Open Benchmark Phases 1 and 2: Windows NT Server 4.0 and Red Hat Linux 5.2 Upgraded to Linux 2.2.6

By Bruce Weiner

(PDF version, 134 KB)

June 30,1999

White Paper Contents

Related Article

PC Week's Story

Results for the Second Benchmark and Phases 1 and 2

This part of the Open Benchmark white paper discloses the results of the Second Benchmark and discusses the results of Phases 1 and 2 of the Open Benchmark.

- **Phase 1**: Mindcraft re-ran the Second Benchmark.
- **Phase 2**: Red Hat engineers used the same software as in Phase 1 but tuned it themselves

Figure 1 shows the results of running the NetBench file-server benchmark for the Second Benchmark and for Phases 1 and 2. You can see that the results are effectively identical even though the Open Benchmark used a different test lab with fewer, much faster client systems.

Figure 1: Second Benchmark vs. Phase 1 and 2 File-Server Performance
(larger numbers are
better)

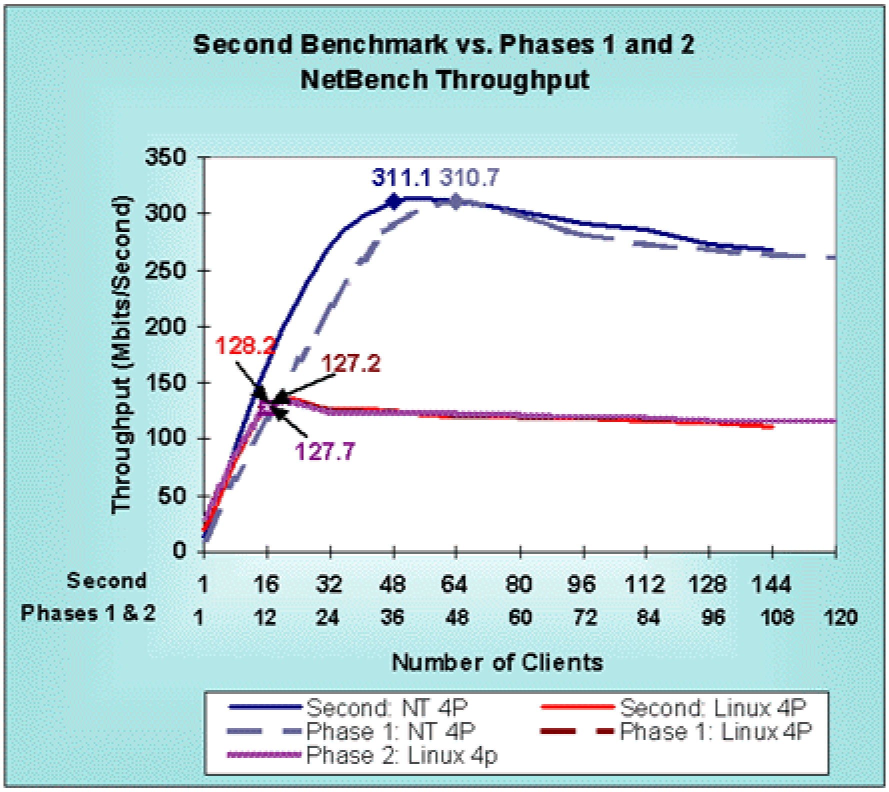
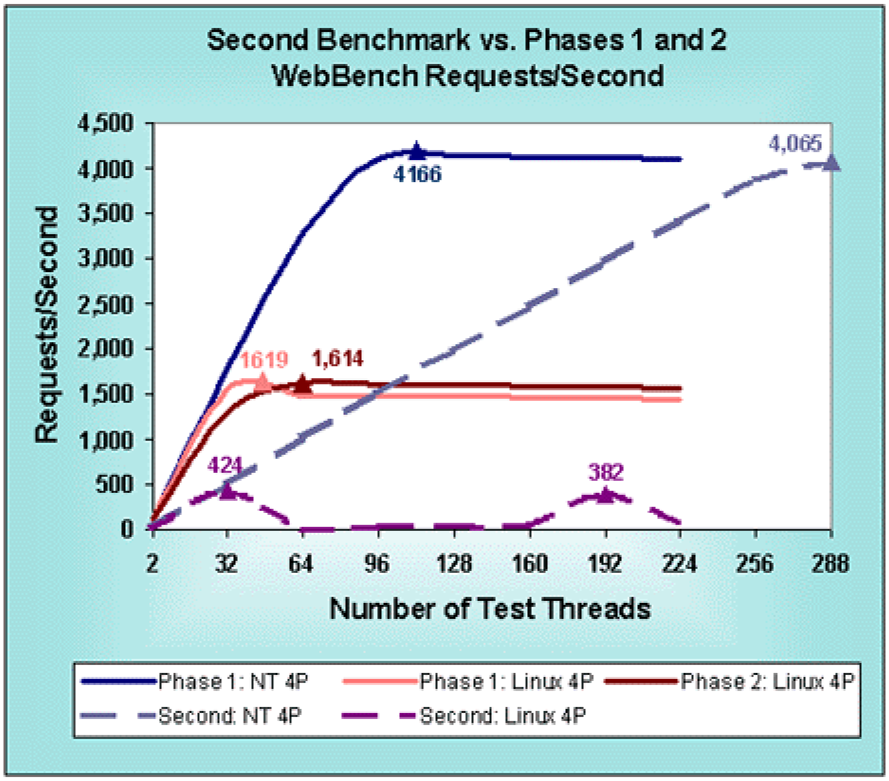


Figure 2 shows the WebBench benchmark test results for the Second Benchmark and Phases 1 and 2. Mindcraft and Red Hat obtained the same Linux/Apache performance in Phases 1 and 2. The Windows NT Server performance difference between the Second Benchmark and Phase 1 is the result of the differences in the test labs. Web-Server Performance Analysis section below provides an analysis of the odd Linux/Apache performance in the Second Benchmark.

Figure 2: Second Benchmark vs. Phase 1 and 2 Web-Server Performance
(larger numbers are better)



The Open Benchmark Phases 1 and 2 show that Mindcraft's Second Benchmark of Windows NT Server 4.0 and Linux 2.2.6 / Samba 2.0.3 / Apache 1.3.6 accurately measured their file- and Web-server performance and was unbiased.

Performance Analysis

Looking at NetBench Results

The NetBench 5.01 benchmark measures file server performance. Its primary performance metric is throughput in bytes per second. The NetBench documentation defines throughput as "*The number of bytes a client transferred to and from the server each second. NetBench measures throughput by dividing the number of bytes moved by the amount of time it took to move them. NetBench reports throughput as bytes per second.*" We report throughput in megabits per second to make the charts easier to compare to other published NetBench results.

Understanding how NetBench 5.01 works will help explain the meaning of the NetBench throughput measurement. NetBench stresses a file server by using a number of test systems to read and write files on a server. A NetBench test suite is made up of a number of mixes. A mix is a particular configuration of NetBench parameters, including the number of test systems used to load the server. Typically, each mix increases the load on a server by increasing the number of test systems involved while keeping the rest of the parameters the same. We modified the standard NetBench NBDM_60.TST test suite to increase the number of test systems to 144 for the Second Benchmark and to 120 for the Open Benchmark. The **NetBench Test Suite Configuration Parameters** show you exactly how we configured the test.

> *NetBench does a good job of testing a file server under heavy load. To do this, each NetBench test system (called a client in the NetBench*

documentation) executes a script that specifies a file access pattern. As the number of test systems is increased, the load on a server is increased. You need to be careful, however, not to correlate the number of NetBench test systems participating in a test mix with the number of simultaneous users that a file server can support. This is because each NetBench test system represents more of a load than a single user would generate. NetBench was designed to behave this way in order to do benchmarking with as few test systems as possible while still generating large enough loads on a server to saturate it.

File Server Performance Analysis

With this background, let us analyze what the results in **Figure 1** mean. The supporting details for Figure 1 are in the **NetBench Configuration and Results** part of this white paper. The two major areas to notice in Figure 1 are:

- **Peak Performance**

  This tells you the maximum throughput you can expect from a file server. NetBench throughput is primarily a function of how quickly a file server responds to file operations from a given number of test systems. So a more responsive file server will be able to handle more operations per second, which will yield higher throughput.

- **Shape of the Performance Curve**

  How quickly a product reaches its peak performance depends on the server hardware performance, the operating system performance, and the client test systems' performance. The part of the throughput performance curve to the left of the peak does not tell us anything of interest because how quickly performance rises to the peak is a function of the speed and number of clients used; this can be seen in the slight performance curve differences for Windows NT in **Figure 1**.

  The performance curve after the peak shows how a server behaves as it is overloaded. If performance drops off rapidly, users may experience significant unpredictable and slow response times as the load on the server increases. On the other hand, a product whose performance is flat or degrades slowly after the peak can deliver more predictable performance under load.

The Windows NT Server 4.0 file-server peak performance shows that Linux/Samba do not take full advantage of the four-processor Dell server. We believe the major reasons for the poor Linux/Samba performance are:

1. A single threaded TCP stack;
2. Large-grained locking in the kernel; and
3. Samba running in user space.

The shapes of the performance curves for both Windows NT Server 4.0 and Linux/Samba indicate that we reached peak performance and went beyond it. Performance for both Windows NT Server 4.0 and Linux/Samba degrades slowly as the load is increased past the peak performance load. So both systems should deliver predictable performance even under overload conditions.

Looking at WebBench Results

In order to understand what the WebBench measurements mean you need to know how WebBench 2.0 works. It stresses a Web server by using a number of test systems to request URLs. Each WebBench test system, also called client, can be configured to use multiple worker threads (threads for short) to make simultaneous Web server requests. By using multiple threads per test system, it is possible to generate a large enough load on a Web server to stress it to its limit with a reasonable number of test systems. The other factor that will determine how many test systems and how many threads per test system are needed to saturate a server is the performance of each test system.

The number of threads needed to obtain the peak server performance depends on the speed of the test systems and the server. It is meaningful to compare the peak server performance measurements from different test beds based on the number of threads, not systems, at each data point. That is why our graphs below show the number of test threads for each data point.

> *WebBench can generate a heavy load on a Web server. To do this in a way that makes benchmarking economical, each WebBench thread sends an HTTP request to the Web server being tested and waits for the reply. When it comes, the thread immediately makes a new HTTP request. This way of generating requests means that a few test systems can simulate the load of hundreds of users. You need to be careful, however, not to correlate the number of WebBench test systems or threads with the number of simultaneous users that a Web server can support since WebBench does not behave the way users do.*

Web-Server Performance Analysis

The primary WebBench 2.0 metric is the number of HTTP GET requests per second the server can satisfy. In addition, WebBench reports the number of bytes per second a Web server sends to all test systems.

We tested both Web servers using the standard WebBench zd_static_v20.tst test suite, modified to increase the number of test threads to 288 (144 system with 2 threads each) for the Second Benchmark and to 240 (120 system with 2 threads each) for Phases 1 and 2. This standard WebBench test suite uses the HTTP 1.0 protocol without keepalives.

With this background, let us analyze what the results in Figure 2 mean (the supporting detail data for this chart is in the **WebBench Configuration and Results** part of this white paper). There are two major areas to look at:

- **Peak Performance**

  This tells you the maximum requests per second that a Web server can handle and the peak throughput it can generate. A more responsive Web server will be able to handle more requests per second, which will yield higher throughput.

- **Shape of the Performance Curve**

  How quickly a Web server reaches its peak performance depends on the performance of the server hardware, the operating system, the Web server software, and the test systems. The part of the performance curve to the left of the peak does not tell us anything of interest since it depends mostly on the test systems. The performance curve after the peak show how a server behaves as it is overloaded.

> The shape of the performance curve after the peak shows how a Web server performs as a function of load. If performance drops off rapidly, users may experience significant unpredictable and slow response times as the load on the Web server increases. On the other hand, a Web server that degrades performance slowly after the peak will deliver more predictable performance under load.

Looking at the WebBench results in **Figure 2**, notice that the performance curves are shifted to the left for Phases 1 and 2 as compared to the Second Benchmark. That is the effect of using faster clients for the Open Benchmark.

Windows NT peak performance is slightly higher in Phase 1 than in the Second Benchmark because we did not have enough clients to drive the server to 100% CPU utilization.

The Linux/Apache performance in Phases 1 and 2 are essentially identical. However, the Linux/Apache performance in the second benchmark exhibited a performance collapse at 32 threads. Why did this happen since Mindcraft used the same Linux and Apache software versions and configurations in the Second Benchmark and in Phase 1?

We used the Linux top command to look at the wait channel before and during the performance collapse. It showed that prior to the collapse Apache was waiting in do_select while after the collapse it was waiting in either wait_for_ or tcp_recvm. There have been several reported problems similar to the performance collapse we found (**karthik**, **van Riel**, **Arcangeli**, **Ezlot**, **Schmidt**, and see **Kegel** for more).

This leads us to conclude that there was an interaction between a Linux bug and the test bed we used for the Second Benchmark that caused the performance collapse shown in **Figure 2**. We verified that the problem was related to Apache by restarting it for the 96-client mix (192 threads). As you can see in Figure 2, performance recovered briefly before collapsing again.

Products Tested

Server System

We used the same Dell PowerEdge 6300/400 for the Second Benchmark and the Open Benchmark. Table 1 shows the system configuration.

Table 1: Dell PowerEdge 6300/400 Configuration

Feature	Configuration
CPU	4 x 400 MHz Pentium II Xeon Cache: L1: 16 KBI + 16 KB D; L2: 2 MB
RAM	2 GB 100 MHz SDRAM ECC
Disks	OS Disk: 9 GB Seagate Cheetah, Model ST39102LC, 10,000 RPM PowerEdge RAID II Adapter, 32 MB cache, RAID 0, BIOS v1.47, stripe size = 64 KB, write policy = writeback, read policy = adaptive, cache policy = directIO, raid across two channels, with one logical drive: Drive D/Data: 8 x 4 GB Seagate Barracuda, Model

	ST34573WC, 7,200 RPM
Networks	4 x Intel EtherExpress Pro 100B Network Interface Cards

Windows NT Server and Linux were each on their own identical disks. We swapped OS disks to change operating systems. The RAID was reformatted each time the operating system was changed.

Software Products and Tuning

Windows NT Server 4.0 File-Server Configuration

We tested using Windows NT Server 4.0 Enterprise Edition with Service Pack 4 installed. We made the following configuration and tuning changes:

- Used 1024 MB of RAM (set maxmem=1024 in boot.ini)
- Server set to maximize throughput for file sharing
- Foreground application boost set to NONE
- Set registry entries:
  HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services:
    - \NDIS\Parameters\ProcessorAffinityMask=0
    - Tcpip\Parameters\Tcpwindowsize = 65535
- Used the NIC control panel to set the following for all four NICs:
    - Receive Buffers = 200 (default is 32; this setting is under "Advanced Settings")
    - NIC speed = 100 Mbit (default is "auto")
    - Duplex=full (default is"auto")
- Spooler service was disabled
- Page file size set to 1012 MB on the same drive as the OS
- The RAID file system was formatted with 16 KB allocation unit size (the /a option of the format command) and an NTFS file system
- Increased the file system log on the RAID file system to 65536 K using the chkdsk f: /l:65536 command
- Used the affinity tool to bind one NIC to each CPU (**ftp://ftp.microsoft.com/bussys/winnt/winnt-public/tools/affinity/)**

Windows NT Server 4.0 Web-Server Configuration

- Used Internet Information Server 4 (IIS 4) as the Web server
- Used the NIC control panel to set the following for all four NICs:
    - Coalesce Buffers = 32 (default is 8)
    - Receive Buffers = 1023
    - Transmit Control Blocks = 80 (default is 16)
    - Adaptive Transmit Threshold = on (default is on)
    - Adaptive Technology = on (default is on)
    - Adaptive Inter-Frame Spacing = 1 (default is 1)
    - Map Registers = 64 (default is 64)
- SMTP, FTP, MSDTC, and Browser services were disabled
- Set registry entries:
  HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services:
    - \InetInfo\Parameters\ListenBackLog=200
    - \InetInfo\Parameters\ObjectCacheTTL=0xFFFFFFFF
    - \InetInfo\Parameters\OpenFileInCache=0x5000
- Using the IIS Manager
    - Set Logging – "Next Log Time Period" = "When file size reaches 100 MB"
    - Set performance to "More than 100,000"
    - Removed all ISAPI filters

  - Removed all Home directory application mappings except .asp
  - Removed permissions for “Application Settings”
- Logs on the F: drive (RAID) along with the WebBench data files
- Server set to maximize throughput for applications when doing WebBench tests
- The other tunes for the file-server configuration were kept

Linux 2.2.6 Configuration

In Phase 1, we tested using Red Hat Linux 5.2 upgraded to the Linux 2.2.6 kernel following Red Hat's instructions (**http://www.redhat.com/support/docs/rhl/kernel-2.2/kernel2.2-upgrade.html**). We made the following configuration and tuning changes:

- Used 1024 MB of RAM (set mem=960M in lilo.conf)
- Used the AMI 1.0 version of the MegaRAID driver for Linux
- Compiled the Linux 2.2.6 kernel using gcc version 2.7.2.3
- Set the following at the end of /etc/rc.d/init.d/rc.local:

```
echo "16384 65535" >
        /proc/sys/net/ipv4/ip_local_port_range
echo "6000" >/proc/sys/fs/file-max
echo "50000" >/proc/sys/fs/inode-max
echo "80 500 64 64 80 6000 6000 1884 2" >
        /proc/sys/vm/bdflush
echo "60 80 80" >/proc/sys/vm/buffermem
```

We have included a separate Web page with all of the **Linux configuration files** that Mindcraft and Red Hat used. See it for Red Hat's Phase 2 Linux tuning.

Samba 2.0.3 Configuration

Mindcraft used the pre-compiled version of Samba 2.0.3 in Phase 1. In addition, we:

- Started Samba manually before each test
- Rebuilt file system on the RAID between NetBench runs using the command mke2fs –b 4096 -R stride=128 /dev/sdb1. Note that mke2fs does not support file systems with block sizes above 4096 bytes.

We have included a separate Web page with the **Samba configuration file** we used. See it for Red Hat's Phase 2 Samba tuning.

Apache 1.3.6 Configuration

- Compiled Apache 1.3.6 using gcc version 2.7.2.3 and glibc 2.0.7
- Started Apache manually before each test

We have included a separate Web page with the **Apache configuration files** we used. See it for Red Hat's Phase 2 Apache tuning.

The Test Lab

Figure 3 shows the test lab at Microsoft we used for the Second Benchmark. There were 144 test systems in the lab made up of two types. Table 2 and

Table 3 show the system configurations. We used 72 Type A systems and 72 Type B systems.

Table 2: Type A Test Systems Configuration

Feature	Configuration
CPU	133 MHz Pentium. All are identical Mitac systems.
RAM	64 MB
Disk	1 GB IDE; standard Windows 95 driver
Network	All systems used Intel E100B LAN Adapter (100Base-TX) using e100b.sys driver version 2.02 Network software: Windows 95 TCP/IP driver.
Operating System	Windows 95, version 4.00.950

Table 3: Type B Test Systems Configuration

Feature	Configuration
CPU	133 MHz Pentium. All are identical Mitac systems.
RAM	64 MB
Disk	1 GB IDE; standard Windows 98 driver
Network	All systems used Intel E100B LAN Adapter (100Base-TX) using e100b.sys driver version 2.02 Network software: Windows 98 TCP/IP driver.
Operating System	Windows 98

Figure 3: Second Benchmark Test Lab

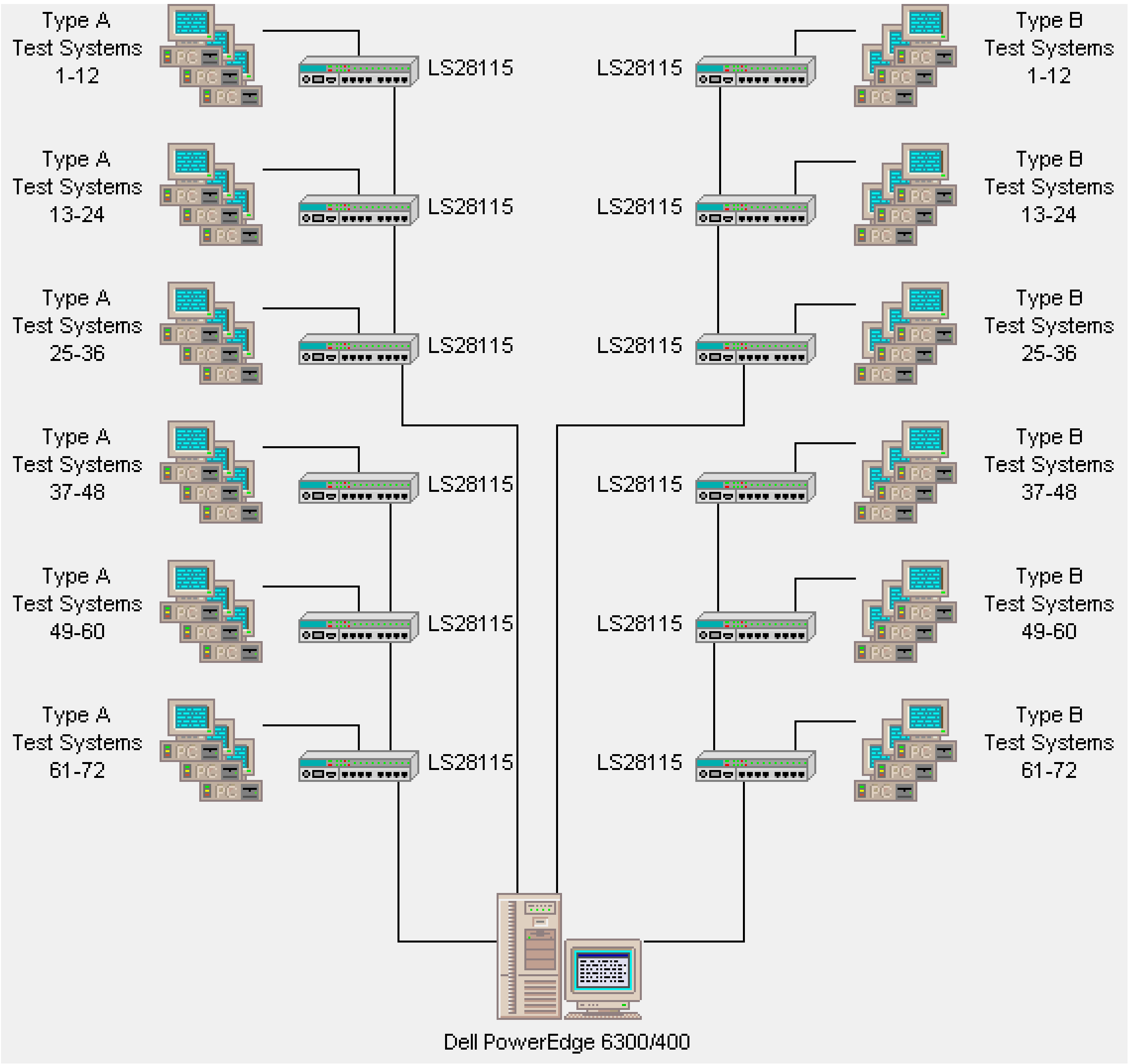
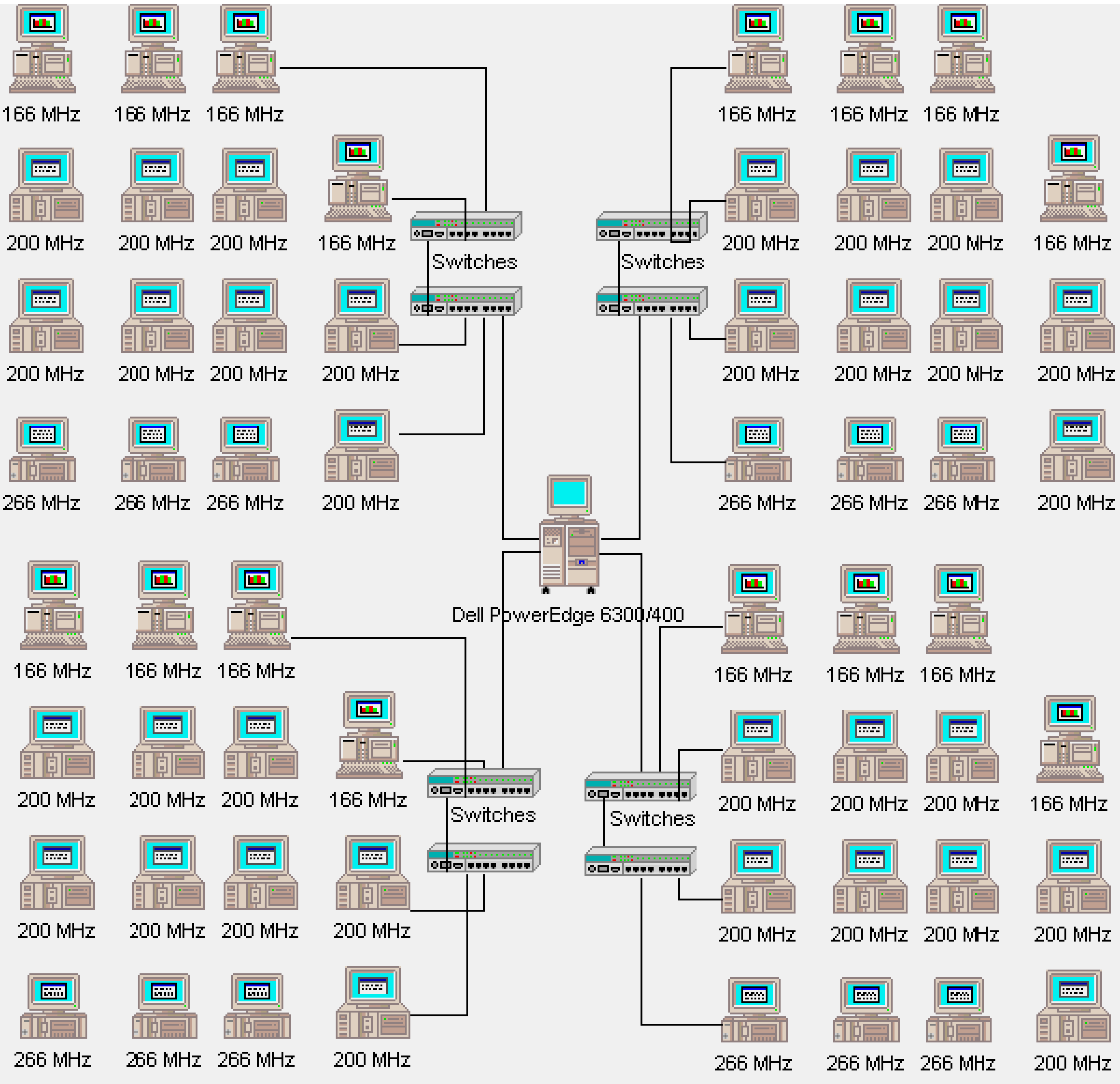


Figure 4 shows the test lab used for the Open Benchmark at ZD Labs. In order to simplify the diagram, each client test system depicted represents two identical systems.

Figure 4: Open Benchmark Lab



Each client system depicted represents two real client systems

◀Overview Phase 3▶

NOTICE:

The information in this publication is subject to change without notice.

MINDCRAFT, INC. SHALL NOT BE LIABLE FOR ERRORS OR OMISSIONS CONTAINED HEREIN, NOR FOR INCIDENTAL OR CONSEQUENTIAL DAMAGES RESULTING FROM THE FURNISHING, PERFORMANCE, OR USE OF THIS MATERIAL.

This publication does not constitute an endorsement of the product or products that were tested. This test is not a determination of product quality or correctness, nor does it ensure compliance with any federal, state or local requirements.

Services Benchmarks Reports Price/Performance Company

Search Contact Us

For more information, ***contact us*** *at:* ***info@mindcraft.com***
Phone: +1 (408) 364-2860
Fax: +1 (408) 364-2862