LEC 10

PART 1

Block Devices = LVM / Flash

- File Systems - (logical volume mgmt)
  - RAID
  - Concatenate / Split (Partition)
  - Snapshot / Migration
    (Entire disk state at a single point in time)

Logical Addr Space

Disk 1

Disk 2

1. Block modified
then snapshot
then write
intercepted 2 snap modified

Similar to copy-on-write
Migration Stall \[ \Downarrow \xrightarrow{\text{still}} \] Still Using older location

Migration done for

old

new

\[ \Downarrow \]

FLASH (NAND - FLASH)

32-128 pages

2-4k Block.

You can only erase & write a large amount at a time
logical -> Physical Addr. Mapping

In Dynamic

Garbage collection & write - evening done together

Flash Translation layer

MLC = Multi-Level Cell,
SLC = Single Level Cell.

Multi-Level

Basic [0 1]

MLC [00 01 10 11] (gray code)

(More complicated prog...
Flash

Read - 20 - 50 ms (for 2-4K)
Write - 100 - 200 ms
Erase <= 2 ms (1/4 - 8MB)

FILE SYSTEMS

namespace - objects - actions
(hierarchical text: names)

objects - sequence of bytes - files
- directories
- special files
PART II

File Systems

<table>
<thead>
<tr>
<th>Name</th>
<th>Block#</th>
<th>Len</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>file1.txt</td>
<td>1</td>
<td>1000</td>
<td>F</td>
</tr>
<tr>
<td>mydir</td>
<td>3</td>
<td>512</td>
<td>D</td>
</tr>
</tbody>
</table>

Block 0

Block 1

Block 2

Block 4

Ancient Unix: Free Block List
Linked List of Free Blocks

Another way = Bitmaps

List of Blks: \[ \overline{011110000} \]
1. Bitmap Search
   \[\rightarrow\] Berkley Fast File Sys.
   \[\rightarrow\] MSDOS.

2. Extent List
   \[\begin{array}{c}
   90-95 \quad 0-25 \\
   25-35 \\
   10-20
   \end{array}\]
   B-tree: for ordered list / lookup.
   \[\rightarrow\] Different from Binary Tree.

   File Organization: contiguous
   \[\rightarrow\] You can't modify file size

3. MSDOS
   \[\rightarrow\] Linked List
   \[\rightarrow\] File Allocation Table.

   \[\begin{array}{c|c}
   \text{In Use} & \text{Next} \\
   \end{array}\]
By allocating spits in a linked list properly we can do caching.

**NODE**

* + Indirect Blocks

No multi-indirection overhead for small files

⇒ Faster for common case of small files

⇒ E.g.

<table>
<thead>
<tr>
<th>Extent List</th>
<th>Physical</th>
</tr>
</thead>
<tbody>
<tr>
<td>local</td>
<td></td>
</tr>
<tr>
<td>[0 199]</td>
<td>[200 299]</td>
</tr>
<tr>
<td>[0-99]</td>
<td></td>
</tr>
</tbody>
</table>
Diff with B-tree & Binary Tree is that the branching factor is many times bigger, like a couple of 100.

File vs. Name of file itself.

File Attributes

- Length
- Timestamp
- Ownership
- Permissions

[Windows NT Namespace]