Characterization of simultaneous multithreading (SMT) efficiency in POWER5

H. M. Mathis
A. E. Mericas
J. D. McCalpin
R. J. Eickemeyer
S. R. Kunkel

Coarse-grained multithreading, the switching of threads to avoid idle processor time during long-latency events, has been available on IBM systems since 1998. Simultaneous multithreading (SMT), first available on the POWER5™ processor, moves beyond simple thread switching to the maintenance of two thread streams that are issued as continuously as possible to ensure the maximum use of processor resources. Because SMT has the potential of increasing processor efficiency and correspondingly increasing the amount of work done for a given time span, the reader might suppose that SMT would exhibit a performance gain for all workloads. This is true for most workloads, but is not true in some exceptional cases. In SMT mode, the processor resources—register sets, caches, queues, translation buffers, and the system memory nest—must be shared by both threads, and conditions can occur that degrade or even obviate SMT performance improvement. The POWER4™ and POWER5 processors have very powerful performance monitor (PM) toolsets that can help the user to determine what is occurring in workloads that may not be providing expected SMT gains. In this paper, the results of measured differences among workloads having large, medium, small, and even negative SMT performance gains are presented along with an approach to investigating workloads to determine the source of SMT performance gain limits.

Introduction

Over the years, the size of POWER*-based server systems has continued to grow. From the uniprocessor systems first shipped in 1990 to the 64-processor POWER5* systems shipping today, the underlying computational building block is the processor core. Major changes in the approach to processor design have fueled this growth. One approach to improving computational performance was to increase the number of *threads*, or independent execution sequences, that a processor could execute concurrently. In 1998 IBM introduced the first multithreaded POWER processor, the RS64-II [1, 2]. The RS64-II chip contains a single processor running two threads, in what is called *coarse-grain multithreading*.

IBM POWER-based systems contained only single-processor chips [1, 3] until 2001, when IBM introduced POWER4*, the first POWER chip containing two single-thread processors [4]. The POWER5 chip [5, 6], introduced in 2004, combines the design concepts of multiple cores and multiple threads. Each chip contains two processors, each running two threads, for a total of four threads per chip. These technological changes have been facilitated by continuously increasing transistor densities, in addition to software improvements that allow scaling to larger numbers of processors and threads.

To understand the multithreaded design of POWER5 better, it would be useful to examine the designs of its predecessors, the RS64-II and the POWER4. One important factor in the design of the multithreaded RS64-II was an increasing concern over rising cache-miss rates driven by new types of applications and languages. At the same time miss rates were rising, the latency of

0018-8646/05/$5.00 © 2005 IBM

cache misses was increasing [7]. As a result, cache misses had begun to make up an increasingly larger portion of execution time. To minimize the amount of processor time wasted on cache misses, the thread mechanism in the RS64-II processor was designed to switch when one was detected. In this way, cache misses could be overlapped between the two threads, resulting in greater system throughput. This is termed coarse-grain multithreading. In general, multithreading can cause cache-miss rates to increase, but the overlap is usually more than enough to overcome this and still achieve significant increases in throughput [8].

POWER4 introduced the basic IBM processor design for out-of-order execution, and this design is continued with POWER5. In this type of design, it is more difficult to switch threads on a cache miss, because at the time of the miss some earlier instructions may not have been performed, while some later instructions may already have been completed. The problem becomes where and how to stop one thread, leaving the thread and its resources in a state that will allow it to be restarted when the switch is made to the other thread. Simultaneous multithreading (SMT) is a way of implementing multiple threads on an out-of-order processor [9–11]. In this design, all threads are simultaneously active so that there is no thread-switch event such as occurs in coarse-grain multithreading. When one thread has a cache miss, the other thread can continue to execute. Because there is no thread-switch overhead, SMT can hide even short-duration stalls in the execution pipeline. If, because of pipeline latency, an instruction from one thread is delayed waiting for a result, or if, because of the misprediction of a branch, a portion of a thread's instructions have been flushed from the execution pipeline, instructions from the other thread can continue to execute. Events such as register dependencies frequently prevent a thread from utilizing the entire processing pipeline to the fullest extent. Such interludes provide an opportunity for the other thread to execute on functional units that would otherwise be idle during a particular cycle.

Along with improvements to POWER processor design came improvements to the performance-monitoring hardware. The POWER5 microprocessor, like its forerunners back to the PowerPC 604* microprocessor, includes on-chip logic to monitor, record, and report key performance events and latencies. The PPC604 performance monitor unit (PMU) contained two counters, and the PPC604e added two more. The POWER3* and POWER4 each contained eight counters. The RS64 processor, which introduced hardware multithreading, contained eight counters, but they had to be shared between two threads. The POWER5 implementation of SMT extends performance-monitoring functionality to the thread level. Each thread has a dedicated six-counter PMU that can be independently configured to monitor more than 300 performance events occurring on the processor or memory system.

At the same time that microprocessor architecture has been evolving, workload characteristics have been changing. The workload mix now includes some traditional commercial workloads that have high cache-miss rates, and important new workloads have been added that have lower cache-miss rates than some of the more traditional commercial workloads [12, 13]. Both types of workloads usually have other pipeline stalls. SMT provides increased throughput in the presence of both cache misses and other pipeline stalls, thereby benefiting multiple classes of workloads.

This paper thus focuses on the performance analysis of thread execution in the POWER5 processor and investigates how a selection of workloads behave on an SMT processor. We begin describing some details of the POWER5 design for SMT and describe how SMT performance is measured. We then describe a set of performance-monitor measurements on POWER4 and POWER5 to demonstrate SMT performance effects. Finally, we summarize factors that affect SMT gain.

POWER5 SMT design and performance

The POWER5 design for SMT recognized that in some cases processor execution pipeline resources that were ample for a single-threaded environment, such as POWER4, would not provide adequate performance for SMT environments and thus would have to be increased and shared or duplicated [6]. Other resources were adequate for both environments but would have to be split in order to make SMT work without increasing processor chip size. Additionally, resources that would have been suitable for POWER4 operation had to be expanded to include bits for thread identification in order for those resources to support SMT.

The following is a brief summary of the design changes made to POWER4 and POWER5 processor resources to support SMT performance. The branch information queue (BIQ) is unchanged in size but is split between the threads. The load reorder queue (LRQ) and store reorder queue (SRQ) are split, but extra virtual entries have been added [6]. Register-renaming resources have been increased to support the additional register requirements of the second thread. The load miss queue (LMQ) is shared, since there is no ordering information among the entries. The global completion table (GCT) had been shared, but since ordering must be maintained within a thread, the design was changed. Parts of the design, such as the branch history table (BHT), have required no changes for SMT. Finally, the associativities of the instruction cache, the data cache, and the data effective-to-real address translation (ERAT) table have been

increased to reduce the amount of cache-line conflict resulting from SMT operation.

Tables containing side-by-side comparisons of the design changes are provided in the analysis sections. The analysis sections containing these comparisons are those on the instruction fetch unit (IFU), instruction decode unit (IDU), instruction sequencing unit (ISU), and load store unit (LSU).

Performance testing during the development cycle

Two sets of assembler-like performance verification program (PVP) test cases were first run on the POWER5 chip shortly after it was powered on in the bring-up laboratory: one set for single-threaded (ST) mode and one set for SMT mode. (In ST mode, only one thread is running, and it can use many of the resources that would normally be reserved for the second thread [6].) These test cases were designed to determine the number of processor cycles required to complete a fixed sequence of instructions, and to validate that the chip operated as specified in the design document. Test cases ranged from as few as one instruction to tens of thousands of instructions. The test cases had already been run on performance models, and this testing provided proof that both the models and the chips executed the instruction sequences using the number of cycles specified in the design. In addition, the results made it possible to determine the "gain" that SMT provided over ST for each test case. The SMT gain was calculated by the following algorithm:

1. $CPI_{\mathrm{SMT}} = (cycles_{\mathrm{SMT}}/instructions_{\mathrm{SMT}})$.
2. $CPI_{\mathrm{ST}} = (cycles_{\mathrm{ST}}/instructions_{\mathrm{ST}})$.
3. $SMT\ gain = 100(2CPI_{\mathrm{ST}}/CPI_{\mathrm{SMT}}) - 100$.

During the early laboratory tests it became apparent that some test cases yielded much higher SMT gain than others. In general, the integer test cases were very simple and yielded higher SMT gains than the floating-point test cases, which were heavily biased toward larger, tightly coded loops.

Although the test-case results were of interest and were subject to considerable analysis, they did not by themselves provide sufficient data to predict whether or how much SMT gain could be realized from a specific workload consisting of hundreds of billions of instructions of all types. Performance for major workloads had been predicted by performance models; now it was time to see how the hardware would perform. Once a laboratory version of Linux** could be booted up, actual benchmarks were run in both ST and SMT modes. Benchmark results from the earliest versions of the chip showed that, in general, integer workloads yielded better SMT gains, while floating-point-intensive workloads yielded smaller SMT gains and even, in some cases, negative ones. This was supported by results from earlier PVP runs. Although some applications appeared to be less amenable and some more amenable to SMT, it was difficult without sufficient data to conclude which application characteristics were favorable and which were not, and which shared resources were most heavily affected by SMT. At the time, however, the version of Linux used for initial testing lacked the capability to make use of the extensive performance-monitoring facilities of the POWER5 processor. Analysis of the workloads had to wait until the laboratory was able to boot AIX*, which could collect data on 127 different groups of processor events.

AIX was booted on several different POWER5 systems once firmware and hypervisor code stabilized. A wide variety of benchmark applications were run, both floating-point and integer, as each new version of the POWER5 chip became available. As in the POWER2*, POWER3, and POWER4 development processes, the hardware performance team collected performance measurements with a set of AIX programs whose binaries remained constant through the bring-up period. These programs formed the regression tests that were used to track performance improvements as chip revisions became available.

The inclusion of SMT capabilities in the POWER5 chip necessitated adding ST–SMT gain measurements to the hardware performance process. ST–SMT gain measurements were accomplished using a two-step process:

1. A copy of a program was bound to each of the physical processors running in ST mode, and the time (T_{ST}) the programs took to complete their work was measured.
2. The system was switched to SMT mode and a copy of the program was bound to each of the two SMT threads (which are also known as logical processors) on each of the two physical processors; again, the amount of time (T_{SMT}) it took both programs to complete was measured. To the operating system each thread appeared as a logical processor.

Each program does a unit of work (W) during its processing, so in the ST case, the amount of work done per unit time by two programs, one on each physical processor, is $2W/T_{\mathrm{ST}}$; in the SMT case, the amount of work done per unit time by four programs, one on each of the two threads on each physical processor, is $4W/T_{\mathrm{SMT}}$. The SMT gain was computed using the following formula:

Table 1 Workloads selected for the study.

Workload	*Computation type*	*SMT gain (%)*
Sentence passing	Integer	41.2
Data compression	Integer	38.6
Programming language	Integer	26.3
3D Multi-grid Solver	Floating-point	21.6
Circuit Routing	Integer	19.8
Seismic Wave Simulation	Floating-point	15.3
Object-oriented Database	Integer	12.5
Neural Network	Floating-point	11.2

$$Gain\,(\%) = 100[(4W/T_{\mathrm{SMT}})/(2W/T_{\mathrm{ST}})] - 100,$$

where W is the number of instructions executed, T_{SMT} is $W\,CPI_{\mathrm{SMT}}/P$, T_{ST} is $W\,CPI_{\mathrm{ST}}/P$, and P is processor frequency in cycles per second. Thus, we obtain the same formula used earlier with PVPs:

$$Gain\,(\%) = 100(2CPI_{\mathrm{ST}}/CPI_{\mathrm{SMT}}) - 100.$$

It was observed that commercial workloads showed higher SMT gain than the computationally intensive, tightly looped applications often found in high-performance computing environments.

Data collection and reduction

This paper analyzes the SMT performance of eight workloads selected on the basis of their previously calculated SMT gain behaviors throughout the POWER5 chip development cycle. These workloads, along with their respective SMT gains calculated during the data collection runs, are shown in **Table 1**.

The instruction streams for the three floating-point workloads, 3D Multi-grid Solver, Seismic Wave Simulation, and Neural Network, respectively contained 66%, 50%, and 40% floating-point instructions. The instruction streams for two integer workloads, Programming Language and Circuit Routing, respectively contained 1% and 9% floating-point instructions, and the other integer workloads had no floating-point instructions. The hardware used in data collection was a 1.65-GHz single-chip dual-processor POWER5 system having a 1.9-MB L2 cache, a 36-MB L3 cache, and 16 GB of memory. Each ST data collection run was obtained by executing two copies of the same workload concurrently, one on each of the single threads on each processor core. Each SMT data collection run was obtained by executing four copies of the same workload concurrently, one on each of the two SMT threads on each processor core. Performance monitor runs were completed for each of the workloads for each of the 127 POWER5 PMU counter event groups.

The resulting data was reduced using SAS**, a statistical analysis program, and then entered into spreadsheets. Results were sorted by processor event group and by the mean value for each metric. The POWER4 data collection effort was exactly like that on the POWER5 system in ST mode; that is, two copies of the same workload were run concurrently on a single-chip, two-core POWER4 system, and data was collected for the 63 POWER4 PMU counter event groups to compare with that collected from the POWER5 system. The data collected from POWER5 contained nearly twice as many data points as that from POWER4, since there are nearly twice as many PMU counter groups for POWER5 as there are for POWER4. The POWER4 data was reduced using SAS and entered into a spreadsheet; as before, it was sorted by processor event group and mean value for the metric, and then merged with the POWER5 data where possible. For many event groups, it was readily apparent that the data values were randomly distributed among the high-, medium-, low-, and negative-gain applications, and had no relationship to the SMT gain of a workload. The sorting order enabled a quick analysis to determine when the negative-gain workload appeared at either low or high values of a measurement for a processor event group, making it a good candidate for further analysis. The sorting also enabled the same quick analysis for high-gain and medium-gain workloads in order to narrow down quickly the number of processor event groups to be considered. This facilitated, for the purpose of this study effort, the selection of data displaying only a direct or an inverse relationship to the degree of SMT gain.

Analysis of the data

The first step in analyzing the data was to narrow down the 127 groups of POWER5 and the 63 groups of POWER4 performance-counter results by eliminating groups that did not contribute to the study. That left 53 POWER4 groups and 115 POWER5 groups, and many of these, such as counts of special-purpose instructions not used by any of the applications, had little or no importance for this paper. Analysis then proceeded with an examination of processor resources, by unit, that were either duplicated or increased in size and shared or split, and how these resources affected POWER5 SMT performance.

Instruction fetch unit (IFU)

Table 2(a) lists the changes that were made to the IFU from POWER4 to POWER5. Instruction prefetch hit ratios were in the range of 0.5 to 0.7. Prefetch miss rates were less than 0.22% for all workloads, indicating the effectiveness of instruction prefetch logic. The data

showed little difference between the ST and SMT modes and provided no evidence of a relationship to SMT gain. Since this event was not collectable on POWER4 PMU, there was no data available to compare with the POWER5 data.

In both POWER4 and POWER5, the branch-prediction mechanism enables speculation, which is the execution of an instruction stream beyond an unevaluated branch instruction. The branch instruction is unevaluated at that temporal point because the data required to evaluate it is not yet available. Without speculation, execution of the instruction stream would be stalled until it could be evaluated. Speculation permits execution to continue on the basis of an "educated guess" by the processor as to which branch path will be taken. When the branch instruction is actually evaluated, it may be the case that the wrong path was taken for speculation; this is referred to as a branch misprediction. After a branch misprediction is discovered, all instructions following the branch are flushed from the execution pipeline and execution begins again along the correct path.

Branch-misprediction flush-rate data values were low and were not significantly different whether the system was in ST mode or SMT mode. This indicates that for POWER5, at least for this set of workloads, no change was needed in the branch hit table to support SMT. Since this event was not collectable on POWER4 PMU, there was no POWER4 BHT data available to compare with the POWER5 data.

Instruction decode unit (IDU)

The instruction fetch buffers (IFBs) hold instructions for group formation prior to dispatch. As shown in **Table 2(b)**, each POWER4 processor core contains a single eight-entry IFB, while the POWER5 processor core contains two six-entry IFBs, one for each thread. The data revealed that the workloads having moderate to high SMT gain also had lower average IFB usage than those workloads with low or negative SMT gains for both the POWER4 and POWER5 systems. POWER4 data was available for comparison, and for both ST and SMT modes the POWER5 system had lower average IFB usage values than the POWER4 system for all workloads except Seismic Wave Simulation, where there was a minimal difference. Data collected with regard to the percentage of time the IFBs were not empty revealed that the POWER4 and POWER5 ST mode behaviors were similar for these workloads. The 3D Multi-grid Solver workload, at the low end of moderate SMT gain and having the smallest number of cycles per instruction for both ST and SMT modes, had the highest utilization.

Data collected on the percentage of time that the IFB contained six fetch groups revealed that the POWER4 and POWER5 ST mode IFB behaviors were similar for these workloads. When the IFB contains six fetch groups, it is full in the case of the POWER5 and it is not full in the case of the POWER4. For both systems the floating-point workloads had the highest percentages for this metric, and for POWER5 the workload having negative SMT gain had the highest percentage for this metric.

Table 2 Comparison of POWER4 and POWER5: (a) IFU resources; (b) IDU resources.

POWER4 (ST only) resource	*POWER5 (ST and SMT) resource*
(a)	
Direct-mapped 64-KB Level 1 instruction cache (L1 I-cache)	Two-way 64-KB L1 I-cache
Four-entry direct-mapped prefetch buffer	Split, two-entry-per-thread prefetch buffer
16-entry branch issue queue (BIQ)	Split, eight-entry-per-thread BIQ
Branch-prediction control	Replicated branch-prediction control
Link stack	Replicated link stack
(b)	
Eight-entry instruction fetch buffer (IFB)	Six-entry IFB per thread

Table 3 Comparison of POWER4 AND POWER5 ISU resources.

POWER4 (ST only) resource	*POWER5 (ST and SMT) resource*
20-entry first-in first-out (FIFO) global completion table (GCT)	20-entry linked-list GCT
80 general-purpose register (GPR), 72 floating-point register (FPR) mappers used to map virtual to real registers	120 GPR, 120 FPR mappers
32-entry condition register (CR) mapper	40-entry CR mapper
24-entry fixed-point exception register (XER) mapper	32-entry XER mapper
20-entry floating-point issue queue (FPQ)	24-entry FPQ

Instruction sequencing unit (ISU)

Table 3 lists resource changes for the instruction sequencing unit from POWER4 to POWER5. In

the POWER5 ST mode, all twenty entries in the GCT are used by one thread, but in SMT mode they are dynamically shared between the two threads using a linked list to relate entries to the thread owners. For the workload having negative SMT gain, the POWER5 GCT was full significantly more of the time in both ST and SMT modes. The percentage of time the GCT was full for this workload in SMT mode was nearly double that for ST mode, reflecting the effect of sharing the twenty GCT entries.

The number of GPR mappers was increased from 80 in POWER4 to 120 in POWER5, and the number of FPR mappers was increased from 72 to 120. Rename mappers in the instruction sequencing unit guarantee each thread 36 GPRs and 32 FPRs, and all are dynamically allocated as required. This enables each POWER5 processor in ST mode to support more fixed-point and floating-point instructions in flight than the POWER4 processor can. These changes had a significant effect on the percentage of time the GPR and FPR mappers were full compared with the POWER4 runs. The GPR mappers were never full during POWER5 ST runs and, for seven of the eight workloads, the POWER5 SMT runs had lower rates of the GPR mappers being full than the POWER4 runs. The highest rates for the GPR mappers being full were for all of the five integer workloads, and the lowest were for the three floating-point workloads.

The highest rates for the FPR mappers being full were for the three floating-point workloads, and these rates were zero for all of the integer workloads. The POWER4 runs had higher FPR mapper full rates than the POWER5 ST runs, presumably because the POWER5 in ST mode has a much larger FPR rename pool. Somewhat surprising was the fact that the POWER4 runs also measured significantly higher FPR mapper full rates than the POWER5 SMT runs, a testament to the effectiveness of POWER5 register-renaming logic.

The number of condition register mappers was increased from 32 in POWER4 to 40 in POWER5. Only one workload, Object-oriented Database, a low-gaining integer workload, managed to fill the CR mappers in ST mode, and then only 1% of the time. For POWER5 in SMT mode, CR mapper utilization was very similar to that for POWER4. For the POWER5 system in SMT mode, five workloads had CR mappers full 2% to 6% of the time; for the POWER4 system, the same five workloads had CR mappers full 1% to 8% of the time.

The POWER5 ISU contains two 18-entry issue queues for fixed-point and load/store instructions, two 12-entry issue queues for floating-point instructions, one 12-entry issue queue for branch instructions, and one 10-entry issue queue for CR-logical instructions. The branch issue and the condition register queues both were never full for any of eight workloads.

The fixed-point queues had highest utilizations for fixed-point workloads and lowest utilizations for floating-point workloads. POWER4 data was available for fixed-point queue 0, but data for queue 1 was not collected. The POWER4 data had slightly lower values than both POWER5 ST and SMT data except for sentence parsing. Showing consistency, the floating-point queues had the highest utilizations for floating-point workloads and the lowest utilizations for fixed-point workloads. The floating-point queues for POWER4 and POWER5 ST and SMT modes had nearly identical data patterns.

Fixed-point execution (FXU) pipelines

The POWER5 processor contains two fixed-point execution pipelines, and both are capable of multiplication and basic arithmetic, logical and shifting operations. One pipeline is additionally capable of division. Instructions are issued out of order with a bias toward oldest operations first, and there is symmetric forwarding between fixed-point and load/store execution pipelines. Not surprisingly, the rates at which both FXUs were busy for both the POWER4 and the POWER5 in ST mode were very similar in all cases. The rates for both the POWER4 and the POWER5 were highest for the five integer workloads, and were very low or zero for the floating-point workloads. The POWER5 SMT rates were less than half of either of those for POWER4 and POWER5 ST mode for the integer workloads.

This phenomenon appears to result from the fact that speculation is treated differently in SMT mode than in ST mode. In SMT mode a thread does not execute as many instructions speculatively beyond an unevaluated branch instruction as it does in ST mode. As a result, fewer instructions are discarded when the branch instruction is evaluated and a branch misprediction is found, and this makes the pipelines appear less busy in SMT mode.

Floating-point unit (FPU) pipelines

The POWER5 processor contains two six-stage floating-point execution pipelines. Both are capable of executing the full set of floating-point instructions, and instructions are issued out of order with a bias toward the oldest instructions. Both IEEE and non-IEEE instruction modes are supported. Floating-point unit utilization is directly related to the percentage of floating-point instructions in the instruction mix.

Floating-point queues had significantly higher rates of being full for the negative SMT gain workload, Neural Network, for the POWER5 in both ST and SMT modes, and for the POWER4, than any other of the workloads. For Neural Network, the full rate for floating-point queue 0 was 54% for both POWER4 and POWER5 in ST mode and 59% for POWER5 in SMT mode. For floating-

point queue 1, the rate was respectively 23%, 26%, and 24% for the POWER4, POWER5 in ST mode, and POWER5 in SMT mode, indicating a dispatch bias toward floating-point queue 0.

Load/store unit (LSU) execution pipelines

The POWER5 design provides two six-stage load/store execution pipelines to handle load/store instructions. Loads and stores are executed in three stages, out of order, with a bias toward the oldest operations first. Stores issue twice, with an address-generation operation and a data-steering operation. Resources shown in **Table 4** are integral to the load/store function.

Data was collected on all of the above caches and queues for each of the eight workloads. Although data was collected from the POWER5 system on cache performance, no POWER4 data was available for comparison. Data available for both POWER4 and POWER5 was available for D-ERAT performance. The POWER5 system in ST mode had lower D-ERAT miss rates on five of the eight workloads than did the POWER4, equal on two of the workloads, and slightly higher on one workload, Neural Network, the workload with negative SMT gain. The POWER5 in SMT mode had lower D-ERAT miss rates on four of the workloads than did the POWER4 system, equal on two and slightly higher on two workloads, Neural Network and Circuit Routing, a workload with moderate SMT gain.

The data collected on the percentage of time the LMQ was full revealed that none of the workloads taxed the LMQ capacity for either POWER4 or POWER5 systems. The data collected on the time an entry spends in the LMQ showed a slight increase as the result of two SMT threads sharing the LMQ. For the POWER5 system in SMT mode, the LMQ was very close to that of the POWER4 system, although there was some variation among the workloads. The negative-gain workload had the greatest LMQ time for both systems and both modes.

None of the workloads significantly taxed the LRQ capacities of either the POWER4 or POWER5 systems. For five of the eight workloads, the POWER4 LRQs were full from 1% to 32% of the time. In ST mode, the POWER5 LRQs were never full. In SMT mode, the LRQs were full for a short time (4% and 7% of the time), but only for 3D Multi-grid Solver and Neural Network. The POWER5 SMT mode LRQ time behavior appeared to have benefited from larger L2 and L3 caches, lower cache latency, and the addition of 32 virtual LRQ entries. It is similar to that for both POWER5 in ST mode and POWER4. The largest value of LRQ time was measured for the negative-gain workload on the POWER5 in ST and SMT modes as well as on the POWER4.

Similarly, neither of the systems' SRQ capacities was taxed significantly by any of the eight workloads. The POWER4 SRQ was full from 1% to 4% of the time for seven of the eight workloads, while the POWER5 SRQ was full in ST mode 1% of the time and in SMT mode 3% of the time, but only for Seismic Wave Simulation. The SRQ data closely paralleled that for the LRQ. The greatest SRQ times were measured for the floating-point workloads, particularly the workload showing negative SMT gain, regardless of system or mode.

The data clearly showed that the LSU had higher utilization in SMT mode. The ST and POWER4 behaviors were very similar, but the highest utilization in all cases was for the negative-gain workload.

Table 4 Comparison of POWER4 and POWER5 LSU resources.

POWER4 (ST only) resource	*POWER5 (ST and SMT) resource*
32-KB two-way set-associative level-1 data cache (L1 dcache)	32-KB four-way set-associative L1 dcache
128-entry two-way effective-to-real address translation (ERAT) table	128-entry fully associative ERAT table
64-entry segment lookaside buffer (SLB)	Replicated 64-entry SLB per thread
32-entry real load reorder queue (LRQ)	16-entry real and 16-entry virtual LRQ per thread
32-entry real store reorder queue (SRQ)	16-entry real and 16-entry virtual SRQ per thread
Eight-entry load miss queue (LMQ)	Eight-entry LMQ with thread control
Interrupts	Interrupts replicated per thread
	Replicated special-purpose registers (SPRs) with thread ID
1.45-MB level-2 cache (L2) on-chip	1.9-MB L2 on-chip
16-MB level-3 cache (L3)	36-MB L3, directory and controller on processor chip

Branch and condition register (CR) pipeline

The POWER5 branch and condition register (CR) pipeline architecture is essentially unchanged from the POWER4 design; however, both threads share the CR in SMT mode. In none of the instances were either POWER4 or POWER5 branch queues full.

The CRs for POWER4 and POWER5 in SMT mode had similar behaviors, with CRs being full a small percentage of the time for five of the eight workloads. The POWER5 in ST mode had the CR full for only one workload, and then for only 1% of the time.

Table 5 Sources and percentages of data from each source by workload and mode.

Workload	*Mode*	*Data source*		
		L2 cache (%)	*L3 cache* (%)	*Main memory* (%)
Sentence parsing	ST	94	6	0
Sentence parsing	SMT	92	8	0
Data compression	ST	100	0	0
Data compression	SMT	100	0	0
Programming language	ST	94	5	0
Programming language	SMT	97	3	0
3D Multi-grid Solver	ST	95	2	3
3D Multi-grid Solver	SMT	88	6	6
Circuit Routing	ST	79	20	1
Circuit Routing	SMT	72	27	1
Seismic Wave Simulation	ST	86	6	8
Seismic Wave Simulation	SMT	82	4	14
Object-oriented Database	ST	85	14	1
Object-oriented Database	SMT	88	11	1
Neural Network	ST	50	50	0
Neural Network	SMT	49	51	0

Discussion of factors affecting SMT gain

Data collected on the percentage of time the FPR mappers were full indicated that for the floating-point workloads, the FPR remapper resources were strained at least some of the time. Consequently, the team examined additional data relating to the floating-point characteristics of these workloads. Looking for the percentage of floating-point multiply–add (FMADD) fused instructions in the instruction mix, the team found that they were respectively 9%, 16%, and 17% for 3D Multi-grid Solver, Seismic Wave Simulation, and Neural Network. Each FMADD instruction requires four floating-point registers. Even if the FMADD instructions are uniformly distributed in the instruction stream, one can expect to see a heavy use of the FPR mapper in SMT mode for Seismic Wave Simulation and Neural Network. They are likely to be processed in clumps, and hitting the maximum number of renames can cause stalls. Nonetheless, the fact that the difference in FMADD percentages between Seismic Wave Simulation and workload 8 were insignificant, and the fact that one had a positive gain and the other had a negative gain appeared to rule out FMADDs as a factor. The team decided to take a detailed look at five of the lowest-SMT-gain workloads: 3D Multi-grid Solver, Circuit Routing, Seismic Wave Simulation, Object-oriented Database, and Neural Network.

The floating-point workload, 3D Multi-grid Solver, traverses the surface and interior points of a cube that requires 16 MB of storage. The footprint for two copies of the workload will fit in the L2 cache. Running four copies of the workload on two POWER5 cores in SMT mode requires double the footprint, and the data for four copies will nearly but not completely fit in the L2 cache. For SMT runs, this results in the workload having to obtain data from the L3 cache and main memory, both of which have greater latency than the L2 cache. The data revealed that 70% of the CPI difference between ST and SMT modes came from load/store-related activities, mostly due to L1 data cache (dcache) misses taking longer, approximately 50% of the extra CPI. As shown in **Table 5,** the percentages of data obtained from L2, L3, and memory in ST mode are respectively 95%, 2%, and 3% and these change to 88%, 6%, and 6% respectively when running in SMT mode. Despite the extra load latencies, the workload exhibits a 20.6% SMT gain. Circuit Routing is an integer workload. For this workload, 92% of the changes in the CPI between ST and SMT runs result from stalls; 83% of the changes in the CPI result from LSU activities, such as dcache misses and ERAT misses, and 5% of the changes result from branch misprediction. The percentages of data obtained from L2, L3, and memory in ST mode are respectively 79%, 20%, and 1%, and these change to 72%, 27%, and 1% respectively when running in SMT mode. This represents a shift of approximately 7% of the loads from the L2 to the higher-latency L3, and this was enough to reduce the SMT gain to less than 20%.

Seismic Wave Simulation is a finite-element simulation. LSU stalls can account for 70% of the changes in the CPI between ST and SMT modes, and 25% of the changes in the CPI are due to FPU stalls. The percentages of data obtained from L2, L3, and memory in ST mode are respectively 86%, 6%, and 8%, and these change to 82%, 4%, and 14% respectively when running in SMT mode. The additional load latencies required by the shifts of loads to L3 and memory were enough to reduce the SMT gain to 15.3%.

Object-oriented Database is an integer workload. Its CPIs, both ST and SMT, were comparatively small. Stalls account for 86% of the changes in the CPI, and 73% of the changes in the CPI result from LSU activities; 17% of the delta results from dcache misses, 17% from FXU activities, 15.7% from rejects, and 3% from branch misprediction. The percentages of data obtained from L2, L3, and memory in ST mode are respectively 85%, 14%, and 1%, and these changed to 88%, 11%, and 1% respectively when running in SMT mode. The percentage of rejects due to having no GCT slots available comprises

11% of the changes in the CPI, the highest for any of the eight workloads.

Neural Network is the only workload in the suite having negative SMT gain on the final POWER5 chip. Neural Network uses a 2-MB array and employs a very tight algorithm that takes large strides across this array. Since the L2 cache of the POWER5 chip is 1.9 MB, two copies of the relevant parts of the array may coexist satisfactorily in the L2, but four copies will not. The fact that four copies of the array footprint do not fit in the L2 cache, combined with the large strides across the data, ensures that cache lines will be replaced frequently, with correspondingly high L1 data-cache-miss rates, L2 miss rates, D-ERAT miss rates, and a high L3 hit rate. The team discovered that 94% of changes in the CPI from ST to SMT resulted from LSU activities: 73% dcache misses and stalls, 15% rejects, mostly due to TLB misses and increases in table walks, and 6% LSU other. The data showed that Neural Network had a huge change in the ERAT reject rate: 5% for ST and 71% for SMT. The negative effects of the LSU and ERAT rejects were the cause of the negative SMT gain. In the final analysis, although the workload was a floating-point workload, this had no bearing on the negative SMT gain.

Summary

The POWER5 is the first IBM chip to employ simultaneous multithreading (SMT). This feature maintains two thread streams that are issued as continuously as possible to ensure maximum use of processor resources. In some instances, implementing SMT required the provision of duplicate copies of some resources or larger queue sizes together with modified resource-allocation algorithms. A broad spectrum of workloads were examined that exhibited varying degrees of SMT gain, ranging from −11% to +41%. In all cases, the biggest single factor affecting SMT gain was the effect of load and store activities. Workloads having lower values for SMT gain tended to have higher load latencies resulting from 1) repeated invalidation and reloading of the same lines of L2 cache memory by each of the two programs running on a processor; 2) higher numbers of loads and stores rejected as a result; and 3) a significantly increased number of data loads from L3 cache and main memory because copies of the two programs' footprints would not fit into the L2 cache simultaneously. However, the study showed that most workloads, with few exceptions, benefited from operating in SMT mode.

References

1. J. M. Borkenhagen, R. J. Eickemeyer, R. N. Kalla, and S. R. Kunkel, "A Multithreaded PowerPC Processor for Commercial Servers," *IBM J. Res. & Dev.* **44**, No. 6, 885–898 (November 2000).
2. S. N. Storino, A. G. Aipperspach, J. M. Borkenhagen, R. J. Eickemeyer, S. R. Kunkel, S. B. Levenstein, and G. J. Uhlmann, "A Commercial Multithreaded RISC Processor," *Digest of Papers, International Solid-State Circuits Conference*, San Francisco, February 1998, pp. 236–237.
3. F. P. O'Connell and S. W. White, "POWER3: The Next Generation of PowerPC Processors," *IBM J. Res. & Dev.* **44**, No. 6, 873–884 (November 2000).
4. J. M. Tendler, J. S. Dodson, J. S. Fields, Jr., H. Le, and B. Sinharoy, "POWER4 System Microarchitecture," *IBM J. Res. & Dev.* **46**, No. 1, 5–26 (January 2002).
5. R. Kalla, B. Sinharoy, and J. M. Tendler, "IBM Power5 Chip: A Dual-Core Multithreaded Processor," *IEEE Micro* **24**, No. 2, 40–47 (March/April 2004).
6. B. Sinharoy, R. N. Kalla, J. M. Tendler, R. J. Eickemeyer, and J. B. Joyner, "POWER5 System Microarchitecture," *IBM J. Res. & Dev.* **49**, No. 4/5, 505–521 (2005, this issue).
7. W. A. Wulf and S. A. McKee, "Hitting the Memory Wall, Implications of the Obvious," *Computer Architecture News* **23**, No. 1, 20–24 (March 1995).
8. R. J. Eickemeyer, R. E. Johnson, S. R. Kunkel, M. S. Squillante, and S. Liu, "Evaluation of Multithreaded Uniprocessors for Commercial Application Environments," *Proceedings of the 23rd Annual International Symposium on Computer Architecture*, IEEE, Philadelphia, May 1996, pp. 203–212.
9. D. M. Tullsen, S. J. Eggers, and H. M. Levy, "Simultaneous Multithreading: Maximizing On-Chip Parallelism," *Proceedings of the 22nd International Symposium on Computer Architecture*, June 1995, pp. 392–403.
10. D. M. Tullsen, S. J. Eggers, J. S. Emer, H. M. Levy, J. L. Lo, and R. L. Stamm, "Exploiting Choice: Instruction Fetch and Issue on an Implementable Simultaneous Multithreading Processor," *Proceedings of the 23rd Annual International Symposium on Computer Architecture*, IEEE, Philadelphia, May 1996, pp. 191–202.
11. N. Tuck and D. M. Tullsen, "Initial Observations of the Simultaneous Multithreading Pentium 4 Processor," *Proceedings of the 12th International Conference on Parallel Architectures and Compilation Techniques*, IEEE, New Orleans, September 2003, pp. 26–35.
12. M. G. Maynard, C. M. Donnelly, and B. R. Olszewski, "Contrasting Characteristics and Cache Performance of Technical and Multi-User Commercial Workloads," *Proceedings of the Sixth International Conference on Architectural Support for Programming Languages and Operating Systems*, October 1994, pp. 145–156.
13. Z. Cventanovic and D. Bhandarkar, "Performance Characterization of the Alpha 21164 Microprocessor Using TP and SPEC Workloads," *Proceedings of the 21st Annual International Symposium on Computer Architecture*, April 1994, pp. 60–70.

Received May 30, 2004; accepted for publication April 14, 2005; Internet publication August 11, 2005

Harry M. Mathis *IBM Systems and Technology Group, 11400 Burnet Road, Austin, Texas 78758 (mathis@us.ibm.com).* Dr. Mathis served 21 years in the U.S. Air Force, retiring as a Lieutenant Colonel in 1985, before returning to graduate school. He received his Ph.D. degree from Texas A&M University in 1989 and then joined IBM in Westlake, Texas, where he worked in the datastream architecture and software performance areas until 1995. Following a two-year stint in the IBM Consulting Group serving business intelligence customers, he joined the IBM hardware performance team in Austin. Dr. Mathis has worked on performance tools and on POWER5 hardware bring-up; he is currently the lead performance member on the hardware bring-up team for a follow-on POWER-series processor.

Alex E. Mericas *IBM Systems and Technology Group, 11400 Burnet Road, Austin, Texas 78758 (mericas@us.ibm.com).* Mr. Mericas is a Senior Technical Staff Member in the IBM Systems and Technology Group. He received a B.S. degree in computer science from the University of New Orleans and a master's degree in computer engineering from National Technological University. He architected the performance instrumentation on POWER4, POWER5, and PowerPC 970* and was team leader for hardware performance on POWER4 and POWER5. Mr. Mericas currently works on future performance instrumentation designs, along with software tools to better exploit available hardware performance data.

John D. McCalpin *IBM Systems and Technology Group, 11400 Burnet Road, Austin, Texas 78758 (mccalpin@us.ibm.com).* Dr. McCalpin received a B.S. degree in physics and an M.S. degree in physical oceanography from Texas A&M University, and a Ph.D. degree in physical oceanography from Florida State University. From 1990 to 1996, he was an Assistant Professor in the College of Marine Studies at the University of Delaware, with a focus on the applied mathematics and computational science of large-scale climate modeling. During this time, Dr. McCalpin developed the STREAM benchmark, which has become the *de facto* industry-standard tool for measuring sustained memory bandwidth in high-performance computers. Leaving academia for industry in 1996, he spent three years at Silicon Graphics, Inc., in performance analysis and system architecture, then joined the IBM POWER microprocessor development team in 1999. Dr. McCalpin's current work involves performance projection methodologies for high-performance computing applications and system architecture research related to cache coherence and memory subsystem design.

Richard J. Eickemeyer *IBM Systems and Technology Group, 3605 Highway 52 N., Rochester, Minnesota 55901 (eick@us.ibm.com)* Dr. Eickemeyer is a Senior Technical Staff Member in the IBM Systems and Technology Group. He is currently the processor core performance team leader for the IBM PowerPC* servers. Prior to this, he worked on performance and architecture of several processors used in AS/400* systems and S/390* systems in Rochester, Minnesota and Endicott, New York. Since joining IBM, he has received awards including a Ninth Plateau IBM Invention Achievement Award, an IBM Outstanding Technical Achievement Award, two IBM Outstanding Innovation Awards, and an IBM Corporate Award. He has also been named an IBM Master Inventor. Dr. Eickemeyer received the B.S. degree in electrical engineering from Purdue University and the M.S. and Ph.D. degrees from the University of Illinois at Urbana–Champaign. His research interests are computer architecture and performance analysis. He is a Senior Member of the IEEE.

Steven R. Kunkel *IBM Systems and Technology Group, 3605 Highway 52 N., Rochester, Minnesota 55901 (srkunkel@us.ibm.com).* Dr. Kunkel received his Ph.D. degree from the University of Wisconsin at Madison in 1987. He then joined IBM in Endicott, New York, doing performance analysis of a vector facility for a mid-range System/390* product. In 1989, he moved to the IBM facility at Rochester, Minnesota, where he currently works. During most of his years in Rochester, he did architecture and performance analysis for AS/400 products, including NUMA, VLIW, caches, MP cache coherency, multithreading, and converting AS/400 to PowerPC Architecture* processors. He is currently a Senior Technical Staff Member doing architecture and performance analysis for iSeries*, pSeries*, zSeries*, and xSeries* servers.