# Mejorando NFS

Gorka Guardiola Francisco J. Ballesteros

{paurea,nemo}@gsyc.escet.urjc.es



Esta ponencia forma parte del III Congreso Hispalinux[1] virtual, celebrado en Noviembre de 2000. Puedes participar en él en la sección de eventos de BarraPunto[2]

**Resumen**

Realizar una copia de un fichero remoto mediante NFS no es todo lo rápido que debiera. Si utilizamos el interfaz tradicional read/write, el fichero atraviesa dos veces la red y ocupa memoria y recursos del ordenador en el que se origina la copia sin que haya necesidad de ello. Hemos intentado resolver este problema añadiendo una operación de copia para NFS. Sin embargo, nos hemos encontrado que, en algunos casos, esto no es tan efectivo como podría parecer a primera vista.

## 1 Introducción

Tradicionalmente cinco llamadas al sistema han compuesto el interfaz que nos ofrece el kernel para el manejo de ficheros, “read, write, open, close y seek”. Aunque es un conjunto completo que nos da todas las operaciones necesarias para manejar ficheros, en algunos casos es especialmente ineficiente.

Cuando estamos haciendo una copia de un fichero entre dos ordenadores remotos, el fichero debe atravesar dos veces la red, una al hacer read al ordenador local (ocupando innecesariamente memoria y CPU) y otra al hacer write de nuevo al remoto.

Para evitar este desperdicio de recursos y hacer más rápida la copia hemos añadido una nueva operación al interfaz de NFS. Se llama copy y tiene como propósito el de copiar el contenido de un descriptor de fichero a otro de forma remota. En la figura 1 se puede ver un ejemplo de uso de copy en comparación con los tradicionales read/write.

Aunque añadir una nueva operación a NFS debería haber sido fácil, nos encontramos con algunos problemas que nos tuvieron bastante tiempo bloqueados.

El primero de ellos lo produjo la forma en la que está diseñado el servidor de NFS. Este servidor atiende peticiones de una en una, lo que significa que si está bloqueado por haber mandado un copy, no puede atender la escritura que éste genera. A causa de esto se puede producir un interbloqueo del servidor consigo mismo que congele la atención de cualquier otra operación de NFS. Mas adelante detallamos cómo resolvimos este problema.

La segunda cuestión con la que nos encontramos es que, sorprendentemente, si el origen y el destino son distintos y remotos, en algunos casos, realiza más despacio la operación de copia usando copy. Mas adelante explicaremos por qué.

[1] `http://congreso.hispalinux.es`
[2] `http://barrapunto.com/eventos`

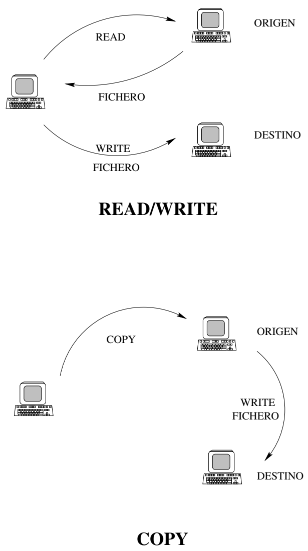


Figura 1: Copia de un fichero con copy y con read/write.

# 2 Introducción básica a NFS

NFS [?] significa “Network File System” y fue creado por SUN para dar acceso remoto transparente a ficheros compartidos a través de redes. En realidad no es más que una forma de dar acceso a través del sistema de ficheros a ficheros remotos. En la terminología de NFS, cuando se da acceso a un árbol de ficheros para que otros ordenadores puedan acceder a él a través de la red, se dice que esta jerarquía se ha *exportado*.

Para su funcionamiento, NFS utiliza un sistema para llamar a funciones a través de la red de forma más o menos transparente llamado RPC, desarrollado por SUN. RPC se basa a su vez en XDR, un sistema de descripción de tipos de datos que permite hacer las RPCs independientes de la arquitectura y plataforma. De esta forma, NFS logra ser independiente de la arquitectura y de la plataforma, cumpliendo así el primero de sus parámetros de diseño.

El segundo objetivo de diseño, que es conseguir que se recupere frente a caídas del servidor, lo logra haciendo éste sin estado y las operaciones idempotentes en la medida de lo posible[3]. Aunque no todas las operaciones pueden ser idempotentes, se intenta que lo sean en la medida de lo posible, por ejemplo, una cache de operaciones recientes. Read y write son idempotentes, así que copy también lo es por haber sido diseñada de forma parecida (enviando la posición u offset del fichero origen y destino) y además, al no ser más que la suma de read y write, dos operaciones idempotentes, a la fuerza tiene que ser idempotente si está bien implementada.

# 3 Cambios que hemos hecho a NFS

Nuestro objetivo, pues, era añadir una operación llamada copy que tuviese iguales resultados que read y write, que cumpliese las restricciones de diseño de NFS y que, a ser posible, fuese compatible hacia atrás con los servidores y clientes sin modificar.

Para ello tuvimos que modificar dos elementos, por un lado el cliente, dentro del kernel, (para lo que utilizamos la versión 2.2.12 del kernel de Linux) para dar acceso a nuestra nueva operación. Por otro lado, tuvimos que modificar también el servidor de NFS para atenderla (para lo que utilizamos la versión 2.2beta37, un servidor de espacio de usuario, pues el de espacio de kernel estaba todavía en estado experimental).

Aunque en un principio pensamos en crear una nueva llamada al sistema para dar acceso a nuestra operación, finalmente nos dimos cuenta de que existe una operación con un interfaz idéntico al que queríamos construir, pero diseñada para trabajar en local. Se llama sendfile y copia de un descriptor de fichero a otro.

[3]Una operación idempotente es aquella en la que la repetición de la operación es exactamente igual a la operación realizada una sola vez.

Fue diseñada con sockets y servidores web en mente, para resolver un problema homomorfo al que nosotros intentábamos resolver. Sirve para evitar cambios de dominio al servir paginas web, copiando de un fichero a un socket sin cambiar de dominio (entre kernel y usuario) ni utilizar memoria innecesariamente.

Tras adoptar la llamada al sistema sendfile, la arquitectura quedó más o menos clara. El punto de entrada en el kernel era la llamada a sendfile, que a su vez llama a una operación del fichero llamada copy si está definida. Esta operación solo esta definida si el fichero es de NFS. La operación de copy llama a la RPC del mismo nombre. El servidor atiende a la petición haciendo read y write de forma local. Si el fichero no es de NFS se copia normalmente, como si se hubiese llamado a sendfile sin que esta tuviese ninguna de nuestras modificaciones.

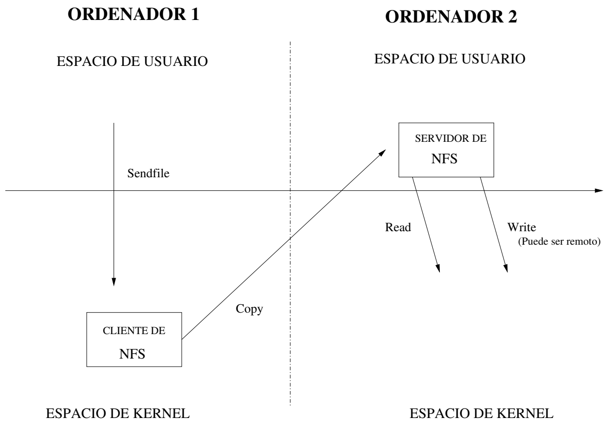


Figura 2: Pasos para una llamada a copy

### 3.1 Compatibilidad hacia atrás

Para poder hacer los nuevos cliente y servidor compatibles hacia atrás, hay dos casos que deben ser resueltos. El caso en el que el elemento modificado es el servidor y en el que es el cliente.

Si hemos modificado el servidor, pero el cliente es un cliente normal de NFS, simplemente nadie llamará a la operación de copy y no habrá ningún problema.

El problema real aparece cuando es el cliente el que ha sido modificado y no existe esa operación en el servidor. En este caso, si no hubiésemos hecho nada, la RPC habría vuelto con un IO error y el cliente se habría quedado reintentando.

Con el fin de resolver este problema, hemos hecho que io error sea sinónimo de operación no implementada. En el caso de un io error, la operación de copy se deshabilita y el cliente deja de usarla, volviéndose automáticamente compatible hacia atrás. Si el error era realmente un io error, el segundo intento fallará y se comportará como el cliente de NFS sin modificar.

Hay un problema con esta aproximación y es que si realmente hay un io error, la operación de copy deja de usarse. La solución a esto sería mantener una cache con el estado del ordenador remoto, mirando si se ha usado o no la operación de copy y si responde a la operación nula (con el fin de si realmente hay un io error) en caso afirmativo.

En el estado actual la operación de copy se deshabilita y habría que volverla habilitar. Esto se puede conseguir haciendo uso de un fichero que creamos en /proc. Aun así, es cierto que se pueden causar problemas en redes con un alto índice de errores, en las que simplemente dejaría de usarse la operación copy.

# 4 Detalles de implementación

Como ya se ha dicho antes, utilizamos un kernel de Linux versión 2.2.12 y un servidor de NFS de espacio de usuario 2.2beta37. No utilizamos el servidor de espacio de kernel porque todavía estaba en estado experimental y tenía todavía algunos errores.

## 4.1 Cambios en el cliente

En el kernel del cliente tuvimos que modificar la llamada al sistema sendfile para que llamase a la operación de copy que añadimos a la estructura de operaciones sobre ficheros. En el caso de un fichero de NFS, este puntero a función apunta a una función llamada `nfs_file_copy`. Esta función, prepara los parámetros, recoge el puntero de fichero y llama a `nfs_file_copy` que se ocupa de construir y llamar a la RPC copy, manejando los posibles errores.

Utilizamos la operación de sendfile porque tenía exactamente el interfaz que necesitábamos. Sin embargo, sendfile fue creada, como ya hemos explicado antes, con servidores web y sockets en mente. Puede suceder, pues, que haya algún detalle de sendfile que no hayamos visto que se encuentre específicamente preparado para funcionar bien en el contexto en el que fue diseñada aunque por ahora, no nos hemos encontrado con ningún problema.

Una parte del kernel que fue especialmente ardua de modificar fueron los encapsulados en XDR de los argumentos. El kernel, con el fin de aumentar la eficiencia de estas traducciones, no utiliza un compilador y hay que hacer estas traducciones a mano. Esta es una tarea difícil, tanto para depurar el código, como por lo fácil que es cometer un error.

Otro cambio interesante que realizamos en el kernel cliente es el añadir soporte para una entrada en el sistema de ficheros de /proc. Esta entrada nos permite cambiar y leer variables en tiempo de ejecución del kernel, pudiendo hacer cosas como habilitar o deshabilitar la llamada a copy. Esta entrada en /proc nos resultó absolutamente fundamental a la hora tanto de depurar como de hacer medidas.

## 4.2 Cambios en el servidor

En el servidor sólo tuvimos que modificar nfsd, el demonio de espacio de usuario que hace de servidor de NFS.

Lo único que tuvimos que hacer es añadir la operación copy al programa RPC del servidor. Como el servidor utilizaba para generar las rutinas de traducción de XDR y los stubs en C el compilador rpcgen, esto fue tarea fácil. El procedimiento que seguimos, fue modificar la descripción del programa en XDR, compilarlo con rpcgen y escribir la implementación de la rutina copy. Esta rutina, lo único que tenía que hacer era llamar a read y write localmente, haciendo uso del sistema de ficheros.

## 4.3 El problema de los bloqueos

La aproximación que hemos descrito hasta ahora, tenía ciertos fallos, que tuvimos que arreglar para que el sistema realmente funcionase. El fallo principal y más difícil de encontrar fueron los interbloqueos que generaba la existencia de un único hilo en el servidor de NFS. Como ya se ha comentado, debido a la existencia de este único hilo, si el write era una petición hecha mediante una RPC al mismo servidor de NFS que estaba atendiendo la de copy, se generaba un bloqueo del hilo de ejecución consigo mismo. Este fallo fue bastante difícil de encontrar porque era un error de diseño y apareció en la fase de depuración.

La solución de este problema, que tampoco es tan complicada una vez se ha conseguido localizar, pasa por el dibujo de todos los diagramas de espera que generan las posibles topologías durante la ejecución de una llamada a copy, como se puede ver en la figura 3.

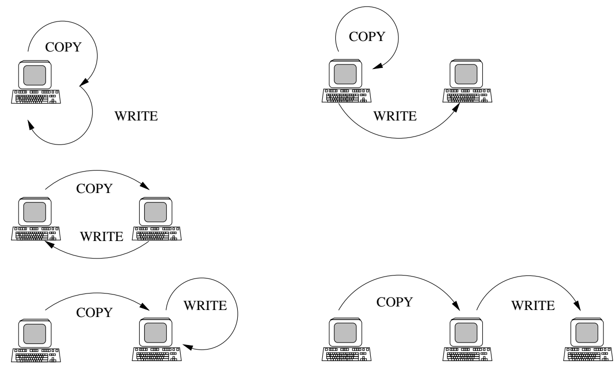


Figura 3: Diagramas de espera para uno dos y tres ordenadores.

Tras ver la figura queda claro que sólo hay dos casos que nos lleven a bloqueo. Ocurre un bloqueo siempre que el origen y el destino estén en la misma máquina. Aunque uno de estos casos (aquel en el que la misma máquina que exporta los ficheros por NFS es la que los monta y es también la que origina el copy) pueda parecer extraño, fue el primero que probamos al hacer el desarrollo en una sola máquina. El segundo (el caso con dos ordenadores y los ficheros origen y destino en el que no es originario del copy) fue el que hizo que no pudiésemos llamar a sendfile en el servidor.

Para resolver ambos problemas hicimos principalmente dos cosas. La primera de ellas fue llamar a la operación de copy solamente en el caso en que ambos ficheros son remotos. En caso contrario, se haría el tradicional read/write, como si se hubiese llamado a sendfile sin modificar.

Lo segundo que hicimos fue llamar a write localmente en lugar de utilizar directamente una RPC. De esta forma, si un ordenador no reexporta un fichero que se ha exportado previamente a sí mismo (un caso completamente anómalo), no puede suceder un bloqueo.

## 4.4 Medidas

Para medir las mejoras en velocidad introducidas por el uso de la nueva operación copy, teníamos que medir cuanto se tardaba en copiar un fichero en los dos casos en que copy funcionaba de forma diferente que sendfile (los dos casos en que ambos ficheros eran remotos). Un factor que había que tener en cuenta también era el uso de la caché (completamente crucial como vimos más adelante).

Para tomar las medidas utilizamos una red a 10Mb/s interconectando ordenadores con procesadores AMD K6/2 a 300MHz.

Para tener en cuenta la importancia de la caché, se hicieron las medidas en unos casos con el fichero en la caché (caché "caliente") y sin el fichero en la caché (caché "fría"). También se hicieron medidas para diferentes tamaños de fichero. Los resultados de las medidas se pueden ver en las figuras 4 a 7

Para el caso de dos ordenadores, es obvio que ganamos más de un orden de magnitud en eficiencia al usar la operación de copia frente a read/write independientemente del estado de la caché. En el caso de tres ordenadores, también obtenemos beneficios si la caché está "fría". Sin embargo, en el caso de caché caliente, contrariamente a la intuición es más lento usar copy que read/write. Esto es debido a la combinación de caché y readahead. A pesar de esto y del alto grado de uso de la caché (tiene un porcentaje de uso del 90% aproximadamente[4] habilitamos el caso de tres ordenadores, porque en media se ganaba en velocidad (aunque fuese poco) y además se reduce el uso de recursos, memoria, red etc.

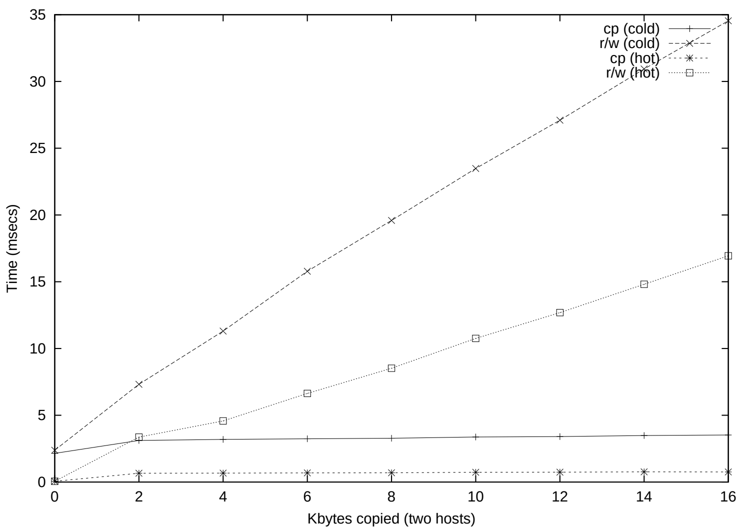


Figura 4: Copia entre dos máquinas usando R/W y Copy—con cache caliente y fria. (Ficheros pequeños)

[4]esto lo medimos con un kernel modificado en un laboratorio

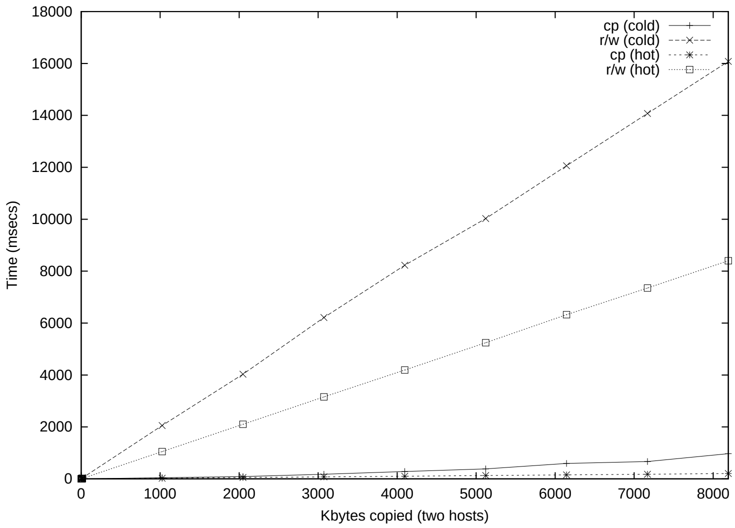


Figura 5: Copia entre dos máquinas usando R/W y Copy—con cache caliente y fria. (Ficheros grandes)

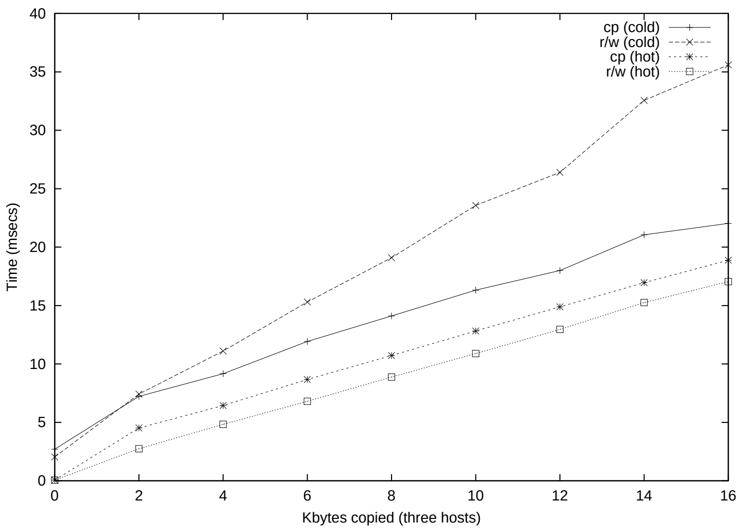


Figura 6: Copia entre tres máquinas con R/W y Copy—cache caliente y fría. (Ficheros pequeños)

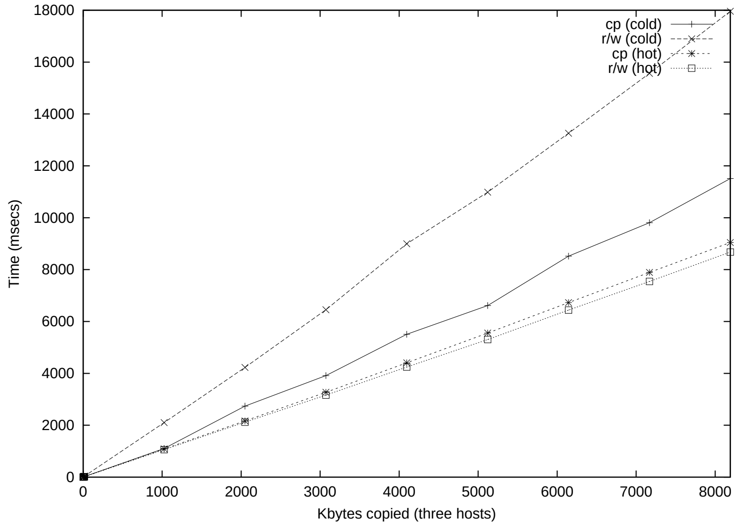


Figura 7: Copia entre tres máquinas con R/W y Copy—cache caliente y fría. (Ficheros grandes)

# 5 Lecciones aprendidas

Al realizar estas modificaciones del kernel de Linux, aprendimos toda una serie de lecciones que hicieron el desarrollo interesante independientemente de los resultados obtenidos.

La primera cosa que nos sorprendió, fue la diferencia entre TCP y UDP. Aunque es algo conocido que UDP es más rápido que TCP, nos dimos cuenta al comenzar la medida que las diferencias eran realmente notables. Finalmente usamos UDP, pues tal y como está construido NFS y las RPCs, no necesitamos de los mecanismos de control de TCP y la diferencia de velocidad es asombrosa.

La segunda cuestión a la que nos enfrentamos al tomar medidas, fue la importancia, en cuanto al retardo que originan, de los cambios de dominio entre espacio de usuario y espacio de kernel. Este retardo, es el que originó la creación de la llamada al sistema sendfile original. En algunos casos (en que hay muchos cambios de dominio) puede llegar a ser comparable al del retardo de una llamada a través de la red.

Nos dimos cuenta de que algo estábamos haciendo mal cuando no teníamos el bucle que tiene la llamada sendfile original, que tiene como fin extender la copia a *count* bytes, (count es el último parámetro de la llamada al sistema). El bucle se encontraba en espacio de usuario y realizábamos muchos cambios de dominio, tantos, que ganábamos mucho menos de lo que esperábamos en el caso a dos.

Una lección que recibimos también fundamental, fue la importancia de las cachés y del readahead. Basta con mirar las medidas para darse cuenta que son realmente fundamentales para el rendimiento, hasta el punto que en el caso de tres ordenadores llegan a hacer read/write, con el fichero atravesando la red dos veces (en teoría), tan rápidos como copy en el que el fichero sólo atraviesa una vez la red.

Nada más terminar la codificación y comenzar las pruebas del sistema, la operación de copy se bloqueaba sin ninguna explicación. Era debido al interbloqueo que se explicó en la sección 4.3. Como ya se ha mencionado antes, este problema fue especialmente largo de resolver, por la dificultad de su localización. En realidad era un problema de diseño que apareció en la fase de depuración. Los interbloqueos son problemas especialmente difíciles de localizar si no se tienen en mente desde un primer momento. Cuando el hilo de ejecución pasa de unas máquinas a otras, hay que tener mucho cuidado.

Otra cuestión interesante ha sido el uso extensivo que hemos hecho de la entrada en /proc. No podríamos haberlo logrado, o al menos no en un tiempo razonable sin el uso de este sistema de acceso a las variables del kernel en tiempo de ejecución. Fue importantísimo para la depuración y absolutamente fundamental a la hora de hacer las medidas.

# 6 Trabajos relacionados

La operación de copia se puede ver como una implementación muy particular de la idea de las llamadas al sistema compuestas, pero a través de la red [**?**]. La idea tras las llamadas compuestas al sistema es muy sencilla. Las llamadas se agrupan de $n$ en $n$ forma que $2n$ cruces de dominio se sustituyen por sólo 2. En un sistema que hace uso de la red, como NFS, una llamada al sistema (como puede ser read), puede incluir un cruce a través de una red, pero la idea es esencialmente la misma. Copy no es más que la agrupación de read y write para disminuir el número de cruces de dominio.

Por otra parte, las cajas son un nuevo interfaz al sistema de ficheros en el que el objeto principal es una caja (debería llamarse sistema de cajas), en lugar de un fichero. La novedad más interesante es que el interfaz sustituye read y write por copy. Para más detalles véase [?].

## 7 Conclusiones

Con la inclusión de la operación de copy hemos conseguido acelerar sensiblemente la copia remota de un fichero de un ordenador a él mismo mediante NFS y disminuir el uso de recursos en general en la copia de ficheros remotos mediante NFS. Además hemos conseguido demostrar que NFS tal y como está construido es ineficiente en algunos casos y que debería ser revisado.

## A El protocolo NFScp

Descripción en XDR de los argumentos y resultados de copy.

```
struct copyargs {
        nfs_fh filedest;
        nfs_fh filesrc;
        u_int offsetdest;
        u_int offsetsrc;
        u_int count;
        host hostname;
};

typedef struct copyargs copyargs;

struct copyokres {
        fattr attributesdest;
        fattr attributessrc;
        u_int count;
        struct {
                u_int data_len;
                char *data_val;
        } data;
};


typedef struct copyokres copyokres;

struct copyres {
        nfsstat status;
        union {
                copyokres reply;
        } copyres_u;
};

typedef struct copyres copyres;
```

Descripción del programa de NFS con la nueva operación de copia.

```
 program NFS_PROGRAM {
        version NFS_VERSION {
                void
                NFSPROC_NULL(void) = 0;

                attrstat
                NFSPROC_GETATTR(nfs_fh) =       1;
```

```
            attrstat
            NFSPROC_SETATTR(sattrargs) = 2;

            void
            NFSPROC_ROOT(void) = 3;

            diropres
            NFSPROC_LOOKUP(diropargs) = 4;

            readlinkres
            NFSPROC_READLINK(nfs_fh) = 5;

            readres
            NFSPROC_READ(readargs) = 6;

            void
            NFSPROC_WRITECACHE(void) = 7;

            attrstat
            NFSPROC_WRITE(writeargs) = 8;

            diropres
            NFSPROC_CREATE(createargs) = 9;

            nfsstat
            NFSPROC_REMOVE(diropargs) = 10;

            nfsstat
            NFSPROC_RENAME(renameargs) = 11;

            nfsstat
            NFSPROC_LINK(linkargs) = 12;

            nfsstat
            NFSPROC_SYMLINK(symlinkargs) = 13;

            diropres
            NFSPROC_MKDIR(createargs) = 14;

            nfsstat
            NFSPROC_RMDIR(diropargs) = 15;

            readdirres
            NFSPROC_READDIR(readdirargs) = 16;


            statfsres
            NFSPROC_STATFS(nfs_fh) = 17;

            copyres
            NFSPROC_COPY(copyargs) = 18;
        } = 2;
 } = 100003;
```

# B Cambios al cliente

A continuación se muestra la rutina de codificación y decodificación de los argumentos de copy para traducirlos a XDR y de vuelta (no usamos ninguna descripción en XDR de copy, puesto que en el kernel no se usa compilador de XDR, sino que se escriben las rutinas a mano).

```
nfs_xdr_copyargs(struct rpc_rqst *req, u32 *p, struct nfs_copyargs *args)
{
```

```
        p = xdr_encode_fhandle(p, args->fhdest);
        p = xdr_encode_fhandle(p, args->fhsrc);
        *p++ = htonl(args->offsetdest);
        *p++ = htonl(args->offsetsrc);
        *p++ = htonl(args->count);
        p = xdr_encode_string(p, args->host);
        req->rq_slen = xdr_adjust_iovec(req->rq_svec, p);

        return 0;
}

nfs_xdr_copyres(struct rpc_rqst *req, u32 *p, struct nfs_copyres *res)
{
        int status,count;
        if ((status = ntohl(*p++)))
                return -nfs_stat_to_errno(status);
        p = xdr_decode_fattr(p, res->fattrdest);
        p = xdr_decode_fattr(p, res->fattrsrc);
        res->count = count = ntohl(*p++);

        return count;
}
```

Aquí se puede observar la parte de `sys_sendfile` que se ocupa de decidir si se usa copy o no.

```
/* Caso remoto en el que se llama a copy*/

                if (nfs_copy_enabled &&
                    is_remote(in_file) && is_remote(out_file)) {
                        size_t (*cpf)(struct file *,struct file*,
                                      size_t,loff_t*,loff_t*);
                        if (!(cpf=in_file->f_op->copy))
                                panic("cp: no copy for remote file");
                        retval = (*cpf)(out_file,in_file,count,dppos,ppos);
                }

/* Caso normal de sendfile si no hemos usado copy*/
                if (!nfs_copy_enabled || retval<0 ||
                    !is_remote(in_file)  || !is_remote(out_file)){
                        read_descriptor_t desc;
                        printk("local sendfile call\n");
                        desc.written = 0;
                        desc.count = count;
                        desc.buf = (char *) out_file;
                        desc.error = 0;
                        do_generic_file_read(in_file,ppos, &desc,
                                             file_send_actor);
                        retval = desc.written;
                        if (!retval)
                                retval = desc.error;
                }
```

# C Cambios en el servidor

Función de atención a la petición de copy.

```
/*
 * max # of read+writes to issue.
 * 1 would be the traditional NFS value.
 * However, latency can be much better if a whole file copy be
 * copied within a single request.
 */
```

```
#define MAX_WORK 8


int nfsd_nfsproc_copy_2(copyargs *argp, struct svc_req  *rqstp)
{
        int      sfd=-1,dfd=-1;
        nfsstat  status=NFS_OK;
        int      limit;
        int      islcl;
        fhcache *sfhc,*dfhc;
        static   char lclhost[NFS_MAXHOSTLEN+1];
        static   int  lclknown=0;
        u_int    nleft;
        struct writeargs warg;
        char     *host=argp->hostname;
        CLIENT   *cp=NULL;

        copyokres *res = &result.copyres.copyres_u.reply;

        if (!lclknown){
                if (gethostname(lclhost,NFS_MAXHOSTLEN)) {
                        Dprintf(L_ERROR,"gethostname failed");
                        return -1;
                }
                lclknown=1;
        }

        islcl=(!strcmp(host,"localhost") || !strcmp(host,lclhost));
        if (islcl)
                Dprintf(D_CALL,"copy to %s (local)",host);
        else
                Dprintf(D_CALL,"copy to %s (remote)",host);

        sfd=getfd(&argp->filesrc,rqstp,argp->offsetsrc,&status,O_RDONLY,&sfhc);
        if (islcl)
            dfd=getfd(&argp->filedest,rqstp,argp->offsetdest,&status,O_WRONLY,
                      &dfhc);
        else {
                warg.file= argp->filedest;
                warg.offset=argp->offsetdest;
                warg.beginoffset=argp->offsetdest;
                warg.data.data_val=iobuf;
                if(!(cp=clnt_create(host,NFS_PROGRAM,NFS_VERSION,"udp"))){
                        Dprintf(L_ERROR,"unable to create client");
                        clnt_pcreateerror(host);
                        status =-1;
                        goto failure;
                }

        }
        if (sfd<0 || (islcl && dfd<0)){
                Dprintf(L_ERROR,"unable to open either src or dst");
                goto failure;
        }

        /* Ok, got descriptors and/or the client connection: do the job */

        nleft=argp->count;
        res->count=0;
        if (argp->offsetsrc == 0 && log_transfers)
                nfsd_xferlog(rqstp, "<", sfhc->path);

        for (limit=MAX_WORK ; limit>0 && nleft>0; limit--) {
```

```
		u_int nbytes = nleft;
		ssize_t   rdres,wrres;
		attrstat *rpcres;


		errno=0;
		if (nbytes > NFS_MAXDATA)
			nbytes=NFS_MAXDATA;
		rdres= read(sfd,iobuf,nbytes);
		if (rdres <0)
			Dprintf(L_ERROR,"read failed");
		else if (!rdres || errno){
			status = NFS_OK;
			if (errno)
				status=nfs_errno();
			break;
		} else
			res->count += rdres;

		if((status=fhc_getattr(sfhc, &res->attributessrc,NULL,rqstp))){
			Dprintf(L_ERROR,"src fh status is not ok");
			goto failure;
		}
		if (islcl) {
			if ((wrres=write(dfd,iobuf,rdres))!=rdres) {
				Dprintf(L_ERROR,"write failed (%d)",errno);
				goto failure;
			}
			if((status=fhc_getattr(dfhc,
					  &res->attributesdest,NULL,rqstp))){
				Dprintf(L_ERROR,"dst fh status is not ok");
				goto failure;
			}
		} else {
			warg.totalcount=warg.data.data_len=rdres;
			if (!(rpcres=nfsproc_write_2(&warg,cp)) ||
			    rpcres->status != NFS_OK){
				clnt_perror(cp, "write failed");
				goto failure;
			}
			memcpy(&res->attributesdest,
			       &rpcres->attrstat_u.attributes,
			       sizeof(res->attributesdest));
			warg.offset += rdres;
			warg.beginoffset += rdres;
		}
		nleft -= rdres;
	}
 failure:
	if (sfd>=0)
		fd_inactive(sfd);
	if (dfd>=0)
		fd_inactive(dfd);
	if (cp)
		clnt_destroy(cp);
	return status;
}
```

# D GNU Free Documentation License

Version 1.1, March 2000

## Preamble

The purpose of this License is to make a manual, textbook, or other written document "free" in the sense of freedom: to assure everyone the effective freedom to copy and redistribute it, with or without modifying it, either commercially or noncommercially. Secondarily, this License preserves for the author and publisher a way to get credit for their work, while not being considered responsible for modifications made by others.

This License is a kind of "copyleft", which means that derivative works of the document must themselves be free in the same sense. It complements the GNU General Public License, which is a copyleft license designed for free software.

We have designed this License in order to use it for manuals for free software, because free software needs free documentation: a free program should come with manuals providing the same freedoms that the software does. But this License is not limited to software manuals; it can be used for any textual work, regardless of subject matter or whether it is published as a printed book. We recommend this License principally for works whose purpose is instruction or reference.

## D.1 Applicability and Definitions

This License applies to any manual or other work that contains a notice placed by the copyright holder saying it can be distributed under the terms of this License. The "Document", below, refers to any such manual or work. Any member of the public is a licensee, and is addressed as "you".

A "Modified Version" of the Document means any work containing the Document or a portion of it, either copied verbatim, or with modifications and/or translated into another language.

A "Secondary Section" is a named appendix or a front-matter section of the Document that deals exclusively with the relationship of the publishers or authors of the Document to the Document's overall subject (or to related matters) and contains nothing that could fall directly within that overall subject. (For example, if the Document is in part a textbook of mathematics, a Secondary Section may not explain any mathematics.) The relationship could be a matter of historical connection with the subject or with related matters, or of legal, commercial, philosophical, ethical or political position regarding them.

The "Invariant Sections" are certain Secondary Sections whose titles are designated, as being those of Invariant Sections, in the notice that says that the Document is released under this License.

The "Cover Texts" are certain short passages of text that are listed, as Front-Cover Texts or Back-Cover Texts, in the notice that says that the Document is released under this License.

A "Transparent" copy of the Document means a machine-readable copy, represented in a format whose specification is available to the general public, whose contents can be viewed and edited directly and straightforwardly with generic text editors or (for images composed of pixels) generic paint programs or (for drawings) some widely available drawing editor, and that is suitable for input to text formatters or for automatic translation to a variety of formats suitable for input to text formatters. A copy made in an otherwise Transparent file format whose markup has been designed to thwart or discourage subsequent modification by readers is not Transparent. A copy that is not "Transparent" is called "Opaque".

Examples of suitable formats for Transparent copies include plain ASCII without markup, Texinfo input format, LaTeX input format, SGML or XML using a publicly available DTD, and standard-conforming simple HTML designed for human modification. Opaque formats include PostScript, PDF, proprietary formats that can be read and edited only by proprietary word processors, SGML or XML for which the DTD and/or processing tools are not generally available, and the machine-generated HTML produced by some word processors for output purposes only.

The "Title Page" means, for a printed book, the title page itself, plus such following pages as are needed to hold, legibly, the material this License requires to appear in the title page. For works in formats which do not have any title page as such, "Title Page" means the text near the most prominent appearance of the work's title, preceding the beginning of the body of the text.

## D.2 Verbatim Copying

You may copy and distribute the Document in any medium, either commercially or noncommercially, provided that this License, the copyright notices, and the license notice saying this License applies to the Document are reproduced in all copies, and that you add no other conditions whatsoever to those of this License. You may not use technical measures to obstruct or control the reading or further copying of the copies you make or distribute. However, you may accept compensation in exchange for copies. If you distribute a large enough number of copies you must also follow the conditions in section 3.

You may also lend copies, under the same conditions stated above, and you may publicly display copies.

## D.3 Copying in Quantity

If you publish printed copies of the Document numbering more than 100, and the Document's license notice requires Cover Texts, you must enclose the copies in covers that carry, clearly and legibly, all these Cover Texts: Front-Cover Texts on the front cover, and Back-Cover Texts on the back cover. Both covers must also clearly and legibly identify you as the publisher of these copies. The front cover must present the full title with all words of the title equally prominent and visible. You may add other material on the covers in addition. Copying with changes limited to the covers, as long as they preserve the title of the Document and satisfy these conditions, can be treated as verbatim copying in other respects.

If the required texts for either cover are too voluminous to fit legibly, you should put the first ones listed (as many as fit reasonably) on the actual cover, and continue the rest onto adjacent pages.

If you publish or distribute Opaque copies of the Document numbering more than 100, you must either include a machine-readable Transparent copy along with each Opaque copy, or state in or with each Opaque copy a publicly-accessible computer-network location containing a complete Transparent copy of the Document, free of added material, which the general network-using public has access to download anonymously at no charge using public-standard network protocols. If you use the latter option, you must take reasonably prudent steps, when you begin distribution of Opaque copies in quantity, to ensure that this Transparent copy will remain thus accessible at the stated location until at least one year after the last time you distribute an Opaque copy (directly or through your agents or retailers) of that edition to the public.

It is requested, but not required, that you contact the authors of the Document well before redistributing any large number of copies, to give them a chance to provide you with an updated version of the Document.

## D.4 Modifications

You may copy and distribute a Modified Version of the Document under the conditions of sections 2 and 3 above, provided that you release the Modified Version under precisely this License, with the Modified Version filling the role of the Document, thus licensing distribution and modification of the Modified Version to whoever possesses a copy of it. In addition, you must do these things in the Modified Version:

- Use in the Title Page (and on the covers, if any) a title distinct from that of the Document, and from those of previous versions (which should, if there were any, be listed in the History section of the Document). You may use the same title as a previous version if the original publisher of that version gives permission.
- List on the Title Page, as authors, one or more persons or entities responsible for authorship of the modifications in the Modified Version, together with at least five of the principal authors of the Document (all of its principal authors, if it has less than five).
- State on the Title page the name of the publisher of the Modified Version, as the publisher.
- Preserve all the copyright notices of the Document.
- Add an appropriate copyright notice for your modifications adjacent to the other copyright notices.
- Include, immediately after the copyright notices, a license notice giving the public permission to use the Modified Version under the terms of this License, in the form shown in the Addendum below.
- Preserve in that license notice the full lists of Invariant Sections and required Cover Texts given in the Document's license notice.
- Include an unaltered copy of this License.
- Preserve the section entitled "History", and its title, and add to it an item stating at least the title, year, new authors, and publisher of the Modified Version as given on the Title Page. If there is no section entitled "History" in the Document, create one stating the title, year, authors, and publisher of the Document as given on its Title Page, then add an item describing the Modified Version as stated in the previous sentence.
- Preserve the network location, if any, given in the Document for public access to a Transparent copy of the Document, and likewise the network locations given in the Document for previous versions it was based on. These may be placed in the "History" section. You may omit a network location for a work that was published at least four years before the Document itself, or if the original publisher of the version it refers to gives permission.
- In any section entitled "Acknowledgements" or "Dedications", preserve the section's title, and preserve in the section all the substance and tone of each of the contributor acknowledgements and/or dedications given therein.
- Preserve all the Invariant Sections of the Document, unaltered in their text and in their titles. Section numbers or the equivalent are not considered part of the section titles.
- Delete any section entitled "Endorsements". Such a section may not be included in the Modified Version.
- Do not retitle any existing section as "Endorsements" or to conflict in title with any Invariant Section.

If the Modified Version includes new front-matter sections or appendices that qualify as Secondary Sections and contain no material copied from the Document, you may at your option designate some or all of these sections as invariant. To do this, add their titles to the list of Invariant Sections in the Modified Version's license notice. These titles must be distinct from any other section titles.

You may add a section entitled "Endorsements", provided it contains nothing but endorsements of your Modified Version by various parties – for example, statements of peer review or that the text has been approved by an organization as the authoritative definition of a standard.

You may add a passage of up to five words as a Front-Cover Text, and a passage of up to 25 words as a Back-Cover Text, to the end of the list of Cover Texts in the Modified Version. Only one passage of Front-Cover Text and one of Back-Cover Text may be added by (or through arrangements made by) any one entity. If the Document already includes a cover text for the same cover, previously added by you or by arrangement made by the same entity you are acting on behalf of, you may not add another; but you may replace the old one, on explicit permission from the previous publisher that added the old one.

The author(s) and publisher(s) of the Document do not by this License give permission to use their names for publicity for or to assert or imply endorsement of any Modified Version.

## D.5 Combining Documents

You may combine the Document with other documents released under this License, under the terms defined in section 4 above for modified versions, provided that you include in the combination all of the Invariant Sections of all of the original documents, unmodified, and list them all as Invariant Sections of your combined work in its license notice.

The combined work need only contain one copy of this License, and multiple identical Invariant Sections may be replaced with a single copy. If there are multiple Invariant Sections with the same name but different contents, make the title of each such section unique by adding at the end of it, in parentheses, the name of the original author or publisher of that section if known, or else a unique number. Make the same adjustment to the section titles in the list of Invariant Sections in the license notice of the combined work.

In the combination, you must combine any sections entitled "History" in the various original documents, forming one section entitled "History"; likewise combine any sections entitled "Acknowledgements", and any sections entitled "Dedications". You must delete all sections entitled "Endorsements."

## D.6 Collections of Documents

You may make a collection consisting of the Document and other documents released under this License, and replace the individual copies of this License in the various documents with a single copy that is included in the collection, provided that you follow the rules of this License for verbatim copying of each of the documents in all other respects.

You may extract a single document from such a collection, and distribute it individually under this License, provided you insert a copy of this License into the extracted document, and follow this License in all other respects regarding verbatim copying of that document.

## D.7 Aggregation With Independent Works

A compilation of the Document or its derivatives with other separate and independent documents or works, in or on a volume of a storage or distribution medium, does not as a whole count as a Modified Version of the Document, provided no compilation copyright is claimed for the compilation. Such a compilation is called an "aggregate", and this License does not apply to the other self-contained works thus compiled with the Document, on account of their being thus compiled, if they are not themselves derivative works of the Document.

If the Cover Text requirement of section 3 is applicable to these copies of the Document, then if the Document is less than one quarter of the entire aggregate, the Document's Cover Texts may be placed on covers that surround only the Document within the aggregate. Otherwise they must appear on covers around the whole aggregate.

## D.8 Translation

Translation is considered a kind of modification, so you may distribute translations of the Document under the terms of section 4. Replacing Invariant Sections with translations requires special permission from their copyright holders, but you may include translations of some or all Invariant Sections in addition to the original versions of these Invariant Sections. You may include a translation of this License provided that you also include the original English version of this License. In case of a disagreement between the translation and the original English version of this License, the original English version will prevail.

## D.9 Termination

You may not copy, modify, sublicense, or distribute the Document except as expressly provided for under this License. Any other attempt to copy, modify, sublicense or distribute the Document is void, and will automatically terminate your rights under this License. However, parties who have received copies, or rights, from you under this License will not have their licenses terminated so long as such parties remain in full compliance.

## D.10 Future Revesions of This Licence

The Free Software Foundation may publish new, revised versions of the GNU Free Documentation License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns. See http:///www.gnu.org/copyleft/.

Each version of the License is given a distinguishing version number. If the Document specifies that a particular numbered version of this License ºr any later version" applies to it, you have the option of following the terms and conditions either of that specified version or of any later version that has been published (not as a draft) by the Free Software Foundation. If the Document does not specify a version number of this License, you may choose any version ever published (not as a draft) by the Free Software Foundation.

## ADDENDUM: How to use this License for your documents

To use this License in a document you have written, include a copy of the License in the document and put the following copyright and license notices just after the title page:

> Copyright © YEAR YOUR NAME. Permission is granted to copy, distribute and/or modify this document under the terms of the GNU Free Documentation License, Version 1.1 or any later version published by the Free Software Foundation; with the Invariant Sections being LIST THEIR TITLES, with the Front-Cover Texts being LIST, and with the Back-Cover Texts being LIST. A copy of the license is included in the section entitled "GNU Free Documentation License".

If you have no Invariant Sections, write "with no Invariant Sections" instead of saying which ones are invariant. If you have no Front-Cover Texts, write "no Front-Cover Texts" instead of "Front-Cover Texts being LIST"; likewise for Back-Cover Texts.

If your document contains nontrivial examples of program code, we recommend releasing these examples in parallel under your choice of free software license, such as the GNU General Public License, to permit their use in free software.