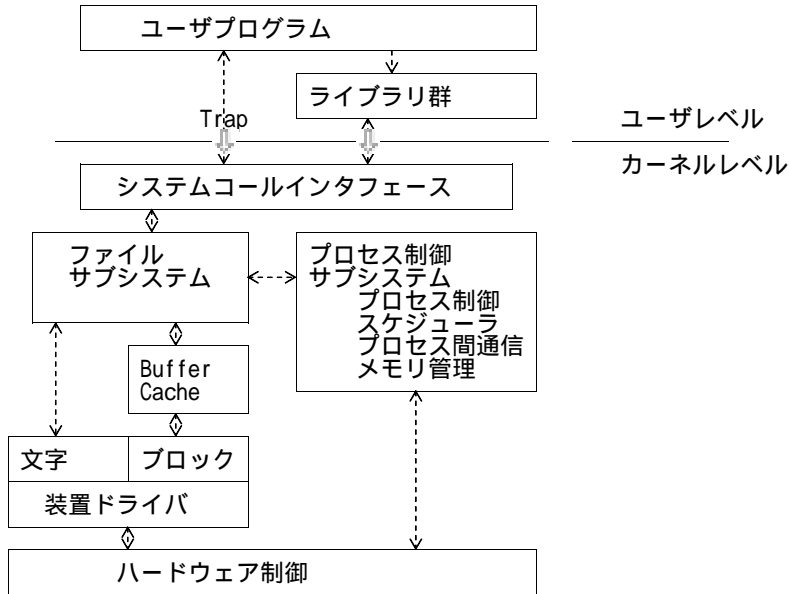


§ 復習: UNIX file system

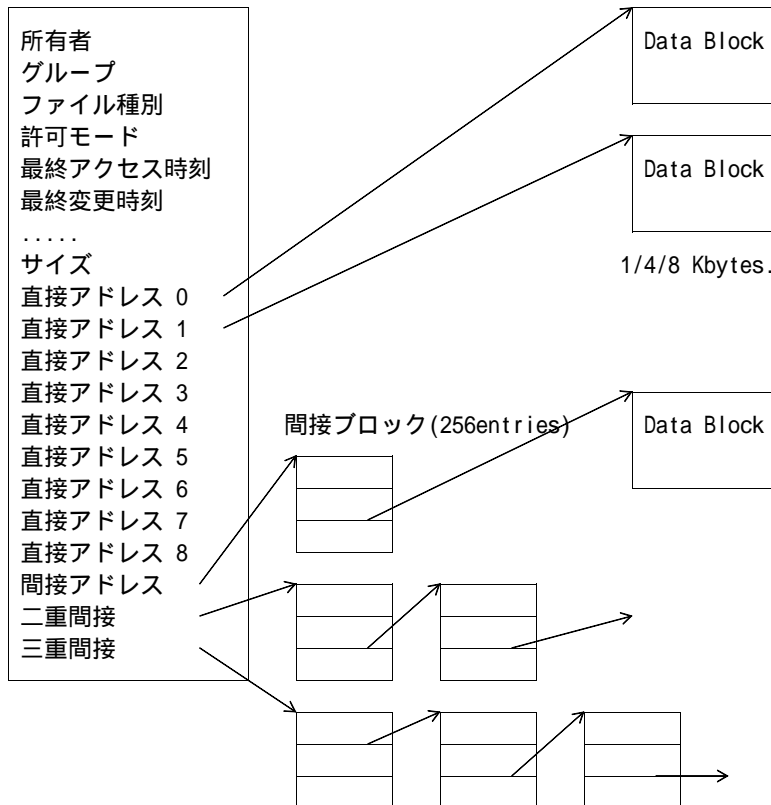
[1] UNIX アーキテクチャ



[2] ファイルシステム

各ブロック記憶装置は、一つ以上のファイルシステムが存在する。  
 各ファイルシステムは『論理装置番号(device number)』で識別する  
 データはブロックとして記録されている。  
 個々のファイルに対応して『i-node』テーブルが存在する。  
 i-nodeテーブルには、各ファイルの制御情報が書かれている  
 『ファイル名』⇒『i-node』⇒『ファイル実体』

[2-1] i-node



[3] バッファキャッシュ

磁気ディスク装置などのアクセスを効率化するために、カーネルは内部バッファを使っている。これを"Buffer cache"と呼ぶ。(本資料では、File cacheと呼ぶ子ともある)

Buffer cacheはソフトウェア構造である。Hardware cacheと混同しないこと。

ブロック装置から読みだしたデータは、バッファキャッシュに保存しておく

ブロック装置に書き出すデータは、先ずバッファキャッシュに書き込み、後でブロック装置に書き込む

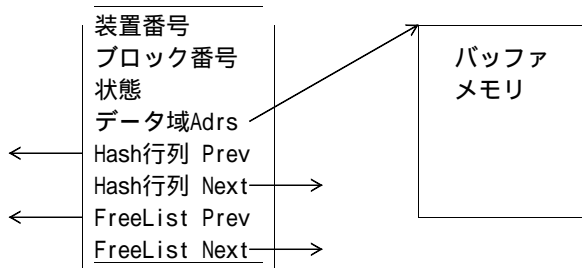
装置番号、ブロック番号のブロックにアクセスするときは、[、]で決まるバッファキャッシュを見つける。

⇒ Hash行列

バッファキャッシュ領域が必要なときは、least recently used (最長不利用)のバッファを利用する。

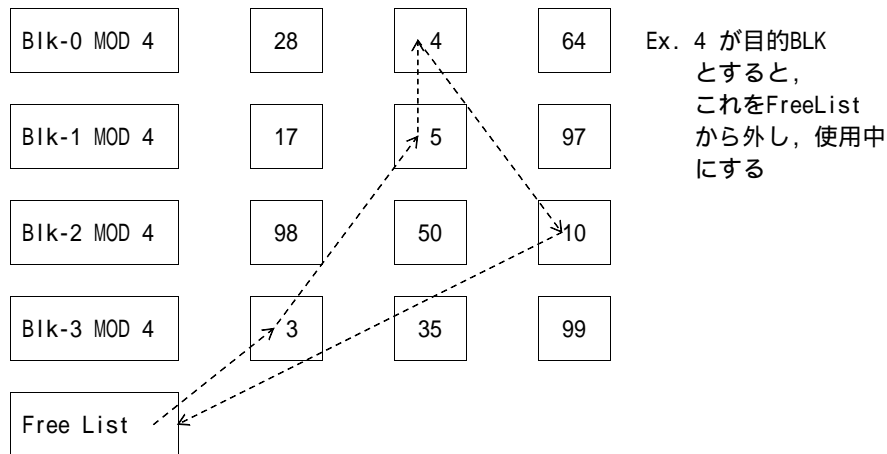
⇒ FreeList

[3-1] バッファ構造

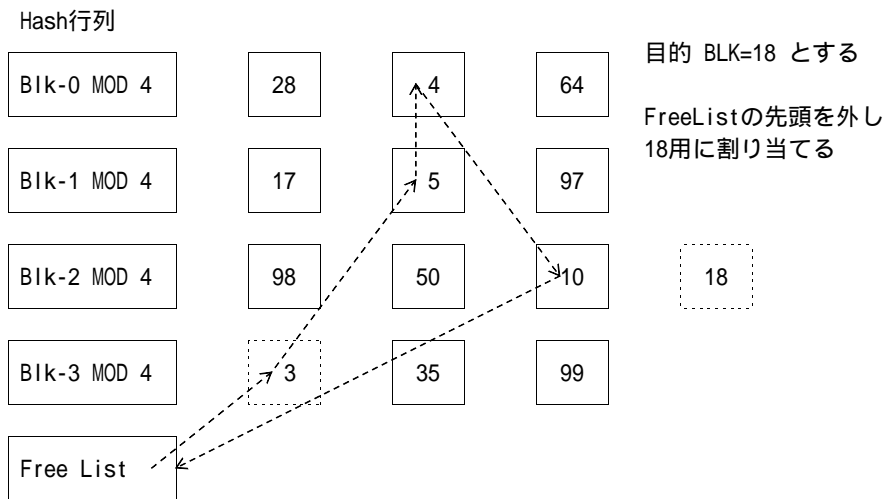


[3-2] バッファキャッシュ管理アルゴリズム

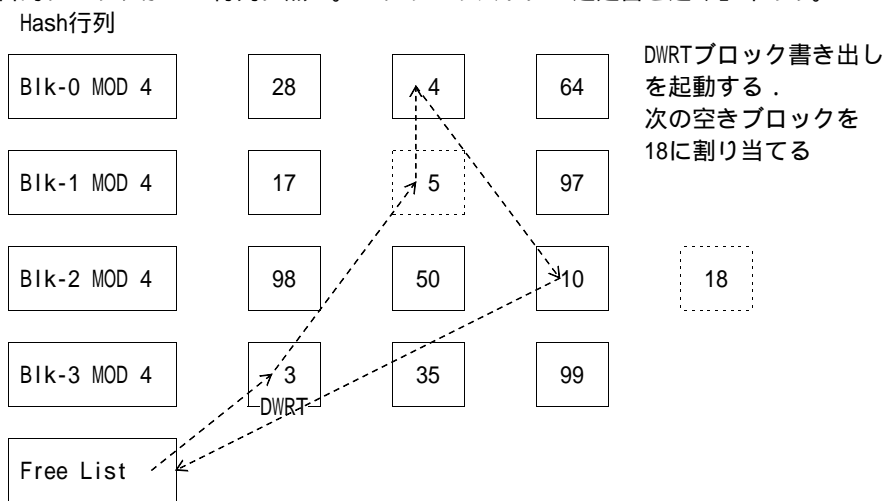
(a) 目的ブロックがHash行列から見付き、かつフリーリストにつながっている場合  
Hash行列



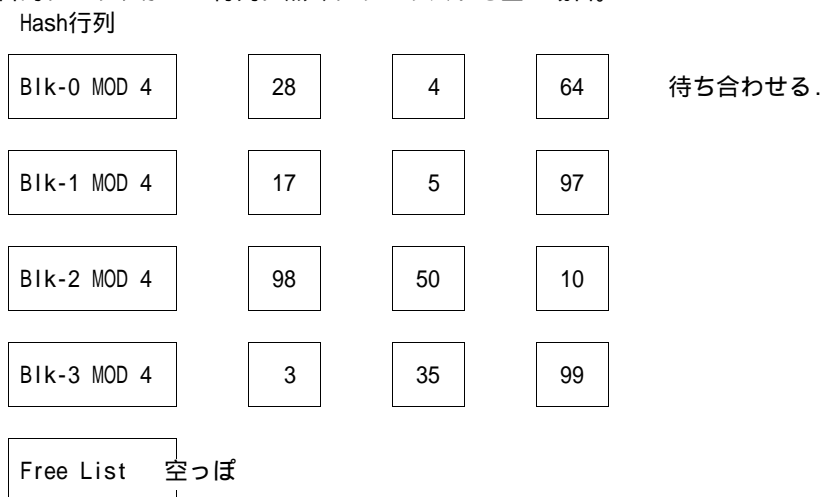
(b) 目的ブロックがHash行列に無い。⇒フリーリストから取り出す。



(c) 目的ブロックがHash行列に無い。⇒フリーリストに『遅延書き込み』印あり。



(d) 目的ブロックがHash行列に無くフリーリストも空の場合。



[参考文献]

M.J.Bach "The design of the UNIX operating system" Prentice-hall  
(訳本 "UNIX カーネルの設計" 共立出版)

## § 分散ファイルシステムの目的

ネットワーク透過性  
ユーザ移動性  
耐故障性  
Scalability

## § 分散ファイルシステムとNaming

位置透過性 Location transparency  
位置独立性 Location independence

ホスト名+ローカル名前  
ローカルな名前空間にリモートのディレクトリをつなぐ NFS  
システム全体に全てのファイルにわたった大域的な名前空間

## § ファイル共有のセマンティクス

### UNIXセマンティクス

クライアントがオープンしているファイルへの書き込みは、同時に同じファイルをオープンしている他のクライアントにも直ちに読める。

### セッションセマンティクス

オープンしているファイルへの書き込みは、ローカルクライアントには直ちに読み込み可能となるが、同時に同じファイルをオープンしているリモートクライアントには直ぐには読み込み可能とはならない。ファイルがクローズされると、これより後にオープンされたセッションに対してのみ反映される。

### 不変共有ファイルのセマンティクス

ファイルが共有ファイルであると宣言されたら、それ以降変更出来なくなる。

### トランザクションセマンティクス

セッション中のファイルのロックにより実現

## § リモートアクセスの手法

### リモートサービス

すべてのファイルアクセスはサーバーによって扱われる。

### キャッシング

アクセスはクライアント側でキャッシュ（主記憶//ローカルディスク）に対して行われる。

○フラッシュ：ダーティーブロックを原本ファイルに書き出す

#### ☐ライトスルー

どのキャッシュでも、変更が加えられたら直ちにサーバーのディスクに書き出す  
低性能、高信頼性。 UNIXセマンティクスが可能

#### ☐遅延書き出し

変更の加えられたキャッシュは、あとでサーバーのディスクに書き出す。

ブロックがクライアントのキャッシュから追い出される時にフラッシュ

⇒高性能

一定の時間間隔(Ex.30秒)で走査し、更新されたブロックのみフラッシュ

クローズ時書き出し ⇒セッションセマンティクス

## § 耐故障性

### Statefullサーバー

サーバーが、個々のサービス要求をまたいでクライアントの情報を保持。

高性能, 高機能

障害時回復には『状態』を元に戻さなければならない。  
⇒クライアントとの対話に基づくかつぶくプロトコル

#### Statelessサーバー

クライアントの要求は一回一回で完了し、クライアントの状態を保持しない。  
クライアントは、サーバーからの返答が得られなければ要求を再送し続ける。

⇒これらの操作は Idempotent (何回繰り返しても 1回と同じ効果)

障害が起きても簡単

要求メッセージが長くなる。

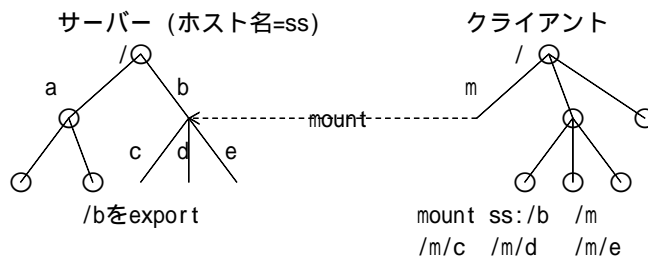
処理が遅くなる

#### ○ファイルの複製

## § SUN NFSの例

分散OSの構成要素というよりも、ファイル共有のためのネットワークサービス、ヘテロジニアス環境でのファイルサービス

#### ○マウントとマウントプロトコル



スーパーユーザだけがマウント操作を行える。

マウントのための通信: マウントプロトコル

マウントされた後は、クライアントマシンのユーザはリモートのディレクトリのファイルをあたかもローカルなディレクトリであるかのようにアクセスできる。

各マシンは自分の名前空間を自由に構成できる

⇒ マシン毎にファイル名前空間は異なりうる。

#### ○ファイルアクセスとNFS プロトコル

ファイル操作のために以下のRPC を提供

ディレクトリ内のファイル探索

ディレクトリエントリの書き込み

リンクとディレクトリの操作

ファイルアトリビュートへのアクセス

ファイルの読み書き

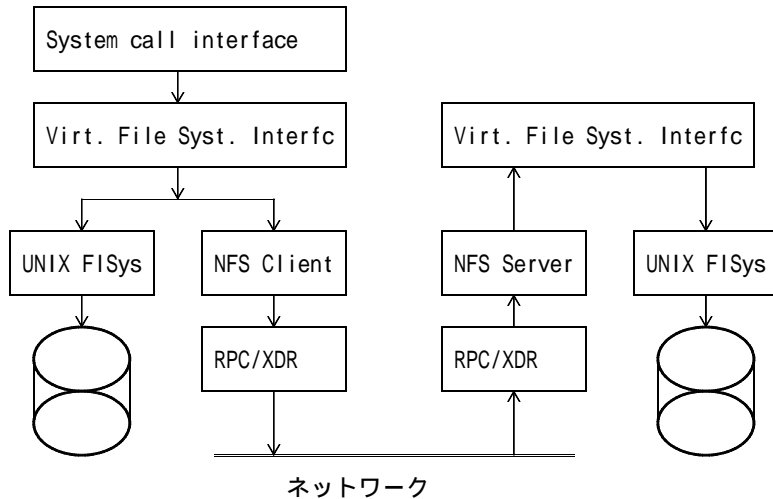
⇒ ファイルのオープンとクローズが無いことに注意

#### ○NSF サーバーは Statelessサーバーである。

サーバー側には、UNIXのオープンファイル表もファイル構造体もない。

耐障害性

○インプリメント



性能向上のためにキャッシュを利用  
 (ファイルブロックキャッシュとファイルアトリビュート=i-node=キャッシュ)  
 先読みと遅延書き出し  
 新しいファイルがあるマシンで作られても、全てのマシンで見られるようになるのは30秒後  
 UNIXセマンティクスはサポートしていない。

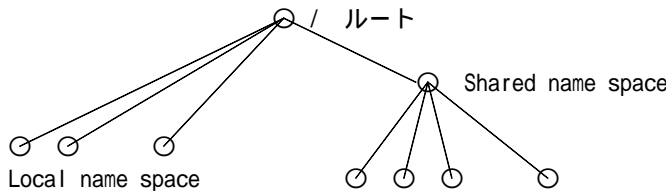
§ ANDREW ファイルシステム

CMU で開発. OSFに導入

○クライアントに単一の位置透明なファイル名前空間を提供

Local name space = その WS のルートであり, WS毎に独立している.

Shared name space



○ファイル操作と共有のセマンティクス

サーバーからファイル全体をキャッシュする。(最新版は64KB Cache)  
 File open 時にサーバーからファイル全体をキャッシュし, Close 時に変更されたファイルの複製をサーバーに送り返す.  
 キャッシュされたファイルを再オープンするときもキャッシュが使われる.  
 Cache された実態(File や directory) は, 他から通知されない限り有効とする.

○セッションセマンティクスとCallback

ClientがFile/directoryをcache したら, サーバーは自分の状態情報にこの事実を記録する. これを Client が, そのファイルのCallbackを持ったという.  
 Serverは他のClientによるそのファイルの変更を許す前に, callbackを持っている Client に通知する.(Client のcallbackを消去)  
 Clientは, そのファイルのCallbackを持っている限り, キャッシュを使える  
 あるClientがFileを更新した後 Closeしたら, このFileをCache している他の全てのクライアントは, そのファイルのCallbackを失う

○インプリメント

Statefull