

Running TUX Web Server for Linux on Dell Servers

By Michael Tiemann

TUX 2.0, a Web server developed by Red Hat, has demonstrated outstanding performance and scalability running on Dell servers. This article describes the development process that culminated in the release of TUX 2.0.

In August 2000, Dell and Red Hat set new Web serving world records in four SPECWeb99 categories: on servers with one, two, four, and eight CPUs.¹ These records not only surpassed previous records by margins of 100 percent to 200 percent, but also showed nearly linear scalability as the numbers of processors increased—a remarkable achievement for any symmetric multi-processing (SMP) operating system.

This article describes TUX, a new architecture from Red Hat that accelerates not only Web server performance, but also a wide range of network-delivered services, including FTP, file services, and network-based storage systems.

The TUX Web Server is an HTTP daemon for Linux®. The TUX Web Server is different from other Web servers in that it runs partially from within the Linux kernel as a module, or kernel subsystem. Given sufficient networking cards, it enables direct scatter-gather direct memory access (DMA) and hardware-based TCP/IP checksums from the page cache (the Linux file data cache) directly to the network, avoiding extra data copies.

Serving content

The TUX Web Server handles static pages and Common Gateway Interface (CGI) scripts directly, and works in concert with kernel modules, user-space modules, and regular user-space Web server daemons to provide dynamic content. Regular user-space daemons

do not need to be altered for TUX to use them to provide content, but in order for TUX to handle and optionally cache dynamic content directly, user-space code has to use a new interface based on the TUX system call.

Static content

Static Web pages are not difficult to serve, but are nevertheless important, since virtually all images and a large portion of HTML pages are static. A regular Web server has little added value for static pages; it is simply a “copy-file-to-network” operation. This can be done very efficiently from within the Linux kernel; for example, the Network File System (NFS) daemon performs a similar task running in the kernel.

Dynamic content

The TUX Web Server handles and optionally caches dynamic content as well. TUX modules, built in kernel or user space (user space is recommended), use (and optionally create) objects that are stored using the page cache. In response to a request for dynamic data, a TUX module may serve a mix of dynamically generated data and cached pregenerated objects, taking advantage of the TUX zero-copy architecture.

This new architecture for providing dynamic content requires a new application programming interface (API). Existing standard

¹ For the complete results, visit http://www.dell.com/us/en/biz/topics/linux_specWeb99.htm

APIs for CGI cannot be mapped to TUX's API easily. Existing applications such as CGI applications for Web servers must be recoded to take advantage of TUX's architecture. TUX can, however, call CGI programs via its CGI module, allowing users to convert selected programs that need TUX's speed to the TUX API and run other programs with the standard CGI interface. TUX can also redirect requests to another Web server such as an Apache server, so that a single site may combine static content, TUX modules, old-style CGIs, and programs written for other Web server APIs.

TUX performance enhancement

Regular Web servers run entirely in user space, which provides security advantages. However, this is detrimental to performance—every time the server moves data, it performs a context switch into the kernel. Prior to the release of TUX, some proprietary kernel-space Web servers that reduced both redundant inbound and outbound data copies had been developed. (The Linux `sendfile` system call reduces only redundant outbound data copies.) Results from some experimental in-kernel Linux Web servers showed promise, but to achieve higher performance goals, Linux needed a new architecture to overcome several limitations.

Removing the global lock

The first problem to solve was the “big kernel lock” problem. The Linux 2.0 kernel, while it supported symmetric multiprocessing (SMP), used a global lock, controlling all kernel functionality. The Linux 2.2 kernel was an improvement, but a considerable portion of the kernel functionality still operated under the big kernel

TUX accelerates not only Web server performance, but also a wide range of network-delivered services, including FTP, file services, and network-based storage systems.

lock. Processes were designed to hold this lock only a short period of time, but holding the lock during a file cache operation, for example, prevented any network traffic from flowing through the kernel, even when there was no direct conflict between the two. Removing the “big kernel lock” was the first and most important step to developing good kernel-level SMP performance.

Implementing fine-grain locking

The second step was to thread individual subsystems, such as the Virtual File System (VFS), the page cache, the timers, and the network stack.

This would allow these kernel subsystems to run in parallel with each other and for multiple CPUs

to be used within each subsystem. To maximize the efficiency of allocating multiple CPUs inside these subsystems, several forms of “affinity” were implemented: timer affinity, interrupt request (IRQ) affinity, and process-processor affinity.

Carefully tuning these affinities minimized the overhead of SMP and maximized the performance of the overall system. Whereas the Linux 2.2 kernel could only use a single CPU when in kernel mode, the Linux 2.4 kernel has shown no difficulty using eight CPUs efficiently across a wide range of network, storage, and server loads.

Implementing a zero-copy infrastructure

The third step was the development of a zero-copy infrastructure. TUX 2.0 now supports true zero-copy disk reads. Whereas TUX 1.0 copied files into a temporary buffer, TUX 2.0 is integrated with the page cache, and thus uses zero-copy block I/O. TUX 2.0 also performs generic zero-copy network writes by using the generic zero-copy TCP framework.

THE HISTORY OF TUX

The history of TUX began in 1999, when a report from an independent lab reported that Apache running on Red Hat® Linux was 100 times slower than a vendor's proprietary Web server and operating system. This came as a surprise because Linux already supported SMP and had a special system call, `sendfile`, to reduce redundant data copies.

Rather than dismissing them immediately, Red Hat investigated these results and drew three conclusions. First, the lab had chosen hardware that was known to support Linux poorly. Second, the lab had configured the proprietary operating system to get the most from the hardware, while they had configured Linux so that its performance was

seriously compromised. Nevertheless, even with more appropriate hardware and the correct operating system and hardware configuration, the Apache and Linux combination was slower than the proprietary combination.

This result was a wake-up call to the Linux community. Linus Torvalds acknowledged that “we had become complacent.” Red Hat took this as a challenge and resolved to put Red Hat Linux at the top of the performance charts. Dell offered to assist, providing hardware expertise, benchmark expertise, and the high-end hardware configurations needed to design and test new ideas.

Where possible, TUX parses input packets directly, enabling zero-copy HTTP-request parsing. Even in RAM-limited situations, TUX now performs full, back-to-back zero-copy I/O. Finally, additional tuning of the RAID infrastructure in Linux keeps performance high, even when TUX is forced to go to disk.

Improved design and functionality

The kernel overhaul was critical to TUX's enhanced performance. The following comparison of TUX to Apache Web server illustrates how the TUX-inspired 2.4 kernel improved baseline performance.

Apache Web server was not designed primarily for speed, but for flexibility and extensibility; therefore, a direct comparison of Apache and TUX performance may be misleading. One powerful and flexible Web serving model uses TUX only for content that requires very high speeds and Apache to handle the remaining HTTP workload.

When Apache runs on Linux, the master Apache listens on port 80 and calls the `accept()` function. Once a browser connects to this server and the TCP handshake has completed, the `accept` system call returns with a new socket. Writing to this socket sends data to the client, whereas reading from this socket grabs the HTTP request headers. Once a request has been accepted by an Apache process and the HTTP request is complete, the Apache process handles this request synchronously—that is, it will execute only this request, and nothing else. While, for example, the server reads a file from disk and sends it to the client, the request blocks all processing of further requests.

Figure 1 summarizes the Apache workflow.

Every operation on Apache can result in an I/O block for different reasons, which are summarized in Figure 2. Apache solves this problem by starting more child processes when there is too much work, but this creates excessive context switching, in addition to the large overhead of coordinating child processes and connection requests. Heavily loaded servers can spend as much as 25 percent of their CPU arbitrating connection requests. Other proprietary Web servers use threads instead of processes, but both models have similar performance limitations.

The TUX architecture represents a new approach to operating system support for network-delivered services. Traditionally, the kernel space and user space are separated for security purposes, leading to a high cost of switching between the two. Traditional approaches often result in either running too much in the kernel and making it vulnerable to security violations, or running too much in user space and duplicating functionality that would be better managed in the kernel.

TUX overcomes this traditional thinking by providing a high-level application-protocol interface in user space, while handling simple tasks purely within the kernel. Only complex requests are

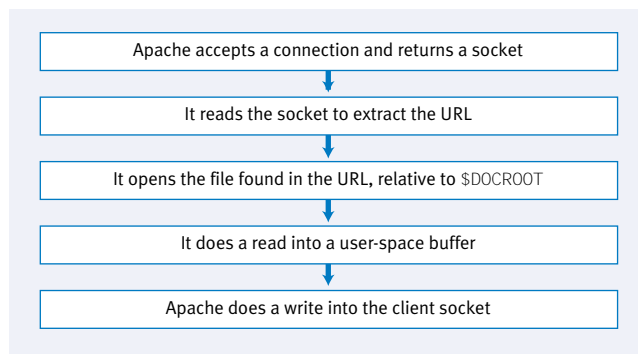


Figure 1. Apache workflow

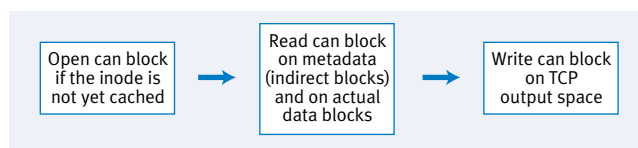


Figure 2. Operation blocking

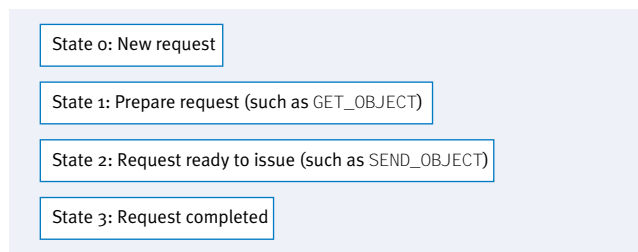


Figure 3. TUX state machine

Queue	Action
Input queue	Valid, but incomplete input; wait for more packets to arrive
User-space queue	Send a request to user space
Cache-miss queue	Handle cache miss now
Postpone queue	TUX itself is done, but more work is needed
Output queue	Send a reply right now
Redirect queue	Send the connection to the secondary Web server
Finish queue	Send a request to logger and be done

Figure 4. TUX queues

passed to user space; other operations such as serving static pages or CGIs are accelerated in the kernel. However, user-space code has to use a new interface based on the TUX system call.

The TUX state machine (see Figure 3) is replicated across several different queues so that, instead of blocking, each part of the Linux kernel can operate asynchronously. By separating each system call into atomic elements that can be queued, TUX threads only block when there is no work to be done, anywhere. Figure 4 lists several types of queues used by TUX. Because TUX threads never

Vendor	Hardware	SPECWeb99 results— number of connections	Web server	Number of CPUs	Remarks
Dell	PowerEdge 8450/700	7500	TUX 2.0	8	World record for 8 CPUs
Dell	PowerEdge 8450/700	7300	IIS 5.0 and Scalable Web Cache (SWC) 3.0	8	
Dell	PowerEdge 8450/700	6387	TUX 1.0	8	
Dell	PowerEdge 6400/700	4200	TUX 1.0	4	World record for 4 CPUs
Compaq®	Alphaserver ES40	2304	Zeus 3.37	4	
Dell	PowerEdge 6400/700	1598	IIS 5.0	4	
Dell	PowerEdge 1550/1000	2765	TUX 2.0	2	World record for 2 CPUs
Dell	PowerEdge 4400/800	2200	TUX 1.0	2	
Dell	PowerApp.web 120	1070	IIS 5.0	2	
Dell	PowerEdge 4400/800	1060	IIS 5.0	2	
Compaq	Alphaserver DS20 6/667	1050	Zeus 3.3.5	2	
Dell	PowerEdge 2400/667	1270	TUX 1.0	1	Previous world record, surpassed by TUX 2.0 on Intel-based hardware
Dell	PowerEdge 2400/667	732	IIS 5.0	1	


Figure 5. SPECWeb99 benchmark results²

block other processes, there is no need to allocate more than one thread per CPU, thereby minimizing scheduling overhead.

But the proof of the system is in the numbers. Figure 5 compares benchmarking results of the new TUX architecture with those of earlier TUX versions and other proprietary Web server and hardware combinations.

TUX's dramatic reduction in resource requirements is perhaps even more impressive than the speed demonstrated in benchmark testing. For the two-CPU TUX 2.0 results, TUX handled over 2,700 connections (a capacity record for two-CPU systems), and served a 9 GB file set using 2 GB of memory. Most other systems represented in the SPECWeb99 results required at least as much system RAM as the file set size.

In the TUX architecture, the kernel does not need to multiplex and demultiplex Web connections through polling (or selecting). This makes it possible to handle tens of thousands to millions of connections, depending on network bandwidth requirements, on machines that might otherwise be limited to hundreds of connections.

The results achieved with TUX as a Web server have been encouraging, but they are only the beginning. Already, Red Hat is working to accelerate FTP and other network services. Those who have built their own servers for proprietary distributed applications have also begun to consider running TUX on Red Hat Linux to improve the scalability and performance of their systems, while preserving their fundamental application platform. 

Michael Tiemann (tiemann@redhat.com) is chief technology officer (CTO) of Red Hat, a leading supplier of Linux and open source solutions. Michael made his first major open source contribution over a decade ago by writing the GNU C++ compiler, the first native-code C++ compiler and debugger. In 1989, Michael co-founded Cygnus Solutions, the first company to provide commercial support for open source software. Michael also serves on several boards, including the Open Source Initiative, the Embedded Linux Consortium, the GNOME Foundation, the Jabber Technical Advisory Board, and the Board of Directors of ActiveState Tool Corp.

FOR MORE INFORMATION

For questions or comments about TUX, please join the tux-list@redhat.com mailing list. For instructions on joining the mailing list, see <http://www.redhat.com/mailling-lists/>

Also visit the Red Hat TUX Web Server product page at <http://www.redhat.com/products/software/ecommerce/tux>

See the Red Hat TUX Web Server Support page at <http://www.redhat.com/products/support/ecommerce/tux>

For the latest development source, see <http://www.redhat.com/products/ecommerce/tux/>

²SPECweb99 is a trademark of the Standard Performance Evaluation Corporation (SPEC). Competitive numbers shown reflect results published on www.spec.org as of 12/31/2000. The comparison presented is based on the top results of Dell and Compaq servers. For the latest SPECweb99 results, visit <http://www.spec.org/osg/web99>