Linux Technology Center Testing

The purpose of this paper is to summarize enterprise test results from the IBM Linux Technology Center Test Team (LTC Test). LTC testing is an integral part of IBM's vision to work with the community to ensure Linux remains enterprise ready. The LTC Test team is responsible for engaging with the Linux community and delivering test suites, documented stress runs, reliability assessment, and verifying defect fixes found by the test team through regression testing.

LTC Test is part of the greater Linux test community whose goal is to ensure that the Linux OS, as delivered by the community to the distros, is as robust and defect free as possible. The work of the test community enables the distros to take the Linux OS, combine it with many other packages into Linux Distributions and test that combination. IBM brands can then take Linux Distributions and system test the appropriate HW/SW product and application stacks before delivering Linux-enabled IBM products to the field. A measure of the success of the LTC Test team's contribution to this process is the activity of the Linux Test Project (LTP) test suite hosted on SourceForge: There are approximately 400 downloads of the test suite per month with over 100 subscribers to the project.

Background

The Linux Technology Center was formed in August, 1999. In late 2000, the LTC team grew from 50 to 250 members. The LTC Test team was added as a critical mission in January, 2001. In just one year, the LTC Test team has accelerated Linux System test. While establishing the lab topology and infrastructure, the team immediately began engaging with the Linux community to deliver robust test suites and patches that focus on the Linux operating system through the Linux Test Project (LTP). The LTP was seeded with 100 test cases developed by SGI™. Since that time, the test suite has grown to almost 600 test cases with contributions from IBM, SGI, OSDL, and Group Bull® along with individual Linux developers. The LTC test team ported many AIX system test cases and enabled several hundred Dynix / PTX test cases, originally developed by Sequent, to run under the SGI PAN test harness.

The team has also developed many test cases, some deriving from defects found during testing and Linux community feedback. Using the LTP, the test team has tested 50 new versions of the kernel. Eight releases of the LTP test suite are hosted on SourceForge, one of the major Linux developer sites. There are approximately 400 downloads of the test suite per month with over 100 subscribers to the project. The Linux Test Project support multiple architectures s/390 (32 & 64-bit), PPC32, and IA32/64 platforms. Another aspect of our testing is working with the Linux Standards Base group and delivering Linux Standards Base test suites to the community.

The test results reported in this document represent results obtained from running tests under specific conditions at IBM's laboratory facility in Austin, Texas. Users may achieve different results depending on their particular installations, configurations, workloads, or other factors. Users are, thus, advised to evaluate the code referenced in this document as is appropriate for their specific installations.

Why is Linux Test Infrastructure Needed in the Linux Community?

As the Linux operating system evolved, there was no formal testing environment available for Linux developers and maintainers to access system level tests to verify that their new code and patches would run in a system environment. There were no test suites available to check for compliance to standards being established to make application development consistent across distributions.

Has Linux improved over the past 12 months? Yes, and here's how.

The overall robustness, reliability of the Linux OS has matured and the following is a summary of enterprise workloads that have been successfully tested on the 2.4 kernel series:

- 1.7 Terabyte files have been created and successfully accessed for over 60 hours
  - 1.2 million 1K blocks successfully read
  - 2.005 billion 1K blocks successfully created
  - 1.241 million read requests
  - 51.383 million write requests
  - 87.891 million 1K blocks read
  - 494.452 million written
- Over 15 million NGPT threads have been successfully created during a 96 hour stress run and utilized by the OS and LTP testcases
- 250 GB databases have been created and stressed for 96 hours with over 12.9 million inserts, queries, and update transactions
  - 4.360 million inserts
  - 4.361 million updates
  - 4.361 million queries
- 7488 hours stress testing against the 2.4 Linux kernel (72 hour runs are executed (twice weekly))
- VMM was stressed for 96 hours on 8-way machines using high memory scenarios that utilized 90 - 99% of real memory
- 168 hour continuous stress runs have been executed against the NFS v2 and NFS v3 concurrently using 2047 MB file sizes
- 161 Linux defects have been opened and115 have been resolved

What is the Linux Test Strategy?

The Test Strategy is to execute test scenarios in a system environment focusing on component stability, integration of component workloads and robustness of the overall system components. Linux reliability runs validate the robustness of the kernel subsystems to support enterprise level computing. The testing strategy is designed to ensure product stability prior to integration. Stress tests are executed during the focus test timeframe. The Test team demonstrates Linux components integrate successfully.

The following areas are tested using the LTP test suite and community workload tools:

- The goal of *Focus Test* is to isolates and validates Linux component and application stability. Linux components are regression tested on new release candidate kernels. Early testing of Linux packages is geared to identify or isolate pervasive or critical defects before packages are released in the Open Source Community. New tests are developed as needed for component testing.
- The goal of *Integration Test* is to merge component workloads and validates component interactions using the integration test stack. See Fig. 1.0 for additional details.
- The goal of *Reliability or Stress Test* is to validate the robustness of the overall system components during 96 hours stress tests.

Focus, integration and reliability test the following components with various configurations as appropriate.

- Memory management
- VMM, Paging space
- Scheduler (process, stack handler)
- Linux Threads
- Next Generation POSIX Threads (NGPT)
- System Calls
- Filesystem / enterprise volume management
- Networking Subsystems
- Applications
- Databases
- Web Servers (Apache, HTTP server)
- Web application servers

Integration Test Stacks

Integration tests create customer scenarios that view the system as a whole.
The integration test efforts utilize stress tests to verify robustness of the product during high system usage. Tests are run using various combinations of the following integration components.

Distros	Threads	Filesystems	Networking	Databases	Web Servers	Architecture
Red Hat	Linux Threads	Ext2	NFSv2	DB2	Websphere	X86
SuSE	POSIX Threads	JFS	NFSv3	MY SQL	Apache	z-Series S/390
TurboLinux		ReiserFS	TCP/ IP v4	Sybase	HTTP	PPC
Caldera		Ext3	TCP/ IP v6	Oracle	Trade2 APP	IA-64
Monta Vista		Bonnie ++	UDP	Postgres	AKstress	
		Dbench	10/100 ETH	Dots		
		Cerebus	GB ETH	T3		
		Iozone	NetPerf			
		Postmark	Volanomark			
		EVMS	Connectathon			
			AIX 5.1			

Table 1.0 Integration Test Stacks

Linux Test Integration Runs

Integration Test merges component workloads and validates OS component interactions. The integration test effort requires stress tests that verify robustness of the product during high system usage. Components of Linux are integrated using the following component stacks.

Test Area	Distro	Kernel	Duration	Processors	Memory	Dasd	Tool	Results
Kernel	Red Hat, SuSE, Turbo	2.4.17	100 hrs.	8-way	12GB	30 GB	LTP	99% Pass
Kernel	SuSE 7.2	2.4.14	140 hrs.	8-way	12GB	30 GB	LTP	99% Pass
WebSphere AES 4.0 (WAS)	Red Hat 7.2 SuSE 7.3	2.4.16 2.4.17	72 hrs.	8-way DB 4-way WAS 4-way WAS	12GB 1 GB 1 GB	4 TB 180 GB 180 GB	Trade 2 akstress	Pages : 17,472,505 Connects: 17,472,492
Filesystems	SuSE 7.3	2.4.17 + JFS 1.0.14	48 hrs	8-way	10GB	35Gb +5.2 Terabyte SAN	lftest	100% pass 1.9 Terabye contigous file succ created 2,005,991,424 (1K) blocks.
Filesystems	SuSE 7.3	2.4.17 + JFS 1.0.14				2Terabytes	Bonnie ++	100% pass Read requests: 1,241,600 Write requests: 51,383,325 Blocks Read(1K): 87,891,700 Blocks Written(1K): 494,452,650
DB2 7.2 Database	SuSE 7.2	2.4.7	96 hrs.	4-way	256MB	64GB	Dots	100% pass Inserts 4360277 Updates 4361666 Queries 4361943

NGPT	Red Hat 7.2	2.4.16+NGPT 1.1.1 Patch	96 hrs	2-way	512MB	4GB	LTP	99% pass test halted
VMM (High memory)	SuSE 7.3	2.4.16	24 hrs	8-way	10GB	12GB	LTP	99% Pass 5GB swap
Networking	SuSE 7.3	2.4.18	96 hours	8-way 8-way	9 GB 9 GB	36GB 36GB	Netpipe LTP	99% pass (12) GB ETH cards, Direct Connect

Table 2.0 Linux Test Integration Runs
Note: All runs were terminated gracefully at the end of duration unless otherwise noted.
Bug reports have been opened for problems that resulted in a success rate of <100%. All testcase failures are documented on ltp.sf.net .

LTP run summary

The following seven diagrams snapshot typical integration test runs:

- **Fig. 01 NGPT and Linux threads Integration runs on a 1-Way**
- **Fig. 02 NGPT and Linux threads Integration runs on a 8-Way**
- **Fig. 03 Database, JFS and EVMS Integration runs on a 8-Way**
- **Fig. 04 NGPT and Apache Integration runs on a 1-Way**
- **Fig. 05 Virtual Memory Manager stress runs on a 8-Way**
- **Fig. 06 LTP Testcase Executions**
- **Fig. 07 NFS integration runs**

NGPT and Linux threads

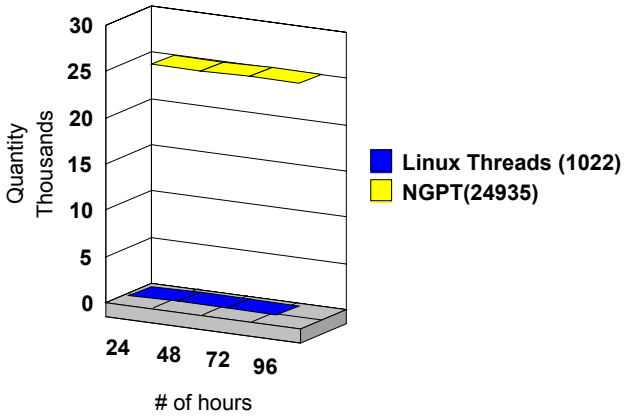
Integration runs on a 1-Way

- Linux Threads
  - Machine: PIII - 866Mhz
  - Kernel: Linux 2.4.18-rc2
  - 1022 Threads created by max thread creation test
  - 1073 processes running on machine
  - 59.6% User CPU utilization
  - 40.3% System CPU utilization
  - 97.81% Memory utilization (256 MB total memory)

- Next Generation POSIX Threads
  - Machine: PIII - 866Mhz
  - Kernel: Linux 2.4.17
  - 24935 Threads created by max thread creation test
  - 55 processes running on machine
  - 99.6% User CPU utilization
  - 00.3% System CPU utilization
  - 98.18% Memory utilization (256 MB total memory)

- Observations
  - NGPT will provide better performance for multi-threaded applications
  - LTP pth_str02 was used as the max thread creation tool
  - The maximum number of threads ranged between 23K -24.9 per process for the LTP pth_str02 testcase
  - The average was < 180 seconds elapsed time to create 24k threads for 4 pth_str02 processes, 896 MB memory
  - Internal thread resource utilization was minimized

Fig. 01



NGPT and Linux threads

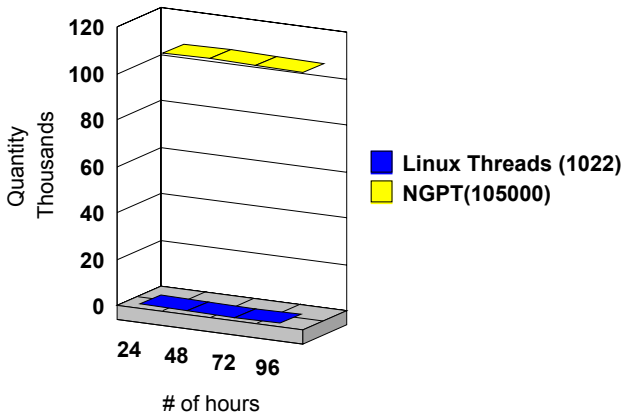
Integration runs on a 8-Way

- Linux Threads
  - Machine: (8)PIII - 700Mhz
  - Kernel: Linux 2.4.17
  - 1022 Threads created by max thread creation test
  - 1073 processes running on machine
  - 59.6% User CPU utilization
  - 40.3% System CPU utilization
  - 97.81% Memory utilization (896 MB total memory)

- Next Generation POSIX Threads
  - Machine: (8) PIII - 700Mhz
  - Kernel: Linux 2.4.17
  - 105000 Threads created by max thread creation test
  - 55 processes running on machine
  - 99.6% User CPU utilization
  - 00.3% System CPU utilization
  - 98.18% Memory utilization (896 MB total memory)

- Observations
  - NGPT will provide better performance for multi-threaded applications in particular on SMP machines.
  - LTP pth_str02 was used as the thread creation tool
  - The maximum number of threads ranged between 23K -24.9 per process for the LTP pth_str02 testcase
  - The average was < 60 seconds elapsed time to create 24k threads for 4 user processes
  - Internal thread resource utilization was minimized

Fig. 02

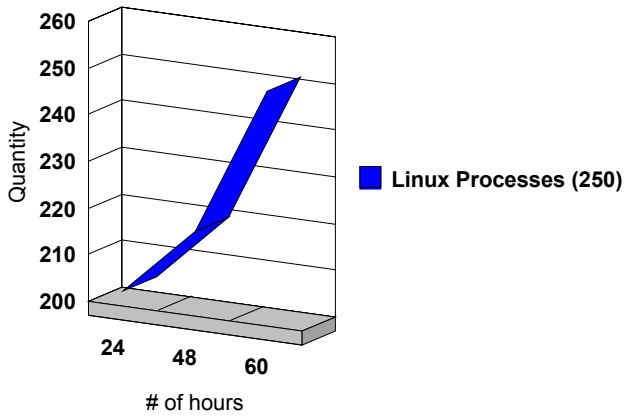


Database, JFS and EVMS

Integration runs on a 8-Way

- Linux Threads
  - Machine: (8)PIII - 700Mhz
  - Kernel: Linux 2.4.18-rc2
  - 218 Processes created by DB2 transactions
  - 46.44 % User CPU utilization
  - 9.83 % System CPU utilization
  - 99.94 % Max Memory Utilization (12112628 KB)
  - 76.97 % Average Memory Utilization during 60 hours run
- Observations
  - Database transactions generate significant stress on Linux tasks (processes)
  - DB2 7.2 using JFS 1.0.15 on EVMS 0.9.0 were used as the integration components during the run.

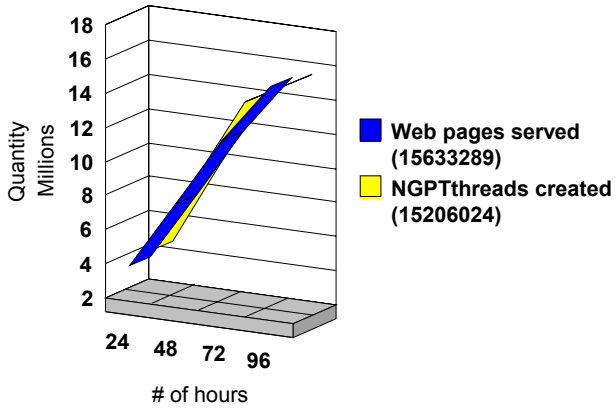
Fig. 03



NGPT and Apache Integration runs on a 1-Way

- Next Generation POSIX Threads, Apache 2.0.28
  - Machine: PIII - 866Mhz
  - Kernel: Linux 2.4.16
  - 15206024 NGPT Threads created
  - 71 processes running on machine
  - 26.08% User CPU utilization
  - 8.67% System CPU utilization
  - 90.69% Memory utilization (256 MB total memory)
  - 40.47% Swap space used (1 GB total swap)
- Observations
  - The results indicate that the ratio for threads vs web pages served is 1:1 plus OS generated tasks
  - Note: Automated Web test tool is been used as the client of the Apache server. 30 virtual clients are created and each client has 3 threads. 15633289 web pages are served by Apache server.

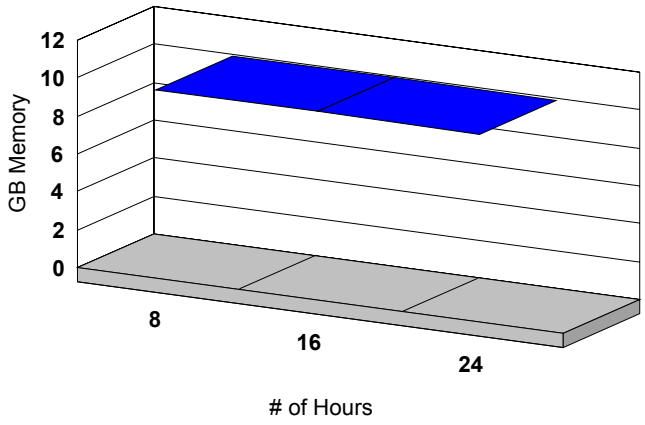
Fig. 04



Virtual Memory Manager stress runs on a 8-Way

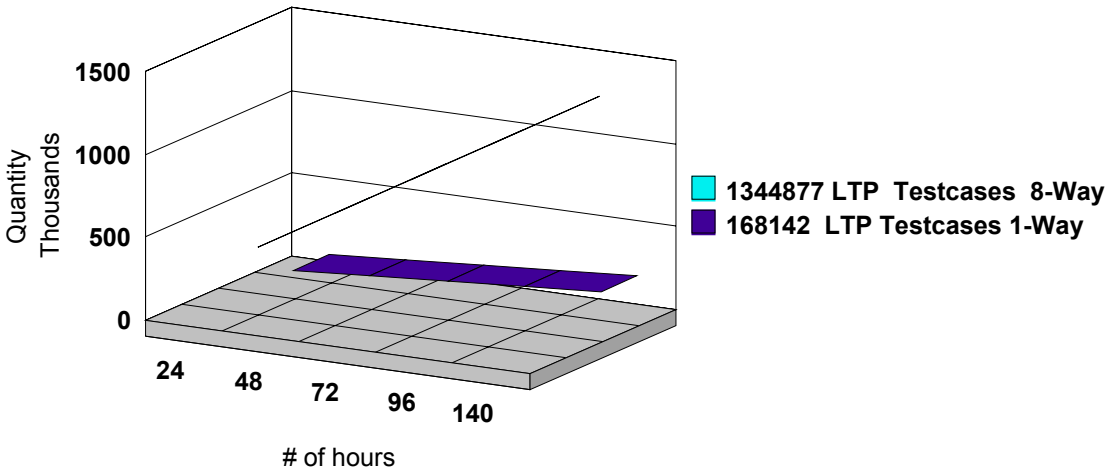
- Linux Memory utilization
  - Machine: (8)PIII - 700Mhz
  - Kernel: Linux 2.4.18-rc4
  - 80 Average Linux Processes(tasks) per second
  - 23.15 % User CPU utilization
  - 68.53 % System CPU utilization
  - 99.77 % Max Memory Utilization (10GB RAM)
  - 2 GB Swap size
  - 97.77 % Average Memory Utilization during 24 hours run
- Observations
  - The Linux 2.4.16 and later kernels successfully ran the following VMM scenarios:
    - Performed memory Stress with Race conditions
    - Simultaneous map/unmap/read
    - Repeated map/write/unmap of a of a large GB size file.
    - Repeated map/write/unmap of a of random size file.
    - Repeated mallocs and frees of random size chunks of memory
    - VMM was successfully stressed for 96 hours on the 2.4.16 and later kernels

Fig. 05



LTP Testcase Executions

Fig. 06



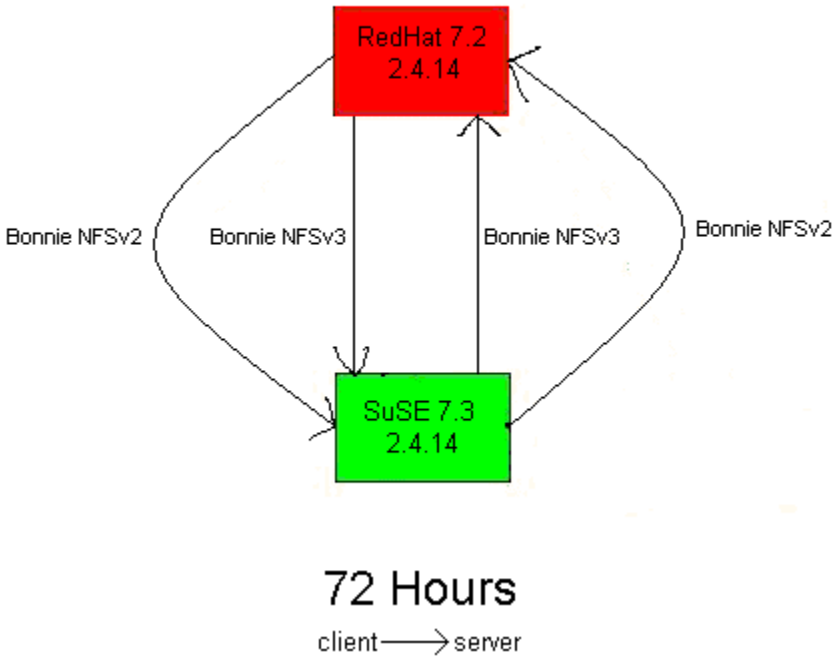
- 8-way
  - Machine: (8)PIII - 700Mhz
  - Kernel: Linux 2.4.17
  - 32.34 % User CPU utilization
  - 21.42 % System CPU utilization
  - 99.05 % Max Memory Utilization (12112628 KB)
  - 45.00 % Average Memory Utilization during the run
- 1-way
  - Machine: PIII - 866Mhz
  - 56.04 % User CPU utilization
  - 21.56 % System CPU utilization
  - 99.11 % Max Memory Utilization (256MB)
  - 47.04 % Average Memory Utilization during the run

- Observations
  - LTP2000207 was the test tool
  - 8-way (8) instances of each test were executed for 140 hrs
  - 1-way (1) instances of each test were executed 140 hrs
  - 7.9 more testcases were executed on the 8-way vs 1-way
  - Linux scheduler successfully balanced the testcase tasks for 96 hours
  - 8-way does provide significant scalability enhancements

Appendix A: NFS integration runs:

The robustness of NFSv2 /v3 is defined by how well it handles large amounts of data and traffic between multiple clients, while other tasks are being performed on the system. LTP kernel tests were run on the server. NFS traffic was generated by various applications and tests that ran on multiple clients.

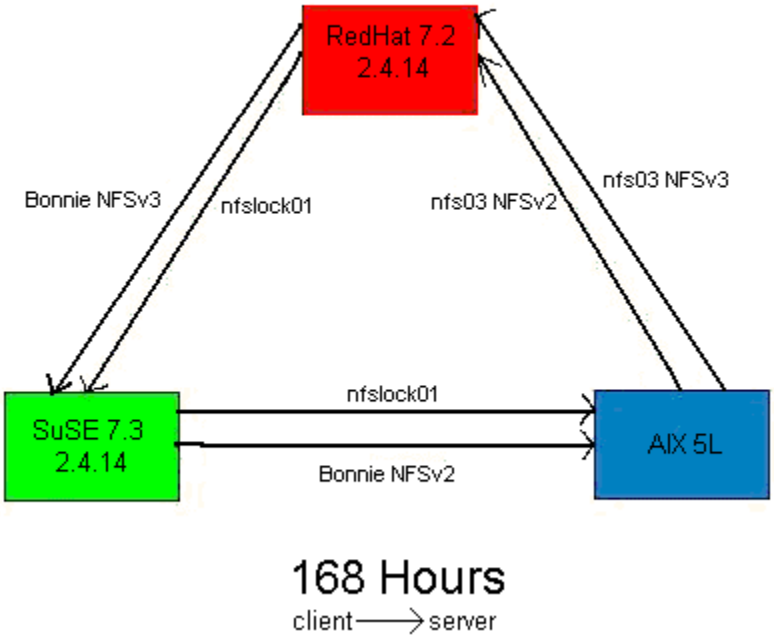
Fig. 07



The following tests were executed for successfully for 72 hours.

- Bonnie - Intense, stressful filesystem and I/O bottleneck benchmark.
- nfslock01 - Two processes open FLOCK_IDATA file simultaneously each one locks odd and even lines of the file simultaneously and fill them with '0's and '1's. After they find eof, the datafiles are compared.
- nfs03 - Rapidly creates and deletes files through multiple processes running in the background. The user may specify the number of subdirectories to create, the number of files to create (per subdirectory), and the number of times to repeat the creation/deletion cycle.

Fig. 08



The following tests were executed for successfully for 72 hours.

- **Red Hat -> SuSE**
  1. Bonnie was able to generate enough traffic to help occupy the CPU at approx. 80% load, with about 30% going to the rpciod process. Bonnie ran successfully without any recorded errors.

  2. The nfslock01 test completed 111586 iterations a success rate of 99%.

  SuSE -> AIX
  1. Bonnie was able to generate enough traffic to help occupy the CPU at approx. 80% load, with about 28% going to the rpciod process and 15% going to the nfsd process. The rpciod deamon is the client-side NFS process daemon, and the nfsd daemon is the server-side NFS process. Bonnie ran successfully without any recorded errors.

  2. The nfslock01 test completed 110134 iterations a success rate of 99%.

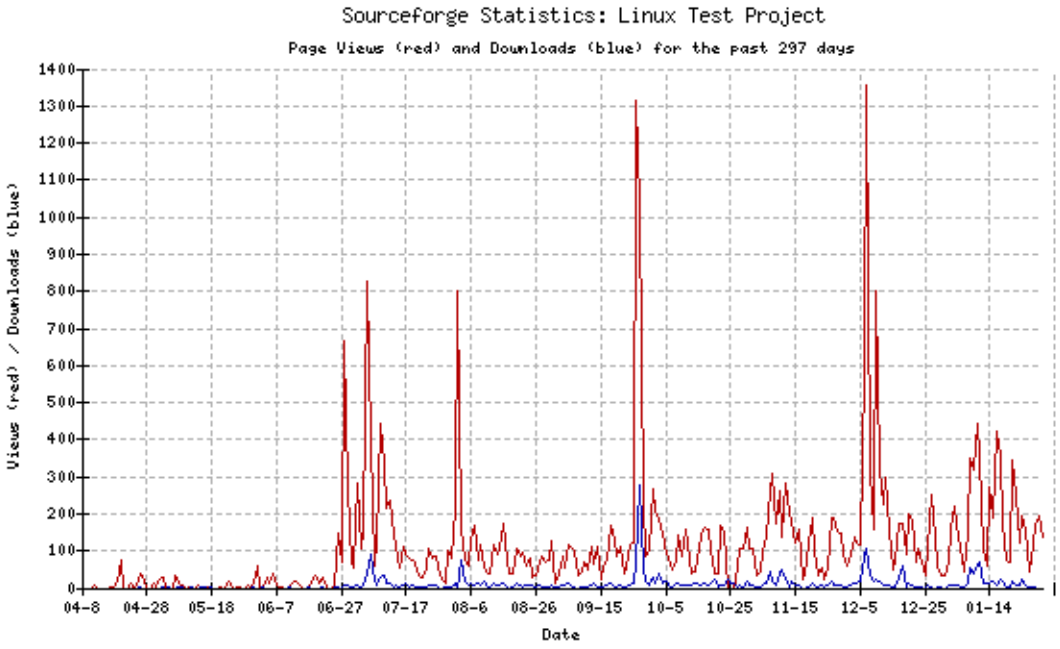
- **AIX -> Red Hat**
  The NFS server daemon, nfsd, occupied 31% of the Red Hat machine's CPU. This high usage was obtained by lowering the number of server daemons from the

default number of 8 down to 1, and running 2 copies of the nfs03 test (one over NFSv2 and the other over NFSv3).

Appendix B: LTP release history on http://ltp.sourceforge.net

- April 2001 – Project moved to SourceForge 100 test cases
- June 2001 – 50 test cases added to the release
- Sept. 2001 – 400 test cases added to the release
- Nov. 2001 – Bug release
- Dec 2001 – Database Open source Test Suite released
- Jan 2002 – Bug release
- Feb 2002 – Cross architecture support IBM s/390, PPC, IA-64

Significant web activity after each release



Appendix C: LTP Tools details

The Linux Test Project is a joint project with SGI™, IBM®, OSDL™, and Bull® with a goal to deliver test suites to the open source community that validate the reliability, robustness, and stability of Linux. The Linux Test Project is a collection of tools for testing the Linux kernel and related features. Our goal is to improve the Linux kernel by bringing test automation to the kernel testing effort. Interested open source contributors are encouraged to join the project.

The LTP has released over 600 tests to the Open Source Community. Several complex IO tests (doio, growfiles, pipeio) and over 25 reliability network tests for remote procedure calls, network file systems, multicast, and various network commands, pthreads, memory, filesystems, disk I/O and test driver (pan). Tools is an area we continue to enhance and several analysis tools for pan output are available. The LTP also contains several tests for commands commonly used in an application development environment.

Enterprise Volume Management System

The EMVS System Test effort utilizes a variety of tools, each of which exercises different aspects of the installed filesystem and, as a result, the EVMS layers upon which the filesystem is installed.

FS_INOD

FS_INOD was originally created for AIX testing and was modified to test Linux filesystems. FS_INOD requires four parameters:

- Volume name upon which FS_INOD runs.

Number of directories FS_INOD creates. These directories are created beneath two parent directories (dir1 and dir2), there are actually twice this number of directories created.
Number of files created in each directory. FS_INOD uses Touch to create these files.

- Number of loops for FS_INOD to execute.

When invoked, FS_INOD creates two parent directories and populates them with the specified number of subdirectories. Then it begins the loop of creating the specified number of files in each directory, using a separate process for each parent directory, and removing all files. FS_INOD executes the specified number of loops, at the end of which it will clean up by removing all files and directories it has created.

The larger the disk, the longer it takes to execute each loop. Experience has shown that 30 loops on an approximately 4GB disk execute for about 72 hours. During this time FS_INOD consumes an average of approximately 15 to 20% of the cpu's cycles, with 100% utilization peaks occurring regularly. Greater cpu utilization percentages can be obtained by running multiple instances of FS_INOD.

FTHRASHER

FTHRASHER, or File Thrasher, is a tool that can hammer a filesystem and utilize significant cpu cycles. It is an internal IBM tool that was originally written to test on the OS/390, and has been modified for use on Linux. FTHRASHER requires the following parameters:

- Size of blocks used to create each test file.
- Number of blocks used to create each test file.
- Number of test files to create.
- Number of seconds between reports.
- Number of processes accessing the test files.
- Ratio of reads to writes, expressed as a percentage.
- Number of hours to run the test.

LFTEST

The Large File Test, created by the LTC Test Team, explores a filesystem's boundaries with regards to the maximum file size that can be created or manipulated. LFTEST is designed to create large files and **lseek** from the beginning of the file to the end of the file after each block write. This test verifies large file support and can be used to generate large files for other filesystem tests to use.

WebSphere Integration Testing

WebSphere is used to test the ability of Linux to withstand the stresses placed on it in a web application server environment. We have taken IBM Websphere 4.0 matched with IBM DB2 and added the Trade2 stock trading application. This scenario is driven with Linux clients using the IBM akstress test tool. By testing in this manner we are able to simulate thousands of clients wanting to perform stock transactions. As a result we can place a heavy load on components of Linux from network communications to file systems.

DOTS

Database Opensource Test Suite, DOTS, is a set of test cases designed to generate stress on database server systems. Dots will stress threads, process scheduling and memory utilization of the OS. Dots will measure database server

reliability and robustness on Linux. DOTS is made up of two test case categories: Basic and Advanced test cases. The primary goals of the Basic Cases are stress testing and 100% JDBC API coverage. The goals of the Advanced Cases are modeling a real-world business application in addition to stress testing on database systems. There are ten test cases in total, all written in Java. The supported (Relational Database Management Systems (RDBMS) are DB2, Oracle, and Sybase.

Next Generation POSIX Threads (NGPT)

NGPT introduces an M:N threading model to the Linux system. This model will provide better performance for multi-threaded applications that utilizes the POSIX pthreads library functionality. This will be particularly true on SMP machines. The goal of this project is to make threading on Linux more robust, more POSIX compliant and more in line with the services provided by commercial Unix operating systems.

Description of available NGPT tests:
test_str01: Creates a large tree of threads
test_str02: Creates a large number of threads sequentially
test_str03: Creates a large tree of threads, each child performs calculations and returns the result to the parent
test_pthread: General test of Pthread API
test_pthread_cancel: Test the pthread_cancel() function
test_pthread_segv: Pthread signal handling test
test_pthread_sig: Test signal handling in pthreads
test_cleanup: Test using cleanup function on thread cancellation

Document Author:
Linda J. Scott
IBM Linux Technology Center

4/4/2002