USENIX Association

Proceedings of the 4th Annual Linux Showcase & Conference, Atlanta

Atlanta, Georgia, USA
October 10–14, 2000

SMP Scalability Comparisons of Linux® Kernels 2.2.14 and 2.3.99

Ray Bryant raybry@us.ibm.com Bill Hartner bhartner@us.ibm.com
Qi He[1] qhe@cc.gatech.edu Ganesh Venkitachalam[2] venkitac@yahoo.com
IBM® Linux Technology Center
IBM Austin

Abstract

The Linux 2.4 kernel should provide significantly better SMP scalability than is available in the 2.2 series. While the development of the 2.4 kernel is still continuing, performance studies on the 2.3.99 kernels can be used as an indicator of what the performance of the 2.4 kernel will be like. In this paper, we compare performance and SMP scalability of Linux 2.2.14 and Linux 2.3.99 on the Intel® 32-bit platform using four benchmarks. Three of these are targeted to specific system components; the fourth is SPECweb99™. SMP scaling for these benchmarks is shown to be significantly better with 2.3.99 than it is with 2.2.14. This should translate into significantly better SMP scaling (and hence additional raw performance on SMP systems) for applications running under Linux 2.4.

Introduction

Just as other authors have compared the function available in Linux 2.4 to that available in Linux 2.2 (see, for example, [Linux 2.4]), the purpose of this paper is to compare the performance likely to be available in Linux 2.4 to the performance delivered by Linux 2.2. We make this estimate by comparing the performance of Linux 2.3.99 kernels to Linux 2.2.14 on the Intel 32-bit platform using a set of four benchmarks. Three of these benchmarks are targeted to specific components of the Linux kernel:

- Volanomark™: a benchmark of a chat room server written in the Java™ language. (scheduler, TCP/IP stack)
- Netperf: a network communications benchmark (TCP/IP stack)
- FSCache: a file system buffer-cache benchmark (file system cache)

The fourth benchmark, SPECweb99 [SPW99] is a benchmark of web server performance.

In addition to comparing the performance of these kernels, we use a standard set of performance tools to analyze the kernels as the benchmarks execute. The resulting data will help us analyze and fix bottlenecks remaining in Linux 2.4.

While benchmarking is of course, imperfect (if not controversial, c. f. [Mindcraft] [ESRFiasco]), we believe this is the only way to obtain reproducible and verifiable comparisons between two kernels. Such a study should be conducted so that another group running the same benchmark with the identical setup will obtain similar results. One must also be careful not to infer conclusions from the benchmark results about unrelated workloads.

In the following sections of this paper, we discuss the following:

- the instrumentation patch we used to analyze kernel performance
- the kernels we compared
- benchmarks employed
- the measurement environment
- the measurement results
- implications of these results pertaining to the performance of the Linux 2.4 kernel

Instrumentation

This paper presents results obtained from the IBM Linux Kernel Trace and Profile Facility patch.[3] This patch implements a profiling facility that profiles the

[1] Author's current address: Georgia Institute of Technology, 328290, Georgia Tech Station, Atlanta, GA 30332.

[2] Author's current address: VMware, Inc., 3145 Porter Drive, Palo Alto, CA 94304.

[3] This code is not available outside of IBM at the present time. If there is sufficient community interest, it may be released to the open source community at some time in the future.

kernel execution. It produces output that is similar to the output produced by the Linux kernel profiling facility or the SGI™ Kernprof [SGIKernprof]. The measurements reported using the IBM Kernel Trace and Profile Facility could, in principle, be duplicated outside of IBM using these other tools.

In addition to timer-based profiling, both the IBM Linux Kernel Trace Facility and the SGI Kernprof Patch support Pentium® performance-counter based profiling. Time-based profiling samples the current instruction pointer at a given time interval (e.g., every 10 ms.). Postprocessing tools use the recorded locations to construct a histogram of the amount of time spent in various kernel routines. In performance-counter based profiling, a profile observation is set to occur after a certain number of Pentium performance counter events [PerfCount]. For example, one could take an observation every 1,000,000 instructions or every 10,000 cache-line misses. Just as time-based profiling shows where the kernel spends its time, an instruction-based profile shows where the kernel executes most of its instructions and a cache-line based profile shows where the kernel takes most of its cache-line misses. These latter types of profiles can provide additional insight into kernel performance.

Kernels Compared

In this paper, we report on comparisons of Linux kernel 2.2.14 and Linux 2.3.99-pre4, pre6, or pre8 as specified in the benchmark data. For some of the benchmarks the kernel.org version of 2.2.14 would not correctly run the benchmark and we had to use the Red Hat™ 6.2 version of the kernel instead (2.2.14-5.0). We were later able to fix this problem by porting the eepro100 ethernet card driver from Red Hat 6.2 to the kernel.org version. The versions of the 2.2.14 kernel used for the SPECweb99 experiments were the kernel.org version of 2.2.14 with the updated version of the eepro100 driver.

The kernels were all built in uniprocessor and multiprocessor versions using gcc version "gcc version egcs-2.91.66 19990314/linux." Kernels were built for machine architecture "686". In some cases we also include measurement data from other kernel versions for comparison purposes. These kernels were also built using the same version of gcc.

A full specification of the configuration options used to build these kernels would be required to completely define the way these kernels were built; these configuration files are available from the authors on request.

Benchmarks Used

Volanomark

Volanomark is a benchmark of a chat room server and is written in the Java language. The benchmark and the chat room server are products of Volano, LLC [Volano]. Occasionally, Volano publishes a report (known as the Volano Report [VReport]) comparing Volanomark performance on a variety of Java implementations and operating systems. The Volanomark measurements reported in this paper use slightly different run rules and parameters than the measurements reported in the Volano Report and are not comparable to the results published there.

We present throughput results using Volanomark 2.1.2 and the IBM Runtime Environment for Linux version 1.1.8 (formally known as part of the IBM® Developer Kit for Linux®, Java™ Technology Edition, Version 1.1.8, and herein referred to as the IBM R/T 1.1.8) that we used in a previous paper [JTThreads]. Further description of Volanomark can be found at [VMark] and in our previous paper. We also present SMP scalability results using Volanomark 2.1.2 and the IBM Runtime Environment for Linux version 1.3 [NewIBMRT] (formally known as part of the IBM® Developer Kit for Linux®, Java™ 2 Technology Edition, Version 1.3, and herein referred to as the IBM R/T 1.3) and IBM R/T 1.1.8.

The principle metric reported by the Volanomark test is "chat server message throughput" in messages per second. All Volanomark results present here are for loopback experiments; in a loopback experiment both the client and server run on the same system.

While Volanomark was developed by Volano, LLC to compare their chat server performance under different Java implementations, we have found it to be useful in the Linux environment to test scheduler and TCP/IP stack performance, particularly if the IBM R/T 1.1.8 or 1.3 is used. Each chat room client causes 4 Java threads to be created. The IBM R/Ts use a separate Linux process to implement each java thread. Thus for a Volanomark run with 200 simulated chat room clients, there will be 800 processes active in the system. Each java thread spends most of its time waiting for data from a communications connection. When data is received, relatively little user space

processing is performed, new messages are sent, and the Java thread then waits for more data. The result is that approximately 60% of the time spent executing the benchmark is in kernel mode.

Netperf

Netperf is a communications benchmark available from [NetPerf]. Netperf includes a number of tests that can be used to measure protocol and network performance.

In this paper we report on the "request response" or "RR" test. The RR test creates a number of socket connections between the client and server. Once the connections are established, messages are transmitted and returned across each connection as quickly as possible. At the end of a fixed time period, the test ends and the clients report the total number of messages exchanged. The principle metric produced by the Netperf RR test is message throughput in messages/second.

FSCache

FSCache is an internal IBM benchmark that measures file system performance for files read out of the buffer cache. (It is our plan to make this benchmark available to the open source community.) FSCache supports both random and sequential read tests; here we report only the results of the random read tests. The principle metric of FSCache is kilobytes/second read from the buffer cache.

This benchmark represents only one aspect of measuring file system performance. However, it is necessary that this component of the file system scale in order for the overall file system to scale well. Thus our results are preliminary indications of overall file-system scalability.

SPECweb99

SPECweb99 was developed by the Standard Performance Evaluation Corporation (SPEC), and is described in detail at [SPWB99]. The version of SPECweb99 we use is the 1.01 version released on 11/23/99. SPECweb99 is designed to test and measure the ability of a particular system and hardware configuration to act as a web server, delivering both static and dynamic content pages to a network of client systems driving the workload. The benchmark measures the number of simultaneous client connections that a server is able to support, while maintaining predefined bit rate and error rate limits. A connection that meets the bit rate and error limits is said to be "compliant"; the primary statistic of the SPECweb99 benchmark is the number of compliant connections.

The benchmark has been designed to favor a web server that is capable of supporting a larger number of relatively slow connections over a web server that is capable of supporting a smaller number of relatively fast connections. The former situation is regarded as being more representative of the web server environment on the Internet and is one of the improvements made to this benchmark from its predecessor (SPECweb96™).

Another improvement in SPECweb99 over SPECweb96 is the inclusion of dynamic as well as static content. Both dynamic GET and POST operations are performed as part of the SPECweb99 workload. The dynamic GETs simulate the common practice of "rotating" advertisements on a web page. The POSTs simulate entry of user data into a log file on the server, such as might happen during a user registration sequence. Dynamic content comprises 30% of the workload and the remainder is static GETs. The dynamic workload is a mixture of POSTs, GETs, GETs with cookies, and a small fraction is due to CGI GETs. The proportions were based on analysis of workloads of a number of internet web sites.

The file access pattern is also modeled after the access patterns found on a number of internet web sites. Files are divided into a number of classes and a Zipf distribution is used to choose files to access within each class. The total number of bytes accessed increases proportionally to the number of connections to the web server. Further details on the file access and HTTP request type distributions can be found in the SPECweb99 FAQ [SPWBFAQ].

The web server software used is independent of the SPECweb99 benchmark. For our test configurations, we used the Zeus 3.3.5a [Zeus] web server with the tuning suggestions as provided by SPEC at [SPTune]. The dynamic content implementation for the Zeus web server was also obtained from the SPECweb99 site [SPWB99]. This implementation is entirely in the C programming language.

Presentation of SPECweb99 Results

The SPECweb99 benchmark is a licensed benchmark of the SPEC organization and its results can only be

used and reported in certain ways defined by the license. In particular, the SPECweb99 statistic can only be reported for a particular system after the result has been submitted to and approved by the SPECweb99 committee. In this paper, it is not our primary goal to report the SPECweb99 statistic, rather, we are interested in using this workload as a basis for comparing the 2.2.14 and 2.3.99 Linux kernels.

Our use of SPECweb99 thus falls into the category of "research" use. At the time of the writing of this paper, the SPECweb99 license did not include a research use clause, although other SPEC benchmarks do include such a clause. Our results are therefore presented under a special agreement with SPEC [SPNote]; we expect that a research usage clause will become part of the license for the SPECweb99 1.02 release.

The terms of the agreement are as follows:

- We agree to provide full disclosure of the details of the benchmark execution. Any details not otherwise provided in this paper can be obtained from the authors.
- Make compliant runs. That is, all benchmark and workload rules will be adhered to.
- The dynamic content implementation will be one from the SPECweb99 site and is thus an implementation that has been audited for conformance to the benchmark rules by the SPECweb99 committee.
- Only relative results will be reported. In our case all results will be scaled by the result for the 2.2.14 UP kernel at the lowest number of connections tested.
- We will not make use of the SPECweb99 statistic (number of compliant connections).

Although we do not report the SPECweb99 statistic, we do report values for the following quantities, as measured by the SPECweb99 client software:

- Average bitrate/second per connection: The average number of bits delivered per connection in Kb/s.
- Average Latency : the average response time per operation in milliseconds.
- Operations/second: The average number of HTTP operations performed across all connections in operations per second.

Here the averages are performed across all connections and throughout the duration of the measurement run as defined by SPECweb99. As specified by the SPECweb99 rules, each run is performed 3 times and the median result of the runs is reported.

Measurement Environment and Experiment Run Rules

Measurements in this paper for all tests except the Volanomark scalability tests are reported for IBM Netfinity® 7000 M10 systems. These are Intel® Pentium® II Xeon™ 4-way SMP systems. For the Volanomark throughput, Netperf, and FSCache tests, the system being measured was a 400 MHZ system with 1.5 GB of RAM. DASD on this system was accessed using an Adaptec® 7858 SCSI driver. For the SPECweb99 test, the system measured was a 450 MHZ system with 4GB of RAM. System data on this system was accessed using an Adaptec 7858 SCSI driver; the SPECweb99 data is accessed via an IBM ServRAID™ device spanning 9 physical disk drives.

The NIC cards in the Netfinity systems are IBM 10/100 EtherJet™ PCI cards; the Intel 8-way uses an Intel Pro 100 ethernet PCI card. In all cases the device driver was the eepro100 driver.

Volanomark Setup

For the Volanomark throughput tests, the server employed was a 400 MHZ Netfinity system. For the Volanomark scalability tests, the server was an Intel Pentium III Xeon 8-way operating at 550 MHZ. The systems were booted with 1GB of RAM and for the N-way speedup test the system was booted with N processors.

Netperf Network and Client Setup

For these tests the client machine was the 8-way Intel Pentium III 550 MHZ system. This client was connected to the server system using a single, 100 MB Ethernet operating in full duplex mode. *vmstat* was run on the server to measure processor utilization. A network sniffer was used to sample Ethernet utilization to make sure it was not becoming saturated and hence a bottleneck. The server machine was the 400 MHZ Netfinity 7000 M10 system.

FSCache Setup

The FSCache test was executed on a 400 MHZ Pentium II Netfinity system. In order to make sure that our measurements report scalability of the buffer cache and are not limited by the bandwidth of the underlying memory system, we first performed a series of experiments with file-system reads replaced by memory-to-memory copies. The reason for choosing this baseline experiment is that the time required to complete a read operation out of the buffer cache is typically dominated by the amount of time required to copy data from the kernel buffer cache to the user-space buffer. If this memory-to-memory copy operation does not scale then the corresponding FSCache benchmark may not scale. These experiments indicated that for file sizes larger than 1MB on the 400 MHZ Pentium II Netfinity system, the FSCache tests would not scale. Additionally when we ran the full FSCache tests, we found that scalability increased if we used file sizes of 128 KB. For this reason, we chose to use the (relatively small) file size of 128KB in our FSCache tests.

SPECweb99 Network and Client Setup

We are currently using a set of 20 clients and four 100 MB Ethernets to run our SPECweb99 experiments. The clients are connected to the server using a 24-port switch (20 ports for the clients and 4 ports for the server).

The network clients are 166-200 MHZ Pentium Pro machines running Microsoft® NT Workstation 4.0 with Service Pack 4. The NT Performance Monitor is used to ensure that client machines do not become the bottleneck. A network sniffer was used to sample network utilization and to ensure that the network does not become a bottleneck for these tests. To balance the workload across the client machines, the benchmark is configured so that the faster clients submit 1.5 times as many requests as the slower clients.

The server machine is a Netfinity 7000 M10 450 MHZ Pentium II 4 CPUs with 4 IBM EtherJet 10/100 ethernet cards and 4 GB of RAM. Since 2.2.14 does not support more than 2GB of RAM, all the experiments for 2.2.14 and most of the experiments for 2.3.99-pre8 were performed with 2GB of real memory; one data point for the 2.3.99-pre8 SMP case was run with 4GB of RAM.

Khttpd was not enabled for any of the runs reported here.

Statistical Properties of the Test Results

In this paper, numbers reported are based on the average of a number of repeated, identical, independent trials. The trials are done without rebooting the system; however the effect of the previous trial on the system is removed as much as possible before the next trial is started. Where there is a warm up effect that results in the first trial being different than the other trials, the results of the first trial are discarded.

Trials are repeated until the tests have "converged". This is stated as follows: "The test converged to an x% confidence interval width at y% confidence." What this means is that the results of a confidence interval estimated based on a Students-T distribution has a width of less than x% of the mean with a y% level of confidence.

After each trial (except for Volanomark), the length of the confidence interval is calculated, and if the length is small enough, the test completes and the average over the trials is reported as the result. In some cases, the trial does not converge after the maximum allowed number of iterations. In these cases, the test is re-run to obtain convergence. For Volanomark, a fixed number of trials is done, and convergence is tested after all trials are completed.

SPECweb99 comes with its own set of run rules and run acceptance criteria [SPWBFAQ]. The results of SPECweb99 reported here are run under the preprogrammed run rules of the test.

Measurements reported here as UP are for a uniprocessor kernel. Measurements reported as 1P are for multiprocessor kernels booted on a single processor. Comparisons between the 1P and UP measurements are particularly useful for evaluating the overhead of SMP synchronization and locking.

For all of the benchmarks discussed here, we define scalability as the ratio of the benchmark statistic for an SMP system to the corresponding benchmark statistic for a UP system. While this results in lower scalability numbers than one might get by dividing the SMP result by the 1P result (since the UP ratio penalizes the SMP system for locking overhead) we regard this as the fairest way to define scalability.

Measurement Results

Volanomark

In Figure 1 we show measurements of the throughput in messages per second for Volanomark with the IBM R/T 1.1.8 and 4 different Linux Kernels. In all cases the kernels are UP. We note that among the kernels shown, Linux 2.3.99-pre4 has the best performance. This is a good indication that Linux 2.4 should outperform Linux 2.2.14 on workloads similar to Volanomark.

In Figure 2 we show the results of a kernel profile measurement of the Linux 2.3.99-pre4 kernel while it is running Volanomark. As previously reported [JTThreads], this profile shows that the largest amount of system time is spent in the scheduler. While Volanomark is admittedly a stress test for the scheduler, it appears that additional enhancements will need to be added to Linux scheduler for workloads with large numbers of threads.

In Figure 3 we show scalability results for Volanomark when run under IBM R/T 1.1.8 and IBM R/T 1.3 for Linux kernels 2.2.14 and 2.3.99-pre4. This figure shows that while running Volanomark, the IBM R/T 1.3 scales better than IBM R/T 1.1.8, and that Linux 2.3.99-pre4 scales better than Linux 2.2.14. Of course, the speedup numbers obtained here are application and environment dependent and should not be taken as general statements of the speedup results an arbitrary workload might achieve while running under the IBM R/T 1.3.

Netperf

In Figure 4, we show the results of a Netperf comparison test between 2.2.14-5.0 (the Red Hat 6.2 Kernel) and 2.3.99-pre8. The horizontal axis represents the number of client threads used to drive the server; each client thread creates one connection to the server. The vertical axis represents the number of Netperf messages sent per second during the test. The four lines on the chart represent data for 2.2.14-5.0 UP and SMP and 2.3.99-pre8 UP and SMP. Note that the benchmark running on the Linux 2.2.14.-5.0 SMP kernel with 4 processors runs more slowly than the benchmark running on the Linux 2.2.14-5.0 UP kernel. Performance of the 2.3.99-pre8 UP kernel is better than the performance of *both* the UP and SMP trials for Linux 2.2.14-5.0. The Linux 2.3.99-pre8 SMP kernel performs dramatically better than all the other kernels and gives SMP scalability of 2.1 on this 4-processor system. Comparing the 2.3.99-pre8 SMP result at 40 connections to the 2.2.14 SMP result at 16 connections (these are the maximum throughputs for each case) shows that the 2.3.99 SMP kernel is able to deliver 3.1x times as many messages as the 2.2.14 SMP kernel on this benchmark.

In Figure 5, we show the results of time-based and instruction-based profiles of the Linux 2.3.99-pre8 kernel while running the Netperf benchmark. The time-based profile shows that the scheduler is again the kernel routine where the most time is spent during the benchmark. Given the very small message sizes used (4 bytes), this is not surprising. The differences between the two profiles indicate that in some routines (scheduler and stext) more instructions are executed per unit time whereas for other routines (speedo_start_xmit) relatively fewer instructions are executed per unit time. These are examples of the kind of differences one can see using a performance-counter event-based profile versus a timer-based profile.

FSCache

In Figure 6, we show FSCache scalability results for Linux 2.2.14 and 2.3.99-pre6. (The initial drop in scalability when going from the UP case to the 1P case reflects the overhead of SMP locking.) For the 2.2.14 case, the size of the buffer used had relatively little effect on the results so we only show the 4096 byte buffer case. For 2.3.99, the size of the buffer did make a significant difference; we show the results for 512 and 4096 byte buffers. As can be seen from Figure 6, Linux 2.2.14 provides no significant performance by adding additional processors for this benchmark, while we see that Linux 2.3.99-pre6 provides as much as a 2.5x increase on this 4-processor system.

SPECweb99

In Figures 7 through 12, we show the results of our SPECweb99 experiments. Figure 7 shows that the bitrate per connection falls off rapidly as the number of connections is increased for all cases except 2.3.99 SMP where only a slight decrease is detected. Similarly, Figure 8 shows that the response latency increases for all cases except 2.3.99 SMP where only a slight increase is detected. In Figure 9, we see that the operations/second handled by the server increases linearly (with the offered load) for the 2.3.99 SMP case

and for all other cases it does not. Finally, in Figure 10 we see an underlying reason for these results. In all cases except 2.3.99 SMP, the system is CPU bound and cannot support additional work.

A plausible question, when comparing the CPU utilization curves for 2.2.14 SMP and 2.3.99 SMP is "Where did all of the extra CPU time come from for 2.3.99?". In Figures 11 and 12, we answer this question using a time-based profile of the kernel while it is running the benchmark. In Figure 11 we see that about 50% of the CPU-time consumed by the kernel was spent in stext_lock. (stext_lock appears in the profile when the kernel is spinning on a spinlock, so about 50% of the time was unavailable to service web requests). Figure 12 shows that less than 4% of the time was spent spinning for locks while running the benchmark under 2.3.99 SMP.

For Figures 7 through 10, the curves drawn represent data for 2GB of RAM. We also include a single data point for 2.3.99-pre8 and 4GB of RAM at 820 connections. This point indicates that the 2.3.99 SMP experiment at 820 connections and 2GB RAM may be memory bound. In Figure 9, the 4 GB data point appears to follow the linear trend established by the 420 through 660 connection data points. Similarly, if we examine the CPU utilization graph in Figure 10, it is apparent that with 4 GB of memory, the system is more fully utilized. If we compare operations per second achieved under 2.2.14 SMP at 500 connections and 2.3.99 SMP at 820 connections and 4 GB RAM (see Figure 9), we see that the latter kernel is able to deliver 1.85 as many operations per second as the former.

At 820 connections, our client network is fully utilized. Thus we are unable to state with certainty that we have reached the peak of the throughput curve for 2.3.99 SMP at 820 connections. We are in the process of obtaining a Gigabit ethernet switch in order to continue these experiments and find a final scaling number for 2.3.99 under this benchmark.

Concluding Remarks

We have presented measurements that show that the SMP scalability of the Linux 2.4 kernel should be significantly better than that of the Linux 2.2.14 kernel. In particular it appears that:

- Linux 2.4 performs better than Linux 2.2.14 for the Volanomark benchmark when run with IBM R/T 1.1.8.
- It remains to be seen whether or not the Linux 2.4 scheduler will still require additional scalability tuning for workloads that require many running threads.
- The scalability of the TCP/IP stack for a multiprocessor Linux 2.4 system is dramatically improved over the scalability of the TCP/IP stack for 2.2.14. This improvement alone should make the Linux 2.4 kernel perform substantially better under benchmarks such as the ones performed by Mindcraft [Mindcraft]. Our Netperf experiments showed that an SMP scalability of 2.1 out of 4 is possible with Linux 2.3.99-pre6. This benchmark performs 3.1 times better under 2.3.99-pre6 SMP than it does on 2.2.14 SMP.
- The SMP scalability of file system buffer cache access has been significantly improved. Our FSCache experiments demonstrated that a scalability of 2.5 out of 4 is possible with Linux 2.3.99-pre6.
- Our SPECweb99 experiments indicate that 2.3.99-pre8 SMP (running the Zeus web server) is able to deliver at least 1.85 times as many operations per second than 2.2.14 SMP. However, we do not yet have a final scalability number for 2.3.99 due to limitations of our current client network.

While these results indicate that Linux 2.4 should provide improved SMP scalability over that available in the current production kernels, much additional work remains before Linux can effectively exploit SMP systems larger than 4-way.

Acknowledgments

Jerry Burke, Scottie M. Brown and George Tracy of IBM were instrumental in setting up the SPECweb99 benchmark and we greatly appreciate their assistance. We also acknowledge the SPEC organization (Particularly Kaivalya Dixit and Paula Smith) for its permission to use the SPECweb99 benchmark in this paper.

References

[Linux2.4]: Wonderful World of Linux 2.4 (Final Draft), Joe Pranevich, http://linuxtoday.com

/news_story.php3?Ltsn=2000-07-17-014-04-NW-LF-KN

[Mindcraft]: Open Benchmark: Windows NT Server 4.0 and Linux, Bruce Weiner, http://www.mindcraft.com/whitepapers/openbench1.html

[ESR Fiasco]:ESR and the Mindcraft Fiasco http://www.slashdot.org/features/99/04/23/1316228.shtml

[SGI Lockmeter]:Kernel Spinlock Metering for Linux, http://oss.sgi.com/projects/lockmeter

[SGI Kernprof]: Kernel Profiling, http://oss.sgi.com/projects/kernprof

[PerfCount]: Intel Architecture Software Developer's Manual Volume 3: System Programming, _http://developer.intel.com/design/pentiumii/manuals/243192.htm

[JTThreads]: Java technology, threads, and scheduling in Linux--Patching the kernel scheduler for better Java performance, Ray Bryant, Bill Hartner, IBM, http://www-4.ibm.com/software/developer/library/java2/index.html

[Volano]: Volano Java Chat Room, Volano LLC http://www.volano.com

[VReport]: VolanoMark Report page, Volano LLC, http://www.volano.com/report.html

[VMark]: Volano Java benchmark, Volano LLC, http://www.volano.com/benchmarks.html

[NetPerf]: Network Benchmarking NetPerf, http://www.netperf.org

[SPWB99]: Web Server benchmarking SPECweb99, Standard Performance Evaluation Corporation, http://www.spec.org/osg/web99

[SPWBFAQ]: SPECweb99 FAQ, Standard Performance Evaluation Corporation, http://www.spec.org/osg/web99/docs/faq.html

[Zeus]: Zeus WebServer, Zeus Technology, http://www.zeustech.net

[SPTune]: SPECweb99 Tuning Description, Standard Performance Evaluation Corporation, http://www.spec.org /osg/web99/tunings

[SPNote]: E-mail communication, Kaivalya Dixit, President of SPEC, the Standard Performance Evaluation Corporation, and Paula Smith, Chair of the SPECweb99 committee, 7/10/2000.

Trademark and Copyright Information

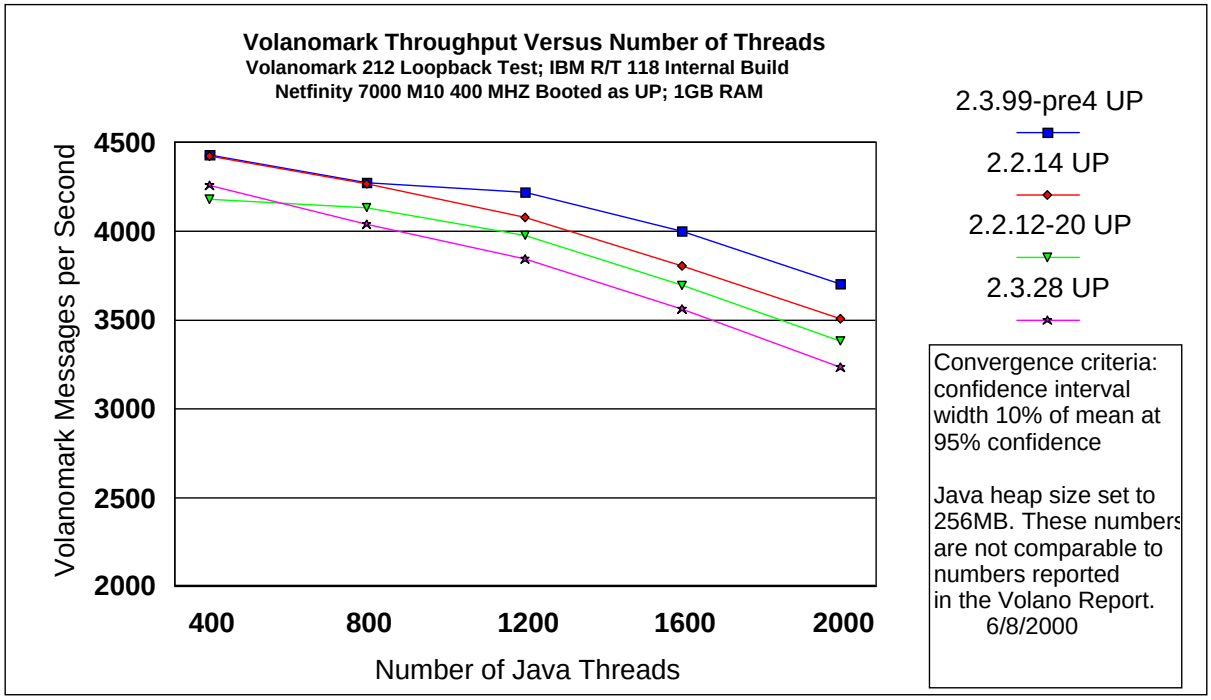

Figure 1

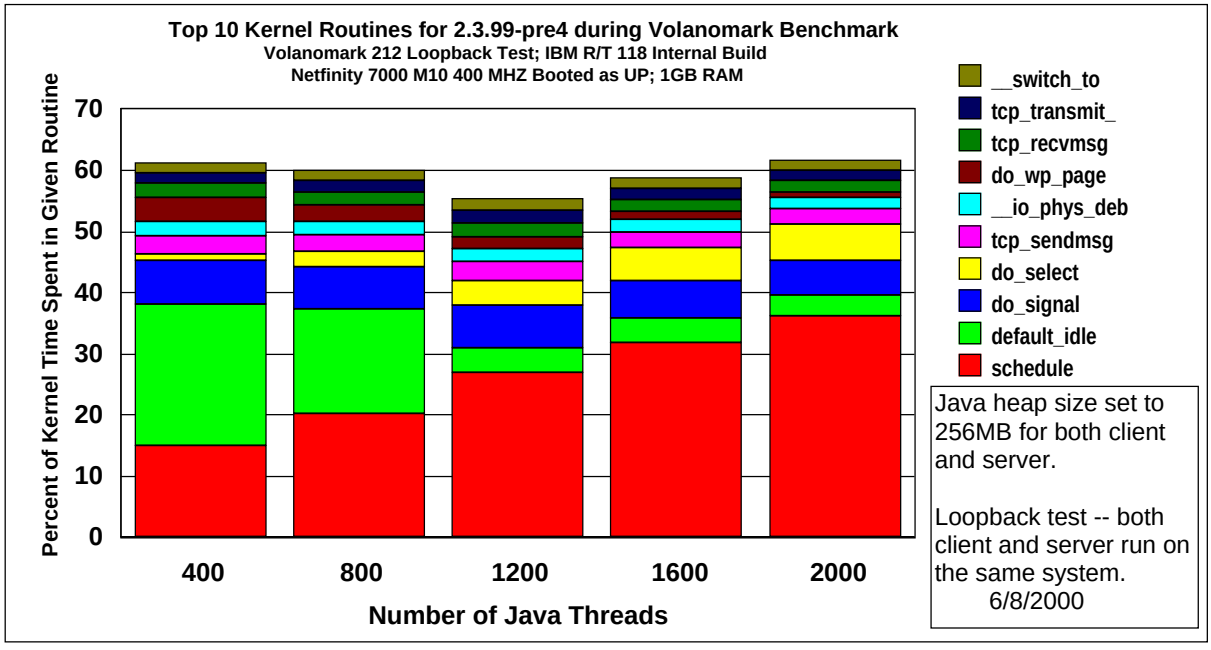

Figure 2

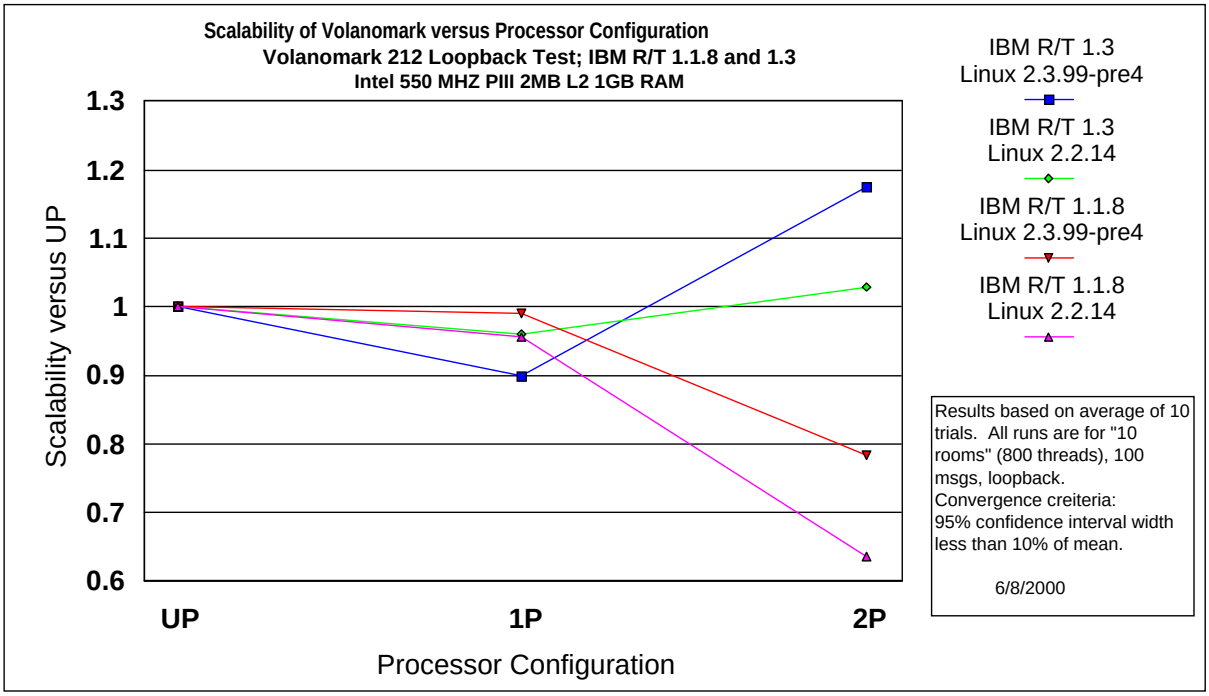

Figure 3

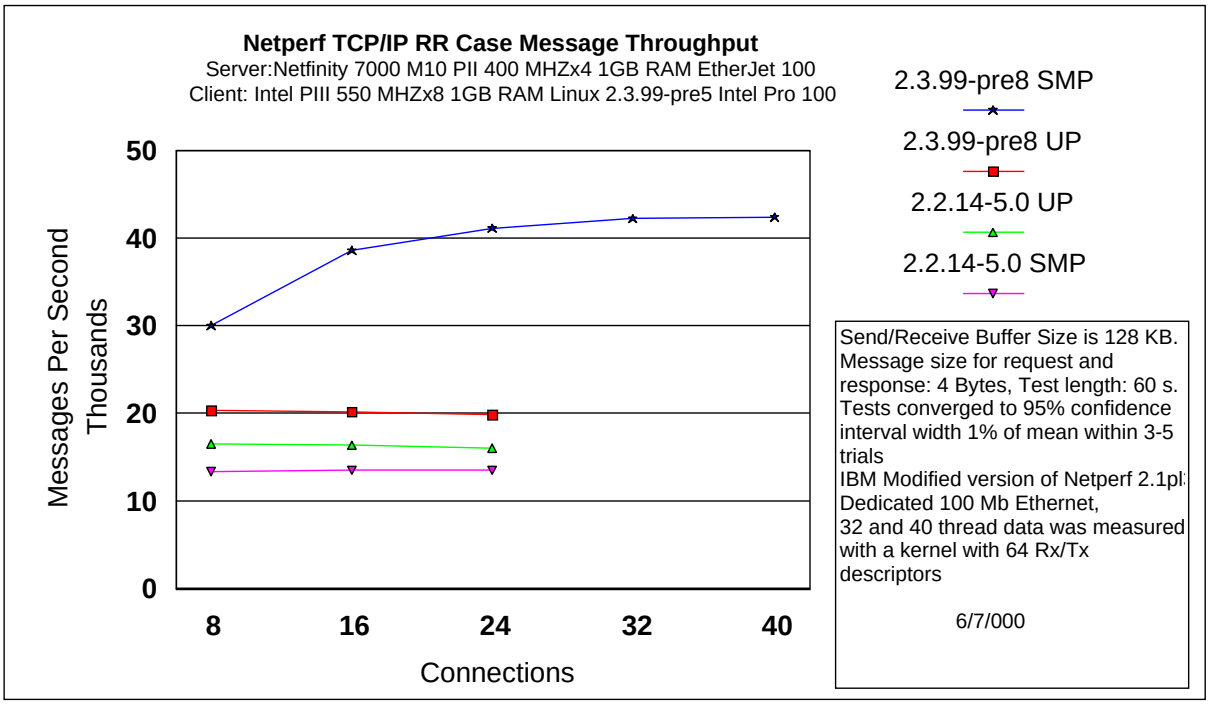

Figure 4

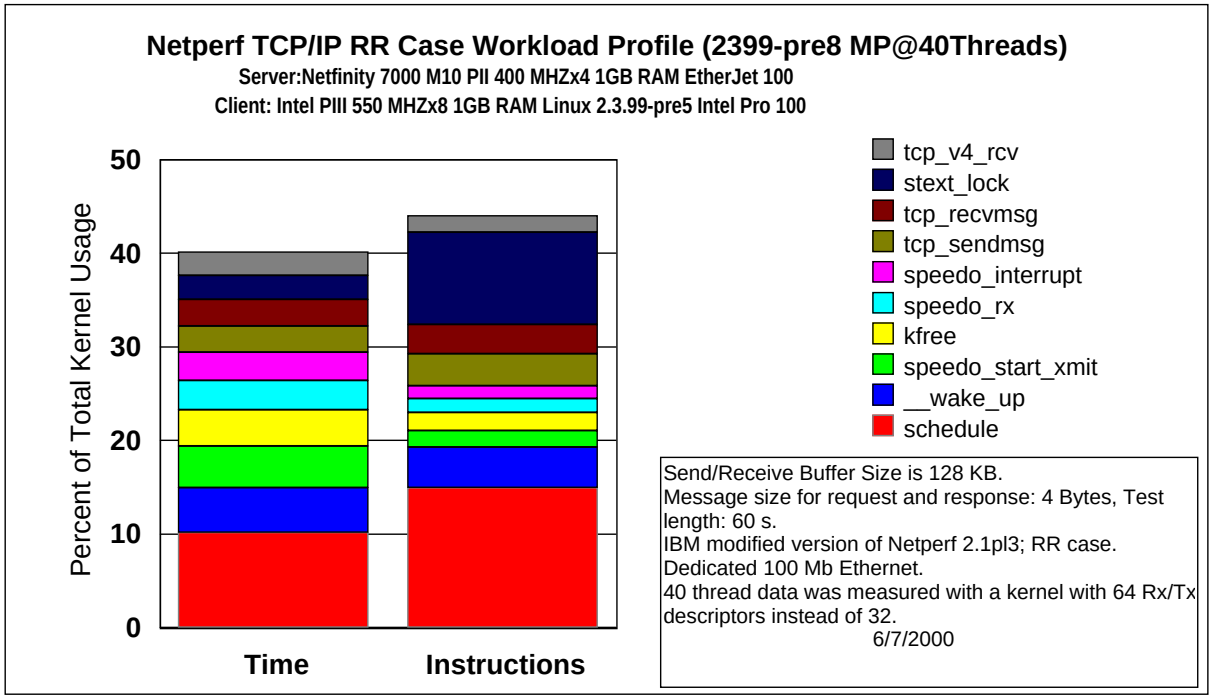

Figure 5

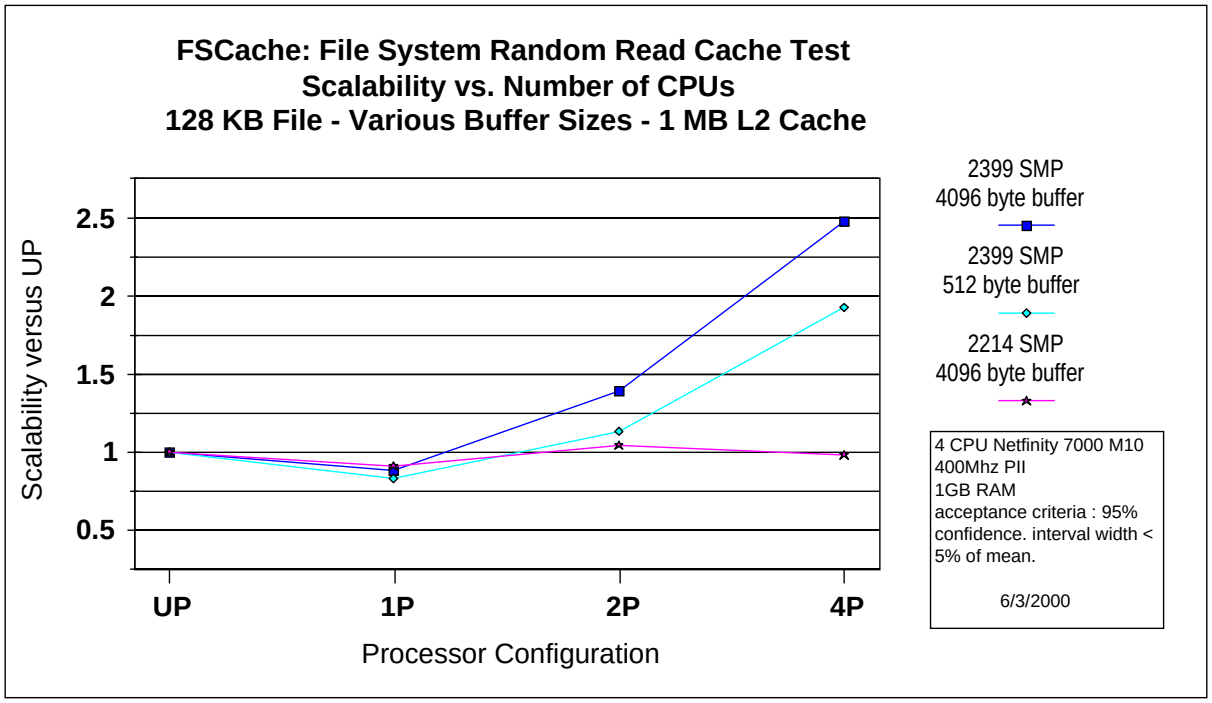

Figure 6

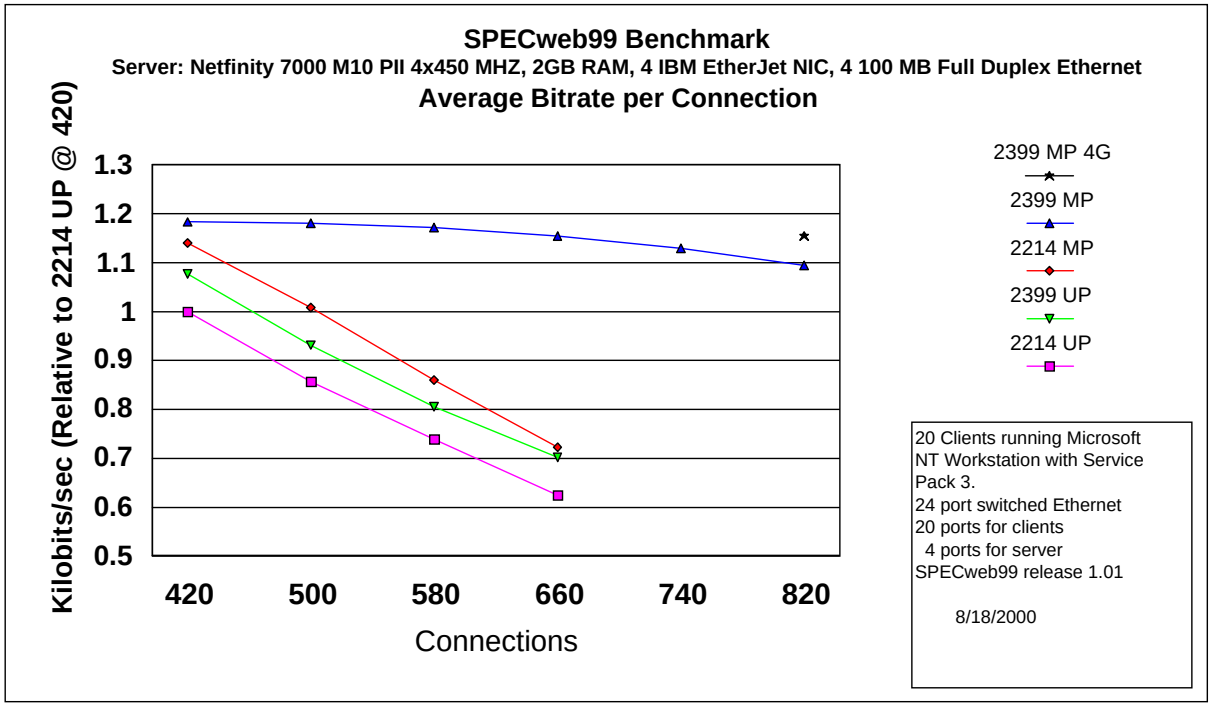

Figure 7

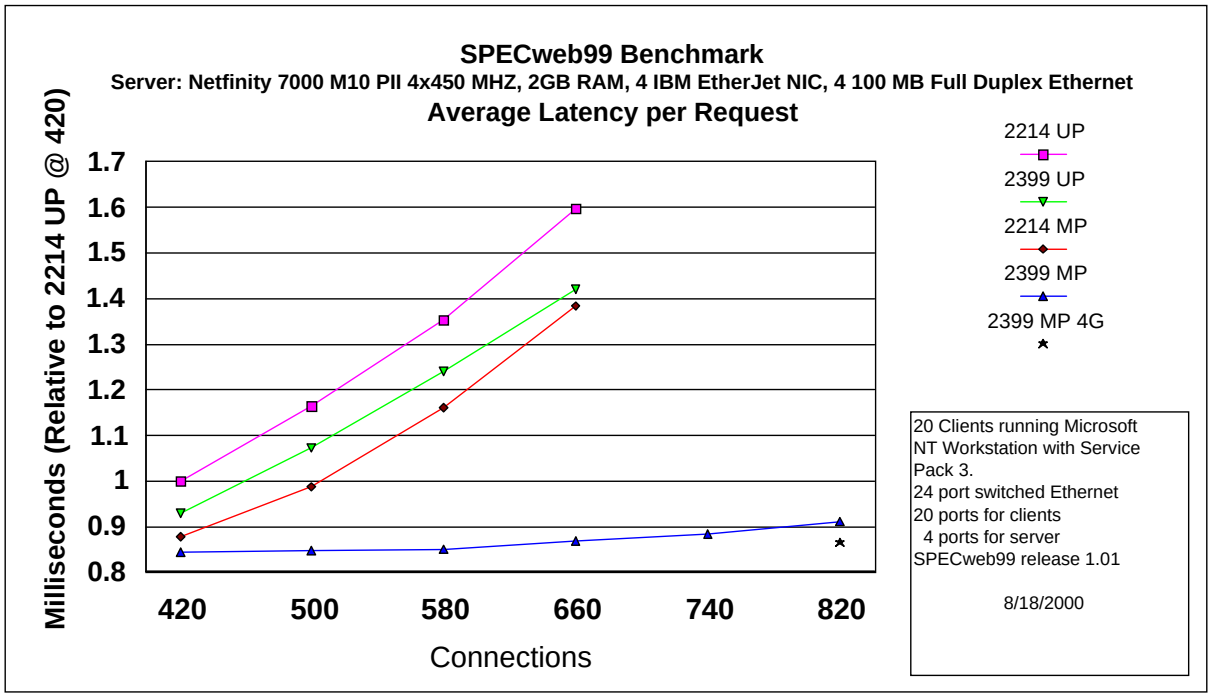

Figure 8

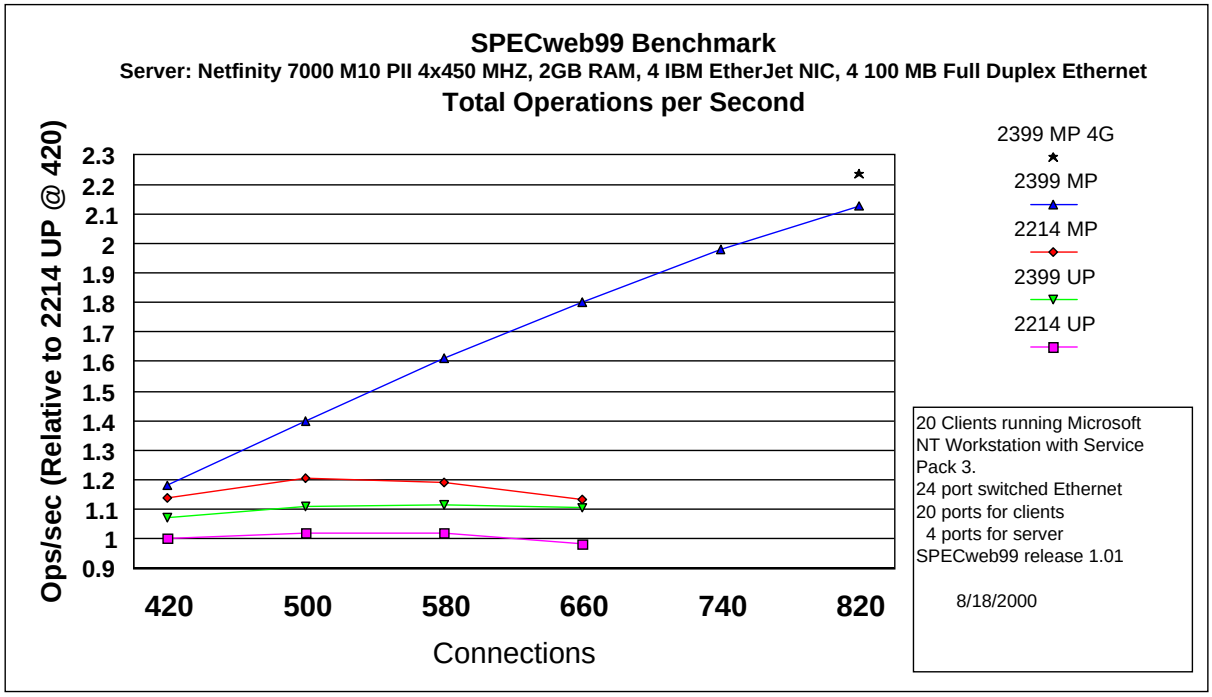

Figure 9

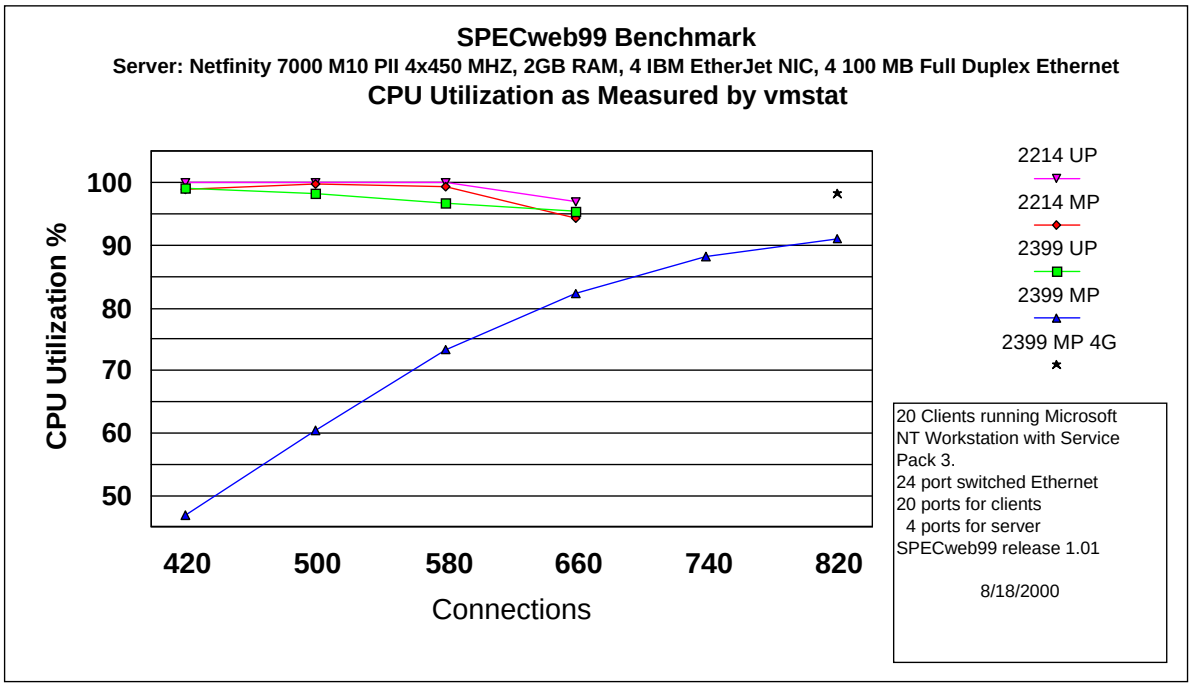

Figure 10

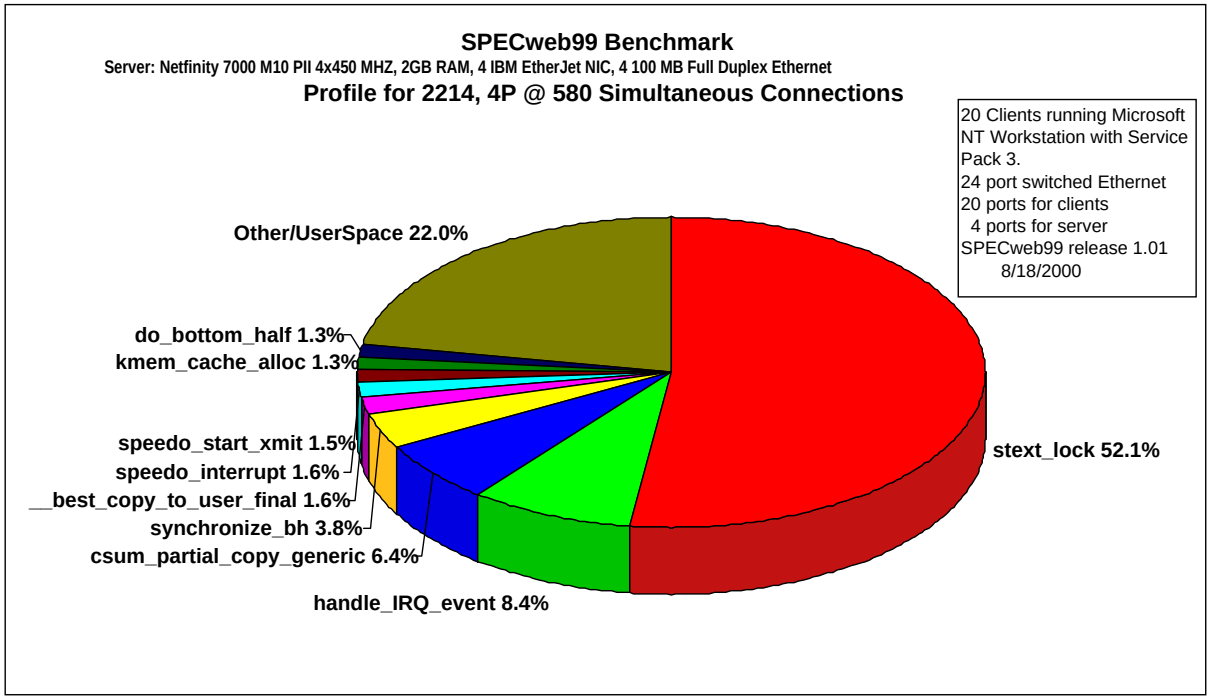

Figure 11

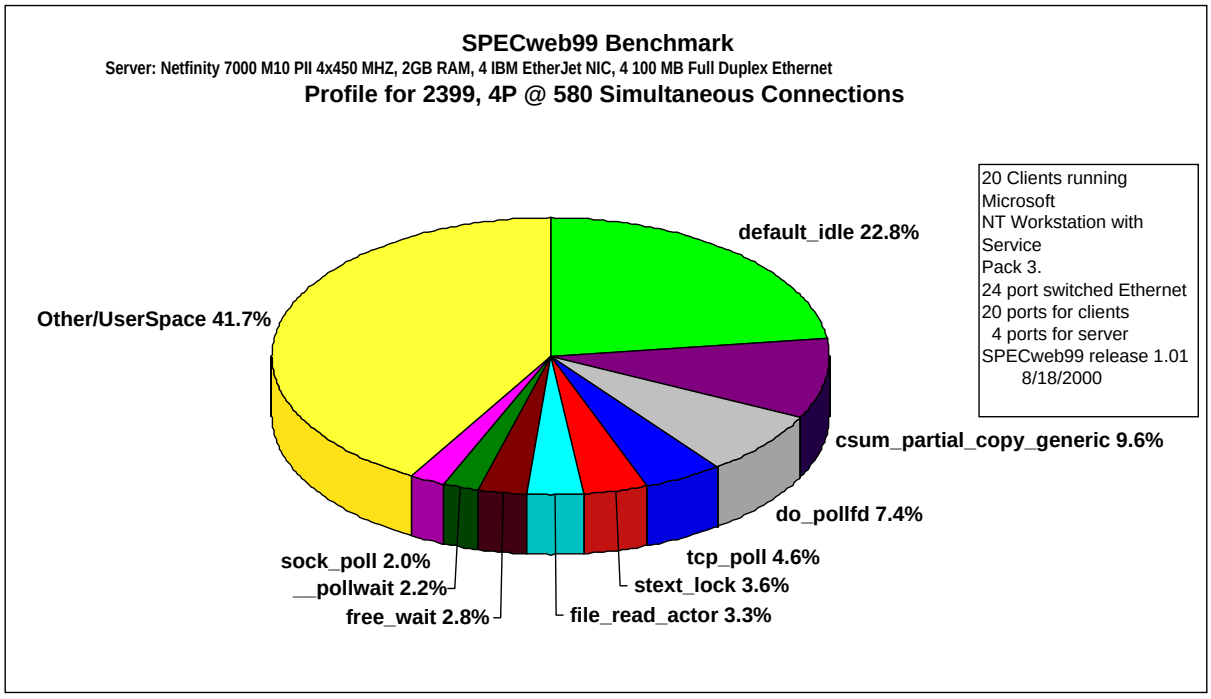

Figure 12