Published: 03/27/68

Obsolete kept only because abstract so brief

Identification

Device Free-Storage Management
J. D. Van Hausen

Purpose

The file system free-storage manager (hereafter called simply free storage) is that part of the DIM which keeps track of the unused storage on all on-line devices. Hyper-records on the devices are withdrawn from or released into the pool of unused storage by calling free storage. Depending upon the entry point used, free storage will either return to the caller an unused hyper-record address or place the address handed it back into its lists.

Another duty of free storage is to complain when the level of unused storage drops below a certain threshold percentage for that device.

Introduction

In the description which follows it is assumed that the reader is familiar with Section BG.10.00, since the basic DIM concepts are presented there. This section first describes the structure of the data bases used by free storage and then the use and internal mechanisms of the free storage procedure.

The free list

The free pool is the set of unused hyper-records on a device. Free storage keeps track of the free pool by keeping a list of addresses; this list is called the free list. There is a separate free list for each device. Only a portion of the free list is in core at any time, in order to be able to satisfy a request without performing any device I/O. Residing in core at any time is some useful portion of the free list for each device. The bulk of the free list resides on the devices themselves, with each entry residing on the device which it describes. This scheme is intended to minimize the core required for the free list without degrading performance.

Since the free pool is a collection of hyper-records arranged randomly over the device, the free pool must be divided into pieces small enough to fit into a hyper-record. These small pieces are called free maps. The size of a hyper-record on a device is a system parameter and most likely will be different on different devices at any one time. Thus rather than change the size of the free maps, they are small enough to fit in the minimum hyper-record, 128 36-bit words. As will be seen later, the free maps are stored in less than 1 per cent of the hyper-records in the free pool, and so the fact that the hyper-records containing free maps are inefficiently used is unimportant.

The address of a hyper-record containing a free map appears in only one place in the free list: in the free map which it contains. Thus there is no danger of handing out a free map hyper-record until its contents have been read into core.

Free maps are linked together into free chains. There may be several free chains in the free list for one device. The free maps which are in core are the ones at both ends of each chain. The map in core at one end of each chain is called the deposition buffer and may only be written onto the device. On the other end is the withdrawal buffer; this buffer may only be read from the device.

There are two reasons for having more than one chain. Since I/O is asynchronous, when I/O is initiated for a buffer, it should not be altered until the "all clear" signal is received from device control. This means that with one chain either no release request or no withdrawal request could be processed while I/O is in progress. With several chains, one may process the request using another chain. The other reason is to permit free storage to be more tolerant of non-recoverable I/O errors. With only one chain, an error would cause the complete collapse of free storage. Having N chains permits free storage to withstand N-1 errors by ignoring those chains which have had errors.

The master sector

The master sector contains data necessary to manipulate the free list for a device. It also contains sufficient information to permit a fairly complete recovery following a catastrophic system shutdown. A copy of the master sector is kept on the device to permit this recovery capability. In core the master sector describes the status of the free list for its associated device.

When the system is shut down normally, the buffers are written out onto the device and the master sector in core is written over the copy on the device. Starting up the free storage procedure is just a matter of reading the master sector which contains the address of the beginning and end of each free chain and then reading in the required portion of each chain. When the system "crashes", the master sector on the device contains addresses of free map hyper-records in each chain; by chasing through each chain, the ends can be found.

The freepool segment

The master sector and all buffers for a device are contained in a wired-down segment whose name is freepool-NN, where NN is the device identification index. The first block of 128 thirty-six bit words contains the master sector, the next M blocks contain the deposition buffers for the M chains, and the next M blocks contain the withdrawal buffers.

The free storage procedure

There are two entry points into free storage; one for withdrawing a hyper-record address and one for releasing a hyper-record address which is no longer needed.

Usage

Free storage as the term is used in this section is really two separate procedures combined into one segment. One procedure is called withdraw; the other release. The rocedures were combined to conserve core space, since free storage must be wired down.

Thus there are two entries to free storage. The corresponding call statements are (1) call free-store $ withdraw (did, add, errcode) and (2) call free-store $ release (did, add, errcode). The declaration of the arguments is as follows:

```
     dcl (did,                    /* device id */

     dcl  add,                    /* address */

          errcode)                /* error code */

          fixed bin (35);
```

If errcode is zero the request has been satisfied; if it is nonzero the request has not been satisfied for any of the following reasons: the device is inoperative, the free pool is empty or there was an error on the one remaining free chain. With each call only one address is withdrawn from or replaced into the list.

The free storage procedure

When a buffer has had I/O initiated on it, free storage must find another chain to use. This it does by using the "nxchain" array in the master sector. The i-th element of "nxchain" gives the chain to be used following the i-th chain. When an error occurs on a chain, the "nxchain" element which pointed to the chain with the error is modified. As an example, consider the case of four chains. Initially the "nxchain" array would look as follows:

```
nxchain (1) = 2

nxchain (2) = 3

nxchain (3) = 4

nxchain (4) = 1
```

If now an error were detected on chain 3, the array would be modified to the following:

```
nxchain (1) = 2

nxchain (2) = 4

nxchain (3) = 4

nxchain (4) = 1
```

Thus after using chain 2, free store uses chain 4 and chain 3 is skipped.

The process of going to the next chain using the "nxchain" array is called commutation. The commutation at the deposition and withdrawal buffers is not synchronized; this simplifies the logic in free storage. Since the "nxchain" array causes free storage to use all usable chains before repeating, all of the chains are maintained at approximately the same length.

Free storage reduces the number of times I/O is needed by using both buffers to make withdrawals and both to make depositions. Without sharing the buffers in this manner a constant overhead of I/O would be required. In a stable system, i.e., one in which the number of depositions is approximately equal to the number of withdrawals, buffers would continually be written onto the deposition end of each chain and read from the withdrawal end of each chain. However, by sharing the duties of the buffers, they act as a small free list which can handle the requirements of a stable system. When the system becomes unstable, commutation and I/O can be brought into play to handle the requests.

Sharing the buffers in this manner has another advantage: two requests for a single device can be processed simultaneously.

In order that the master sector on the device be usable for recovery it must be updated occasionally so that the data it contains will not be so completely outdated as to be useless. When the commutation process at the withdrawal end has covered all usable chains the master sector is rewritten. At this point the percentage of the device in the free pool is recalculated by using the length of each chain, the size of each map and the number of usable chains. If the percentage has dropped below a certain threshold for the device, a call is made to device-distress with the device identification.

<u>PL declarations</u>

```
          dcl  /* master sector */

               msec based (ptr),

                 2 miow,

                   3 ok bit (9),

                   3 pending bit (9),

                   3 error bit (18),

                 2 diow (16),

                   3 ok bit (9),

                   3 pending bit (9),

                   3 error bit (18),
2 wiow (16),
  3 ok bit (9),
  3 pending bit (9),
  3 error bit (18),
2 allocatedsectors fixed bin (17),
2 catastropheflag bit (1),
2 numusablechains fixed bin (17),
2 dchain fixed bin (17),
2 wchain fixed bin (17),
2 diolock bit (36),
2 wiolock bit (36),
2 dbuflock bit (36),
2 wbuflock bit (36),
2 dend (16) fixed bin (35),
2 dndx (16) fixed bin (17),
2 wend (16) fixed bin (35),
2 wndx (16) fixed bin (17),
2 nxchainlock bit (36),
2 nxchain (16) fixed bin (17);
```

miow, diow, wiow - master sector, deposition buffer, and withdrawal buffer I/O words.

ok - set on by device control when I/O has been successfully completed.

pending - set on by free storage when initiating I/O and set off by device control when the I/O has been completed either successfully or not.

error - set on by device control when there has been an I/O error.

allocatedsectors - the number of hyper-records in this partition.

catastropheflag - set on at system initialization time and set off at system shutdown time. This is used to check for catastrophic shutdowns.

numusablechains - the number of chains which have had no errors.

maxchain - the number of chains including those with errors.

dchain, wchain - the index of the chains currently being used for depositions and withdrawals.

diolock, wiolock - used to keep two processors from sensing an I/O completion.

dbuflock, wbuflock - used to lock buffers whenever they are to be modified by withdrawals, depositions or I/O.

dend, wend - used by the free storage initialization procedure. These contain the addresses of the ends of the chains.

dndx, wndx - these are the indices giving the next available slot in the deposition buffer and the next available address in the withdrawal buffer.

nextchainlock - keeps two processors from changing the next chain array.

nextchain - is used to get the next usable chain.

When an error occurs on any chain, the next-chain array is modified so that the chain is no longer used.

```
dcl  /* buffers and free maps */

     buf (16) based (ptr),

      2 checksum bit (36),

      2 fill_1 bit (1),
          2 seqnum bit (35),

          2 fill_2 bit (1),

          2 chainnum bit (35),

          2 backadd bit (18),

          2 frwdadd bit (18),

          2 add (248);
```

checksum - contains the Multics standard checksum of the entire buffer excluding checksum.

seqnum - contains a number which is incremented by one for each map going from the withdrawal to the deposition end of the chain.

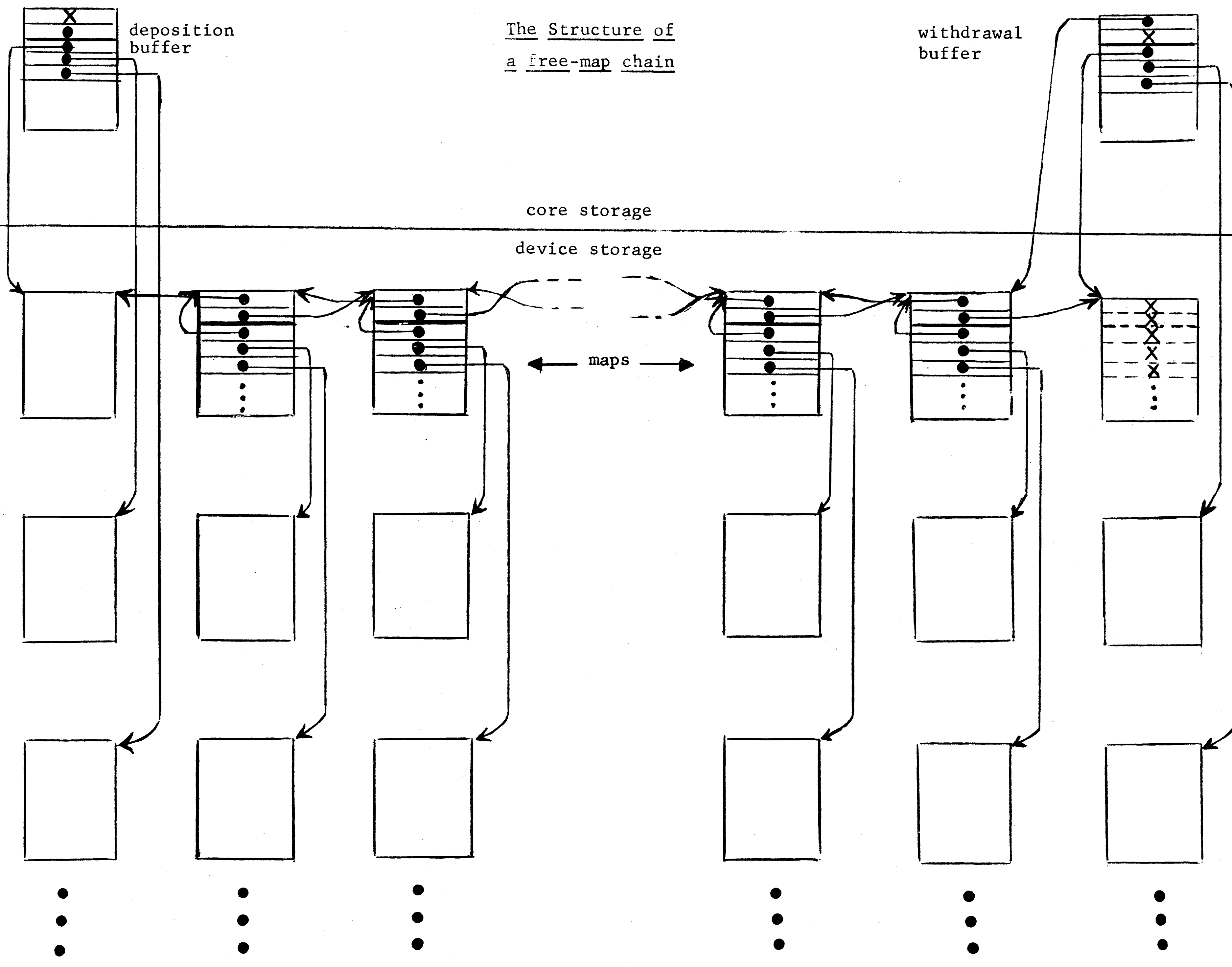
chainnum - the index of the chain in which this buffer occurs.

fill_1, fill_2 - used to right adjust the two 35-bit strings in the 36-bit machine words.

backadd, frwdadd - the backward and forward linking pointers.

add - the free hyper-record addresses, (add (1) is a self-pointer).

The Structure of a free-map chain


deposition buffer
withdrawal buffer
core storage
device storage
maps