

# Characterization of Web Server Workloads on Three Generations of IBM PowerPC Microarchitectures

Pattabi Seshadri and Lizy K. John

University of Texas at Austin

Department of Electrical and Computer Engineering (UT-ECE)

In Collaboration With: Alex Mericas, Processor Performance, Advanced CEC Engineering,  
IBM Austin Server Group

## Abstract

The past decade has seen the development of several out of order, dynamically scheduled superscalar microprocessors. While they have been commercially successful, there has been some concerns on the effectiveness of out of order execution for commercial workloads. Commercial workloads including web server, E-commerce and data base workloads are thought to have a behavior different from SPEC CPU type of workloads and it is doubted by many that out of order execution will not be very effective for these workloads. The aim of this work is to discover whether current microarchitectural design features, speculative and out of order execution in particular, can be effective for applications such as web servers.

To this end, one SPECint workload (gcc), one SPECfp workload (art), and three different types of web server workloads—a HTML page return, a CGI script, and a Java servlet—were run on three generations of IBM PowerPC microarchitectures—the PowerPC 604e (an out-of-order machine), the PowerPC RS64-II (an in-order machine), and the POWERIII (a highly aggressive out-of-order machine)—in order to characterize these workloads on three very different microarchitectures that vary in complexity. Measurements were made using the built-in hardware performance monitor on each machine.

What we found was that the out-of-order POWERIII processor is successful in obtaining fewer execution cycle counts than the in-order RS64-II in web server and SPEC workloads. The speedup is noteworthy in both SPEC workloads (1.38 in gcc and 1.76 in art) and two of the three web server workloads (1.28 in java, and 1.52 in cgi). The html workload incurs nominally fewer cycles on the in-order RS64-II. However, considering the faster clock of the RS64-II, the POWERIII actually performs worse than the RS64-II for all workloads but art (where the POWERIII takes 13% less time). For the other workloads, the POWERIII takes 0% , 11%, 18% or 57% more time. We also found that the 604e, despite slightly worse but comparable branch prediction rates compared to the POWERIII, being a 32-bit machine with less cache and execution bandwidth, does not perform nearly as well as the other two machines for any of the workloads. Presuming that the POWERIII clock frequency is not as high as that of the RS64-II due to the complexity introduced by the out-of-order mechanisms, it seems that aggressive out of ordering is not beneficial to web server workloads. If there were other implementation issues other than out-of-order complexity that resulted in lowering POWERIII's clock frequency, the observation has to be appropriately modified. Also we need to perform more studies with benchmarks such as SPECweb99 and SPECJBB 2000 to confirm the findings.

## INTRODUCTION

The past decade has seen the development of several out of order, dynamically scheduled superscalar microprocessors. While they have been commercially successful, there has been some concerns on the effectiveness of out of order execution for commercial workloads. In 1998,

Keeton et al [Keeton 1998] conducted a study on On-Line Transaction Processing (OLTP) workloads on the Intel PentiumPro processor. They examined the effectiveness of out of order execution, branch prediction, superscalar issue and retire, caching and multiprocessor scaling, using the performance monitoring counters on the PentiumPro. They concluded that out of order execution is only moderately effective for OLTP workloads. It was also seen that PentiumPro's branch prediction scheme is not nearly as effective for database workloads as it is for SPEC workloads. At about the same time, Radhakrishnan and Rawson [Ra98] conducted a study of three web server workloads on two x86 microarchitectures, the Pentium and the PentiumPro. The Pentium is an in order two-issue processor while the PentiumPro uses speculative, out of order execution. The PentiumPro was seen to lower the program execution time and cycles per instruction, illustrating exploitation of speculation and dynamic scheduling.

Thus past research has not been very conclusive about the usefulness of out of order execution techniques on commercial workloads. It is extremely important to examine these results on other platforms and confirm or contradict the previous findings. Three of IBM's recent PowerPC processors, the PPC 604e, Power PC RS64-II, and the POWER III cover a nice spectrum of PPC machines from in-order microarchitectures to aggressively out of order and speculative microarchitectures. In this project, we undertook the task of examining the effectiveness of branch prediction and out of order execution on these three IBM PowerPC microarchitectures. The primary objective of our study is to measure and identify components of processor performance (CPI), and study the effects of speculative and out-of-order execution on resource usage and overall performance.

## EXPERIMENTAL SETUP

This section gives an overview of the three PowerPCs and the three web server workloads used in this study. The PowerPC 604e<sup>1</sup> is a 32-bit, superscalar, out-of-order issue, speculative execution, in order completion machine. It has two single cycle integer units, one multiple cycle integer unit, one branch unit, one condition register unit, one load/store unit, and one pipelined three stage floating point unit. It has a six stage pipeline. The PowerPC 604e can fetch, dispatch, and retire up to four instructions per cycle. For branch prediction, it uses a 64 entry BTAC and a 512 entry, two-bits-per-entry branch history table. It has a 32 KB, four way set associative L1 instruction cache, a 32 KB, four way set associative, non-blocking L1 data cache, and a 256 KB unified L2 cache. It also has two Block Array Translation Units, one for instructions and one for data, that provide quick address translation to large contiguous irregularly sized blocks of memory. The processor clock is 332 MHz.

The PowerPC RS64-II<sup>2</sup> is a 64-bit, superscalar, in order, speculative execution machine and is targeted specifically for commercial applications such as web servers. It has one single cycle integer unit, one multiple cycle integer unit, one four stage pipelined floating point unit, one branch unit, and one load/store unit. It has a five stage pipeline. The RS64-II can fetch, dispatch, and retire up to four instructions per cycle. Unlike the POWERIII and PowerPC 604e, it does not employ any form of dynamic branch prediction. Rather, it prefetches up to 8 instructions from the branch target into a branch target buffer during normal execution, predicts the branch not taken, continues to fetch from the current instruction stream, and then, once the branch is resolved in the dispatch stage, either continues fetching from the current instruction stream with no penalty or flushes the instructions fetched after the branch and starts fetching from the branch target buffer, with a penalty of at most one and often zero cycles. The PowerPC RS64-II also has a 64KB,

---

<sup>1</sup> IBM Corporation and Motorola Corporation, 1998.

<sup>2</sup> Borckenhagen and Storino, 1999.

direct mapped L1 instruction cache, a 64KB two-way set-associative L1 data cache, and a 4MB, four way set associative unified L2. The processor clock is 340 MHz.

The POWERIII<sup>3</sup>, the most complex and aggressive of the three processors, is a 64-bit, superscalar, out-of-order and speculative execution, in-order completion machine. It is targeted mainly for scientific and commercial applications. It has two single cycle integer units, one multiple cycle integer unit, one branch/ condition register unit, two load/store units, and two three stage pipelined floating point units. It can fetch up to four, dispatch up to eight, and complete up to four instructions in the same cycle. It has a 256 entry BTAC and a 2048 entry (bits per entry not available) branch history table for use in branch prediction. Its memory system consists of a 32 KB, 128 way set associative, two way interleaved L1 instruction cache, a 64 KB, 128 way associative, four way interleaved L1 data cache, and a 4MB, four way set associative unified L2. All caches are nonblocking. The POWERIII is designed with separate buses to memory and L2 for greater memory bandwidth. To the same end, it can perform up to four data cache operations in the same cycle (with certain restrictions). The POWERIII also employs a data prefetching mechanism which detects sequential data access patterns and prefetches cache lines to match these patterns. The processor clock is 222 MHz.

The three web server workloads used in this work (HTML, Java servlet and CGI) are representative of real world static and dynamic web server workloads. These are the same simple workloads used by Radhakrishnan and Rawson<sup>4</sup> and Radhakrishnan and John<sup>5</sup>. To understand the execution characteristics of these three workloads, it is necessary to understand how each type of web server workload generates the data that is sent to the HTTP client. An HTML page return is static. In other words, when an HTML server receives an HTTP request for an HTML page, it simply sends the requested page, which is already on disk or in memory, to the client that requested it. CGI scripts and Java servlets are dynamic in that once an HTTP request is received, some computation is done by the server to create data on the fly that is then sent back to the client. Of course, the exact mechanisms by which CGI scripts and Java servlets dynamically create data are different. In the case of this experiment, the CGI script is processed by code that is part of the Apache Web Server executable, while the java servlet is a piece of java code that is interpreted by a Java Virtual Machine. Figure 1 shows the instruction counts for the three web workloads.

Since each of these web server workloads have different mechanisms by which they service client requests, it is to be expected that they exhibit different performance characteristics. In previous workload characterization work done on the Intel PentiumPro and Pentium using the same workloads as will be used in this study<sup>6</sup>, the Java servlet was shown to have the lowest CPI, the lowest instruction and data cache miss rates, and the highest frequency of parallel instruction issue. It was followed in performance in all of these categories by the CGI script and then by the HTML server. Also, interestingly, Radhakrishnan and Rawson found that, while each of the server workloads exhibited treelike branch behavior with lower than average predictability, the Java servlet had a higher branch prediction ratio (and, as to be expected, a higher ratio of instructions completed to instructions dispatched) than the other two workloads. This is surprising since interpreted Java programs typically have higher branch misprediction rates.

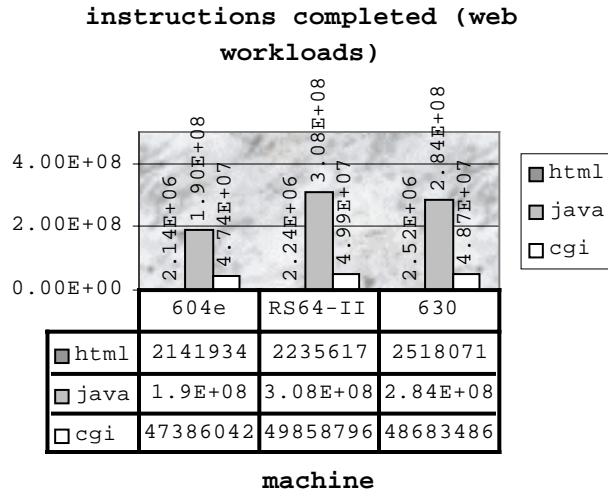
---

<sup>3</sup> Papermaster, Dinkjian, Mayfield, Lenk, Ciarfella, O'Connell, and DuPont, 1998.

<sup>4</sup> Radhakrishnan and Rawson, 1998.

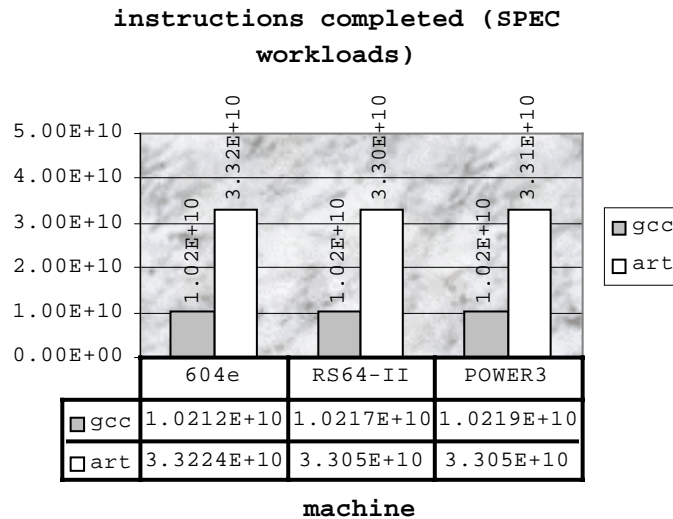
<sup>5</sup> Radhakrishnan and John, 1999.

<sup>6</sup> Radhakrishnan and Rawson, 1998.



**Figure 1: Instructions completed for the web workloads.**

Also, since design decisions for microprocessors are often guided by SPEC results, we will characterize the performance of these three machines on one SPECint 2000 workload (gcc) and one SPECfp 2000 workload (art). We will then use this data to determine if it is justifiable to use SPEC performance data to create design points for web server microprocessors. Figure 2 displays the instructions completed counts for the SPEC workloads.



**Figure 2: Instructions completed for the SPEC workloads.**

## METHODOLOGY

For the web server workloads, client requests will be generated by an automatic client generator, which will be run on a remote machine. The client generator allows the selection of the number

of simultaneous client requests and the total number of requests. For the purposes of this study, all of the requests in a given run will be simultaneous, since the load on the server is determined by the number of simultaneous client requests. Each workload will be run with 20 simultaneous client requests. The web server software used will be Apache Web Server 1.3.12, the CGI script interpreter built into Apache Web Server, and Apache JServ 1.1.2 (a module of Apache Web Server) with JDK 1.1.8 from IBM and JSDK2.0 from Sun Microsystems. The files requested by the clients will be simple, example pages included in the Apache software. The reasons for this are twofold: to allow quicker measurements and to ensure that one type of server workload is not more complex than the other. The essential characteristics of each type of workload should not depend on the size of the requested files. Further, these three workloads are reasonably representative of real world web transactions, as they include both static and dynamic transactions.

Performance measurements will be made using the built in performance monitors in each machine. A performance monitor is hardware built into the machine that can collect various measurements during execution such as cycles, instructions completed, instructions dispatched, branch mispredictions, branch frequency, access latency in each level of cache, etc. IBM has developed an API that can be used to program and collect data from any PowerPC performance monitor. Calls from this API will be inserted into the web server code to interface with the performance monitor.

Since each performance monitor has a limited number of counters, and there are far more relevant events to measure than there are counters, several runs are required to gather all of the desired measurements. The arrangement of measurements was designed to count as many measurements which are relevant to each other as possible in the same run. For example, the number of branches was counted simultaneously with the number of branches mispredicted. One series of runs to collect all of the necessary measurements will be henceforth referred to as a set. Ten sets of measurements were run for each web workload on each machine, and five sets were run for each SPEC workload on each machine (since the variation in performance monitor counts from run to run was experimentally found to be very small for the SPEC workloads). Thus,  $(10 \times 3 + 5 \times 2) \times 3$ , or 120 measurement sets, were run in total, 40 cycles on each machine.

Once the results were collected, a trimmed mean was taken from the five or ten values for each measurement (i.e., the highest and lowest of the values was discarded, and a mean was taken from the remaining values).

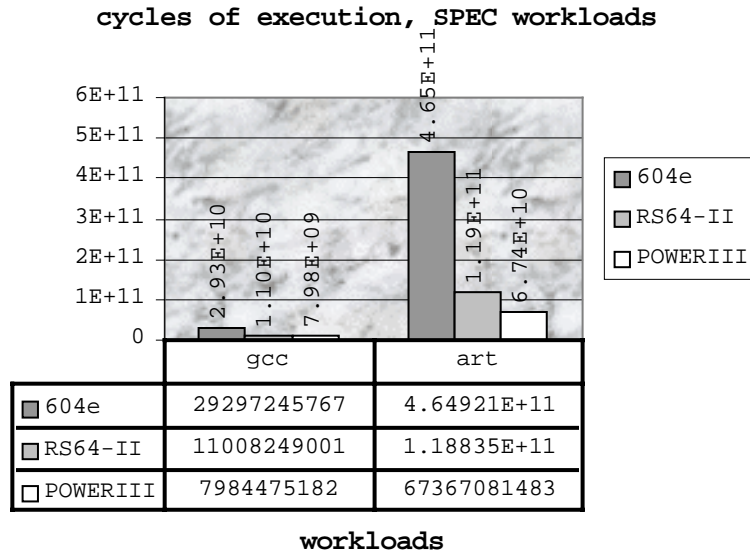
## **RESULTS**

### **Overall Performance**

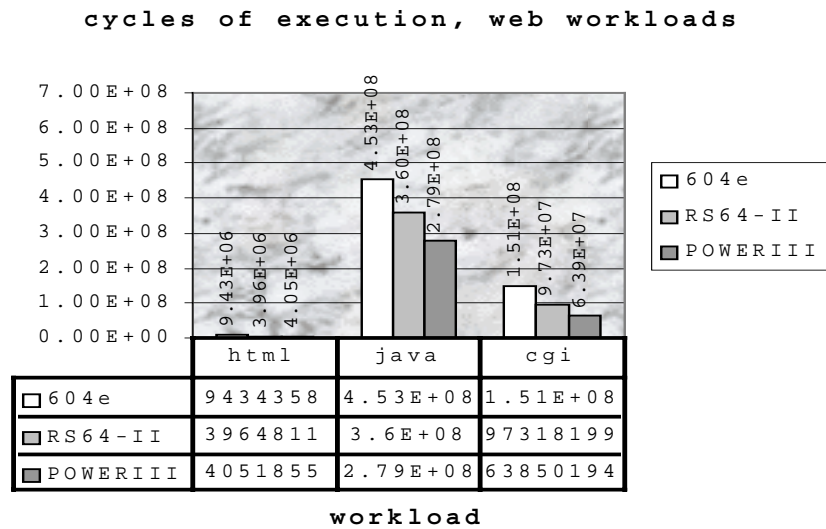
Figure 3 shows the cycles of execution of the web workloads and Figure 4 displays the cycles of execution for the SPEC workloads. POWERIII with its out of order mechanisms is seen to accomplish the tasks in fewer processor cycles. The speedup is noteworthy in both SPEC workloads (1.38 in gcc and 1.76 in art) and two of the three web server workloads (1.28 in java, and 1.52 in cgi). The html workload incurs nominally fewer cycles on the in-order RS64-II. It is also clear that the 604e cannot come close to the performance of either the RS64-II or the POWERIII on any of the workloads. The 604e, of course, is a 32-bit machine while the other two are 64-bit machines, and this can be expected.

The RS64-II processor that we used in our study has a faster clock than the POWERIII (340MHz on the RS64-II versus 222MHz on the POWERIII). Hence it is important to compare actual execution time rather

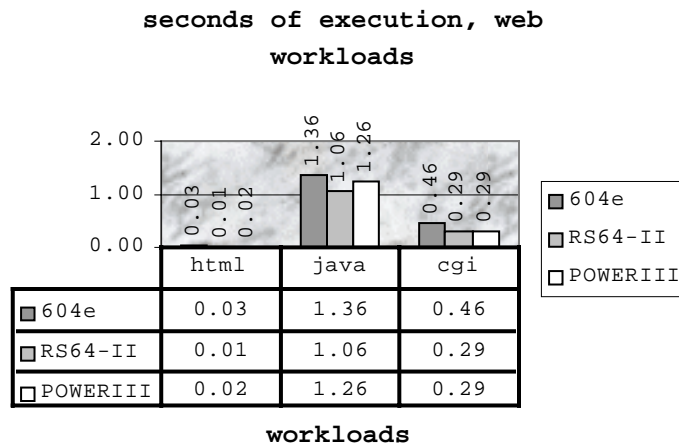
than clock cycles. Figure 5 compares the actual execution time for the web workloads, and Figure 6 compares the actual execution time for the SPEC workloads. The POWERIII actually performs worse than the RS64-II for all workloads but art (where the POWERIII takes 13% less time). For the other workloads, the POWERIII takes 0%, 11%, 18% or 57% more time. Presuming that the POWERIII clock frequency is not as high as that of the RS64-II due to the complexity introduced by the out-of-order mechanisms, it seems that aggressive out of ordering is not beneficial to web server workloads. If there were other implementation issues other than out-of-order complexity that resulted in lowering POWERIII's clock frequency, the observation has to be appropriately qualified. Also, the RS64-II has a larger instruction cache than the POWERIII. The impact of that factor will be analyzed in detail later.



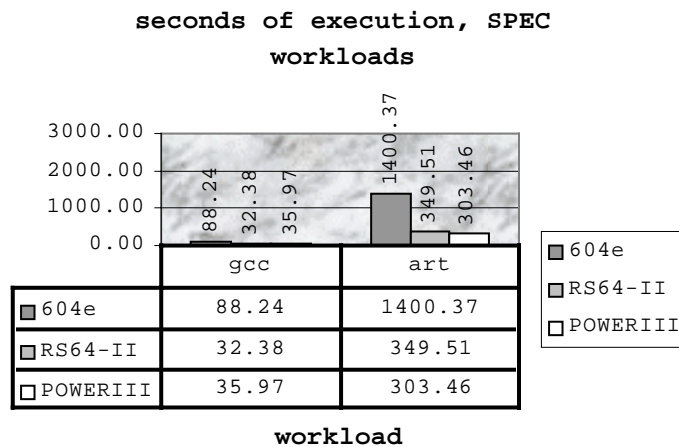
**Figure 3: Cycles of execution for the web workloads.**



**Figure 4: Cycles of execution for the SPEC workloads.**

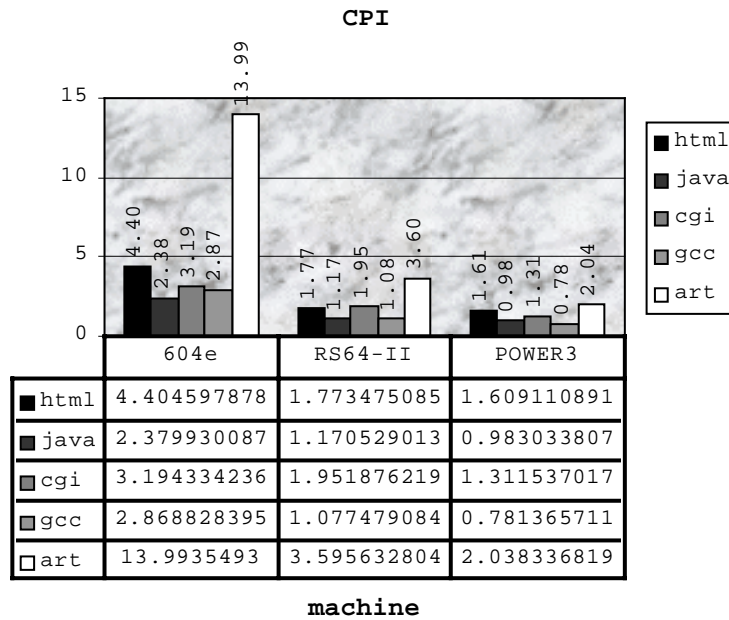


**Figure 5: Execution time (seconds) for the web workloads.**



**Figure 6: Execution time (seconds) for the SPEC workloads.**

Figure 7 displays the CPI for each machine on all of the workloads. While it looks as though the RS64-II has a significantly worse CPI than the POWERIII on the SPEC workloads and the CGI script (henceforth referred to as cgi), it comes reasonably close to the POWERIII's CPI on the HTML (henceforth html) and Java servlet (henceforth java) workloads. Thus, while an in-order design cannot quite get better CPI figures than an out-of-order design, its simpler architecture allows a faster clock, which can give it better performance.



**Figure 7: CPI for the workloads.**

Table 1 gives a side-by-side comparison of the performance of each machine. The “Perf. Margin” columns show the percentage difference in IPC between the 604e and RS64-II and the POWERIII, while the “Execution Time” columns show the percentage difference in execution time between the POWERIII and the other two machines. A positive number indicates that POWERIII is performing better. There are 3 negative numbers in the last column indicating lower performance of POWER III. The 604e, as noted above, obviously cannot match the POWERIII’s performance for any workload, the smallest performance margin the POWERIII has being .59. The RS64-II, on the other hand, lags quite far behind the POWERIII on gcc, art, and cgi (with .27, .43, and .33 performance margins respectively), but for java and html, its performance is comparable (.09 and .16 performance margins, respectively).

workload	POWERIII IPC	604e IPC	RS64-II IPC	POWERIII Perf. Margin Over 604e	POWERIII Perf. Margin Over RS64-II	POWERIII Execution Time Margin Over 604e	POWERIII Execution Time Margin Over RS64-II
html	0.62	0.23	0.56	0.63	0.09	0.36	-0.57
java	1.02	0.42	0.85	0.59	0.16	0.08	-0.19
cgi	0.76	0.31	0.51	0.59	0.33	0.37	0.00
gcc	1.28	0.35	0.93	0.73	0.27	0.59	-0.11
art	0.49	0.07	0.28	0.85	0.43	0.78	0.13

**Table 1: Performance comparison of the 604e and RS64-II to the POWERIII.**

These results suggest that, since the RS64-II is an in-order machine and the POWERIII is an out-of-order machine, out-of-order execution might not have as great of an advantage over in-order execution as commonly thought, at least for some web workloads. In addition, as aforementioned, the higher clock rate of the RS64-II gives it equal or better performance in real time than the POWERIII for all workloads but art (understandable, since the POWERIII has two floating point units compared to one for the RS64-II).

But it is not enough to simply look at the raw performance data and declare that in-order execution compares favorably to out-of-order execution on web workloads. The POWERIII and RS64-II differ not only in that one is an in-order machine and one is an out-of-order machine, but also in instruction cache size, data cache associativity, branch prediction strategy, dispatch width, and number of execution units, among other features. Further, while we can say with certainty that a simpler out-of-order machine such as the 604e can never compete either with a more complex in-order machine like the RS64-II or a highly aggressive out-of-order machine like the POWERIII, it would be instructive to see exactly why the 604e cannot compete on these workloads. Therefore, it is necessary to look at the branch, cache, and dispatch behavior and resource stalls of these three machines in order to ascertain how these performance vectors vary from workload to workload, how they contribute to performance, and to what extent they depend on the fact that the machine is in-order or out-of-order.

### Branch Behavior

First we look at branch behavior. Table 2 gives a comparison of the branch prediction accuracies of the three machines, along with the POWERIII's performance margin (in terms of IPC) over the 604e and RS64-II for each workload.

wkld	POWERIII IPC	604e IPC	RS64-II IPC	POWERIII Perf. Margin Over 604e	POWERIII Perf. Margin Over RS64-II	POWERIII Branch Pred. Accuracy	604e Branch Pred. Accuracy	RS64-II Branch Pred. Accuracy
html	0.62	0.23	0.563865	0.63	0.09	0.91	0.88	0.36
java	1.02	0.42	0.854315	0.59	0.16	0.95	0.93	0.30
cgi	0.76	0.31	0.512328	0.59	0.33	0.96	0.95	0.48
gcc	1.28	0.35	0.928092	0.73	0.27	0.97	0.97	0.27
art	0.49	0.07	0.278115	0.85	0.43	0.99	0.99	0.34

**Table 2: Branch prediction accuracy and performance margins.**

One seemingly surprising detail of this table is that the branch prediction accuracy of the RS64-II has no correlation with the IPCs for each workload or with the relative performance compared to the POWERIII. However, it must be kept in mind that the RS64-II does not employ a typical dynamic branch predictor; rather, it prefetches instructions from the target instruction stream, always predicts not taken, then, upon resolving the branch, either continues fetching along the not taken path or feeds the instructions from the target stream buffer into the instruction stream. This strategy, which seeks to minimize the branch misprediction penalty rather than predict branches more accurately, makes the actual branch prediction rather irrelevant, hence the arbitrary manner in which not taken is always predicted.

On the other hand, if we look at the branch prediction accuracy of the POWERIII in comparison to its IPC and performance margins over the other two machines, there can be seen a direct correlation between branch prediction accuracy and performance margin over the RS64-II. The better the branch prediction accuracy, the better the POWERIII performs compared to the RS64-II. The same general trend can be seen in comparison with the 604e, except that the performance margins for java and cgi are lower than that for html.

These data give several key insights. One is that branch prediction is likely a major bottleneck for the POWERIII, which is to be expected, since it is a highly aggressive machine. Accordingly, these data also show that for workloads (specifically html and java) where branch prediction success is lower (below 96%), an in-order machine such as the RS64-II can perform comparably to an out-of-order machine like the POWERIII. To put it another way, an out-of-order machine cannot take full advantage of its ability to execute out of order when it cannot predict branches with a very high success rate.

Another interesting note is that, while the 604e has slightly worse branch prediction success than the POWERIII, as is to be expected due to the larger BTAC and history table of the POWERIII, its performance lags further behind than could be expected from its branch prediction numbers alone. The 604e, of course, is a 32-bit machine while the other two are 64-bit machines, and this can be expected to affect performance considerably due to the comparative lack of cache and execution bandwidth of the 604e. If this hypothesis is true, resource stalls would likely play a large role. Table 3 shows the resource stalls for the 604e and RS64-II. Resource stall counts were not available for the POWERIII, and at the time of this writing, data on the dispatch and load/store stalls for the art workload on the 604e had not been collected. We see here that, indeed, resource stalls seem to play a much larger role on the 604e than the RS64-II. On the art workload, where performance is especially poor for the 604e, FPU stalls occur for the great majority of the execution cycles. Also, dispatch stalls seem to play a huge role on every workload, as do load/store stalls for cgi and gcc.

wkld	604e LSU Stalls / Cycles	604e Disp Stalls / Cycles	604e Integer Unit Stalls / Cycles	604e FPU Stalls/ Cycles	RS64-II Integer Stalls/ Cycles	RS64-II FPU Stalls / Cycles
html	0.24416	0.90302	0.36831	0.00000	0.14438	0.00002
java	0.22792	0.81577	0.59427	0.00004	0.25312	0.00017
cgi	0.51118	0.92163	0.46036	0.00011	0.11637	0.00003
gcc	0.67110	0.92036	0.17217	0.00002	0.16098	0.00023
art	N/A	N/A	0.00139	0.87645	0.07657	0.14397

**Table 3: Resource Stalls for 604e and RS64-II .**

Table 4 shows the top two causes of both load/store unit (LSU) and dispatch stalls for the 604e. These ratios are the resource stall counts over the number of cycles. It seems that for the LSU stalls, poor data cache and store bandwidth are problems, while for the dispatch unit, poor instruction cache bandwidth (which starves the dispatch unit of instructions) and resource stalls (causing no execution unit to be available) are problems. Our hypothesis of why the 604e performs so poorly seems to be correct. Better instruction and data cache bandwidth, along with more execution units, might solve this machine's performance problems. Despite being an out-of-order machine, these hindrances seem to prevent the 604e from utilizing most of its resources.

wkld	LSU Stalls--BIU/Cache Busy	LSU Stalls--Store Queue Full	Disp Stalls--No Execution Unit Avail.	Disp Stalls--Waiting on Instructions
html	0.06	0.09	0.32	0.55
java	0.05	0.09	0.42	0.55
cgi	0.12	0.22	0.64	0.55
gcc	0.67	0.41	0.57	0.12
art	N/A	N/A	N/A	N/A

**Table 4: Top two causes of LSU and Dispatch Stalls for the 604e.**

### Cache Behavior

Branch behavior apparently has a significant performance impact on the three machines in this study, but cache behavior is another important consideration.

wkld	POWERIII IPC	604e IPC	RS64-II IPC	POWERIII Perf. Margin Over 604e	POWERIII Perf. Margin Over RS64-II	POWERIII IC Miss Rat.	RS64-II IC Miss Rat.	604e DC Miss Rat.	RS64-II DC Miss Rat.
html	0.62	0.23	0.56	0.63	0.09	0.07	0.03	0.05	0.03
java	1.02	0.42	0.85	0.59	0.16	0.04	0.01	0.04	0.01
cgi	0.76	0.31	0.51	0.59	0.33	0.04	0.02	0.06	0.03
gcc	1.28	0.35	0.93	0.73	0.27	0.02	0.01	0.07	0.02
art	0.49	0.07	0.28	0.85	0.43	0.0004	0.0002	0.57	0.24

**Table 5: Comparison of cache behavior with raw performance.**

Table 5 displays the correlation between cache performance and overall performance for the three machines.

### Instruction Cache Behavior

The RS64-II, having an instruction cache twice the size of the other two machines (64 KB compared to 32 KB), expectedly has the superior IC miss ratio. The POWERIII, as one would predict due to the fact its IC is half the size of the RS64-II's, has an IC miss ratio which is roughly twice that of the RS64-II for every workload. The 604e's instruction cache miss rates were not available, but since it too has a 32 KB IC, one would expect performance similar to the 630. However, it has different associativity (4-way compared to 128-way) and different bus width (16 bytes compared to 32 bytes), so the numbers could be different. However, one would not expect them to vary greatly.

Comparing the RS64-II and POWERIII's IC miss ratios, a clear trend emerges: the POWERIII's performance advantage diminishes as the IC miss ratio for both workloads increases. This is

most likely because the larger the IC miss ratios for both machines, the more the POWERIII, with a miss ratio roughly twice that of the RS64-II, is punished. This seems to work in tandem with the correlation between low branch performance and small performance margin for the POWERIII to allow the in-order RS64-II to perform comparably well on the html and java workloads.

That being said, the question remains whether branch prediction success or IC miss rates are more responsible for the fact that the RS64-II performs comparably to the POWERIII for these workloads. The likely answer is that both are significant contributors to the two machines' comparative performance. As far as the contribution of branch misprediction rates to the POWERIII's performance advantage (or lack thereof) over the RS64-II, Table 6 shows that, indeed, except for cgi, the proportion of instructions flushed to instructions dispatched increases as branch prediction accuracy gets worse, and furthermore, this ratio is higher than the branch misprediction ratio, except for on the art workload. This suggests that branch mispredictions are causing major pipeline delays for the POWERIII, which can have up to 32 instructions pending during a branch misprediction. Table 6 is an indication of the speculation factor and wasted work due to speculation in these workloads. Most work is wasted in the html workload.

wkld	Br. Pred. Acc.	Instrs Flushed	Instrs Flushed / Instrs Disp
html	0.91	841005.50	0.29
java	0.95	52228317.88	0.17
cgi	0.96	10709814.50	0.21
gcc	0.97	1197749941.50	0.11
art	0.99	464571662.33	0.01

**Table 6: Comparison of branch prediction accuracy with ratio of instructions flushed to instructions dispatched for the POWERIII.**

As for instruction cache miss rates, poorer branch prediction could be associated with poorer IC performance in that many of the instruction cache misses, especially for the workloads with high IC miss ratios, could be caused by indirect branches, subroutines, and if-then statements, which cause large jumps in code space. Certainly, it would be expected that the web workloads would have more of these branching structures than the SPEC workloads, which typically run in tight loops, and thus higher IC miss ratios, which is borne out by the data. Certainly, higher IC miss rates can also cause significant execution delays. So instruction cache performance for these workloads is closely tied to branch predictability, and both are significant contributors to the large advantage of the POWERIII over the RS64-II for cgi, gcc, and art, as we had hypothesized

### **Data Cache Behavior**

Now we turn our attention to the data cache miss ratios. Data cache information was not available on the POWERIII. Also, data cache miss counts were available on the 604e, but data cache references were not, so the 604e's DC miss rates were calculated by considering each load/store instruction dispatched as a cache reference. Having a data cache half the size of the RS64-II's, one would expect the 604e to have about twice the data cache miss ratio. Indeed, as Table 5 attests, the 604e's DC miss rates are twice as high or worse for all of the workloads.

These high miss ratios help to explain the poor performance of the 604e. The POWERIII data cache, with its 64 KB size, 24 byte bus, and 128 byte line size, is similar to the RS64-II's 64 KB, 16 byte bus, 128 byte line data cache, and one would expect their performance to be similar. The associativities are different (2-way for the RS64-II compared to 128-way for the POWERIII), but, especially given how the data cache performance is so good for the RS64-II for all workloads but art, this difference is unlikely to cause a significant performance disparity.

Given the low DC miss rates for the RS64-II and the lack of correlation between the RS64-II's data cache miss rates and relative performance to the POWERIII (except for in art, where FPU resource conflicts are likely to play a larger role), it looks as if data cache performance is not a significant factor for these web workloads. The fact may be that these workloads do not exert sufficient cache pressure, whereas actual web server applications may. We wish to continue these studies with SPECweb99 and SPECJBB workloads in order to investigate this issue.

### Dispatch Bandwidth

Table 7 gives dispatch efficiency numbers for the three machines. Some of the metrics were not available on the POWERIII. We can see that the 604e has a much higher proportion of cycles where zero instructions are dispatched than either the RS64-II or the POWERIII, by a factor no lower than 1.13 (for the cgi workload as compared to the POWERIII) and as high as 2.42 (for the gcc workload as compared to the RS64-II). Comparing the 604e's proportion of cycles of  $\geq 2$  instructions dispatched to that of the RS64-II, the RS64-II beats it by a factor ranging from 1.8 to 5. Obviously, the 604e is not taking full advantage of its superscalarity, and this is directly related to the large proportion of dispatch stalls as detailed in Table 3 and Table 4. The relative performance of the 604e in this area to the other two machines is poor for all of the workloads, so this is not a characteristic of the web workloads per se—the 604e just does not have the execution bandwidth to perform comparably to the other two machines on any workload.

machine	metric	html	java	cgi	gcc	art
604e	% Cycles 0 Inst. Disp.	0.85	0.76	0.81	0.80	0.95
	% Cycles 1 Inst. Disp.	0.08	0.11	0.09	0.08	0.02
	% Cycles 2 Inst. Disp.	0.05	0.08	0.06	0.09	0.02
	% Cycles 3 Inst. Disp.	0.02	0.05	0.04	0.03	0.00
	% Cycles 4 Inst. Disp.	0.003	0.004	0.005	0.003	0.001
	% Cycles < 2 Inst. Disp.	0.93	0.86	0.90	0.88	0.98
	% Cycles $\geq 2$ Inst. Disp.	0.07	0.14	0.10	0.12	0.02
	Inst. Disp/ Cyc.	0.24	0.44	0.33	0.36	0.44
RS64-II	% Cycles 0 Inst. Disp.	0.57	0.46	0.64	0.33	0.78
	% Cycles 1 Inst. Disp.	0.23	0.30	0.19	0.24	0.11
	% Cycles 2 Inst. Disp.	0.10	0.16	0.08	0.30	0.06
	% Cycles 3 Inst. Disp.	0.08	0.21	0.09	0.11	0.04
	% Cycles 4 Inst. Disp.	0.01	0.01	0.01	0.01	0.00
	% Cycles < 2 Inst. Disp.	0.81	0.66	0.82	0.58	0.90
	% Cycles $\geq 2$ Inst. Disp.	0.19	0.34	0.18	0.42	0.10
	Inst. Disp. / Cyc.	0.71	1.07	0.64	1.23	0.71
POWERIII	% Cycles 0 Inst. Disp.	0.72	0.56	0.72	0.42	0.79
	Inst. Disp./ Cyc.	0.71	1.12	0.81	1.36	0.64

Table 7: Dispatch efficiency numbers.

Now we compare the RS64-II and the POWERIII. The RS64-II has a significantly lower proportion of cycles with 0 instructions dispatched than the POWERIII. This is most likely because of the branch strategy of the RS64-II, which seeks to minimize branch misprediction penalties. There is never a situation in the RS64-II where all of the pending instructions are flushed, and resource stalls are not a very significant performance bottleneck as shown in Table 3, so there naturally is not a high proportion of instructions where zero instructions are dispatched. The POWERIII, on the other hand, has quite a high proportion of these cycles, and this is, as detailed in the previous section on branch performance, likely because of the high branch misprediction penalties for the machine. There is not a direct correlation between a high proportion of zero-dispatch branches and a high branch misprediction ratio for the POWERIII. The high proportion of zero instructions dispatched for art is more likely due to resource stalls, since the workload puts a lot of strain on the FPU. If one compares the web workloads to gcc, they have a higher proportion of zero-dispatch cycles. Yet, the high zero-dispatch number for cgi is strange, because its branch prediction success is quite good for the workload, and judging by Table 3, it does not seem to cause a higher amount of resource stalls in the RS64-II, and neither does it cause significantly higher cache miss rates.

Despite the RS64-II's lower zero-cycle dispatch numbers, it has a lower number of instructions dispatched per cycle than the POWERIII for every workload. This is probably due to the fact that, while the POWERIII is punished heavily for branch mispredictions, when its predictor works it has a high execution bandwidth, while the RS64-II does not often dispatch zero instructions, but does not often dispatch three or four either because it is in-order and cannot take full advantage of the code's ILP. Nevertheless, it must be noted that the POWERIII can theoretically dispatch up to eight instructions per cycle, while the RS64-II can only dispatch up to four. So the POWERIII is coming nowhere near its dispatch bandwidth potential. This seems to indicate that the extra superscalarity designed into the POWERIII (two LSUs, two FPUs compared to one and one for the RS64-II) does not give it much of a performance edge because the dispatch restrictions are too strict. Thus, branch performance contributes more to the performance gap between the POWERIII and the RS64-II than superscalarity.

This difference between instructions dispatched per cycle for the two machines, as expected, is less for java and html (where the ratio of POWERIII instructions dispatched per cycle to the RS64-II's is 1 and 1.05, respectively) than for cgi, gcc, and art (1.27, 1.11, and 1.11 respectively). This is most likely due to the poor branch predictability of java and html as described in the section on branch behavior.

Overall, the dispatch bandwidth numbers confirm what the raw performance and branch performance data indicate—that the POWERIII has less of an advantage over the RS64-II for web workloads.

## SUMMARY

From the observations made above, several conclusions can be drawn:

- The out-of-order POWERIII processor is successful in obtaining fewer execution cycle counts than the in-order RS64-II in web server and SPEC workloads. The speedup is noteworthy in both SPEC workloads (1.38 in gcc and 1.76 in art) and two of the three web server

- workloads (1.28 in java, and 1.52 in cgi). The html workload incurs nominally fewer cycles on the in-order RS64-II.
- Considering the faster clock of the RS64-II, the POWERIII actually performs worse than the RS64-II for all workloads but art (where the POWERIII takes 13% less time). For the other workloads, the POWERIII takes 0%, 11%, 18% or 57% more time than the in-order RS64-II.
  - If the same clock rates can be achieved on in-order and out-of-order processors, the out-of-order machine will have better performance, though its performance advantage will be narrow for workloads like html and java.
  - Instruction cache performance is very critical for web server workloads. The larger instruction cache on the RS64-II is seen to be significantly beneficial to its performance compared to the POWERIII.
  - The benefits of an out-of-order machine are completely eliminated if the machine has poor IC and DC bandwidth and poor superscalarity, as the 604e does.
  - An in-order machine can perform comparably to a highly aggressive out-of-order machine for workloads which exhibit less predictable branches, such as html and java; the POWERIII's IPC performance margin over the RS64-II is .09 and .16, respectively, for these workloads, while its IPC performance margin on the other three workloads ranges from .27 to .43. The RS64-II's unique branch prediction scheme also likely helps it avoid the problems of low branch predictability by endeavoring to reduce branch misprediction penalties rather than branch misprediction rates.
  - Since html and java are two of the most commonly used web processing methods, these results speak to the fact that out-of-order execution, while it works extremely well for SPEC-type workloads with high branch predictability, gives much less of an advantage for web processing.
  - However, if dynamic branch prediction methods can be improved to yield near-perfect branch misprediction ratios for workloads such as HTML and Java servlets, which do not contain many loop branches in proportion to the total number of branches, out-of-order execution can reap much greater benefits over in-order execution.
  - Branch prediction is key to the success of a highly aggressive out-of-order machine such as the POWERIII; since it is so aggressive, it incurs a large penalty from branch mispredictions. Branch misprediction rates above 4% cause performance problems for an out-of-order machine relative to an in-order machine such as the RS64-II.

Also worth mentioning is the fact that according to the branch performance numbers, the java workload does not have the best branch predictability of the three web workloads, but rather the cgi workload does, which contradicts the findings of Radhakrishnan and Rawson<sup>7</sup>. This seems to agree with the fact that java code has a lot of indirect branches to subroutines which do not have good spatial locality, which would hurt its branch performance despite the JVM having a lot of loops. However, our numbers do confirm their finding that the java workload has the best IC and DC miss ratios of the three, with cgi next and then html. Also, as Radhakrishnan and Rawson also found, java seems to have the best overall performance, with cgi next and then html, except on the RS64-II, where the order of html and cgi is reversed.

Hopefully these results can provide some guidelines to the architects of the next generation web servers. Future research consists of experimenting with workloads that give more cache pressure, evaluating POWER IV performance, etc. We are starting to experiment with SPECweb99 and SPECJBB2000 workloads. As soon as POWER IV is available, we would like to continue our studies on that platform.

---

<sup>7</sup> Radhakrishnan and Rawson, 1998.

## REFERENCES

Radhakrishnan, R. and Rawson, F. L. III, "Characterizing the Behavior of Windows NT Web Server Workloads Using Processor Performance Counters", Presented at the First Workshop on Workload Characterization, November 1998, Dallas, Texas, Also appears in Workload Characterization: Methodology and Case Studies, IEEE Computer Society Press, edited by Lizy Kurian John and Ann Marie Maynard, pp. 76-85.

Radhakrishnan, R. and John, L. K., "A Performance Study of Modern Web Applications," Proceedings of the European Parallel Processing Conference (Euro-Par 99), Lecture Notes in Computer Science, Springer, pp. 239-247.

Papermaster, M., Dinkjian, R., Mayfield, M., Lenk, P., Ciarfella, B., O'Connell, F. and DuPont, R. "POWERIII: Next Generation 64-bit PowerPC Processor Design", White Paper, IBM Corporation, 1998.

Borkenhagen, J. and Storino, S., "4th Generation 64-bit PowerPC-Compatible Commercial Processor Design", White Paper, IBM Corporation, <http://www.rs6000.ibm.com/resource/technology/nstar.html>, January 1999.

"PowerPC 604e Microprocessor User's Manual", IBM Corporation and Motorola Corporation, 1998.

[Keeton98] Kimberly Keeton, David A. Patterson, Yong Quiang He, Roger C. Raphael, and Walter E. Baker, "Performance Characterization of a Quad PentiumPro SMP Using OLTP Workloads", Proceedings of the International Symposium on Computer Architecture, Barcelona, Spain, 1998, pp. 15-26.