March 26

Simona Boboila

File Systems

API

Semantics

Implementation - code
  - disk layout

2 Key issues in FS design: performance reliability

Disk layout:

1) directory lookup structure

2) identification of blocks making up file file structure

3) free block management / block allocation

File Allocation Table

superblock  F.A.T.  root dir

contains the info:
  block size
  volume size (in blocks)

$FAT\ size (in \ blocks) = \left\lceil \frac{volume\ size}{FAT\ entries\ per\ block} \right\rceil$

free / EOL / ptr

  EOL
  F
  EOL
  F
  F
1) linear search of fixed-sized entries

   name, start block

   8+3

   performance: random access

2) file structure

   linked list w/ external pointer array

   problems: vulnerable to fragmentation

Lookup structure: linear search

   B-tree or other tree

→ B-tree:

   \( \frac{K}{2} \ldots K \)

   from bottom up, split nodes in 2

   using the file system directory structure
file structure: linked list
  i-node (indirect block)
  extent list

  i-nodes

  extent list
  [1000 - 1500]
  [1210 - 12753]
  [200 - 310]
  [0 1000 - 1500]
  [500 -]
  [1143 -]

  free block management: linked list
  allocation table
  bit map: Berkeley Unix
  extent list - de-fragments space naturally

  original Unix
extents:
- you merge regions when they are freed
- can be maintained as a B-tree

Overhead:

```
/usr/local/bin/program
```

= 10 disk accesses just for one single file due to caching
modern OS:

block cache

File system

phys pages

VM

Block device

Problems: non-disk FSs
- virtual memory

VFS:
- file system

inode

Kernel

Read file

Write page