SMP System Interconnect Instrumentation for Performance Analysis

Lisa Noordergraaf and Robert Zak

Sun Microsystems
{lisa.noordergraaf, robert.zak}@sun.com

Abstract

The system interconnect is often the performance bottleneck in SMP computers. Although modern SMPs include event counters on processors and interconnects, these provide limited information about the interaction of processors vying for shared resources. Additionally, transaction sources and addresses are not readily available, making analysis of access patterns and data locality difficult. Enhanced system interconnect instrumentation is required to extract this information.

This paper describes instrumentation implemented for monitoring the system interconnect on Sun Fire™ servers. The instrumentation supports sophisticated programmable filtering of event counters, allowing us to construct histograms of system interconnect activity, and a FIFO to capture trace sequences. Our implementation results in a very small hardware footprint, making it appropriate for inclusion in commodity hardware.

We also describe a sampling of software tools and results based on this infrastructure. Applications have included performance profiling, architectural studies, and hardware bringup and debugging.

1 Introduction

In a modern SMP, the system interconnect is often the performance limiting component. Unfortunately, system interconnect performance is extremely difficult to analyze due to the complex interactions of various processors and devices which utilize the interconnect for main memory accesses, cache to cache transfers, and I/O. Analysis of performance is further impaired by the lack of detailed information available.

Modern SMPs generally contain hardware performance monitors. Typically, hardware event counters are incremented based on a limited set of processor-visible events (e.g. cache misses, pipeline stalls), including, in some cases, system interconnect transactions. While excellent for isolating performance issues within the processor pipeline and cache hierarchy, these simple event counters do not provide enough information to allow detailed analysis of many performance issues.

Specifically, raw event counts provide no information about the sources of the events, nor information about the memory addresses involved or snoop states of the caches. Information about the sequencing of various transactions is also unavailable. To extract this type of information, more detailed instrumentation of the system interconnect is required.

The complete set of system interconnect information for an SMP is a timestamped record of every system transaction that occurs during the execution of a given OS segment or application. Unfortunately, due to very high system transaction rates and relatively long application duration, storage of such a complete record is impractical.

Still, partial or incomplete traces represent an important source of information for system designers and performance analysts, and have been shown to be an effective method of exploring memory system behavior[1]. Unfortunately, traditional sources of traces have significant drawbacks.

External hardware monitoring boards which can be inserted into the system interconnect to monitor and record transactions are becoming increasingly costly to produce as system interconnect speeds increase, and are also not available to the wider performance community. Despite substantial progress in the field of high-speed system simulators[2], analysis of actual hardware systems offers key advantages, including support of very large applications, memory images, and storage images, high speed operation, and direct mapping of measured results to the actual production hardware and software environment. Software instrumentation of applications can introduce substantial overheads, and only information about the particular executable which has been instrumented is available - this makes this technique generally unsuitable for evaluating complex multiprogrammed workloads and operating system activity.

0-7695-1524-X/02 $17.00 (c) 2002 IEEE

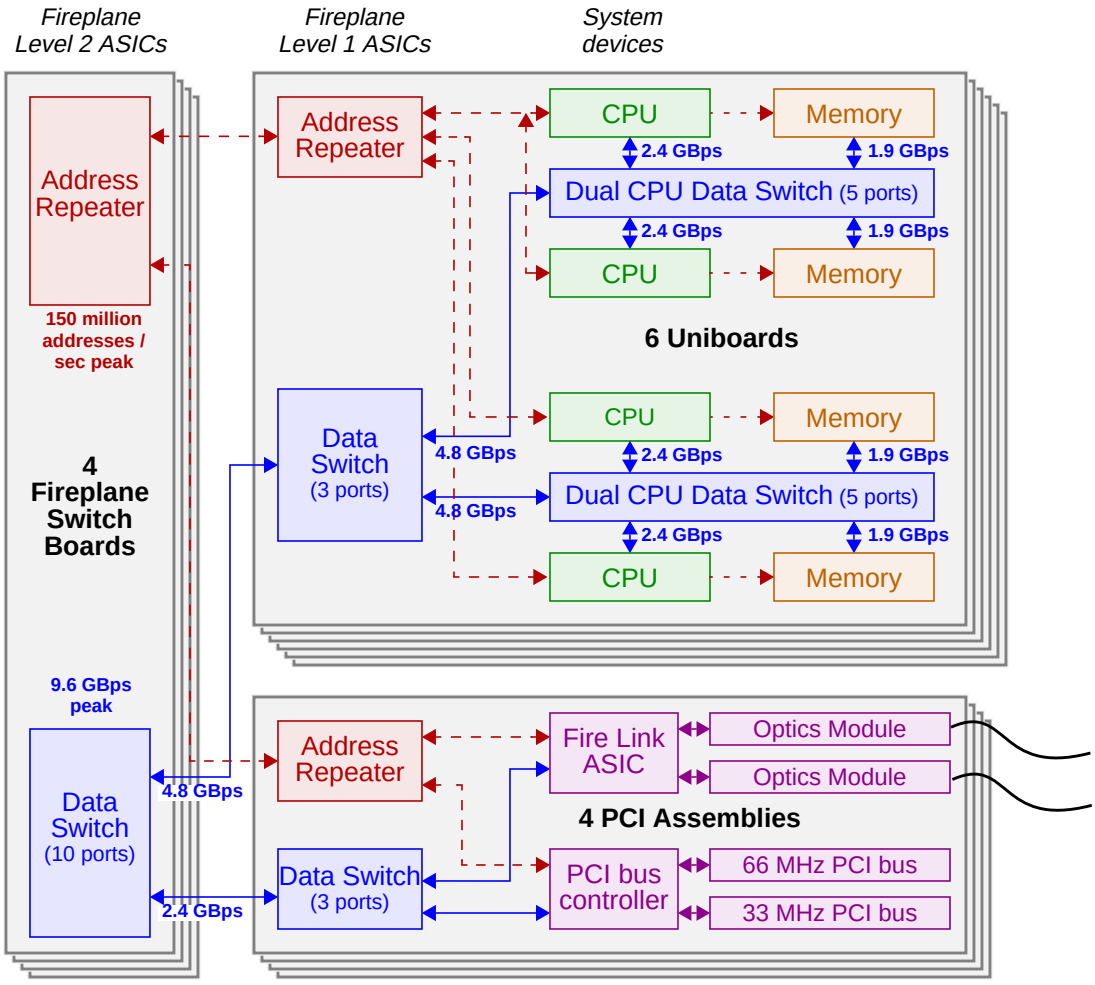

FIGURE 1. Mid-range Interconnect (Sun Fire 3800-6800 Servers). On systems configured with Sun Fire Link, the Fire Link ASIC and optics modules replace the 2nd PCI bus controller and associated PCI assemblies on the System I/O board.

An alternative to trace-based analysis involves the use of histogramming mechanisms, which allow for the gathering of statistics such as the distribution of transaction types, addresses accessed, and the relative contribution of various system devices to overall system interconnect usage. Histogramming mechanisms generally consist of logic for selecting events of interest and determining which "bucket" any event falls into, and one or more banks of memory used as a histogram array, with individual memory locations functioning as histogram "buckets" and collecting event counts. As with hardware-based trace collection, histogramming mechanisms used to date have required expensive, custom-built hardware, and have not been practical for inclusion in commodity chipsets.

This paper describes a set of embedded hardware instrumentation mechanisms for the system interconnect that are part of the Sun Fire™ Link hardware[3], a high-bandwidth, low-latency clustering interconnect for Sun Fire servers (see Figure 1). These embedded mechanisms have the advantage over external instrumentation both in cost and in universal availability, thus making the information available to the general performance community.

Our instrumentation includes a set of counters supporting the histogramming of system interconnect transactions within a given system interval, and a FIFO buffer for recording a limited sequence of consecutive transactions in time. Both allow programmable filtering of the system interconnect stream based on transaction parameters (i.e. source device, physical address range, transaction type, cacheline snoop state) which allows performance analysis tools to isolate and examine system effects due to critical transaction sequences.

As it is not practical to devote any significant amount of memory solely to performance data collection, we instead rely on software-based sampling techniques to construct our histograms and traces. In effect, our performance instrumen-

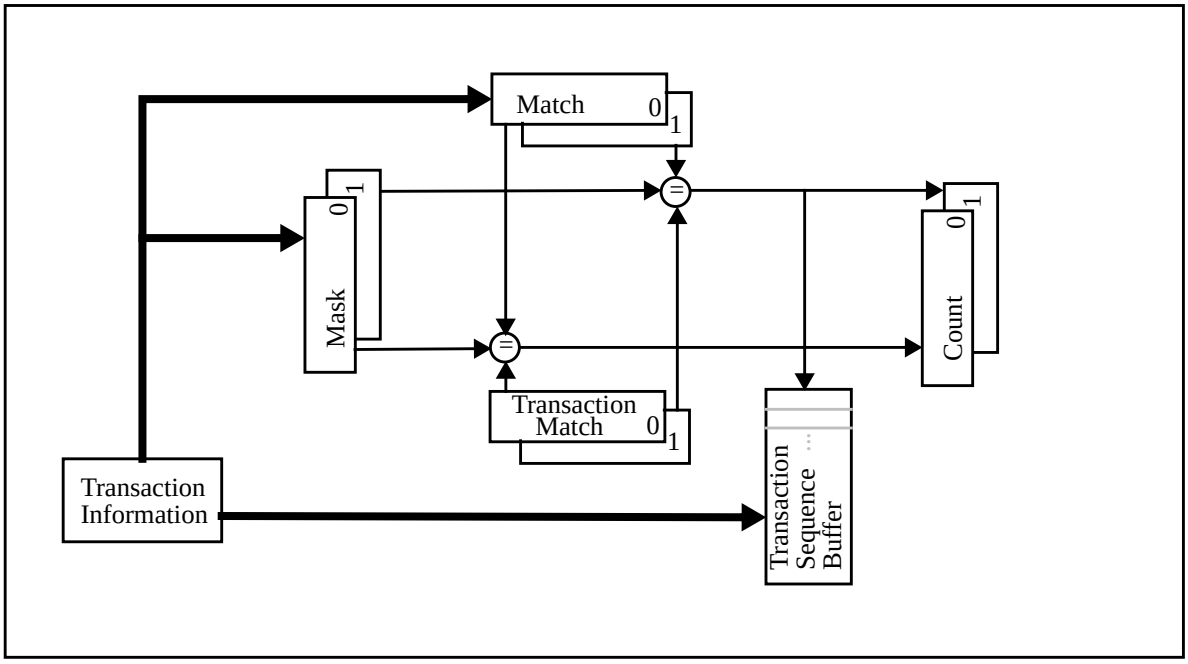

FIGURE 2. Fire Link Performance Instrumentation. The Mask register determines which bits of the transaction information are to be tested against the value specified in the Match register. A seperate Transaction Match register allows for the selection of a single type or any combination of transaction types. When a match occurs, the corresponding count register is incremented. In the case of count 1, transactions are also stored in the Transaction Sequence Buffer.

tation collects trace samples suitable for further analysis, albeit of a much shorter duration than advocated by previous work. Though all the implications of these shorter-duration traces have yet to be explored, it is clear that the available data is still sufficient to be extremely useful to performance analysts.

The remainder of this paper is organized as follows. Section 2 provides a description of the implementation and capabilities of the Sun Fire Link performance instrumentation. Section 3 explores some uses of the histogramming counters, while Section 4 describes tools built utilizing the transaction sequence buffer. Finally, Section 5 discusses related work, and Section 6 offers some conclusions.

2 Performance Instrumentation

We describe three mechanisms for dynamically summarizing the complete system interconnect transaction history: Transaction Filtering, Event Histogramming, and Transaction Sequence Sampling.

2.1 Transaction Filtering

System interconnect transactions typically contain both a transaction type and a physical address. In the Sun™ Fireplane interconnect found in Sun Fire servers, the transactions also include information on cache-coherence states, the source device of the transaction, and a transaction identifier[4]. The Fire Link performance instrumentation allows system transactions to be filtered based on combinations of these parameters.

The Fire Link filter function allows a unary mask to be applied for a namespace of 19 transaction types, which allows for filtering transactions of a single type or any arbitrary set of transaction types.

Parameters with very large domains, such as physical addresses[1], require something other than a unary filter. Our implementation allows such values to be filtered based on a vector of three-valued “trits”, where each “trit” has a value 1, 0 or X (Don’t Care). For example, a 4-trit vector "110X" produces a filtered stream of all transactions with address values of either 1101 or 1100. The Fire Link performance instrumentation allows binary filtering of the physical address, the source id of the transaction, and the transaction identifier.

Taken together, these mechanisms provide a highly flexible and dynamic method of targeting large or small regions of physical memory, individual or sets of source devices, and arbitrary combinations of transaction types. These parameterizable filters act as the first stage for the histogramming counters and transaction sequence buffers, described below.

1. For Sun Fireplane systems, the physical address spans 42 bits.

2.2 Histogramming Counters

Each Fire Link ASIC contains two sets of independent histogramming counters (see Figure 2). Each counter has an associated set of filter parameters, and is incremented each time a transaction occurs on the system interconnect which meets the filter criteria. This allows the precise creation of a "histogram" of system interconnect transactions which occur during sampled system intervals. Counters from multiple ASICs can combined to increase the number of simultaneously active events. In addition, the filters can be dynamically changed, and the counter values sampled to increase the apparent number of counters via time-multiplexing.

2.3 Transaction Sequence Buffers

Each Fire Link ASIC provides a buffer capable of storing a record of 8 transactions. The entry for each transaction includes the physical address, transaction type, transaction identifier, and cache coherence information for Fireplane system transactions. The filtering mechanism described earlier can be used to limit the captured sequence to "interesting" transactions.

When enabled by software, the transaction buffer stores transactions which match the filter criteria. Software may use status information in the FIFO to determine whether an over-run has occurred. Thus, a software tool, by repeatedly sampling the buffers can accumulate an accurate (if incomplete) transaction trace comprising segments of at least 8 consecutive transaction sequences.

3 Using the Histogramming Counters

In this section, we present examples of ways in which the histogramming counters have been used for performance analysis, and discuss how they were integrated into existing Sun performance counter tools.

3.1 Workload Analysis

Workload analysis, which we loosely define here as analyzing the number and types of different transactions required by particular workloads, is useful in many different aspects of performance analysis. A profile of transactions can provide a good overall measure of whether a particular workload is CPU, IO, or memory intensive, information which can be useful in deciding how to approach the performance tuning process.

During the hardware qualification of the Sun Fire™ servers, the Sun Fire Link performance instrumentation was used to construct tools to examine the total Fireplane utilization and transaction distribution achieved by the various applications in a test suite. The filters controlling which transaction types were counted by the histogramming counters were changed dynamically in software throughout the duration of the application. A histogram of the types of transactions was then be accumulated through sampling the histogramming counters. Examples of some raw outputs of this process for applications from the SPEComp[5] and NAS Parallel Benchmark[6] suites are shown in Figure 3.

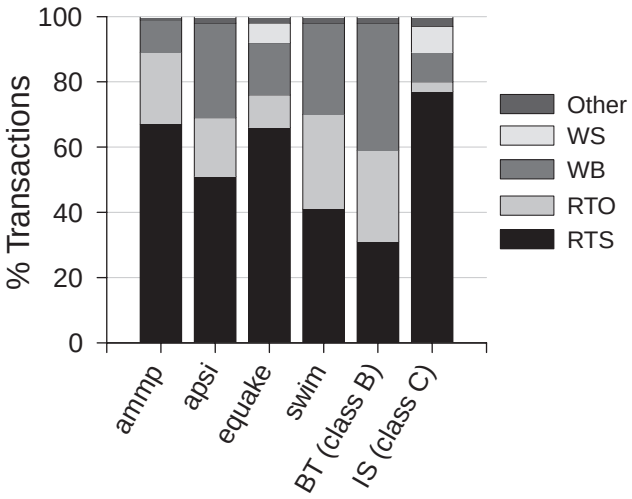

FIGURE 3. Transaction distribution for 6 different applications. All applications were run using 16 CPUs on a Sun Fire 6800. Transactions observed were *Read To Share* (RTS), *Read To Own*(RTO), *Write Back*(WB), and *Write Block IO* (WBIO).

These histograms allowed for the assessment of the relative frequency of transactions over the total runtime of the workload. Using this information, we were able to customize the loads presented to the system, and choose a mix of applications with different system interconnect footprints to run simultaneously to obtain a desired system load.

3.2 Analysis of System Interconnect Utilization

It is often desirable to be able to quickly determine the types of demands being placed on the system interconnect. At its most basic level, this information can tell the performance analyst if the system interconnect itself is a performance bottleneck. More detailed analysis of these demands can reveal the sources of congestion on the interconnect, and serve to focus the attention of the performance analyst towards those areas where attention should provide the greatest benefit.

To make this type of information easily available to end users, we have added support for the Fire Link performance counters to the busstat(1M) command[7], a command line tool in the Solaris™ Operating Environment that provides access to the bus-related hardware performance counters on the system. Calculations of interconnect utilization, and the percentage of traffic due to such things as IO, interrupts,

Event Name	Description
sfi_hstgrm_all_trans	A count of all transactions seen on the address bus.
sfi_hstgrm_int	All Interrupts
sfi_hstgrm_local_int	Interrupts from the local board
sfi_hstgrm_rmt_clu_incm_int	Interrupts generated by this RSM[a] device
sfi_hstgrm_rmt_ssm_incm_int	Interrupts generated by this SSM[b] device
sfi_hstgrm_io	All I/O transactions
sfi_hstgrm_rmt_ssm_incm_io	I/O from this RSM device
sfi_hstgm_cohrnt	All coherent transactions
sfi_hstgrm_rmt_clu_incm_cohrnt	Incoming coherent transactions from this RSM device
sfi_hstgrm_rmt_ssm_otg_cohrnt	Outgoing coherent transactions from this SSM device
sfi_hstgrm_rmt_ssm_incm_cohrnt	Incoming coherent transactions from this SSM device

TABLE 1. Safari Histogramming Counter busstat(1m) Events

a. The Sun Fire Link interconnect exports *remote shared memory* (RSM) model that supports low latency kernel bypass messaging[3].

b. Systems with more than 24 CPUs use multiple snooping coherence domains. This protocol is called *Scalable Shared Memory* (SSM), and described in more detail in [4].

coherence traffic, and Fire Link interconnect traffic, is easily done via the busstat interface. Table 1 provides a list of events that are currently available via busstat.

Often just being able to locate “hot spots” in the system is a great help to performance analysts, but there are many other possible uses of this information, one of which is scheduling algorithms. Traditional scheduling algorithms look primarily for free CPU cycles when determining where to schedule threads and processes to run. But in today’s systems, available bandwidth within a system board can be a limiting factor in performance. Information gathered about which system boards have particularly high bandwidth demands at a given point in time could be funneled to scheduling algorithms to assist in deciding where to schedule threads and processes.

4 Using the Transaction Sequence Buffer

In this section of the document, we discuss some uses of the transaction sequence buffer.

4.1 A Simple Cache Conflict Detection Tool

Since the UltraSPARC-III processors use physically indexed caches, it is possible to determine the cache blocks involved with each transaction based on the physical address captured by the transaction sequence buffer. In this section, we describe a prototype of a tool intended to detect capacity and conflict misses in external cache (E$). The tool works by analyzing writeback (WB) transactions seen on the Fireplane address bus, which are indicative of either a capacity or conflict miss in the E$; the physical address associated with the writeback is the address of the data being evicted from E$.

First, the transaction filtering mechanism is programmed to insure that only information about writebacks are seen and recorded by the transaction sequence buffer. The transaction sequence buffer is then sampled constantly by software throughout the course of the workload of interest. Information about each writeback transaction is recorded and saved to file. If desired, mappings of virtual to physical address mappings for processes of interest is collected for later use.

Once the workload has finished running and the writeback information has been recorded, post-processing scripts are used to analyze the collected data. For a given processor, a simple script computes the cache block index associated with each writeback, and compiles statistics on how many writebacks are associated with each cache block.

Figure 4 presents an example of such information displayed as a simple scatter plot. Note that while cache usage is fairly uniform, there are a few cache blocks which show excessive numbers of writebacks. Data extracted from the transaction buffer indicates that we counted 1,822 writebacks from cache block 2967, a number which is clearly well above the average. Our trace file reveals that though data evicted from

this cache block included references from 220 different pages, a single one of those pages accounted for 1,385, or 76%, of those writebacks. We can also see that 3 addresses within the page account for almost all of the writebacks. Information from a utility used to examine the mapping of virtual to physical addresses at runtime reveals that these addresses are part of the process stack. Figure 5 shows excerpts from our tool output related to this particular cache block.

Note that this information is generally sufficient for determining whether the majority of writebacks are caused by conflict or capacity misses. If conflict misses are a problem, the address information will make it obvious which addresses are repeatedly contending for the same cache block. In our example, the large number of writebacks is a result of capacity misses due to the data set in use exceeding our cache size.

Since the system interconnect exposes displaced cache lines, in systems with n-way associative cache hierarchies, the address being displaced can only be isolated to *n* potential cache blocks. In practice, this is not limiting because the displacement itself is what causes the performance impact, and the tuning exercise should address conflicts amongst any of the n competing cache blocks.

4.2 Analyzing cache-to-cache transfer behavior

Since the transaction sequence buffer captures the snoop results associated with each read transaction, it is possible to use this information to differentiate between transactions serviced by main memory and those serviced by processor caches; see Figure 6 on page 8 for one such example. This information can provide significant insights into SMP application behavior, and has direct implications for cache system architecture.

4.3 Analyzing Data Access Patterns and Locality

One of the major goals of this performance instrumentation was to assist in analyzing application locality, and provide information on data access patterns for application tuning purposes. There are a number of possible techniques for using the Fire Link performance instrumentation for these purposes.

Buck and Hollingsworth[8] have developed techniques for searching through the address space to isolate memory bottlenecks. These same search techniques can potentially be used to isolate areas of memory which are experiencing high numbers of remote (off-board) accesses.

An alternative approach to finding memory regions subject to high numbers of remote accesses involves programming the transaction filtering mechanism in such a way that only coherency operations are recorded. Data collected can then be analyzed to determine locality characteristics, and statistics on locality of particular addresses or address regions may be accumulated over time. This approach has the advantage of providing information as to the source of remote accesses, and in effect, allows us to characterize and understand data access patterns.

5 Related Work

We believe our implementation of performance instrumentation is unique in that its small hardware footprint makes it practical to include in commodity hardware. In the past, this information has typically been collected via external hardware probes or other mechanisms too expensive for use in commodity chipsets. We note here some previous examples of data collection mechanisms in this section, as well as pointing out some related software and tools.

Histogramming counters are certainly not a new idea. One of the earliest uses of histogramming counters that we are aware of is the ADAM hardware monitoring device for the Sigma 7 processor[9], which used two cooperating processors to construct histograms and small address traces.

Emer and Clark describe a performance monitoring system originally developed for the VAX 11/780[10], and later updated for the VAX 8800[11]. The monitor requires the use of a special histogram count board, and also a processor interface board.

The Stanford DASH (Directory Architecture for SHared Memory) multiprocessor project included a performance monitor which could be programmed to provide histogramming capabilities[12][13]. The DASH system does not use a separate external board to house instrumentation features. Instead, over 20% of the DASH directory controller on each node is devoted to performance monitoring hardware.

The SHRIMP (Scalable High-performance Really Inexpensive Multiprocessor) multicomputer developed at Princeton, included a performance monitor[17] with tracing and histogramming capabilities. A special monitoring board on each SHRIMP node is used to captures information about incoming packets.

The SMiLE[15] project (Shared Memory in a LAN-like Environment) includes a hardware monitor[16] that is implemented as a second PCI card which can be added to each node. The monitor snoops an internal link on the SCI adapters in order to monitor all incoming and outgoing packets. It includes a counter module and also a ring buffer in which remote memory access information may be recorded. A software infrastructure has been constructed for extracting

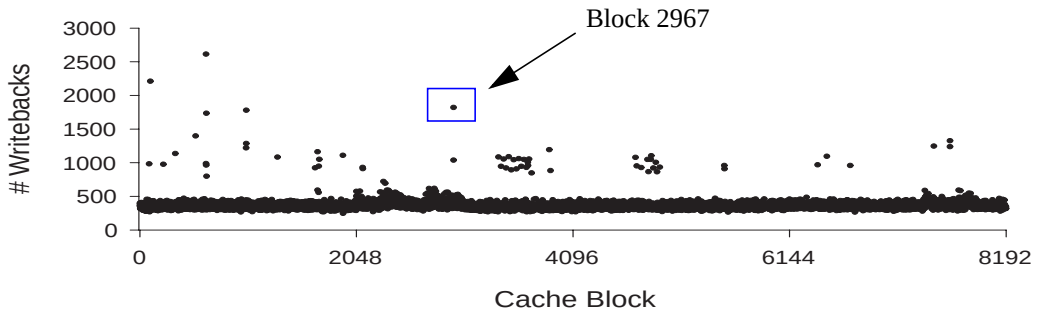

FIGURE 4. Per-cache block writeback activity during SPEComp SWIM application (1 processor).

```
CacheBlock: 2967        Writebacks: 1822
Writebacks by Page:
Physical Page       # WBs        Virtual Page
0x4096172000        5              127EBE000
0x4097172000        2              1276C2000
0x4097572000        3              11E30A000
0x4097d72000        2              14D98E000
0x4098d72000     1385       FFFFFFFF7FFFE000
0x4099972000        2              1286BA000
0x4099d72000        2              15395E000
0x409a572000        4              111372000
...

CacheBlock: 2967
Writebacks by Address:
Physical Address   #WB        Physical Page   Virtual Page
0x4098d72e00      401         0x4098d72000    FFFFFFFF7FFFE000
0x4098d72e40      360         0x4098d72000    FFFFFFFF7FFFE000
0x4098d72e80      613         0x4098d72000    FFFFFFFF7FFFE000
0x4098d72ec0        2         0x4098d72000    FFFFFFFF7FFFE000
0x4098d72f00        4         0x4098d72000    FFFFFFFF7FFFE000
0x4098d72f40        2         0x4098d72000    FFFFFFFF7FFFE000
0x4098d72f80        2         0x4098d72000    FFFFFFFF7FFFE000
0x4098d72fc0        1         0x4098d72000    FFFFFFFF7FFFE000

Segment 64 Per Page Info for   [ stack ]
Vaddr              Paddr                 Bank
---------------   ------------------   ----------------
FFFFFFFF7FFF6000   0x000000209c56a000   /N0/SB2
FFFFFFFF7FFF8000   0x00000000cad6c000   /N0/SB1
FFFFFFFF7FFFA000   0x00000020ab56e000   /N0/SB2
FFFFFFFF7FFFC000   0x000000409e570000   /N0/SB5
FFFFFFFF7FFFE000   0x0000004098d72000   /N0/SB5
```

FIGURE 5. Detailed information describing activity from cache block 2967 in Figure 4.

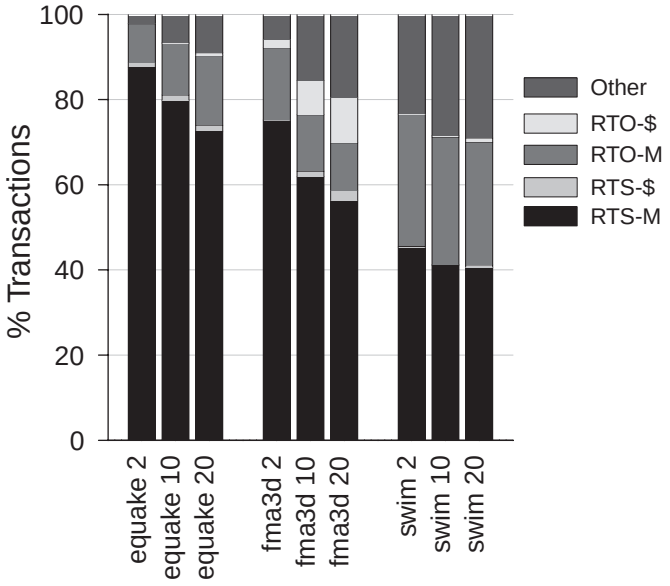

FIGURE 6. Cache-to-cache and memory-to-cache transfers. This figure details the percent of RTS and RTO transactions that resulted in cache-to-cache (RTS-$, RTO-$) transactions, as opposed to transfers from main memory (RTS-M, RTO-M). Results are shown for runs of 3 specOMP benchmarks run using 2, 10, and 20 threads.

meaningful information from the low-level data provided by the performance monitor, including a mechanism for physical to virtual address translation. Additional work has been done in the area of tools for visualizing this data[17].

A thorough discussion of techniques for trace-driven analysis of memory simulation may be found in[1], which also discusses various trace collection and reduction techniques.

6 Conclusions

This paper presents a hardware efficient set of integrated performance instrumentation for the system interconnect in SMP systems. This instrumentation includes a general transaction filtering mechanism, and a set of histogramming counters and transactions sequence buffers which operate on the filtered transaction stream. We also show how information from these hardware mechanisms can be interpreted by a set of performance tools to enable application and OS tuning, as well as architectural analysis.

As even “small” systems tend toward SMP designs (based, for example, on multi-core chip-multiprocessors), these types of hardware mechanisms and tools will become increasingly important in achieving high overall system performance.

Acknowledgments

We would like to thank Eugene Loh, John Sopka, Bruce Hatton, Becky Gill, and Andy Boughton for their comments on early versions of this paper. We would also like to thank Matt Mergener, who developed and provided assistance using the transaction profiling tool; Josh Simons for his suggestion that we use the transaction sequence buffers to explore cache behavior; Mike Johnson for his work gathering data for this paper; Alan Charlesworth for supplying the Sun Fire Server Architecture diagram; and Andy Boughton, Hien Nguyen, and Don Morrier for their expertise and willingness to always answer questions on the use of these counters.

We would also like to acknowledge our colleagues, Hien Nguyen, Monica Wong-Chan, Erik Hagersten, and Anders Landin for their contributions to the design of this performance instrumentation.

References

[1] R. A. Uhlig and T. N. Mudge, “Trace-Driven Memory Simulation: A Survey,” *ACM Computing Surveys*, Vol. 29, No. 2, June 1997.

[2] P. S. Magnusson, M. Christensson, J. Eskilson, D. Forsgren, G. Hållberg, J. Högberg, F. Larsson, A. Moestedt, B. Werner, "Simics: A Full System Simulation Platform," *IEEE Computer*, February 2002, pp. 50-58.

[3] S. J. Sistare, C. J. Jackson, “Ultra-High Performance Communication with MPI and the Sun Fire Link Interconnect,” Proceedings of Supercomputing 2002, Nov. 2002.

[4] A. Charlesworth, “The Sun Fireplane system interconnect.” In *Proceedings of SC2001*, Nov. 2001.

[5] .V. Aslot, M. Domeika, R. Eigenmann, G. Gaertner, W. B. Jones, and B. Parody, “SPEComp: A new benchmark suite for measuring parallel computer performance,” *Workshop on OpenMP Applications and Tools, WOMPAT 2001, Lecture Notes in Computer Science*, vol. 2104, pp. 1-10, 2001.

[6] D. H. Bailey, E. Barszcz, J. Barton, D. Browning, R. Carter, L. Dagum, R. Fatoohi, S. Fineberg, P. Frederickson, T. Lasinski, R. Schreiber, H. Simon, V. Venkatakrishnan,and S. Weeratunga, “The NAS Parallel Benchmarks,” Technical Report RNR-94-007, NASA Ames Research Center, 1994.

[7] “busstat(1M)”, Solaris 8 Reference Manual Collection, man pages section 1M: System Administration Commands. Sun Microsystems. [Online]. Available: http://docs.sun.com

[8] B.R.Buck and J. K. Hollingsworth, “Using hardware performance monitors to isolate memory bottlenecks,” *Proceedings of Supercomputing 2000*, Nov. 2000.

[9] J. Hughes and D. Cronshaw, “On using a hardware monitor as an intelligent peripheral,” Performance Evaluation Review, Vol. 2, No.4, December 1973.

[10] J. S. Emer and D. W. Clark, “A characterization of processor performance in the VAX-11/780,” in *Proceedings of the 11th Annual International Symposium on Computer Architecture*, 1984, pp. 301-310.

[11] D. W. Clark, P. J. Bannon, and J. B. Keller, “Measuring VAX 8800 performance with a histogram hardware monitor,” in *Proceedings of the 15th Annual International Symposium on Computer Architecture*, 1988, pp. 176-185.

[12] D. Lenoski, J. Laudon, K. Gharachorloo, W. Weber, A. Gupta, J. Hennessy, M. Horowitz, M. Lam, “The Stanford Dash Multiprocessor”, *IEEE Computer*, Vol. 25, No. 3, March 1992.

[13] D. Lenoski, J. Laudon, T. Joe, D. Nakahira, L. Stevens, A. Gupta, J. Hennessy, “The DASH Prototype: Logic Overhead and Performance,” *IEEE Transactions on Parallel and Distributed Systems*, Vol. 4, No. 1, pp. 41-61, January 1993.

[14] M. Martinosi, D. W. Clark, and M. Mesarina, “The SHRIMP performance monitor: Design and applications,” in *Proceedings of SIGMETRICS Symposium on Parallel and Distributed Tools*, 1996, pp. 61-69.

[15] W. Karl, M. Leberecht, M. Schulz, “Supporting shared memory and message passing on clusters of PCs with SMiLE”, *CANPC ’99*, (together with *HPCA-5*), 1999. Published in *Lecture Notes in Computer Science*, vol. 1602, Heidelberg:Springer Verlag.

[16] R. Hockauf, W. Karl, M. Leberecht, M. Oberhuber, and M. Wagner, "Exploiting spatial and temporal locality of accesses: A new hardware-based monitoring approach for DSM Systems," in *Proceedings EuroPar ’98*, 1998.

[17] Jie Tao, Wolfgang Karl, and Martin Schulz, “Visualizing the Memory Access Features of Shared Memory Applications on NUMA Architectures,” in *Proceedings of the 2001 International Conference on Computational Science (ICCS)*, 2001, pp. 861-870.