Multimedia Storage Servers: A Tutorial

D. James Gemmell
Harrick M. Vin
Dilip D. Kandlur
P. Venkat Rangan
Lawrence A. Rowe

Continuous Media Recording and Playback

- Continuous media: audio and video
  - Sequence of media quanta
  - Convey meaning only when presented in time

- **Recording**
  - Input device produces data (camera, etc.)
  - Continuously stored to prevent buffer overruns

- **Playback**
  - Server retrieves data from disk
  - Prevent output device (display, etc.) from starving

Single Stream Playback

- Sequence of retrievals of media blocks from disks
- Deadlines = scheduled playback times
- Retrieval is bursty
- Must buffer data before playback
- Adequate buffers protect against starvation
- Tradeoffs: prevent starvation, minimize buffer space and latency
- Larger buffers ==> longer latency
  - Read more data before transfer can begin
- Small buffers work well
  - Disk data rates much higher than stream rates

Multistream retrieval

- Different streams may access different parts of file simultaneously
- Simplistic: one stream per disk head
  - Most disks are not parallel transfer disks
  - Equivalent of one stream per disk (one stream per disk arm)
- Multiplex disk heads among streams
- Meet real-time delivery requirements of all streams
- Ensure that guarantees can be met by admission control: limiting total streams allowed

Disk Scheduling Algorithms

- Traditional file systems:
  - Reduce seek times and rotational latency
  - High throughput, fair access to each stream
  - Example: Scan -- order all seeks, sweep across
- **Real-time scheduling of tasks with deadlines:**

1. Earliest deadline first (EDF)
   - No optimizations to reduce seek distances
2. Scan-EDF: For accesses with same deadlines, use scan to minimize seek distances
   - Effective if deadlines are batched
3. Rounds: During each round, retrieve block for each stream

Round Length and Latency

- Round = total time to fetch a block for every stream
- **Round-robin: order of servicing streams is fixed from round to round**
  - Latency between retrievals of successive blocks for one stream is the round length
  - Playback can begin after first full round
- **Scan: Within round, order reads by seek distances**
  - Scanning across the disk: sometimes will service a stream at end of one round, beginning of next
  - Need more buffer space: two fetches per stream
  - More efficient: round is shorter
- **Grouped sweeping scheme: each round partitioned into groups; scan within groups**

Buffer Requirements

- During a round: retrieve at least as much as consume
- **Fetch enough for each stream during round that it can play until it fetches again**
- Streams may have different playback rates
  - Waste buffer space if always fetch the same amount for each stream
  - (Have to fetch for worst case/highest bandwidth)
- Fetch amount proportional to consumption rate
- **FIFO buffers work best**
- Topping up: if buffer isn't empty, just fill it--don't fetch entire block
- Read varying amounts; may then have to seek

Admission Control

- Determine whether a new stream can be serviced without affecting those already in service
- Strict: allow no missed deadlines
- Some applications can tolerate missed deadlines (lost video frame, lost audio packet) occassionally
- **Quality-of-Service Categories**

1. Deterministic: guarantee all deadlines are met
   - Must assume worst-case seek/rotation times
2. Statistical: guarantee deadlines met with probability
   - Exploits variations in block access times from disk, compression rates (e.g., avg. rotation)
3. Background: no guarantees; service when time

- During a round: service deterministic streams before statistical before background
- **Example: variable-rate compression; media blocks will compress into different sized quanta**
- Deterministic: Worst-case compression statistics would decide how much to fetch on each cycle
- Or, could record compression ratios achieved and fetch actual data
- Statistical: Fetch amount based on statistics on compression rates
- **Background and statistical: sometimes miss deadlines**
  - May play back information late
  - May drop media blocks (skip frames, samples)

Contiguous vs. Scattered Data Placment

- **Contiguous: Store media file as continuous file**
  - Only one seek at start of file
  - Subject to fragmentation
  - Hard to edit (large copies)
  - Useful for read-only systems

- **Scattered: Blocks of file in various disk locations**
  - Seek overhead to move between blocks
  - Avoid fragmentation and copy overheads
  - Efficient if choose a fairly large block size
  - Read one block each round (intrafile seeks during a round)

Constrained & Log-Structured Placements

- **Constrained data placement:**
- Constrain the layout of blocks on disk to bound the time spent on intrafile seeks
- Does not aim to avoid them, but limit their effect
- Attractive when block size must be small (e.g., conventional file system with 4KByte page sizes)
- Don't use scan--fetch all blocks for a stream together

- **Log-Structured Placement**
- Group writes together and append to log
- Optimized for writes; read performance can be poor

Data Striping and Interleaving

- Misuse of terms (striping = interleaving)
- Really distinguishing between RAID Level 3 style and RAID Level 5 style accesses
- Group of disks acting in parallel (seek, rotate, transfer simultaneously) vs. disks more independent

- **RAID Level 3 style: for each stream, fetch blocks from each disk on each round**
  - Ensures balanced load on disks
  - Requires more buffer space
- **RAID Level 5 style: read larger block from one disk, then jump to next disk**
  - Balanced load if starting positions are offset

Tertiary Storage and Hierarchies

- Hand-wavy
- "Economics will dictate tertiary storage"
- Options:
- **Store beginning of files on disk, rest on tape**
  - Tape too slow to support this today
- **Intelligent caching strategies**
- User access patterns highly predictable for some applications
- **Distributed systems: servers at various geographical locations, moving multimedia data between them**

File System Metadata

1. Linked list of pointers to blocks: inefficient random access
2. File allocation table: entry per block, big memory
3. Unix-style I-nodes (separate inode for each file, pointers to blocks): indexes
4. Hybrid style: linked list + index structure
   - Efficient streaming without having to fetch metadata blocks
   - Also possible to do random access

- Multiple media formats: video, audio, text

Editing

- Media components immutable
- **Edit by manipulating pointers to media component files or to portions of files**
- **Small insertions: pointer manipulation may not be feasible for maintaining continuous playback**
  - Copy operations are time-consuming
- **Insert/delete an integral number of blocks; just alter the file map**
- Maintain partially filled blocks; distribute data among blocks after insertion/deletion

Interactive control

- **Pause/resume hard**
- If multicasting, interferes with shared stream

- **Fast-forward:**
- Playback at higher rate than normal
- Playback at normal rate, but skip some frames
- Maintain separate, highly compressed file (e.g., MPEG-2 standard suppports highly compressed D frames that have no interframe dependency)
  - Additional storage space
- Categorize blocks as relevant or irrelevant to FF
  - Normal retrieve all; during FF: only relevant