



Bisoños Usuarios de GNU/Linux de Mallorca y Alrededores | Bergantells Usuaris de GNU/Linux de Mallorca i Afegitons

Ext2, ReiserFS and XFS Benchmarks (129376 lectures)

Per Ricardo Galli Granada, [gallir](http://mnm.uib.es/gallir/) (<http://mnm.uib.es/gallir/>)

Creado el 22/05/2001 04:46 modificado el 22/05/2001 04:46

First of all, there is no a clear winner, XFS is better in some aspects or cases, ReiserFS in others, and both are better than Ext2 in the sense that they are comparable in performance (again, sometimes faster, sometimes slightly slower) but they a journaling file systems, and you already know what are their advantages... And perhaps the most important moral, is that Linux buffer/cache is really impressive and affected, positively, all the figures of my compilations, copies and random reads and writes. So, I would say, buy memory and go *journalled* ASAP...

After our site was [Slashdotted](#)⁽¹⁾ when [Beowulf](#)⁽²⁾ reported some very simple tests among ReiserFS, XFS, JFS and Ext2, we received comments from Hans Reiser, his collaborators and other readers in Internet. The major criticism was that the report was in Spanish, so here you have new ones in English (rapidly written, without running any spelling or grammar checker, we've got bored after many *mkfs...mount...time time time umount... :-)*.

[Mongo Benchmarks](#)⁽³⁾

The original author of the previous test, Guillem Cantallops, carried up new benchmarks ([XFS vs. ReiserFS vs. JFS vs. Ext2](#)⁽³⁾) but this time using the Mongo script provided by Hans Reiser. The first benchmark was done on a slow CPU machine (Cyrix MII 233MHz, 128MB SDRAM, Samsung SV0844A hard disk) and we noted that, although ReiserFS was the fastest for reading small size files (up to 100 or 1000 bytes), it was extremely slow for bigger ones (the files' average size of a vanilla Red Hat or Debian distribution is about 16 KB).

We weren't very convinced of the validity of the benchmark results due to the old CPU, so Guillem repeated the benchmarks ([XFS, ReiserFS, Ext2](#)⁽³⁾) on a Pentium III, 800 Mhz, 512MB SDRAM and a Seagate ST330621A hard disk. Although the results are relatively comparables to the previous benchmark, they are much closer to each other in the second test.

Linux Kernel Compilation

I also wanted to test the three file systems but compiling the kernel. I run three times the test, which consisted of a *cp*, *make bzImage*, *make clean* and *rm -r*.

According to some comments from Hans Reiser that the use of tails for storing data belonging to other files can lower the performance (they have implemented a new layout policy called **Plan A** that missed the 2.4 code freeze and will be in Reiser4), we tested with the default mount option (using tails) and with *mount -o notail*.

The average of the three execution are as follows:

Command	Kernel Compilation (time in seconds)							
	XFS		ReiserFS (notail)		ReiserFS		Ext2	
	CPU	Real	CPU	Real	CPU	Real	CPU	Real
cp -a /usr/linux /mnt/	8.57	29.40	4.81	5.99	5.49	6.45	2.55	17.90



make bzImage	289.24	292.067	289.33	291.14	289.38	297.27	288.99	293.69
make clean	1.00	1.37	0.50	0.50	0.50	0.50	0.44	0.46
rm -rf /mnt/linux	2.14	11.41	1.06	1.07	1.05	1.05	0.19	0.19

The test was done on the same machine: Pentium III, 800 Mhz, 512MB SDRAM and a Seagate ST330621A. For every run, the partition (7GB) was cleaned (*rm -rf **), unmounted and mounted again. Because the availability of the data in the Linux cache may affect the time measured for *cp -a*, I repeated the command a couple of times before doing the real measurements (there was a huge variance with the first time).

Random lseeks, reads and writes

The previous tests have shown me that the three file systems, besides the very slow copy in XFS, are very close in terms of wall-clock and CPU times. But I wanted to try also what would happen in a "small database real case", so to speak. The difference of the access pattern of a database is that most RDMS do a lot of lseek and read on files of different size (table data and indices) and sporadically the write new values into the DB *pages*. Sometimes, a *fsync* follows immediately after a write and the files may increase their size very slowly when new data is inserted.

So, I've created five files of different sizes: 1, 10, 100, 250 and 500 MBytes (I could try with bigger sizes, but the benchmarks would take hours to complete...):

```
#!/bin/sh
for i in 1 10 100 250 500
do
    echo $i
    dd if=/dev/zero of=/mnt/$i.dat bs=1M count=$i
    chmod ugo+rw /mnt/$i.dat
done
```

Then I run a small program *random.c* (find it at the end of the page) that cycled (100 times) through the five files, and for each cycle, it accessed 100 times to random (*lseek*) locations of the file, *read* a buffer of 16KB and then in the 25% of the cases, it *lseek*'ed back to the previous position, *wrote* the buffer and the forced a *fsync* of the file.

To obtain the times, I first mounted the file system, created the files with the previous script and run three times the program before to get the cache populated as fair as possible. Then I run the *random* program three times and took the average of the three executions:

Read/Write/Fsync Tests in seconds (random.c)			
	XFS	ReiserFS (notail)	Ext2
CPU	1.30	1.55	1.16
Real	62.50	66.91	65.32

Conclusions

I noted disparate results among the different base file size [Mongo benchmarks](#)⁽⁴⁾, so I wanted to see what would happen in a real scenario (at least for us) where the Linux VFS and cache techniques can improve enormously the global performance of the system.

In the case of the kernel compilation, Ext2 has a very low performance for copying files, for the other tests ReiserFS with notail option is the winner but the times are very close to Ext2 and XFS, the difference is less than 2% for *make bzImage*.

In the last case, where I mixed *lseek*, *read*, *write* and *fsync* on files of different sizes, the winner is XFS but for a very small difference, less that 8% compared to ReiserFS.



Analysing all benchmarks, it seems that for the common cases, ReiserFS and XFS have a better performance than Ext2 and with the added value of a *journalled* file system.

What can I say? If you are a home user or own a small server and a relatively fast CPU, use ReiserFS or XFS, both were very stable in our tests and the differences are almost inexistent.

--ricardo

```
random.c

#include <sys/types.h>
#include <sys/stat.h>
#include <fcntl.h>
#include <unistd.h>
#include <stdlib.h>
#include <stdio.h>
#include <string.h>
#include <errno.h>
#define NFILES 5
#define NCYCLES 100
#define NTRIES 100
#define BSIZE 16536
char *filenames[NFILES] = {"1.dat", "10.dat", "100.dat", "250.dat", "500.dat" };
char *dir = "/mnt/";

main()
{
    char buffer[BSIZE];
    int cycle, try, fd, i, fsize, bytes;
    char filename[256];
    struct stat st;
    srand(time(NULL));
    for(cycle=0;cycle<NCYCLES;cycle++) {
        sprintf(filename, "%s%s", dir, filenames[rand() % NFILES]);
        if((fd=open(filename, O_RDWR) < 0) {
            fprintf(stderr, "Couldn't open file %s, stop\n", filename);
            exit(0);
        }
        fstat(fd, &st);
        fsize=st.st_size;
        printf("Trying %s, size: %d\n", filename, fsize);
        for(try=0; try<NTRIES; try++) {
            i = rand() % fsize;
            seek(fd, i, SEEK_SET);
            bytes = read(fd, buffer, BSIZE);
            // printf("read %d bytes\n", bytes);
            if (rand() % 4 == 0) {
                lseek(fd, i, SEEK_SET);
                bytes = write(fd, buffer, BSIZE);
                fsync(fd);
                if (bytes < 0) {
                    fprintf(stderr, "Error %s\n", strerror(errno));
                }
            }
        }
    }
}
```

Lista de enlaces de este artículo:

1. <http://slashdot.org/article.pl?sid=01/05/10/1747213>
2. <http://bulma.net/avanzada.phtml?btnBuscar=Buscar&chkAutor=on&strTexto=Beowulf>



3. <http://bulma.net/body.phtml?nIdNoticia=648>
4. <http://bulma.net/static/mongo/>

E-mail del autor: gallir_ARROBA_uib.es

Podrás encontrar este artículo e información adicional en: <http://bulma.net/body.phtml?nIdNoticia=642>