*Estrategia*

Comienza le ejecución del programa esperando la conexión de uno o más PPDs. Una vez conectado el PPD se habilita la conexión de los PFSs lo cual el PRAID entra en funcionamiento.

Cuando se conecta más de un disco la sincronización se lleva a cabo según otro disco para agilizar la sincronización (no se hacen los pedidos a todos los discos activos, sino que solo a uno). La elección del disco en el cual se realizara la sincronización es según el disco que tiene menor cantidad de pedidos en ejecución.

Los pedidos de lectura realizados por los PFS serán distribuidos entre los discos activos según la función **uno\_y\_uno** que es similar a Round Robín.

*Estructuras*

typedef struct \_sectores\_t{

uint32\_t sector;

struct \_sectores\_t \*sgte;

} \_\_attribute\_\_ ((packed)) **sectores\_t**;

La utilizamos en una lista para llevar el control de los sectores que ya fueron sincronizados. Una vez que va sincronizan se va liberando.

También la utilizamos para tener una lista de los sockets de los discos caídos en ejecución, que si los liberábamos y levantábamos rápidamente comenzaba a recibir pedidos que eran para el disco anterior.

struct **mensaje\_cola**{

uint32\_t type;

uint32\_t sector;

}\_\_attribute\_\_ ((packed));

Estructura que se creó al comienzo, pero luego fue dejada sin uso.

typedef struct \_pedido{

nipc\_socket sock;

uint8\_t type;

uint32\_t sector;

struct \_pedido \*sgte;

} \_\_attribute\_\_ ((packed)) **pedido**;

Estructura para llevar el control de que pedidos se hicieron a cada disco. Para saber a que PFS se debe devolver el sector solicitado.

typedef struct \_disco{

uint8\_t id[20];

nipc\_socket sock;

pthread\_t hilo;

int32\_t pedido\_sincro;

int32\_t sector\_sincro;

sectores\_t \*ya\_sincro\_start;

sectores\_t \*ya\_sincro\_end;

uint32\_t cantidad\_pedidos;

pedido \*pedidos\_start;

pedido \*pedidos\_end;

int32\_t encolados;

struct \_disco \*sgte;

} \_\_attribute\_\_ ((packed)) **disco**;

Estructura disco, contiene todos los datos necesarios para llevar el control del disco. Identificador del disco, socket asignado, hilo asignado, numero de sector sincronizado, lista de pedidos ya sincronizados, cantidad de pedidos en ejecución y lista de pedidos.

typedef struct \_lista\_pfs{

nipc\_socket sock;

sem\_t semaforo;

sectores\_t \*escrituras;

sectores\_t \*lecturas;

struct \_lista\_pfs \*sgte;

} \_\_attribute\_\_ ((packed)) **lista\_pfs**;

Estructura del PFS para controlar los pedidos realizados por cada PFS y el socket asignado.

typedef struct datos{

nipc\_socket sock\_control;

lista\_pfs \*pfs\_activos;

disco \*discos;

uint32\_t max\_sector;

sem\_t semaforo;

sectores\_t \*sock\_x\_liberar;

}**datos**;

Estructura principal del sistema, lista de disco, lista de PFS, cantidad máxima de sectores, lista de socket a liberar al finalizar el programa.

typedef struct config\_t{

uint8\_t server\_host[1024];

uint16\_t server\_port;

uint16\_t console;

char log\_path[1024];

log\_output log\_mode;

} \_\_attribute\_\_ ((packed)) **config\_t**;

Estructura del archivo de configuración.

*Funciones*

void **agregar\_disco**(datos \*\*info\_ppal, uint8\_t \*id\_disco,nipc\_socket sock\_ppd);

Agrega un disco nuevo a la lista y crea un hilo para escuchar las respuestas del PPD.

void **listar\_pedidos\_discos**(disco \*discos);

Función de control para ver los pedidos asignados a cada disco.

void **listar\_discos**(disco \*discos);

Función de control para ver los discos activos en el sistema.

nipc\_socket **menor\_cantidad\_pedidos**(disco \*discos, int32\_t sector);

Devuelve el socket del disco con menor cantidad de pedidos en ejecución y que tiene el sector solicitado sincronizado.

nipc\_socket **uno\_y\_uno**(datos \*\*info\_ppal, int32\_t sector);

Devuelve el socket del disco siguiente en la lista de disco y que tiene el sector solicitado sincronizado. (Round Robín)

uint8\_t \***distribuir\_pedido\_lectura**(datos \*\*info\_ppal,nipc\_packet mensaje,nipc\_socket sock\_pfs);

Crea el pedido según el mensaje y ejecuta una de las funciones anteriormente descriptas para distribuir el pedido entre los discos.

void **distribuir\_pedido\_escritura**(datos \*\*info\_ppal,nipc\_packet mensaje,nipc\_socket sock\_pfs);

Crea el pedido según el mensaje y se los envía a todos los discos.

void \***espera\_respuestas**(datos \*\*info\_ppal);

Función del hilo que se crea al registrar un nuevo PPD. Se encarga de escuchar todos los mensajes enviados por ese PPD y responder al PFS indicado o registrar su sincronización.

void **liberar\_pfs\_caido**(datos \*\*info\_ppal,nipc\_socket sock\_pfs);

Al perder la conexión con un PFS se lleva a cabo esta función que libera los sockets y libera todos los recursos consumidos en memoria.

uint16\_t **limpio\_discos\_caidos**(datos \*\*info\_ppal,nipc\_socket sock\_ppd);

Al perder la conexión con un PPD se lleva a cabo esta función para registrar los socket en la estructura principal y libera todos los recursos consumidos en memoria.

void **insertar**(sectores\_t \*\*lista, uint32\_t sector);

Función utilizada para insertar los pedidos ya sincronizados, para poder recorrerlos más rápidamente.

int32\_t **config\_read**(config\_t \*config);

Lee el archivo de configuración