ELEN627 Lecture 8

- Review of last class
- Memory issues

Multimedia - memory

- What’s different

- Video takes a lot of memory for buffering
  - Retrieval of 200KB/s every 0.5 seconds needs 2 buffers of 100KB each
    - Higher requirements with VBR
    - Can use up memory pretty fast

- Typically sequential reads
  - Normal buffer management policies may not work well

- To guarantee availability
  - Buffers have to be reserved - pinned in memory
Memory pinning

- To make sure buffer is not stolen by another application
- Make the page not replaceable

Video and memory

- Typically data access from disk involves two copies
  - Disk DMAs data into kernel address space
  - Kernel copies data into user’s buffer
- May have to copy it back to kernel for network delivery
- Data copying can be a major overhead
- Need to use shared address spaces
- More efficient retrieval and memory usage
- Use copy-on-write semantics for sharing
Storing video in memory

- Large memory needed
  - Expensive

- Provides more bandwidth than disk
  - Can be useful for high demand movies

Storing video in memory

- Don’t store the entire movie

- Store only the peaks in VBR streams
  - Can reduce the demand variance
  - Retrieve data from disk normally
  - During high demands, retrieve from memory

- With little memory, can improve stream throughput
Video Buffer management

• File systems cache data retrieved from cache
  – Caching normally improves performance

• Use LRU management
  – Kick out LRU block to make room for a new block
  – Works well for typical file applications
  – Not so well for sequential accesses

Buffer management for video

• BASIC
  – For each block, compute an offset-ratio
  – Replace blocks with largest offset-ratio

• Offset-ratio: time at which this block may be accessed

• Example: Stream reading block 1 of file at 200KB/sec
  – Block size 32KB
  – Block 10 from that same file has an offset of 9*32 = 288KB
  – Offset ratio = offset/stream rate = 288/200 = 1.44 secs.

• Replace blocks that will be used farthest in the future
  – Can predict because of sequential playback

• Similar to Optimal block replacement policy
Video buffer management

- DISTANCE
  - Compute distance when block may be used
  - Replace block with the largest distance

- Distance is measured in terms of # of buffers

- Client 1 consuming block 10 of file1, client 2 block 2 of file1
  - Distance is 8

- Same as BASIC if rates are same
  - Optimal if all the stream rates are same

Figure 1 The change in miss ratios of LRU, MRU, BASIC, DISTANCE and optimal buffer replacement algorithms when inter-arrival time of clients is changed. The number of buffers is n = 2400, thus, the buffer cache is 76.8 MB.

Miss ratios vs. arrival rates
Integrated memory management

- Handle video and regular files
- What is the best way to manage the cache

Figure 2: The change in miss ratios of LRU, MRU, BASIC, DISTANCE and optimal buffer replacement algorithms when the number of buffers, and then the size of the buffer cache, is changed. Inter-arrival time of clients is 10 minutes.

Miss ratios vs. number of buffers
Multimedia accesses

- Mostly sequential accesses
  - Caching not much use

- Accessing large datasets
  - Can throw out a lot of data from file cache
  - LRU doesn’t work well

- Need better mechanisms

- Individual users can decide not to cache

- What about Internet servers?

Caching video/audio

- Least Frequently Used

- AVI_MRU = MRU policy for audio and video
  - Caching dependent on file type
  - Put audio, video files at the end of LRU chain

- SIZE_MRU = MRU policy for large accesses
  - Caching dependent on file size
  - Put files above some threshold size at the end of LRU chain

- SpacexAge policy
  - Replace file with largest SpacexAge product
  - Difficult to implement
  - Use approximations
Evaluation of caching policies

- Traces from NCSA

<table>
<thead>
<tr>
<th>Type of request</th>
<th># of req.</th>
<th># of blocks</th>
<th>Avg. Req. size in blocks</th>
<th># of files accessed</th>
</tr>
</thead>
<tbody>
<tr>
<td>Text</td>
<td>300,329</td>
<td>1,645,016</td>
<td>4.21</td>
<td>14,361</td>
</tr>
<tr>
<td>Images</td>
<td>544,389</td>
<td>1,247,139</td>
<td>2.29</td>
<td>8,137</td>
</tr>
<tr>
<td>Audio</td>
<td>703</td>
<td>8,939</td>
<td>12.72</td>
<td>106</td>
</tr>
<tr>
<td>Video</td>
<td>749</td>
<td>89,040</td>
<td>118.88</td>
<td>90</td>
</tr>
<tr>
<td>Total</td>
<td>936,170</td>
<td>2,990,134</td>
<td>3.19</td>
<td>22,694</td>
</tr>
</tbody>
</table>

Performance measures

- Hit Ratio

- Block Service time
  - block miss rate * block miss penalty

- Request Service time
  - Request miss rate * Request miss penalty

- Block service time ignores spatial locality on disk

- Request service time ≤ n * block service time
**Spatial locality on disk**

- Depends on a number of factors
  
  - The file system allocation policy
  - The disk utilization

<table>
<thead>
<tr>
<th>Characteristic</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>% of single-block files</td>
<td>63.0%</td>
</tr>
<tr>
<td>% of Multiple block files</td>
<td>37.0%</td>
</tr>
<tr>
<td>% of blocks in multi-block files</td>
<td>92.4%</td>
</tr>
<tr>
<td>% of multi-block files allocated contiguously</td>
<td>75.8%</td>
</tr>
<tr>
<td>Average # of contiguous blocks in broken multi-block files</td>
<td>7.755</td>
</tr>
</tbody>
</table>

**Simulations**

- Used traces from NCSA’s web server
- Simulate several caching policies
- Assumed on an average 4 blocks are contiguous on disk
- Measured
  
  - Hit rate
  - Block service time
  - Request service time
Impact of disk locality

Request Hit ratios
Summary of results

- Possible to improve performance by policies other than LRU
- LFU works well with smaller cache memories 1-16MB
- SpaceXAge consistently performs better than LRU
- SpaceXAge performs best with 16-256MB cache size
- Above 256MB, all policies are the same
- SpaceXAge is easier to implement than LFU
- Benefit of SpaceXAge improves with increased video usage and increased disk data rates