

# Installing and Configuring the **TUX** Web Server for Optimal **Performance**

By David J. Morse

In June 2000, Dell was the first vendor to publish SPECweb99 results with TUX 1.0, an innovative new Web server that demonstrated outstanding performance across the entire line of Dell® PowerEdge® servers. The latest version, TUX 2.1, is now integrated as a system service in Red Hat® Linux® 7.1 and 7.2. This article explains how to install, configure, and optimize TUX to obtain the best performance on Dell PowerEdge servers.

With the Internet becoming more popular each day, the ability of Web servers to handle large amounts of traffic becomes increasingly important. Traditional Web servers handle HTTP requests in user mode, which inherently incurs overhead in sending out Web pages. To send an HTML page or image, the Web server must make a system call to the kernel, which reads the file into a buffer and passes the data back to user space. This interaction, called context switching, results in unnecessary processing time being spent by the server. The TUX architecture, developed by Red Hat, eliminates excessive context switching by moving the Web server into the kernel.<sup>1</sup>

This article explains how to install and configure TUX to obtain the best performance possible on Dell® PowerEdge® servers. This information applies to customers who want to implement a highly scalable, stand-alone Web server and to customers who already have the Apache Web server set up and would like TUX to serve as a front-end accelerator.

## Hardware and software recommendations for TUX

For optimal performance of TUX, Dell recommends the following system configuration:

- ▶▶ **Software.** Red Hat® Linux® 7.1 or 7.2, which can be factory-installed on PowerEdge servers or downloaded from <http://www.redhat.com>.
- ▶▶ **Server.** Any PowerEdge server; Red Hat Linux is certified across the entire line of Dell servers.
- ▶▶ **Networking.** A supported network interface card (NIC); all of the internal NICs that ship with Dell servers are supported by Red Hat Linux. TUX can also take advantage of a feature known as zero-copy networking if the NIC and the Linux driver support it. Zero-copy networking eliminates unnecessary CPU cycles by utilizing features in the Linux 2.4 kernel, such as off-loading TCP/IP checksums and sending data directly to the NIC instead of copying it into a kernel buffer. At the time of writing, only a few NICs, such as the Alteon® ACEnic, the 3Com® 3C985, the

The TUX architecture eliminates excessive context switching by moving the Web server into the kernel.

<sup>1</sup>For more information on the TUX architecture, see “Running TUX Web Server for Linux on Dell Servers” by Michael Tiemann in Dell *Power Solutions*, Issue 1, 2001.

```
[root@pe2500 /root]# mount /mnt/cdrom
[root@pe2500 /root]# ls /mnt/cdrom/RedHat/RPMS/tux*
/mnt/cdrom/RedHat/RPMS/tux-2.1.0-2.i386.rpm
[root@pe2500 /root]# rpm -ivh /mnt/cdrom/RedHat/RPMS/tux-2.1.0-2.i386.rpm
Preparing...      ##### [100%]
 1:tux            ##### [100%]
```

Figure 1. Sample output from installing TUX

Intel® PRO/1000, and the Broadcom® BCM5700, support zero-copy networking.

- ▶ **RAID controller.** Although not required, a hardware RAID (redundant array of independent disks) controller can help increase disk I/O throughput when TUX reads Web pages off the disk and logs Web server requests. Red Hat Linux 7.1 ships with drivers for versions 2 and 3 of the Dell PowerEdge RAID Controller (PERC/2 and PERC/3).

### Installing TUX

TUX should already be installed if the PowerEdge server came pre-installed with Red Hat Linux 7.1 or above; otherwise, an administrator would need to perform a server-class installation of Red Hat

Linux. Check to determine whether TUX is installed by issuing the command `rpm -q tux`. The result may resemble the following:

```
[root@pe2500 /root]# rpm -q tux
tux-2.1.0-2
```

A similar response indicates that TUX is installed. However, if “package tux is not installed” appears, mount the Red Hat Linux CD-ROM Disc 1 and install the TUX package using the command `rpm -ivh /mnt/cdrom/RedHat/RPMS/tux-2.1.0-2.i386.rpm` (modified as necessary to reflect the version of the RPM package). Figure 1 shows a sample result from installing the TUX package.

### Configuring TUX

The recommended (and default) configuration is to use TUX as a front-end Web server listening on port 80 and to use a back-end Web server (such as Apache) on port 8080 for answering requests TUX does not understand (such as Perl or PHP pages). This setup requires a modification to the Apache default configuration file `httpd.conf`: change the line that reads “Port 80” to “Port 8080”.

Most TUX parameters can be changed in the `/etc/sysconfig/tux` file with a text editor. Figure 2 lists the currently supported parameters.

Edit the parameters to suit the particular environment. Initially, the only parameter that may need to be changed is `DOCRROOT`. Set it to the directory where the Web pages reside—preferably on a software or hardware RAID partition to allow for more space and redundancy.

### Starting and stopping TUX

TUX ships as a system service in Red Hat Linux 7.1 and above, so it can be easily started and stopped via the `service` command:

- ▶ **To manually start TUX:** `service tux start`
- ▶ **To manually stop TUX:** `service tux stop`
- ▶ **To restart TUX (after a configuration change):** `service tux restart`
- ▶ **To automatically start TUX on boot:** `chkconfig tux on`

Parameter	Default value	Description
TUXTHREADS	Number of CPUs in the system	Number of independent kernel threads TUX uses to answer Web requests; cannot be greater than the number of CPUs in the system
DOCRROOT	/var/www/html/	Directory from which TUX serves Web pages
CGIROOT	DOCRROOT (above)	Directory in which TUX runs CGI programs; TUX spawns a process with this directory as the top-level directory for security purposes
DAEMON_UID/ DAEMON_GID	nobody/nobody	The user and group under which TUX runs; set to “nobody” for security purposes
CGI_UID/CGI_GID	nobody/nobody	The user and group under which TUX runs CGI programs; set to “nobody” for security purposes
MAX_KEEPA_LIVE_ TIMEOUT	30	Maximum number of seconds the server waits for the next request before the Keep-Alive connection is closed

Figure 2. TUX parameters in `/etc/sysconfig/tux`

To ensure that TUX is operating correctly, manually start TUX and then request a Web page; for example, run `lynx http://localhost/index.html` from the server itself or access the URL `http://server-IP/index.html` from a client browser, where `server-IP` is the IP address of the TUX server. If the request does not work, make sure the file being requested (`index.html`) is in the `DOOROOT`, verify the networking setup, and look for any error messages at the bottom of `/var/log/messages`.

## Tuning TUX for performance

After configuring a base TUX setup, administrators can tune numerous parameters to get the maximum performance from the Web server. TUX already performs well by default, but administrators looking for even more performance should try modifying the parameters listed below.

Although `/etc/sysctl.conf` is usually the preferred location to tweak parameters, the TUX hierarchy in `/proc` does not exist until the kernel loads the TUX module. Therefore, the system should load the TUX module first, followed by the tweaks; this is best accomplished in `/etc/rc.d/rc.local` so the parameters are adjusted every time the system starts. See the online version of this article at <http://www.dell.com/powersolutions> for a sample `rc.local` script.

### Adjusting default TUX parameters

The default settings for TUX should be sufficient for most environments; however, the settings may require adjustments to suit a particular environment. Figure 3 summarizes the most common adjustments.

Change these settings by echoing the desired value into the appropriate parameter:

```
echo value > /proc/sys/net/tux/parameter_name
```

The sample `rc.local` script in the online version of this article (<http://www.dell.com/powersolutions>) contains an example of changing TUX parameters.

### Binding TUX threads to specific IP addresses

As mentioned earlier, TUX creates  $N$  kernel threads, where  $N$  is the number of CPUs in the system. By default, TUX threads listen to any IP address (0.0.0.0). However, in a multiple-network environment, certain TUX threads can be forced to listen only on certain IP addresses. To do this under stock Red Hat Linux 7.1 and 7.2, each octet in the IP address must be converted to its hexadecimal equivalent and then strung together. For example, 192.1.1.32 would be represented as `c0010120` (192 = `c0` in hex, 32 = `20` in hex). This process has been made easier in the latest version of TUX, available via a kernel update (2.4.9-12 or later) from <http://www.redhat.com/errata>.

For a PowerEdge 2500 with two processors and two network cards with IP addresses 192.1.1.32 and 192.1.2.32, the two TUX threads are bound via the commands shown in Figure 4.

### Binding NIC IRQs to specific CPUs

A new feature in the Linux 2.4 kernel—interrupt request line (IRQ) affinity—allows the binding of IRQs to specific CPUs to reduce unnecessary cross-bus traffic. To identify the IRQ used by each NIC, run the `ifconfig` command and note the “Interrupt” number. Figure 5 shows an example where `eth0` uses IRQ 24 and `eth1` uses IRQ 25.

Parameter	Default value	Description	Adjustment
<code>max_connect</code>	10000	Maximum number of simultaneous connections	Decrease if TUX is overloaded; increase if underloaded
<code>max_output_bandwidth</code>	0 (unlimited)	Maximum bandwidth that Keep-Alive requests can use (bytes/sec)	Set to throttle the bandwidth
<code>max_keepalives</code>	10000	Maximum number of open Keep-Alive connections	Decrease if TUX is overloaded
<code>max_backlog</code>	2048	Maximum size of the SYN backlog of the TUX socket (bytes)	Increase to handle more connections, but not too high to prevent SYN floods
<code>logging</code>	0	Determines whether TUX logs HTTP requests	Set to 0 (disabled) if no logging is necessary (reduces disk writes); set to 1 (enabled) to analyze traffic patterns
<code>compression</code>	0	Determines whether to send a gzipped version of the file if the client supports gzip encoding and a .gz version exists in the same directory as the uncompressed version	Set to 1 (enabled) to decrease client download time, thereby allowing more connections if the data is highly compressible (for example, text or HTML)

Figure 3. Configurable TUX parameters

```
# For kernels prior to 2.4.9-12
echo c0010120 > /proc/net/tux/0/listen/0 # 192.1.1.32
echo c0010220 > /proc/net/tux/1/listen/0 # 192.1.2.32

# For kernels 2.4.9-12 and later
echo http://192.1.1.32:80 > /proc/net/tux/0/listen/0
echo http://192.1.2.32:80 > /proc/net/tux/1/listen/0
```

Figure 4. Commands binding TUX threads to specific IP addresses

```
[root@pe2500 /root]# ifconfig
eth0      Link encap:Ethernet HWaddr 00:60:CF:20:21:C9
          inet addr:192.1.1.32 Bcast:192.1.1.255 Mask:255.255.255.0
          [...]
          Interrupt:24 Base address:0xc000

eth1      Link encap:Ethernet HWaddr 00:60:CF:20:21:CF
          inet addr:192.1.2.32 Bcast:192.1.2.255 Mask:255.255.255.0
          [...]
          Interrupt:25 Base address:0x8000
```

Figure 5. Sample output from the `ifconfig` command

An IRQ can be bound by placing a bitmask of the CPU(s) into the appropriate IRQ resource. For this example, the following commands bind `eth0` to CPU0 and `eth1` to CPU1:

```
echo 1 > /proc/irq/24/smp_affinity
echo 2 > /proc/irq/25/smp_affinity
```

Binding threads to IP addresses and NIC IRQs to CPUs can provide a very efficient, linear binding scenario as illustrated in Figure 6.

### Binding CGI execution to specific CPU(s)

TUX can also restrict Common Gateway Interface (CGI) execution to a specific processor or group of processors. By default, TUX spreads the load evenly across the CPUs of a symmetric multi-processing (SMP) machine, but it may be necessary to limit the effect of CGI programs on system load. To bind CGI execution, echo

the CPU bitmask into the `cgi_cpu_mask` parameter. For example, the following command binds CGI execution to the first processor:

```
echo 1 > /proc/sys/net/tux/cgi_cpu_mask
```

Another possible reason to bind CGI execution to specific CPU(s) would be to rectify CPU load imbalances, although this should not be necessary if NIC IRQs and TUX threads are correctly bound, as explained earlier.

### Adjusting Linux TCP/IP stack parameters

Many times, when one performance bottleneck is removed, it exposes another bottleneck somewhere else. TUX practically removes the Web server application as a bottleneck, but the network stack may need tuning to keep pace. Figure 7 lists the suggested network parameters to tune, along with the values that Red Hat and Dell have found to be the most beneficial. These values allocate large amounts of networking buffers, so the server should have a sufficient amount of RAM (512 MB or more is recommended).

To change these parameters, echo the desired values into `/proc/sys/net/location/parameter_name`. For example:

```
echo "30000000 30000000 30000000" > \
    /proc/sys/net/ipv4/tcp_mem
```

The sample `/etc/rc.d/rc.local` shell script in the online version of this article (<http://www.dell.com/powersolutions>) contains all of the network optimizations.

### Optimizing disk parameters for improved I/O performance

If the system uses a Dell PERC/2 or PERC/3 hardware RAID controller, create the volumes with the largest chunk size available (usually 64 KB or 128 KB), enable read and write caching, and choose a RAID type that provides an acceptable level of redundancy (that is, anything above RAID-0).

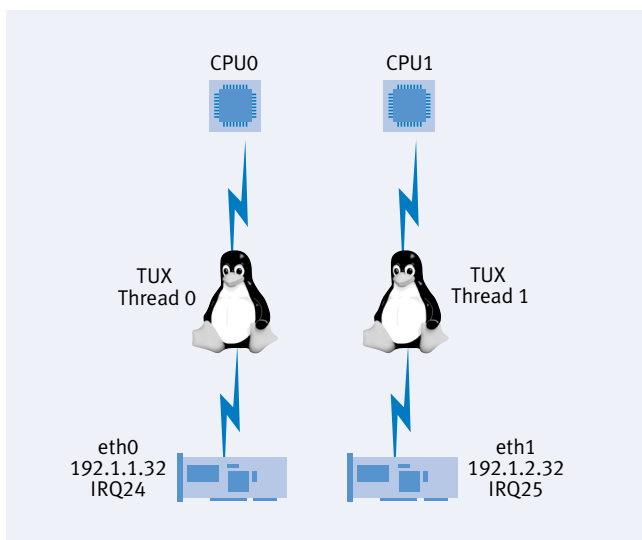


Figure 6. Example of an optimal TUX binding configuration

Parameter	Location*	Description	Default	Tuned value
tcp_timestamps	ipv4	TCP timestamp support	1 (on)	0 (off)
tcp_max_tw_buckets	ipv4	Pool size of TCP time-wait sockets	180000	2000000
tcp_rmem	ipv4	TCP read-buffer space (bytes)		
		Minimum	4096	30000000
		Default	87380	30000000
tcp_wmem	ipv4	TCP write-buffer space (bytes)		
		Minimum	4096	30000000
		Default	16384	30000000
tcp_mem	ipv4	TCP buffer space (bytes)		
		Minimum	195584	30000000
		Default	196096	30000000
hot_list_length	core	Maximum number of cached Linux network buffers	128	10000
		Maximum and default socket buffer sizes (bytes)	65535	10000000
		Maximum number of unprocessed input packets before the kernel starts dropping them	300	300000

\*Subdirectory of /proc/sys/net

Figure 7. Network stack adjustments for Linux

```

LABEL=/boot    /boot          ext2           defaults      1 2
LABEL=/        /              ext2           noatime,nodiratime 1 1
LABEL=/www     /www           ext2           noatime,nodiratime 1 2
LABEL=/logs    /logs         ext2           noatime,nodiratime 1 2
none          /proc         proc          defaults      0 0
none          /dev/shm      tmpfs         defaults      0 0
/dev/sda2     swap          swap          defaults      0 0
/dev/cdrom    /mnt/cdrom    iso9660       noauto,owner,kudzu,ro 0 0

```

Figure 8. Typical /etc/fstab file with added noatime and nodiratime parameters

For software RAID on Linux, RAID-1 provides the best read seek performance, which is crucial in RAM-limited situations. A large chunk size, such as 4 MB, also helps read performance; to set chunk size, specify “chunk-size 4096” in the /etc/raidtab file when creating the array. For more information, see “The Software RAID HOWTO” at <http://www.linuxdoc.org/HOWTO/Software-RAID-HOWTO.html>.

For both software and hardware RAID, set the noatime and nodiratime parameters for the partitions where the TUX log file and Web pages reside. The noatime parameter prevents the

file system from updating the inode access times on files, while nodiratime does the same for directories. Both parameters speed up read/write performance. Figure 8 shows a typical /etc/fstab file with these parameters added.

Red Hat Linux 7.2 introduces ext3, a new journaling file system that provides higher availability, better data integrity, and potentially higher throughput than ext2. Applying the noatime and nodiratime parameters is actually more beneficial for ext3 because every five seconds, ext3 writes metadata twice and synchronizes it. Also, the default journaling type (ordered)

on the log file partition can be changed to writeback; the trade-off is a certain degree of data integrity. To make this change, add “data=writeback” to the appropriate line in /etc/fstab.

### Compiling TUX into a monolithic kernel

By default, Red Hat Linux 7.1 and 7.2 ship with TUX (as well as many other components) as a loadable module; this approach allows for dynamic insertion and removal as the service is needed. However, if a server will be dedicated to Web serving, it may make sense to build TUX, as well as the necessary SCSI, RAID, and NIC drivers, statically into the kernel. This customization requires recompiling the kernel and an intimate knowledge of the system’s components; note that this task is not trivial. When compiling TUX into a monolithic kernel, do not compile TUX as a module: make sure the kernel configuration file contains the line “CONFIG\_TUX=y” (not “m” or “n”). For more information on building a custom kernel, see the Red Hat Linux Customization Guide.

### A high-performance Web server solution

TUX utilizes Linux 2.4 kernel enhancements such as zero-copy networking and IRQ affinity, which allow Dell PowerEdge servers to scale out with unsurpassed performance as more CPUs are added. Figure 9 illustrates this scalability by showing Dell’s record-breaking TUX SPECweb99 results.<sup>2</sup>

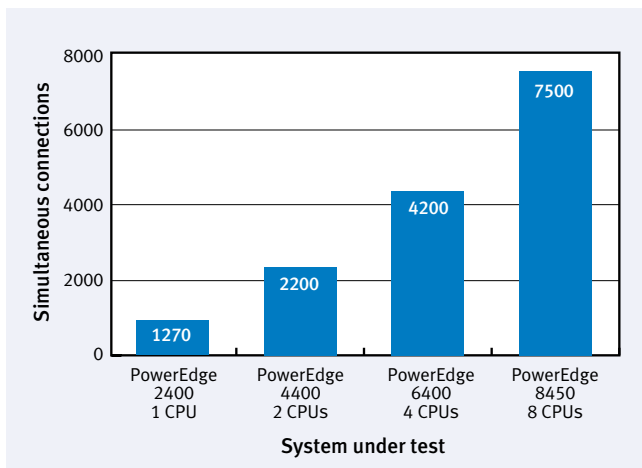


Figure 9. SPECweb99 results obtained with TUX (November 2000)

TUX utilizes Linux 2.4 kernel enhancements such as zero-copy networking and IRQ affinity, which allow Dell PowerEdge servers to scale out with unsurpassed performance as more CPUs are added.

eTesting Labs, working closely with the Dell System Performance team, showed that TUX performed approximately three times faster than Apache, proving that a TUX-Apache combination is a very attractive solution for high-traffic Web sites.<sup>3</sup> Red Hat Linux 7.1 and 7.2 integrate TUX as a standard system service, allowing easy integration with an existing Apache setup for a seamless, high-performance Web server solution. ☞

### Acknowledgments

Thanks to Ingo Molnar at Red Hat for his invaluable Linux advice and his excellent development work on TUX, and Michael K. Johnson at Red Hat for his comments and suggestions on this article.

**David J. Morse** ([david\\_j\\_morse@dell.com](mailto:david_j_morse@dell.com)) is a senior performance engineer for the Dell System Performance and Analysis Lab. He specializes in Web server performance and is responsible for running the industry-standard SPECweb® Web server benchmark across the line of Dell PowerEdge servers. Prior to joining Dell, he spent two years at NCR in the performance integration and testing group. David has a B.S. in Computer Engineering from the University of South Carolina and is a Red Hat Certified Engineer (RHCE).

### FOR MORE INFORMATION

#### Linux on Dell servers:

<http://www.dell.com/linux>

#### TUX manual:

<http://www.redhat.com/docs/manuals/tux>

#### TUX support:

<http://www.redhat.com/services/techsupport/application/tux.html>

#### Latest TUX patch:

<http://people.redhat.com/~mingo/TUX-patches>

#### TUX mailing list:

<http://www.redhat.com/mailling-lists/tux-list>

#### Red Hat Linux Customization Guides:

<http://www.redhat.com/docs/manuals/linux>

<sup>2</sup>SPECweb is a trademark of the Standard Performance Evaluation Corporation (SPEC). For the latest results, visit <http://www.spec.org/osg/web99>.

<sup>3</sup>For the complete article, see “TUX: Built for Speed” at <http://www.eweek.com/article/0,3658,s%253D702%2526a%253D7480,00.asp>.