Executive Summary

In an effort to assess the progress Lunar Linux has made toward becoming a viable enterprise platform, the Lunar team ran a number of benchmark tests to directly compare Lunar Linux to Red Hat Linux.

Ultimately, the goal of these tests was to enable an accurate characterization of any operative advantages Lunar might provide over other Linux distributions. The Lunar development group had long perceived these advantages, but prior to this series of tests had no evidence beyond anecdotal testimony from developers and users which indicated a distinct interactive speed enhancement when using Lunar. The results strongly verified these initial observations. The advantages highlighted in the testing are mainly due to Lunar's distributed application management system (AMS provided by Lunar's core code) and the simplified customization routines it provides which enable user software configuration and optimization.

The benchmarks used were selected to characterize web-service performance and local computational tasks. They included comparisons of bash shell and perl performance, the Mysql database and the apache web server. Perl is commonly used for system administration, webservices and data-reduction tasks. Apache and Mysql are the most widely deployed web server/SQL database (when used on Linux machines with PhP this is commonly called LAMP) combination and are both noted for their speed and simplicity. The Bash shell is the default shell provided by both Lunar and Red Hat Linux.

Lunar Linux aims to improve performance by simplifying and improving configuration and dependency management. Lunar makes it easy for IT departments to keep their systems up to date, and its local build routine allows applications to be optimized/customized for local hardware configurations. When using Lunar, optimization of applications may be accomplished without resorting to hand-building applications, and IT staff members can easily track and document that work.

Lunar principally optimizes applications and libraries, taking a fairly conservative approach with the Linux kernel and core libraries. Some performance benefits could also be realized here, however we have found that the degree of return for the effort required to accomplish this is small, usually not better than 2-5%, while we routinely accrue 20-50% (or better) performance gains through our approach to building server applications that are locally tuned and optimized for their runtime environment.

Test descriptions and results

Database and web server

Mysql random table inserts This benchmark is designed to test the performance of reads and writes to the database as a table becomes fully populated. Generally database writes take longer, however as the table fills, the number of reads between writes increases.

Lunar Linux consistently runs this benchmark 100% faster than Mysql on RedHat.

Apache/php, using mysql cookie tracking This benchmark runs a many-users performance test against a web page requiring a database record insert for every page access.

Under moderate loads, Lunar and Redhat show nearly identical performance. At high load levels, RedHat performance is approximately 14% higher-- however both the database and apache services repeatedly locked, requiring reboot to correct and leaving corrupted filesystems. Lunar.....

Unixbench and lmbench Unixbench is an accepted benchmark used for testing performance on Linux and Unix(tm) servers. UnixBench provides information on application level performance. Lmbench tests low-level performance of a wide variety of system characteristics and can be useful in pinpointing the causes of higher level performance problems.

Lunar Linux averages 30% faster than RedHat Linux on Unixbench. Lunar is also generally faster than RedHat in the lmbench. The differences appear largely due to Redhat's customization of it's version of the Linux kernel and glibc.

Perl Perl has been the lingua-franca of CGI from the beginning of the widespread use of the Internet, It has also been the most widely used tool for system administration and data manipulation.

Perl performance is compared using the **perlbench** benchmark in addition to two routines which I have used for evaluating perl performance for a decade.

Using the perlbench routine, Lunar Linux's perl build performs at about 25% faster than perl on Redhat Linux. A larger performance is found in the data reduction and fibonacci tests, both of which run 30% faster on Lunar.

Benchmark details

System specs

CPU: dual P3/866 / 256kb cache Model, IBM netfinity 5100 server
HDD: SCSI adaptec aic7x 160mb/s w/ 1 18G hotswap disks fully allocated per OS
RAM: 256 mb / swap never activated during benchmarks
Lunar optimization:
-O3 -mcpu=pentium3 -march=pentium3 -mmmx -msse -mfpmath=sse,387Note

Unixbench results

Unixbench is a well known benchmark suite used to compare filesystem, scripting and process overheads. In this table, larger numbers indicate better performance.

Table: UnixBench microbenchmarks. File copy throughput is in megabytes per second. The other UnixBench microbenchmarks are in microseconds per loop iteration (or milliseconds for the shell scripts benchmark).

Microbenchmark	Lunar	RedHat	Difference
file copy 4KB	280.7	228.6	23%
file copy 1KB	301.7	173.7	74%
file copy 256B	322.5	144.7	122%
pipe	279.1	297.3	-6%%
pipe switching	279	n/a	n/a
process creation	318.8	409.5	-22%
execl	396.0	369.4	7%
shell scripts (8)	650.0	135.0	381%

Lmbench results

Lmbench measures low-level kernel and glibc performance. It is an important tool for understanding system-limited performance when application-level tuning avenues have been exhausted. In this table, smaller numbers indicate better performance.

Table: lmbench microbenchmarks. Measurements are in microseconds. Measurements below the bar represent round-trip latency for various forms of IPC.

Microbenchmark	Lunar	RedHat	Difference
null I/O	0.73	0.71	-3%
stat	2.5	4.7	85%
open/close	3.8	6.1	60%
0KB create	40.1	63.9	59%
0KB delete	8.4	21	150%
fork	303	292	-3%
execve	1002	1050	5%
sh	4971	4131	-17%
pipe	8.8	7.2	-18%
AF_UNIX	27.2	11.4	-58%
TCP	60	35.1	-41%
TCP connect	143	78	-45%

See appendix for Unixbench and Lmbench detail data

Apache web server and Mysql database

In their default configurations, Apache2 and Mysql on Lunar served pages 8% faster than RedHat after accounting for wrong pages transmitted under RedHat due to database connection failures. We found that on average 15% of pages were corrupted due to these errors.

We were able to fix this bottleneck in RedHat by increasing the connection-limit, this however resulted in severe server instability. The mysql database would begin to encounter errors after just 2 passes (15 minutes) of the stress-test and after 2 hours, the apache process stopped responding to queries. Neither service could be stopped by the standard control scripts, and after killing the mysql service (mysqld), the host filesystem was left with unremovable data, filling 500mb of space due to open filehandles.

The nominally faster RedHat performance is predicted by the low-level lmbench benchmark results. We assume that RedHat has put considerable work into optimizing Glibc and the linux kernel, particularly in the area of the network code layer. We also know that RedHat has backported NPTL multi-thread code changes from the 2.5 development kernel branch. It seems that these optimizations have introduced some stability problems.

Perl

Perlbench

In the perlbench benchmark test the optimized version of perl proved to be on average 25% faster than the perl interpreter in included with RedHat Linux. Lunar's Perl was consistantly faster than RedHat's, and individual tests showed Lunar giving performance enhancements ranging from 10-47%.

See appendix for raw Perlbench report data

Additional Benchmarks

In addition to this, we performed two pure cpu benchmarks, one a fibonacci caclulator, the other an edge-case, in calculating a moving average from a random data file. This was posted to comp.lang.perl in '94.

See: '94 results for various architectures

Fibonacci - simply calculates any fibonacci number, where the fibonacci sequence is defined as ... N + (N-1) + (N-2) ...

The recursive fibonacci algorithm is a handy benchmark simply because it generates a very large number of subroutine calls for relatively small values of N.

Data Reduction This script benchmarks the time to calculate a 60-value moving average on a data set, generating a new data set approximately 10x smaller. This is a handy pre-processor for time-series data, but has it's own computational cost.

Fibonacci results:

```
          n:   28    32     32(opt'n by Larry Wall)
redhat perl:   3.6s  25.1s  17.1
lunar  perl:   2.7s  19.3s  13.3
 difference:   0.75  0.77   0.77
```

Moving avg calc results:

```
redhat perl:   17.0s
lunar  perl:   12.4s
 difference:   0.73
```

Comments

These simple perl benchmarks are cpu-bound, the data-reduction script requires perl to fit into system cache and buffers 60 lines of input data to do processing.

Mysql - a perl script inserting random rows into a table

Forrest D. Whitcher, the developer who ran this benchmark test, wrote the following script in 1993 in oraperl to bench inserts into an Oracle DB.

The plots are elapsed seconds between 100k iterations through the loop. It takes about 2.1 million queries to 90% populate the table.

All the plots but the last two represent one client and server running on one system. Protocol latency seems to predominate and the perl benchmark running concurrent on the server machine will use slightly more CPU than the mysqldb process.

For this reason, after running the script locally on both Redhat and Lunar we ran the client side script on another 2xSMP linux system (with a slightly faster perl compiled with the Intel(r) C compiler). With this approach we were able to load the database engine to utilize 70-100% of the CPU.

Summary results

```
Time to insert 900k records into db best (worst) time

 bench:  1 client  4 clients  6 clients
------------------------------------------------------------
redhat:  42/(51)m  31m        n/a
 lunar:  24(28)m   15m        10(12)m

inserts/sec
redhat: 357(294)   483        n/a
 lunar: 625(535)   1000       1500(1250)

queries/sec
redhat: 833(686)   1129       n/a
 lunar: 1458(1250) 2333       3500(2916)
```

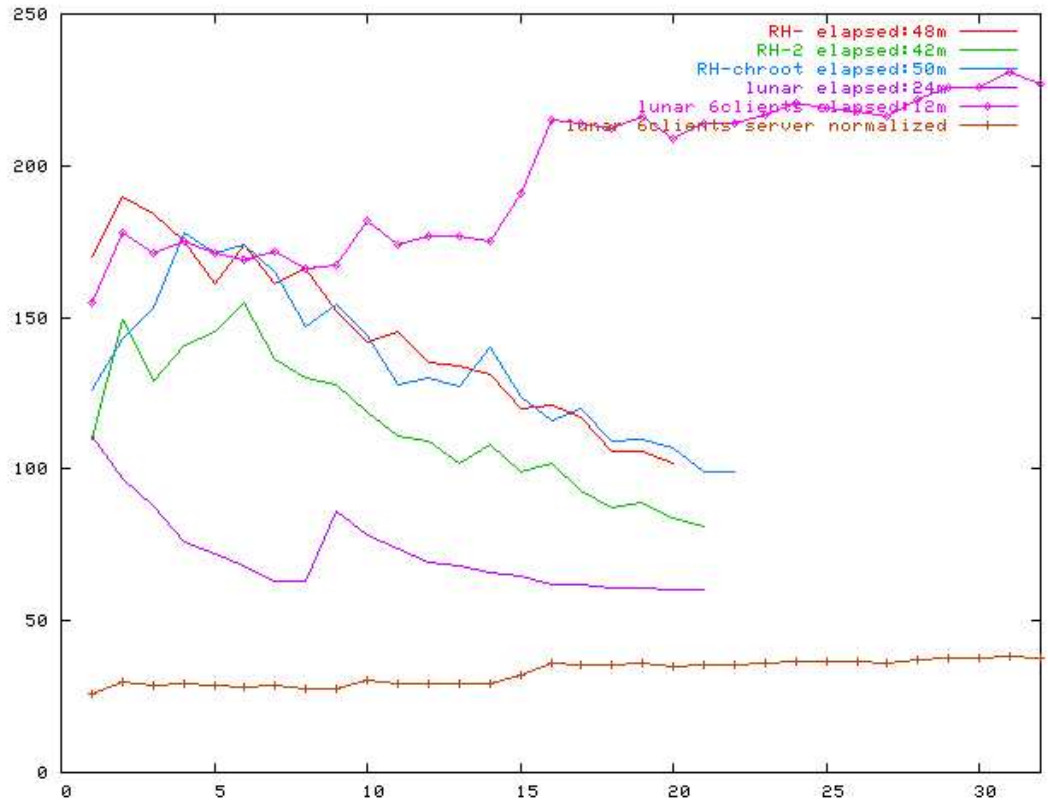
Comments

Interactive performance on RedHat suffered substantially, whether using the stock kernel or a vanilla linus-tree kernel when the mysql daemon was busy at more than 25% of cpu. For example:

```
time uname -a
Linux w2 2.4.20 #2 SMP Sat May 31 19:40:03 EDT 2003 i686 i686 i386 GNU/Linux
real    0m2.178s
user    0m0.000s
sys     0m0.000s
```

Interactive performance on Lunar (linus-tree kernel), on the other hand, remained instantaneous at all times even when running against 6 remote clients with both CPU's runnning at 100% utilization.

Graph: time per 100,000 db queries during bench run



Code

perl fibonacci

```
#!/usr/bin/perl
$n = @ARGV[0];
$f=&fib($n);
print "$nth fib = $f\n";
sub fib {
    local ($n)=$_[0];
    if ($n==0) {return (0);}
    elsif($n==1) {return(1);}
    return (&fib ($n-1) + &fib($n-2));
}
```

Larry Wall's optimized fibonacci

```
sub fib { $_[0] == 0 ? 0 : $_[0] == 1 ? 1 : &fib($_[0]-1) + &fib($_[0]-2) }
```

perl moving averages on random data

bnchtest.sh

```
#!/bin/sh
uname -a > result.dat
date >> result.dat
perl mkdata.pl 100000 > 100k.dat 2>> result.dat
perl movavg.pl 100k.dat > /dev/null 2>> result.dat
perl movavg.pl 100k.dat > /dev/null 2>> result.dat
perl movavg.pl 100k.dat > /dev/null 2>> result.dat
date >> result.dat
rm -f 100k.dat
```

mkdata.pl

```
#!/usr/local/bin/perl
($utime, $stime, $cutime, $cstime) = times();
print (STDERR "compile: user = $utime, system = $stime\n");
$ndata = $ARGV[0];

for ($i=0; $i<=$ndata; ++$i){
        $r1 = rand();
        $r2 = rand();
```

```
        printf "%d %f %f\n", $i, $r1, $r2;
        }
($utime, $stime, $cutime, $cstime) = times();
print (STDERR "run: user = $utime, system = $stime\n");
```

movavg.pl

```
#!/usr/local/bin/perl
$sumi=$sumo=$avgi=$avgo=0;
($utime, $stime, $cutime, $cstime) = times();
print (STDERR "compile: user = $utime, system = $stime\n");

while (<>)
 {

 ($hr, $inp, $outp) = split(' ',$_, 999);
 if($. <= 60)
  {
#  printf "%f, %f, %f\n", $hr, $inp, $outp ;
  unshift (@imv_avg, $inp);
  unshift (@omv_avg, $outp);
  }
 else
  {
  $nil = pop (@imv_avg);
  $nil = pop (@omv_avg);
  unshift (@imv_avg, scalar($inp));
  unshift (@omv_avg, scalar($outp));

 if (($. / 10) == int($. / 10))

  {
   $sumi = $sumo =0;
  for ($i=1;$i < 21;$i++)
   {
   $sumi += scalar ($imv_avg[$i] * (20/$i));
   $sumo += scalar ($omv_avg[$i] * (20/$i));
#   printf "%d", $i;
   }
  $avgi =  $sumi / 60;
  $avgo =  $sumo / 60;
  printf "%f %f %f\n", $hr, $avgi, $avgo;
  }
 }
}
($utime, $stime, $cutime, $cstime) = times();
print (STDERR "run: user = $utime, system = $stime\n");

!
```

mysql insertion bench

```
#!/usr/bin/perl -w

require "ctime.pl";
use DBI;

#use strict;
use vars qw($dbh $hostname $opt_user $opt_password $opt_help $opt_host
            $opt_socket $opt_port $host $version);


$dbh=$host=$opt_user= $opt_password= $opt_help= $opt_host= $opt_socket= "";
$opt_port=0;
if ($opt_host eq '')
{
  $hostname = "w2";
}
else
{
  $hostname = $opt_host;
}
$opt_user='bench';
# ask for a password if no password is set already
if ($opt_password eq '')
{
  system "stty -echo";
  print "Password for user $opt_user to connect to MySQL: ";
  $opt_password = <STDIN>;
  chomp($opt_password);
  system "stty echo";
  print "\n";
}
srand (time());

# make the connection to MySQL
$dbh= DBI->connect("DBI:mysql:testdb:host=$hostname:port=$opt_port:mysql_socket=$
opt_socket",$opt_user,$opt_password, {PrintError => 0}) ||
```

```
    die("Can't make a connection to the mysql server.\n The error: $DBI::errstr");

my $rowdata="Just another row in the table";
$count = 0;
while (1){
    my $r_num = int(rand((1024*1024)));
    my $select = $dbh->prepare ( "
                    SELECT rindex, time, rdate
                    FROM random
                    WHERE rindex = $r_num
                    " );
                $select->execute();
    my @row = $select->fetchrow_array();
    if (undef ($row[1])) {
        $nohit = 0;
    }
    else {
        $nohit = 1;
    }
    #print "@row\n";
    if ( $nohit >= 0 ) {
        $now = time();
        $count++;
        my $insert = $dbh->do ( "INSERT INTO random set rindex=$r_num,time=$now, rdata=\"$rowdata\"" );
        if ($count %100000 == 0) {
            $dattime = &ctime($now);
            print "inserting $count $now ,$dattime record\n";}
    }
    else {
        print ("collided at $r_num $row[1]\n");
    }
}
$dbh->disconnect;
```

Authors: Forrest D. Whitcher, Charles S. Mead, Editor: Suzanne Burns

Copyright © 2003 FW Systems LLC, Lunar Linux.org *All Rights Reserved*

Notes:

data-reduction perl bench 1994 (25k datafiles)

sec/real	cpu Arch.	Results sys Mfgr	cpu Clock	Notes
23/25	(2)R4400	SGI	150/75	
24/44	ALpha	DEC	?	
25/28	PA-RISC	HP	66	
29/30	Power2	IBM	66	(128 bit memory width)
34/36	(2)R4400	SGI	100/50	
70/72	i486	?	66	(cache=256-Interactive)
70/72	(2)MC88100	DGen	?	
70/78	SPARC10	SUN	?	
78/86	i486	?	50	(linux)
82/83	(2)R3000	SGI	40	(cache=256k/1MB)
87/240	(6)SPARC	Solbourne	33	(120 users, heavily loaded)
98/102	MC88100	modcomp	?	
156/169	SPARC2	SUN	33	
160/183	POWER	IBM(320)	22	{64 bit memory width)
na/175	i486	HP	33	(no cache-os/2)
184/187	PA-RISC	HP(847)	?	
223/246	MC68040	HP	?	
290/340	MC68030	HP	?	
344/360	i386	?	25	(Interactive)
na/375	i386	IBM	25	(cache=128-os/2)
464/540	SPARC1	SUN	?	
1100/1200	i386	?	40	(noFP)

perl "hints/linux.sh" edited to compile perl with these optimizations

Appendix, Raw benchmark report results

lmbench 3.0 data

Summary of lmbench results

```
commband_c (bigger is better)
Lunar     559 901 37  430 556 240 176 555 213
RedHat    680 354 168 443 555 238 175 555 213.5
-------------------------------------------------------------------------------
 Host                 OS  Pipe AF      TCP    File   Mmap  Bcopy  Bcopy  Mem    Mem
                              UNIX          reread reread (libc) (hand) read   write
 --------- ------------ ---- ----   ----   ------ ------ ------ ------ ----   -----
                       1.2158 3.9005 4.4596 1.0296 0.9990 0.9929 0.9956 0.9991 1.0007


commlatent_c  (smaller is better)
Lunar    2.2 8.8 27.2 60.0 143
RedHat   1.1 7.2 11.4 35.1 78
-------------------------------------------------------------------
 Host                 OS 2p/0K  Pipe   AF       TCP   TCP
                         ctxsw         UNIX           conn
 --------- ------------ ----- -----   ----   -----  ----- ----- ----- ----
                       0.5147 0.8201 0.4203 0.5854 0.5461

cswitch_c  (smaller is better)
Lunar    2.2 4.4  34.33 14.7 94 28.8 95
RedHat   1.1 3.4  2.55  6.38 97 25.6 97.5
-------------------------------------------------------------------------
 Host                 OS  2p/0K 2p/16K 2p/64K 8p/16K 8p/64K 16p/16K 16p/64K
                          ctxsw  ctxsw  ctxsw ctxsw  ctxsw   ctxsw   ctxsw
 --------- ------------ ------ ------ ------ ------ ------ ------- -------
                       0.5147 0.7818 0.0745 0.4323 1.0351 0.8908 1.0248

processor_c  (smaller is better)
Lunar    864 0.40 0.73 2.5 3.8 30.9 1.0 3.4 303 1002 4971
RedHat   864 0.43 0.71 4.7 6.1 28.3 1.1 3.2 292 1050 4131
-----------------------------------------------------------------------------
 Host                 OS  Mhz   null   null          open   slct   sig    sig    fork   exec   sh
                              call    I/O   stat   clos   TCP    inst   hndl   proc   proc   proc
 --------- ------------ ----   ----   ----   ----   ----   ----   ----   ----   ----   ----   ----
                       1.0000 1.0795 0.9677 1.8585 1.6064 0.9154 1.0583 0.9363 0.9659 1.0481 0.8311

vmlatent_c  (smaller is better)
Lunar    40.1 8.4 119 35 527 0.86 1.8 22.4
RedHat   63.9 21  183 44 864 0.45 2.7 20.3
-------------------------------------------------------------------------------
 Host                 OS   0K File      10K File     Mmap    Prot   Page   100fd
                         Create Delete Create Delete Latency Fault  Fault  selct
 --------- ------------ ------ ------ ------ ------ ------- ----- ------- -----
                       1.5915 2.5432 1.5310 1.2563 1.6401 0.5298 1.5069 0.9056
```

Lunar Linux shows subtantially slower performance than Redhat Linux in some of the network and context-switching latency tests, probably due to kernel/glibc tuning by RedHat. however Lunar offers substantially better performance in an array of other measurements, notably in the VM and filesystem operations are significantly faster when running Lunar.

Lunar Linux Unixbench results

```
  BYTE UNIX Benchmarks (Version 4.1.0)
  System -- Linux w2 2.4.20 #1 SMP Sun Jun 1 13:31:40 EDT 2003 i686 unknown unknown GNU/Linux
  Start Benchmark Run: Tue Jun 10 13:36:07 EDT 2003
   10 interactive users.
   13:36:07 up 33 min, 10 users,  load average: 0.15, 1.28, 3.68
  lrwxrwxrwx    1 root     root            4 May 29 13:41 /bin/sh -> bash
  /bin/sh: symbolic link to 'bash'
  /dev/sdd2              482249    325458    131891  72% /
Dhrystone 2 using register variables     1919747.4 lps   (10.0 secs, 10 samples)
Double-Precision Whetstone                  504.2 MWIPS (10.0 secs, 10 samples)
System Call Overhead                     405435.7 lps   (10.0 secs, 10 samples)
Pipe Throughput                          370154.5 lps   (10.0 secs, 10 samples)
Pipe-based Context Switching             111631.5 lps   (10.0 secs, 10 samples)
Process Creation                           4017.4 lps   (30.0 secs, 3 samples)
Execl Throughput                           1702.7 lps   (29.8 secs, 3 samples)
File Read 1024 bufsize 2000 maxblocks    318357.0 KBps  (30.0 secs, 3 samples)
File Write 1024 bufsize 2000 maxblocks   221466.0 KBps  (30.0 secs, 3 samples)
File Copy 1024 bufsize 2000 maxblocks    119486.0 KBps  (30.0 secs, 3 samples)
File Read 256 bufsize 500 maxblocks      126660.0 KBps  (30.0 secs, 3 samples)
File Write 256 bufsize 500 maxblocks     100533.0 KBps  (30.0 secs, 3 samples)
File Copy 256 bufsize 500 maxblocks       53369.0 KBps  (30.0 secs, 3 samples)
File Read 4096 bufsize 8000 maxblocks    507736.0 KBps  (30.0 secs, 3 samples)
File Write 4096 bufsize 8000 maxblocks   277066.0 KBps  (30.0 secs, 3 samples)
File Copy 4096 bufsize 8000 maxblocks    162786.0 KBps  (30.0 secs, 3 samples)
Shell Scripts (1 concurrent)               1949.7 lpm   (60.0 secs, 3 samples)
Shell Scripts (8 concurrent)                390.0 lpm   (60.0 secs, 3 samples)
Shell Scripts (16 concurrent)               199.0 lpm   (60.0 secs, 3 samples)
Arithmetic Test (type = short)           214153.4 lps   (10.0 secs, 3 samples)
Arithmetic Test (type = int)             217260.6 lps   (10.0 secs, 3 samples)
Arithmetic Test (type = long)            217277.3 lps   (10.0 secs, 3 samples)
```

```
Arithmetic Test (type = float)          229278.5 lps   (10.0 secs, 3 samples)
Arithmetic Test (type = double)         229267.4 lps   (10.0 secs, 3 samples)
Arithoh                                3999483.8 lps   (10.0 secs, 3 samples)
C Compiler Throughput                     485.0 lpm   (60.0 secs, 3 samples)
Dc: sqrt(2) to 99 decimal places        54762.0 lpm   (30.0 secs, 3 samples)
Recursion Test--Tower of Hanoi          31214.4 lps   (20.0 secs, 3 samples)


                    INDEX VALUES
TEST                                        BASELINE     RESULT      INDEX

Dhrystone 2 using register variables        116700.0  1919747.4      164.5
Double-Precision Whetstone                      55.0      504.2       91.7
Execl Throughput                                43.0     1702.7      396.0
File Copy 1024 bufsize 2000 maxblocks         3960.0   119486.0      301.7
File Copy 256 bufsize 500 maxblocks           1655.0    53369.0      322.5
File Copy 4096 bufsize 8000 maxblocks         5800.0   162786.0      280.7
Pipe Throughput                              12440.0   370154.5      297.6
Pipe-based Context Switching                  4000.0   111631.5      279.1
Process Creation                               126.0     4017.4      318.8
Shell Scripts (8 concurrent)                     6.0      390.0      650.0
System Call Overhead                         15000.0   405435.7      270.3
                                                                   =========
     FINAL SCORE                                                       277.7
```

Redhat 9 Unixbench results

```
  BYTE UNIX Benchmarks (Version 4.1.0)
  System -- Linux w2 2.4.20-8smp #1 SMP Thu Mar 13 17:45:54 EST 2003 i686 i686 i386 GNU/Linux
  Start Benchmark Run: Tue Jun 10 11:59:54 EDT 2003
   4 interactive users.
   11:59:54  up  1:54,  4 users,  load average: 0.15, 0.04, 0.27
  lrwxrwxrwx    1 root     root            4 May 21 16:02 /bin/sh -> bash
  /bin/sh: symbolic link to bash
  /dev/sda10             4127076    468504   3448928  12% /home
Dhrystone 2 using register variables     1831148.3 lps   (10.0 secs, 10 samples)
Double-Precision Whetstone                  482.8 MWIPS (10.0 secs, 10 samples)
System Call Overhead                     391557.0 lps   (10.0 secs, 10 samples)
Pipe Throughput                          369903.0 lps   (10.0 secs, 10 samples)
Pipe-based Context Switching             142648.5 lps   (10.0 secs, 10 samples)
Process Creation                           5159.6 lps   (30.0 secs, 3 samples)
Execl Throughput                           1588.4 lps   (29.8 secs, 3 samples)
File Read 1024 bufsize 2000 maxblocks    313947.0 KBps  (30.0 secs, 3 samples)
File Write 1024 bufsize 2000 maxblocks    98309.0 KBps  (30.0 secs, 3 samples)
File Copy 1024 bufsize 2000 maxblocks     68801.0 KBps  (30.0 secs, 3 samples)
File Read 256 bufsize 500 maxblocks      126649.0 KBps  (30.0 secs, 3 samples)
File Write 256 bufsize 500 maxblocks      31522.0 KBps  (30.0 secs, 3 samples)
File Copy 256 bufsize 500 maxblocks       23949.0 KBps  (30.0 secs, 3 samples)
File Read 4096 bufsize 8000 maxblocks    497549.0 KBps  (30.0 secs, 3 samples)
File Write 4096 bufsize 8000 maxblocks   204977.0 KBps  (30.0 secs, 3 samples)
File Copy 4096 bufsize 8000 maxblocks    132571.0 KBps  (30.0 secs, 3 samples)
Shell Scripts (1 concurrent)                345.3 lpm   (60.0 secs, 3 samples)
Shell Scripts (8 concurrent)                 81.0 lpm   (60.0 secs, 3 samples)
Shell Scripts (16 concurrent)                41.0 lpm   (60.0 secs, 3 samples)
Arithmetic Test (type = short)           218855.6 lps   (10.0 secs, 3 samples)
Arithmetic Test (type = int)             225708.2 lps   (10.0 secs, 3 samples)
Arithmetic Test (type = long)            225826.3 lps   (10.0 secs, 3 samples)
Arithmetic Test (type = float)           228865.5 lps   (10.0 secs, 3 samples)
Arithmetic Test (type = double)          228978.5 lps   (10.0 secs, 3 samples)
Arithoh                                 4015630.2 lps   (10.0 secs, 3 samples)
C Compiler Throughput                       443.0 lpm   (60.0 secs, 3 samples)
Dc: sqrt(2) to 99 decimal places          56781.9 lpm   (30.0 secs, 3 samples)
Recursion Test--Tower of Hanoi            32163.7 lps   (20.0 secs, 3 samples)


                    INDEX VALUES
TEST                                        BASELINE     RESULT      INDEX

Dhrystone 2 using register variables        116700.0  1831148.3      156.9
Double-Precision Whetstone                      55.0      482.8       87.8
Execl Throughput                                43.0     1588.4      369.4
File Copy 1024 bufsize 2000 maxblocks         3960.0    68801.0      173.7
File Copy 256 bufsize 500 maxblocks           1655.0    23949.0      144.7
File Copy 4096 bufsize 8000 maxblocks         5800.0   132571.0      228.6
Pipe Throughput                              12440.0   369903.0      297.3

Note: redhat did not complete the Pipe-based
Context Switching test

Process Creation                               126.0     5159.6      409.5
Shell Scripts (8 concurrent)                     6.0       81.0      135.0
System Call Overhead                         15000.0   391557.0      261.0
                                                                   =========
     FINAL SCORE                                                       204.1
```

Perlbench results

```
Lunar) perl-5.008
        path        = /usr/bin/perl
        cc          = cc
        optimize    = -O3 -mcpu=pentium3 -march=pentium3 -mmmx -msse \
        -mfpmath=sse,387
        ccflags     = -D_REENTRANT -D_GNU_SOURCE -fno-strict-aliasing \
        -D_LARGEFILE_SOURCE -D_FILE_OFFSET_BITS=64
        usemymalloc = n

Redhat) perl-5.008
        path        = /mnt/d1/usr/bin/perl
        cc          = gcc
        optimize    = -O2 -march=i386 -mcpu=i686 -g
        ccflags     = -D_REENTRANT -D_GNU_SOURCE -DTHREADS_HAVE_PIDS \
        -DDEBUGGING -fno-strict-aliasing -I/usr/local/include \
        -D_LARGEFILE_SOURCE -D_FILE_OFFSET_BITS=64 -I/usr/include/gdbm
        usemymalloc = n
```

	Lunar	Redhat
arith/mixed	100	79
arith/trig	100	81
array/copy	100	79
array/foreach	100	77
array/index	100	88
array/pop	100	77
array/shift	100	79
array/sort-num	100	83
array/sort	100	89
call/0arg	100	74
call/1arg	100	76
call/2arg	100	79
call/9arg	100	76
call/empty	100	68
call/fib	100	75
call/method	100	74
call/wantarray	100	74
hash/copy	100	82
hash/each	100	78
hash/foreach-sort	100	91
hash/foreach	100	84
hash/get	100	90
hash/set	100	87
loop/for-c	100	87
loop/for-range-const	100	86
loop/for-range	100	88
loop/getline	100	73
loop/while-my	100	75
loop/while	100	83
re/const	100	69
re/w	100	88
startup/fewmod	100	81
startup/lotsofsub	100	83
startup/noprog	100	74
string/base64	100	74
string/htmlparser	100	76
string/index-const	100	77
string/index-var	100	78
string/ipol	100	87
string/tr	100	92
AVERAGE	100	80