity and the dispersiveness of the addresses. As is typical of all the SPECint95 traces, the memory access patterns of the compress trace are not as regular as the multimedia benchmark traces. The multimedia traces exhibit short bursts of heavy memory activity at regular intervals. In general, the multimedia benchmarks showed more regular patterns than the SPECint95 traces.

This method provided a mechanism to visualize the paging and transaction behavior of the algorithms for each Benchmark. Since the traces contain some 20,000 to 40,000 refresh intervals it is necessary to further process this information to focus on more precise memory page behavior. To do this, we organized the refresh intervals into buckets. These buckets were categorized according to how many DRAM transactions occurred in each interval. Graphs of the open page hit ratios and memory activity distributions

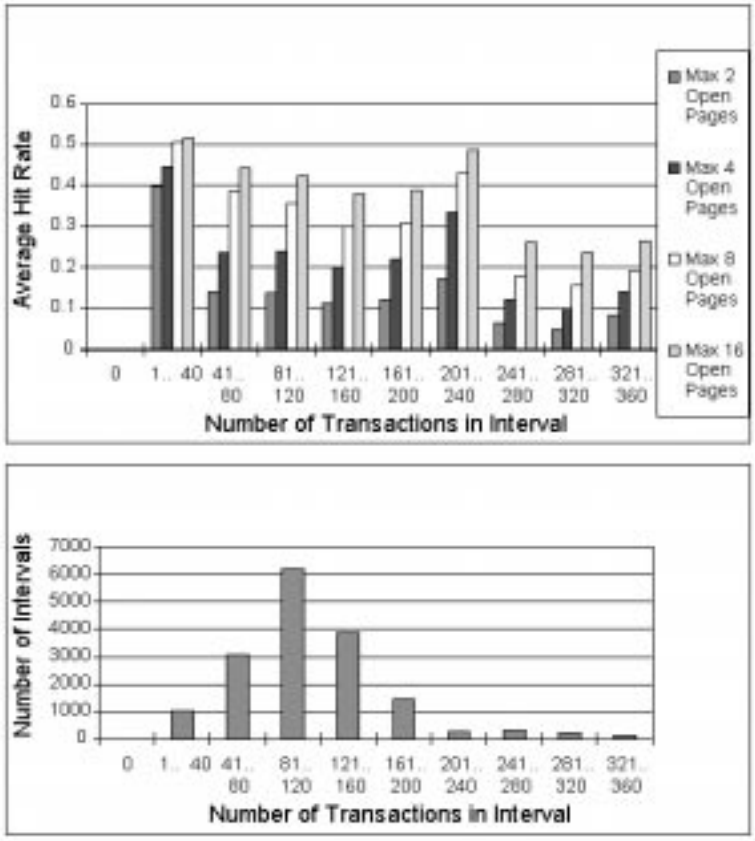

Figure 5. Open page hit ratios and number of transactions per refresh interval for compress using a maximum of 2, 4, 8, and 16 open pages.

for compress, InfiniD, and Freehand are shown in figures 5, 6, and 7.

In these figures, we combine the transaction and the open-page hit information. We take refresh intervals with approximately the same memory transaction traffic and analyze them together using the buckets. Each of these figures are shown as a pair of graphs. The first graph depicts the open-page hit ratio assuming 2,4,8 and 16 open page registers in the open-page retention buffer. The second graph simply shows the number of refresh intervals that occurred in each bucket. For example, in the first graph of Figure 6 all refresh intervals containing zero transactions are placed in the first bucket, while all refresh intervals containing one to forty transactions were placed in the second bucket. Correspondingly in the second graph of figure 6, we can observe the number of refresh intervals for each bucket. As depicted in the graph there are over 2000 refresh intervals that do not have any memory transactions. Additionally we can observe that the largest number of refresh intervals (approx. 12000) contain 41 to 80 memory transactions each.

Used together, these two graphs allowed us the ability to sift out the periods of time with unusually heavy, very little, or no memory activity. Because some of the benchmarks periodically had refresh intervals with no memory activity, without this further analysis, results obtained would have been biased towards lower demands on the memory controller.

A further analysis using the bucket method revealed that the proportion of refresh intervals with no DRAM activity ranged from 0% to 50% for the simulations run with a 512kB L2 cache. For some of the traces, these periods of time with no DRAM activity were occupied with ROM or

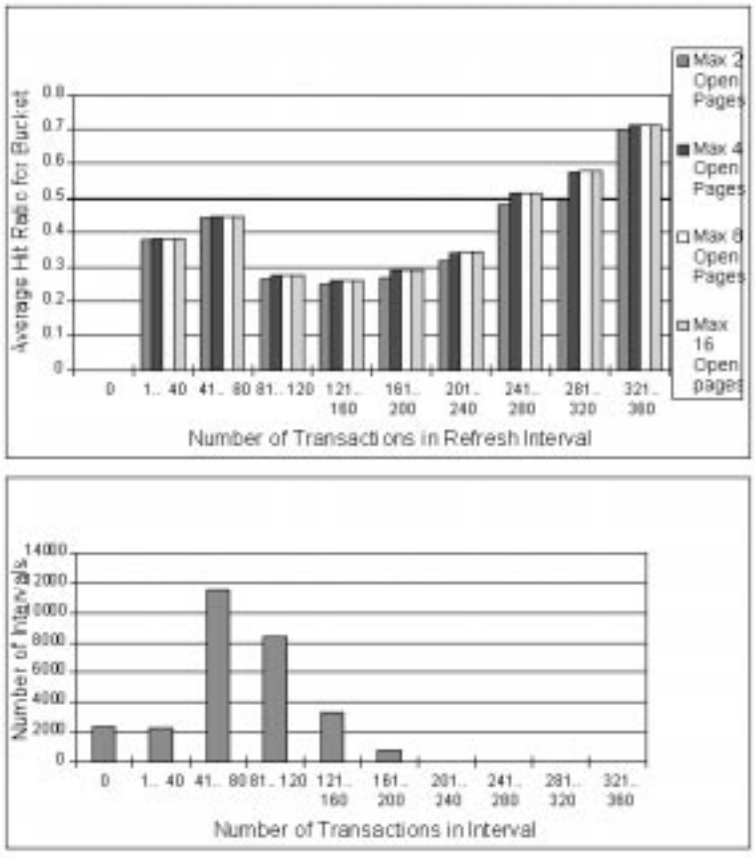

Figure 6. Open page hit ratios for the Adobe Freehand benchmark using a maximum of 2, 4, 8, and 16 open pages.

I/O accesses. For others, the working set lay almost completely in the cache.

4 Results

For all simulations, a DRAM configuration with 16 banks and various maximum numbers of open pages are used. The page miss ratios and number of open pages before refreshes for each simulation were compared in order to determine how much performance increased by using a large number of open pages.

4.1 Open Page Retention

In order to determine an optimal number of open pages to be supported in hardware, simulations were run using a maximum of 2, 4, 8, and 16 open pages. As illustrated in figures 6 and 7, the page hit ratios of the traces running on MacOS showed little or no improvement when a maximum of more than four open pages were used. This is due to the fact that not all banks of memory were used by the programs. Because of the virtual to physical page coloring algorithm used by MacOS, the range of the physical memory used by a program does not occupy very many banks. Because of the distribution of the working set, the high density of accesses to a small range of the physical addresses caused a significant amount of page contention. This results in a large set of page close, page open sequences.

For most of the benchmarks running under AIX, the page hit ratios tended to level off at 8 open pages and there was little or no benefit to using 16 open pages. Only compress and vortex frequently used more than 8 open pages. The

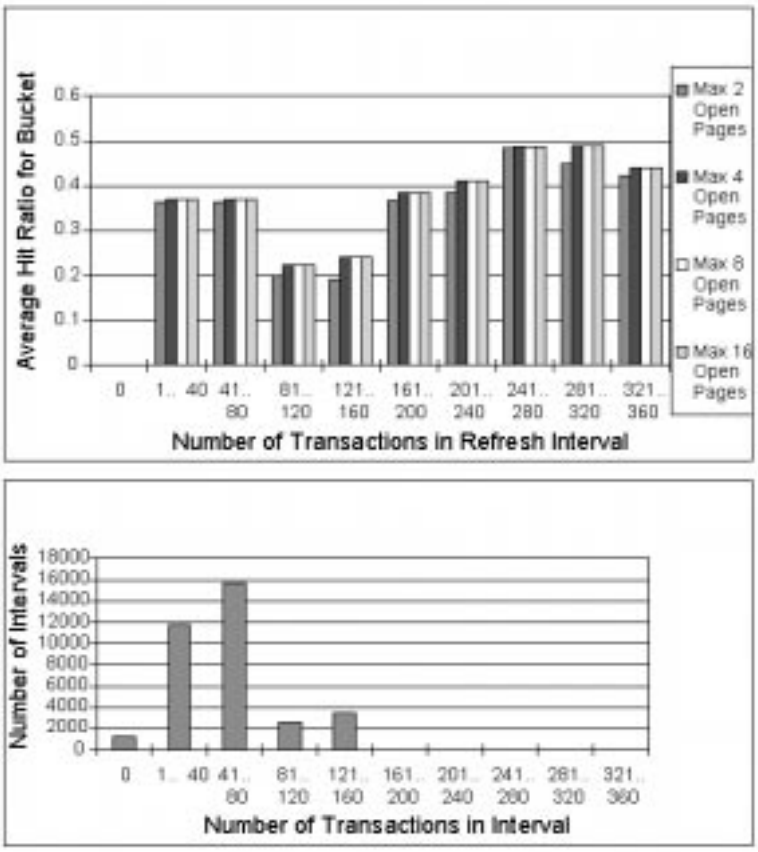

Figure 7. Open page hit ratios for the InfiniD benchmark using a maximum of 2, 4, 8, and 16 open pages.

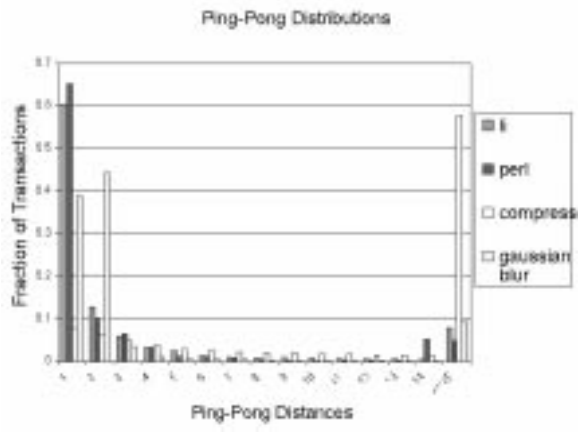

Figure 8. Ping-pong distributions for li, perl, compress, and gaussian blur.

increase in open page usage as compared to the MacOS benchmarks is due to the fact that the physical pages used by the programs were allocated randomly throughout memory, which increased the number of banks used within each refresh interval.

The differences in the results obtained from the benchmarks running under the two different operating systems shows how heavily the virtual to physical OS page coloring algorithm affects the benefits of supporting several open pages.

The effects of different page replacement algorithms on the performance were found to be inconsequential. Simulations using a maximum of four open pages were run using LRU, Round Robin, and Random DRAM replacement. There was very little difference in the miss rates among these replacement algorithms; LRU had a one percent higher average hit ratio than Round Robin and Random replacement.

4.2 Benefits of Out-of-order Scheduling

The ping-pong metric was used to study the affects of out-of-order scheduling. For all of the benchmarks studied, 10% to 60% of all memory transactions were to the same page as the previous transaction. The proportion of transactions to unused pages varied widely among the benchmarks. The distribution of ping-pong distances in the benchmarks li, perl, compress, and Adobe Photoshop gaussian blur are shown in figure 8. The first group (ping-pong distance of 1) shows the percentage of transactions that were to the same page as the previous transaction, and the last group (ping-pong distance greater or equal to 15) shows the percentage of transactions to pages which had not been accessed within the previous 14 transactions. This graph shows the most extreme cases of ping-pong activity for all benchmarks. Li and perl showed the most regularity in page accesses. The gaussian blur frequently had accesses which alternated between 2 pages. Compress had the most highly irregular access pattern; over 50% of the memory transactions were to pages that were not recently referenced. The proportion of transactions with ping- pong distances between 2 and 15 ranged between 25% and 50% for each benchmark. Overall, 37% of the memory transactions had ping-pong distances in the range of 2 to 15. If these transactions could be re-ordered to be adjacent then they would benefit more from a memory controller's open page policy.

5 Conclusions

The results of this study have allowed us to evaluate the degree of hardware complexity in the memory controller that will be efficiently utilized by workloads. It also provides a rational basis for the determination of the number of open pages to expect within a refresh cycle and hence the size of the register set in the controller to support that activity.

Because of the impact of the range of the working set size on the memory performance, the physical page allocation algorithms of different operating systems should be investigated further. In addition, systems with more than one processor sharing memory or one processor with multiple cores executing more than one instruction stream may have a wider combined working set[1]. If such systems are to be used, the consolidated access patterns of multiple threads should be investigated.

References

[1] Flynn, Michael J. *Computer Architecture: Pipelined and Parallel Processor Design*. Jones and Barlett Publishers, Inc., Sudbury, MA, 1995.

[2] *MPC106 PCI BridgeMemory Controller User's Manual*. Motorola Inc., 1997.

[3] *PowerPC 604e RISC Microprocessor User's Manual*. Motorola Inc., 1998.