The Volano Report

John Neffenger

May 30, 2003

1 Introduction

This report provides test results of the VolanoMark™ version 2.5 benchmark program on 17 Java virtual machines and four operating systems. VolanoMark is a pure Java server benchmark characterized by long-lasting network connections and high thread counts. Since 1997, VolanoMark has accurately predicted the performance and connection limitations of the various Java platforms for customers running VolanoChat in their production environments.

See the VolanoMark Benchmark[1] page for background information and instructions on how to run these tests yourself. The VolanoMark benchmark creates client connections in groups of 20 and measures the time required by the clients to take turns broadcasting a set of messages to the group. At the end of the test, it reports a score as the average number of messages transferred by the server per second. All tests ran identical copies of VolanoMark 2.5 on identical hardware. Because the tests compare Java version 1.3 and 1.4 virtual machines, the benchmark uses the synchronous Java input and output methods available in both versions and does not use the `java.nio` package found only in Java 1.4.

2 Performance

The performance tests were executed over the local loopback interface with the Java command options shown in Figure 2. The operating system was rebooted before each set of tests for a particular Java virtual machine. The client benchmark ran four times, but the VolanoMark server was not restarted before each run of the client benchmark.

Figure 1 shows the final score for each Java platform. Table 1 shows the test results in detail. Each test result is the server throughput in messages per second with 200 concurrent loopback connections. The score is the average of the last three results. Bigger numbers are faster.

BEA JRockit 8.1 on Linux and Blackdown Java 1.4.1 on FreeBSD failed to run the tests, probably due to incompatibility with the operating system versions I chose. BEA JRockit 8.1 is supported on Red Hat Enterprise Linux AS/ES/WS 2.1 (kernel 2.4.9, glibc 2.2.4) but not on the Red Hat Linux 8.0 system I used (with kernel 2.4.18, glibc 2.3.2). BEA JRockit on Linux seems to be much more dependent on specific Linux kernel and glibc versions than the Java ports for Linux from Blackdown, IBM, and Sun. The Blackdown Java ports are written for

[1] `http://www.volano.com/benchmarks.html`

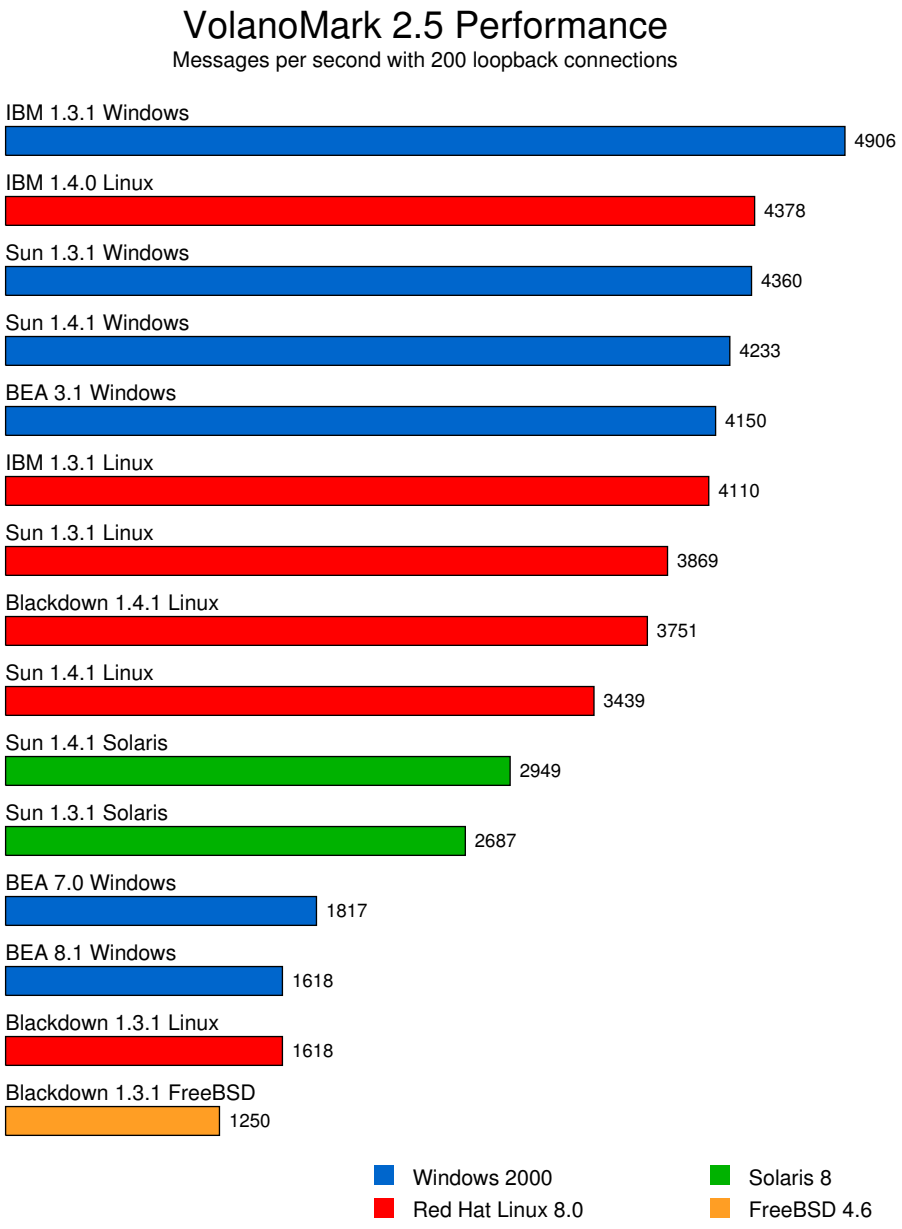

Figure 1: VolanoMark 2.5 loopback performance scores

Java Platform	Test Results				Score
IBM 1.3.1 Windows	4647	4892	4886	4941	4906
IBM 1.4.0 Linux	4080	4291	4478	4364	4378
Sun 1.3.1 Windows	4160	4325	4365	4390	4360
Sun 1.4.1 Windows	3983	4125	4307	4268	4233
BEA 3.1 Windows	4088	4102	4171	4177	4150
IBM 1.3.1 Linux	3918	4112	4110	4109	4110
Sun 1.3.1 Linux	3727	3851	3873	3882	3869
Blackdown 1.4.1 Linux	3513	3619	3783	3850	3751
Sun 1.4.1 Linux	3296	3364	3540	3414	3439
Sun 1.4.1 Solaris	2835	2937	2947	2963	2949
Sun 1.3.1 Solaris	2595	2665	2699	2697	2687
BEA 7.0 Windows	1820	1817	1815	1818	1817
BEA 8.1 Windows	1464	1556	1649	1650	1618
Blackdown 1.3.1 Linux	1620	1597	1630	1628	1618
Blackdown 1.3.1 FreeBSD	1266	1252	1251	1248	1250
BEA 8.1 Linux	Hangs with low CPU usage (~ 6%)				
Blackdown 1.4.1 FreeBSD	HotSpot Virtual Machine Error				

Table 1: VolanoMark 2.5 loopback performance results (messages per second)

Linux, so it is not too surprising that the Java 1.4.1 port is not yet working under the Linux binary compatibility of FreeBSD.

3 Network Scalability

The network scalability tests were executed over a 100-Mbps Ethernet cross-over cable connection with the Java command options shown in Figure 4. The operating systems on both sides were rebooted before each set of tests for a particular Java virtual machine. The VolanoMark server was not restarted between each run of the client benchmark at increasing connection levels.

Table 2 and Table 3 show the test results in detail. Each test result is the server throughput in messages per second for the corresponding number of concurrent connections, starting with 1,000 connections and increasing by 1,000 up to 10,000 connections. If the Java platform failed to run all ten tests, the error which caused the failure is shown in the last column. Figure 3 shows the final score for each Java platform. The score is the maximum connection level completed without errors. Bigger numbers are better and indicate greater network scalability. The order of the Java platforms is preserved from their performance scores.

4 Tricks

The main trick in getting these Java platforms to handle the required number of threads is to reduce the size of the thread stack with the -Xss option. I generally set the stack size to 32 kilobytes when permitted by the Java virtual machine on startup. Otherwise, I set the stack size to the minimum required by the virtual

BEA 3.1 Windows -Xgc:gencopy -Xthinthreads -Xmx64m

BEA 7.0 Windows -Xgc:gencopy -Xthinthreads -Xmx64m

BEA 8.1 Linux -Xgc:gencopy -Xthinthreads -Xmx64m

BEA 8.1 Windows -Xgc:gencopy -Xthinthreads -Xmx64m

Blackdown 1.3.1 FreeBSD -green -Xmx64m

Blackdown 1.3.1 Linux -green -Xmx64m

Blackdown 1.4.1 FreeBSD -server -Xmx64m

Blackdown 1.4.1 Linux -server -Xmx64m

IBM 1.3.1 Linux -Xmx64m

IBM 1.3.1 Windows -Xmx64m

IBM 1.4.0 Linux -Xmx64m

Sun 1.3.1 Linux -server -Xmx64m

Sun 1.3.1 Solaris -server -Xconcurrentio -Xmx64m

Sun 1.3.1 Windows -server -Xmx64m

Sun 1.4.1 Linux -server -Xmx64m

Sun 1.4.1 Solaris -server -Xconcurrentio -Xmx64m

Sun 1.4.1 Windows -server -Xmx64m

Figure 2: Command options for the loopback performance tests

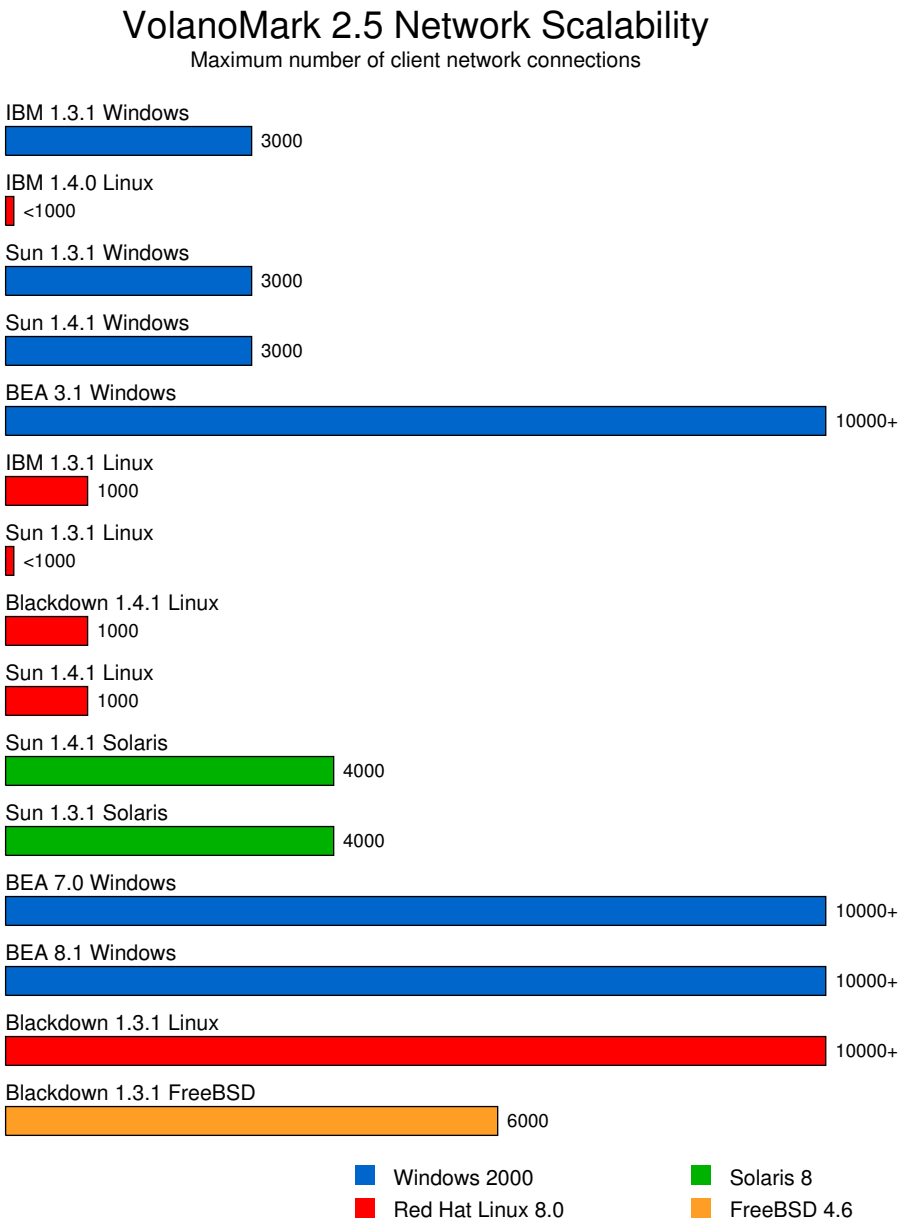

Figure 3: VolanoMark 2.5 network scalability scores

Java Platform	1000	2000	3000	4000	5000
IBM 1.3.1 Windows	5473	6234	5132		
IBM 1.4.0 Linux					
Sun 1.3.1 Windows	5263	5482	4282		
Sun 1.4.1 Windows	5457	5997	5899		
BEA 3.1 Windows	5939	5045	4568	4713	3993
IBM 1.3.1 Linux	4303				
Sun 1.3.1 Linux					
Blackdown 1.4.1 Linux	4306				
Sun 1.4.1 Linux	3099				
Sun 1.4.1 Solaris	3452	4513	4364	3877	
Sun 1.3.1 Solaris	2840	3344	3093	2899	
BEA 7.0 Windows	3097	1485	1065	787	579
BEA 8.1 Windows	1311	1664	1064	776	592
Blackdown 1.3.1 Linux	2047	1289	1188	668	789
Blackdown 1.3.1 FreeBSD	1574	901	642	556	577
BEA 8.1 Linux					
Blackdown 1.4.1 FreeBSD					

Table 2: VolanoMark 2.5 network scalability results (messages per second): 1,000–5,000 concurrent connections

Java Platform	6000	7000	8000	9000	10000
IBM 1.3.1 Windows					
IBM 1.4.0 Linux					
Sun 1.3.1 Windows					
Sun 1.4.1 Windows					
BEA 3.1 Windows	4352	3428	3069	3646	3563
IBM 1.3.1 Linux					
Sun 1.3.1 Linux					
Blackdown 1.4.1 Linux					
Sun 1.4.1 Linux					
Sun 1.4.1 Solaris					
Sun 1.3.1 Solaris					
BEA 7.0 Windows	469	349	293	255	224
BEA 8.1 Windows	460	346	292	252	223
Blackdown 1.3.1 Linux	513	451	283	446	246
Blackdown 1.3.1 FreeBSD	437				
BEA 8.1 Linux					
Blackdown 1.4.1 FreeBSD					

Table 3: VolanoMark 2.5 network scalability results (messages per second): 6,000–10,000 concurrent connections

BEA 3.1 Windows -Xgc:gencopy -Xthinthreads -Xmx256m -Xss32k

BEA 7.0 Windows -Xgc:gencopy -Xthinthreads -Xmx256m -Xss32k

BEA 8.1 Linux -Xgc:gencopy -Xthinthreads -Xmx256m -Xss32k

BEA 8.1 Windows -Xgc:gencopy -Xthinthreads -Xmx256m -Xss32k

Blackdown 1.3.1 FreeBSD -green -Xmx256m -Xss32k

Blackdown 1.3.1 Linux -green -Xmx256m -Xss32k

Blackdown 1.4.1 FreeBSD -server -Xmx256m -Xss96k

Blackdown 1.4.1 Linux -server -Xmx256m -Xss96k

IBM 1.3.1 Linux -Xmx256m -Xss32k

IBM 1.3.1 Windows -Xmx256m -Xss32k

IBM 1.4.0 Linux -Xmx256m -Xss32k

Sun 1.3.1 Linux -server -Xmx256m -Xss32k

Sun 1.3.1 Solaris -server -Xconcurrentio -Xmx256m -Xss64k

Sun 1.3.1 Windows -server -Xmx256m -Xss32k

Sun 1.4.1 Linux -server -Xmx256m -Xss96k

Sun 1.4.1 Solaris -server -Xconcurrentio -Xmx256m -Xss64k

Sun 1.4.1 Windows -server -Xmx256m -Xss32k

Figure 4: Command options for the network scalability tests

Java Platform	Level	Error
IBM 1.3.1 Windows	3,000	OutOfMemoryError
IBM 1.4.0 Linux	0	Segmentation fault
Sun 1.3.1 Windows	3,000	OutOfMemoryError
Sun 1.4.1 Windows	3,000	OutOfMemoryError
BEA 3.1 Windows	10,000	
IBM 1.3.1 Linux	1,000	OutOfMemoryError
Sun 1.3.1 Linux	0	OutOfMemoryError
Blackdown 1.4.1 Linux	1,000	Hangs (0% CPU)
Sun 1.4.1 Linux	1,000	OutOfMemoryError
Sun 1.4.1 Solaris	4,000	Segmentation fault
Sun 1.3.1 Solaris	4,000	Segmentation fault
BEA 7.0 Windows	10,000	
BEA 8.1 Windows	10,000	
Blackdown 1.3.1 Linux	10,000	
Blackdown 1.3.1 FreeBSD	6,000	Hangs (0% CPU)
BEA 8.1 Linux	0	Hangs (~ 6% CPU)
Blackdown 1.4.1 FreeBSD	0	HotSpot VM Error

Table 4: VolanoMark 2.5 network scalability test: highest successful level and subsequent error

machine: 64 kilobytes for the HotSpot Server VM on Solaris and 96 kilobytes for the version 1.4.1 HotSpot Server VM on Linux and FreeBSD.

4.1 JRockit on Windows

BEA JRockit has a unique High Performance Threading System, also called *Thin Threads*, which is enabled by the -Xthinthreads option. By mapping all Java threads onto a smaller set of native threads, BEA JRockit is the only Java virtual machine that can scale past the 4,000-connection barrier of VolanoMark on Windows. BEA JRockit developers recommended that I use the -Xgc:gencopy option to select the Generational Copying Garbage Collector, and my own tests have shown it to be the best choice for VolanoMark as well.

Table 5 shows the performance differences between the BEA native and Thin Thread implementations. With the large difference in performance between these two models in JRockit versions 7.0 and 8.1, BEA now forces you to choose between superb network scalability (with Thin Threads) and superb performance (with native threads). With JRockit 3.1, we got both at the same time. Unfortunately, JRockit 3.1 is no longer available from BEA.

4.2 HotSpot on Solaris

Server-side Java on Solaris has never quite been the same since Sun switched from the ExactVM in Java version 1.2.2 on Solaris to the HotSpot VM in Java version 1.3 and later. Yet there are some tricks for the HotSpot VM that can help make up the difference. You can find important information and excellent suggestions at the Sun Web site, Performance Documentation for the Java HotSpot

Java Platform	Threading	Test Results				Score
BEA 3.1 Windows	Native	4103	4110	4156	4183	4150
BEA 3.1 Windows	Thin	4088	4102	4171	4177	4150
BEA 7.0 Windows	Native	3799	3791	3790	3797	3793
BEA 7.0 Windows	Thin	1820	1817	1815	1818	1817
BEA 8.1 Windows	Native	3791	3786	3794	3777	3786
BEA 8.1 Windows	Thin	1464	1556	1649	1650	1618

Table 5: Performance comparison of JRockit threading options (messages per second)

Java Platform	Threading	Test Results				Score
Sun 1.2.2 Solaris	Default	3215	3171	3175	3142	3163
Sun 1.3.1 Solaris	Default	1779	1799	1882	1786	1822
Sun 1.3.1 Solaris	/usr/lib/lwp	2132	2123	2158	2135	2139
Sun 1.3.1 Solaris	-Xconcurrentio	2595	2665	2699	2697	2687
Sun 1.4.1 Solaris	Default	2619	2797	2799	2817	2804
Sun 1.4.1 Solaris	/usr/lib/lwp	2688	2751	2812	2807	2790
Sun 1.4.1 Solaris	-Xconcurrentio	2835	2937	2947	2963	2949

Table 6: Performance comparison of Solaris threading options (messages per second)

VM.[2] In particular, the Solaris HotSpot VM has an option called -Xconcurrentio. It apparently does two things: it enables lightweight process-based (LWP-based) synchronization rather than the Solaris thread-based synchronization, and it disables thread-local allocation buffers (TLABs). LWP-based synchronization is the default in the Java 1.4 HotSpot VM, but must be enabled for the Java 1.3 HotSpot VM. Thread-local allocation buffers are enabled by default only in the Solaris SPARC Edition of the HotSpot Server VM. Note that all tests for this report were performed on the Intel Edition of Solaris.

Table 6 shows the performance differences of the Solaris HotSpot Server VM running with its default threading and synchronization options, using the alternative one-to-one thread library found in `/usr/lib/lwp`, and specifying the -Xconcurrentio option. The performance results of the Sun version 1.2.2_12 ExactVM on Solaris (native threads, sunwjit) are also included for comparison. The -Xconcurrentio option gives a big performance boost for VolanoMark, with little or no reduction in the network scalability. All combinations of the HotSpot Server VM on Solaris managed to complete the network tests up to 4,000 concurrent connections without errors, while the Sun 1.3.1 Solaris virtual machine with its default threading options completed the tests up to 6,000 concurrent connections.

[2] `http://java.sun.com/docs/hotspot/index.html`

5 Conclusions

When choosing the best Java platform for network servers similar to VolanoMark (and VolanoChat), you can ignore the performance scores. All of the Java platforms which complete the performance test are fast enough to support high-traffic Web sites on recent hardware, such as a 500 MHz Intel Pentium III with 256 megabytes of memory. What matters most is the connection capacity.

At 10,000 connections, BEA JRockit 3.1 on Windows has 14 times the throughput of any other Java platform—by far, the best network scalability I have ever tested. While other Java vendors waited for better threading support in the operating system or new programming interfaces for the application, the JRockit team solved the Java threads problem right where it originated. The results are remarkable, and BEA made a wise purchase.

BEA JRockit 3.1 is no longer available, so the best Java platform today is Blackdown 1.3.1 on Linux. The Blackdown 1.3.1 port scales easily to 10,000 connections, keeping pace with the latest from BEA even without a just-in-time compiler. For years, the Blackdown Project has taken the Sun Java source code and made it better. They provide fixes to bugs that Sun won't fix;[3] they provide security updates to known vulnerabilities[4] months before the official updates from Sun; and real people provide assistance on a lively mailing list. Finding that level of support from other Java vendors is difficult, even when you're paying for it. The volunteers at Blackdown are a great asset to the Java community.

The good news is that Dan Kegel's C10K Problem[5] has been solved in Java 1.3, even when using the original blocking Java input and output methods, and even when using modest hardware. The bad news is that the Java vendors seem to have abandoned that achievement in Java 1.4. Blackdown 1.4.1 no longer offers the `-green` Classic VM option that allows so many connections with the 1.3.1 release. The BEA High Performance Threading System (Thin Threads) is no longer so highly performing in JRockit versions 7.0 and 8.1 compared to the earlier 3.1 release. Tower Technology, with its fast and scalable TowerJ product, simply went out of business. However, there is still cause for hope. The *new I/O* model in Java 1.4 allows some server applications to reduce their thread count to single digits; the Native POSIX Thread Library[6] in Red Hat Linux 9 promises to solve the problems of highly-threaded Java applications on Linux once and for all; and of course, increases in hardware performance have made some of these problems simply evaporate. With these changes in hand, Java developers can start making headway in the C100K Problem.

A Hardware

The server machine under test is a Dell OptiPlex GX1p with a 500 MHz Intel Pentium III processor, 512 kilobytes of L2 cache, 384 megabytes of 100 MHz ECC SDRAM, a 3Com EtherLink 10/100 PCI TX NIC (3C905B-TX), and Phoenix ROM BIOS PLUS Version 1.10 A08.

[3] `http://developer.java.sun.com/developer/bugParade/bugs/4427986.html`
[4] `http://www.volano.com/security/volano-2001-1.html`
[5] `http://www.kegel.com/c10k.html`
[6] `http://people.redhat.com/drepper/nptl-design.pdf`

The client side driving the network scalability tests is a Dell OptiPlex GX110 with a 1 GHz Intel Pentium III processor, 256 kilobytes of full-speed L2 cache, 512 megabytes of 100 MHz non-ECC SDRAM, a 3Com EtherLink 10/100 PCI TX NIC (3C905B-TX), and Dell Computer Corporation BIOS A05, 9/18/2000. The client test driver was executed under Microsoft Windows XP Professional Version 5.1.2600 Service Pack 1 Build 2600 running BEA WebLogic JRockit Virtual Machine Build 3.1.5-CROSIS-20020617-1325 with Thin Threads and the Generational Copying Garbage Collector.

Note that the client side must run on a much more powerful system than the server side because it simulates up to 10,000 clients on just one machine and must saturate the server side's CPU usage.

B Operating Systems

B.1 FreeBSD 4.6-RELEASE

FreeBSD 4.6-RELEASE with Linux binary compatibility:

```
FreeBSD gx1p.test.volano.com 4.6-RELEASE FreeBSD 4.6-RELEASE #0:
↪ Tue Jun 11 06:14:12 GMT 2002 murray@builder.freebsdmall.com:
↪ /usr/src/sys/compile/GENERIC  i386
```

Contents of `/etc/sysctl.conf`:

```
# Increase file descriptor limits from their default of 5479.
kern.maxfiles=15360
kern.maxfilesperproc=10240
```

Installed the linux_base-7.1_2 port with an emulated Linux roughly equal to Red Hat Linux 7.1 (kernel 2.4.2, glibc 2.2.2):

```
Linux gx1p.test.volano.com 2.4.2 FreeBSD 4.6-RELEASE #0:
↪ Tue Jun 11 06:14:12 GMT 2002 murray@ i386 unknown
glibc-2.2.2-10
```

B.2 Microsoft Windows 2000 Advanced Server

Microsoft Windows 2000 Advanced Server with Service Pack 3:

```
Microsoft Windows 2000 Advanced Server
Version 5.0.2195 Service Pack 3 Build 2195
```

Optimized the performance for background services (rather than applications) under Start, Settings, Control Panel, System, Advanced, Performance Options, Application response. This change specifies that all programs receive equal amounts of processor resources.

Added the following HKEY_LOCAL_MACHINE registry keys under:

```
SYSTEM \ CurrentControlSet \ Services \ Tcpip \ Parameters
```

Name	Type	Default	New Value
MaxUserPort	REG_DWORD	5,000	65,534
TcpMaxDataRetransmissions	REG_DWORD	5	32

where:

MaxUserPort determines the highest port number TCP can assign when an application requests an available user port from the system. Typically, ephemeral ports (those used briefly) are allocated to port numbers 1024 through 5000.

TcpMaxDataRetransmissions determines how many times TCP retransmits an unacknowledged data segment on an existing connection. TCP retransmits data segments until they are acknowledged or until this value expires.

B.3 Red Hat Linux 8.0

Red Hat Linux 8.0 (kernel 2.4.18, glibc 2.3.2) with all updates from the Red Hat Network:

```
Linux gx1p.seattle.volano.com 2.4.18-27.8.0 #1
↪ Fri Mar 14 06:45:49 EST 2003 i686 i686 i386 GNU/Linux
kernel-2.4.18-27.8.0
glibc-2.3.2-4.80.6
```

Added to `/etc/pam.d/login`:

```
session    required     /lib/security/pam_limits.so
```

Added to `/etc/security/limits.conf`:

```
*               soft    nofile          1024
*               hard    nofile          10240
```

B.4 Sun Solaris 8

Sun Solaris 8 with all recommended Solaris and J2SE (Java 2 Platform, Standard Edition) Patch Clusters:

```
SunOS gx1p 5.8 Generic_108529-19 i86pc i386 i86pc
```

Added to `/etc/system`:

```
set rlim_fd_max = 10240
set rlim_fd_cur = 1024
```

C Java Virtual Machines

C.1 BEA 3.1 Windows

For Microsoft Windows 2000.

```
java version 1.3
Java(TM) 2 Runtime Environment, Standard Edition
↪ (build "1.3.1_CROSIS-20020620-1430")
BEA WebLogic JRockit Virtual Machine
↪ (build 3.1.5-CROSIS-20020617-1325)
```

C.2 BEA 7.0 Linux

For Red Hat Linux Advanced Server 2.1 (kernel 2.4.9, glibc 2.2.4).

```
ERROR: The pthread library is unknown. Are you running a supported
Linux distribution?
```

C.3 BEA 7.0 Windows

For Microsoft Windows 2000 with Service Pack 2.

```
java version "1.3.1_06"
Java(TM) 2 Runtime Environment, Standard Edition (build 1.3.1_06)
BEA WebLogic JRockit(R) Virtual Machine
↪ (build 7.0sp2-1.3.1_06-win32-GARAK-20030130-0938,
↪ Thin Threads, Generational Copying Garbage Collector)
```

C.4 BEA 8.1 Linux

For Red Hat Enterprise Linux AS/ES/WS 2.1 (kernel 2.4.9, glibc 2.2.4).

```
java version "1.4.1_02"
Java(TM) 2 Runtime Environment, Standard Edition (build 1.4.1_02)
BEA WebLogic JRockit(R) Virtual Machine
↪ (build 8.1-1.4.1_02-linux32-borg.appeal.se-20030320-1059,
↪ Thin Threads, Generational Copying Garbage Collector)
```

C.5 BEA 8.1 Windows

For Microsoft Windows 2000 with Service Pack 2 or higher.

```
java version "1.4.1_02"
Java(TM) 2 Runtime Environment, Standard Edition (build 1.4.1_02)
BEA WebLogic JRockit(R) Virtual Machine
↪ (build 8.1-1.4.1_02-win32-CROSIS-20030320-1110,
↪ Thin Threads, Generational Copying Garbage Collector)
```

C.6 Blackdown 1.3.1 FreeBSD

For systems running Linux kernel version 2.2.16 or newer and glibc version 2.1.3 or newer.

```
java version "1.3.1"
Java(TM) 2 Runtime Environment, Standard Edition
↪ (build Blackdown-1.3.1-02b-FCS)
Classic VM (build Blackdown-1.3.1-02b-FCS, green threads, nojit)
```

Modified `/usr/local/j2sdk1.3.1/bin/.java_wrapper` by changing `expr` in the following two lines:

```
link=`expr "${ls}" : '.*-> \(.*\)$'`
if expr "${link}" : '/' > /dev/null; then
```

to the Linux-compatible version, `/usr/compat/linux/usr/bin/expr`.

C.7 Blackdown 1.3.1 Linux

For systems running Linux kernel version 2.2.16 or newer and glibc version 2.1.3 or newer.

```
java version "1.3.1"
Java(TM) 2 Runtime Environment, Standard Edition
↪ (build Blackdown-1.3.1-02b-FCS)
Classic VM (build Blackdown-1.3.1-02b-FCS, green threads, nojit)
```

C.8 Blackdown 1.4.1 FreeBSD

For Linux kernel version 2.4.18 or newer and glibc version 2.2.5 or newer.

```
Java HotSpot(TM) Server VM warning: Cannot read /proc/self/stat.
#
# HotSpot Virtual Machine Error, Internal Error
# Please report this error at
# http://www.blackdown.org/cgi-bin/jdk
#
# Java VM: Java HotSpot(TM) Server VM (Blackdown-1.4.1-01 mixed mode)
#
# Error ID: 4F533F4C494E55580E4350500334
#
Abort trap - core dumped
```

C.9 Blackdown 1.4.1 Linux

For Linux kernel version 2.4.18 or newer and glibc version 2.2.5 or newer.

```
java version "1.4.1"
Java(TM) 2 Runtime Environment, Standard Edition
↪ (build Blackdown-1.4.1-01)
Java HotSpot(TM) Server VM (build Blackdown-1.4.1-01, mixed mode)
```

C.10 IBM 1.3.1 Linux

For Red Hat Linux 7.1, 7.2, 7.3, 8.0, and Red Hat Linux Advanced Server.

```
java version "1.3.1"
Java(TM) 2 Runtime Environment, Standard Edition (build 1.3.1)
Classic VM (build 1.3.1, J2RE 1.3.1 IBM build cxia32131-20021102
↪ (JIT enabled: jitc))
```

C.11 IBM 1.3.1 Windows

For Microsoft Windows 2000 Professional with Service Pack 2 or higher, and Windows XP Professional.

```
java version "1.3.1"
Java(TM) 2 Runtime Environment, Standard Edition (build 1.3.1)
Classic VM (build 1.3.1, J2RE 1.3.1 IBM Windows 32 build
↪ cn131-20021107 (JIT enabled: jitc))
```

C.12 IBM 1.4.0 Linux

For Red Hat Linux 7.3 and Red Hat Linux Advanced Server.

```
java version "1.4.0"
Java(TM) 2 Runtime Environment, Standard Edition (build 1.4.0)
Classic VM (build 1.4.0, J2RE 1.4.0 IBM build cxia32140-20020917a
↪ (JIT enabled: jitc))
```

C.13 Sun 1.3.1 Linux

For Linux kernel version 2.2.12 and glibc version 2.1.2-11 or later, with specific support for Red Hat Linux 6.2. Also tested on Red Hat Linux 6.1 and 7.1.

```
java version "1.3.1_07"
Java(TM) 2 Runtime Environment, Standard Edition (build 1.3.1_07-b02)
Java HotSpot(TM) Server VM (build 1.3.1_07-b02, mixed mode)
```

C.14 Sun 1.3.1 Solaris

For Solaris 2.6, Solaris 7, Solaris 8, and Solaris 9 Operating Environments with the full set of required patches.

```
java version "1.3.1_07"
Java(TM) 2 Runtime Environment, Standard Edition (build 1.3.1_07-b02)
Java HotSpot(TM) Server VM (build 1.3.1_07-b02, mixed mode)
```

C.15 Sun 1.3.1 Windows

For Microsoft Windows 95, 98 (1st or 2nd edition), NT 4.0 with Service Pack 5, ME, 2000 Professional, 2000 Server, 2000 Advanced Server, and XP operating systems.

```
java version "1.3.1_07"
Java(TM) 2 Runtime Environment, Standard Edition (build 1.3.1_07-b02)
Java HotSpot(TM) Server VM (build 1.3.1_07-b02, mixed mode)
```

C.16 Sun 1.4.1 Linux

For Linux kernel version 2.2.12 and glibc version 2.1.2-11 or later. Also tested on Red Hat Linux 7.2 (kernel 2.4.9-31) and Red Hat Linux 7.3 (kernel 2.4.18, glibc 2.2.5).

```
java version "1.4.1_02"
Java(TM) 2 Runtime Environment, Standard Edition (build 1.4.1_02-b06)
Java HotSpot(TM) Server VM (build 1.4.1_02-b06, mixed mode)
```

C.17 Sun 1.4.1 Solaris

For Solaris 7, Solaris 8, and Solaris 9 Operating Environments with the full set of required patches.

```
java version "1.4.1_02"
Java(TM) 2 Runtime Environment, Standard Edition (build 1.4.1_02-b06)
Java HotSpot(TM) Server VM (build 1.4.1_02-b06, mixed mode)
```

C.18 Sun 1.4.1 Windows

For Microsoft Windows 98 (1st or 2nd edition), NT 4.0 (with Service Pack 5 or later), ME, XP, and 2000 (with Service Pack 2 or later).

```
java version "1.4.1_02"
Java(TM) 2 Runtime Environment, Standard Edition (build 1.4.1_02-b06)
Java HotSpot(TM) Server VM (build 1.4.1_02-b06, mixed mode)
```

D Revisions

May 30, 2003 First published.

June 2, 2003 Corrected the system requirements for Blackdown 1.4.1 on Linux and FreeBSD.

E Notices