SeFS: Unleashing the Power of Full-text Search on File Systems

USENIX FAST '07 (WiP)

Stergios V. Anastasiadis

(joint work with G. Margaritis)

U. Ioannina, Greece

Motivation

- Full-text search in modern systems often used for
  - Email
  - Application help files
  - Log files
  - Any file that contains text
  - ...
- Maybe full-text search should
  - Receive the attention it deserves from system designers
  - Be made available as general system service to developers

File System Features

- File size
  - Most files are small BUT
  - Most bytes are in large files
- File lifetime
  - Is highly variable across different systems
  - Varies from minutes to years
  - Has median age = tens of days
- User expectations
  - Perceive the file system as a reliable “storage medium”
  - Anticipate changes to be made visible almost immediately

Attempt #1: Information Retrieval

- Upside
  - Online support of Boolean queries and dynamic updates
  - Mature technology (first ACM-SIGIR in 1978)
- Downside
  - Technology initially developed for article archives
  - “Dynamic update” mainly means addition of new articles
  - Indexing structures biased from decade-old studies to serve the above assumptions

Index Maintenance in IR

- Inverted files
  - Map terms to term positions in documents (posting lists)
- Decades ago
  - Updated infrequently to include new articles
  - Contiguously stored on disk to minimize query time
- Recently
  - Updated dynamically to include new articles BUT
  - Treating document changes as insertions/deletions
  - Use complex relocation techniques to preserve contiguity

Question

- Why not allocate posting lists on fixed-size blocks?
  - Avoid data relocation during inserts/appends
  - Amortize disk seeks over large block sizes
  - Simplify system structure without major performance penalty
- Several I/O demanding systems based on blocks
  - Database systems
  - The Google File System (chunks of 64MB)
  - Video streaming storage
  - …

Attempt #2: Web Search

- Upside
  - Technology can handle large data sets
  - Search results quite close to user expectations
- Downside
  - The web is perceived as unreliable; infrequent updates ok
  - Distributed nature make stats gathering difficult
  - Dedicated hardware devoted to indexing
- Bottom line
  - Despite commonalities, file systems differ from the web
  - Exploit strengths without adopting weaknesses

Attempt #3: Relational Databases

- First approach
  - Store all system metadata on a relational database system

  E.g. SRB/SDSC, SCFS/MIT, Amino/Stony Brook
  - Ok for ftp-like services
  - BUT maybe too heavyweight for fine-grain accesses

- Why?
  - File systems custom-developed/optimized for handling their metadata

Relational Databases (cont'd)

- Second approach
  - Keep system metadata on custom file-system structures
  - BUT maintain user metadata in a database
  - Maybe ok but still insufficient for full-text search

- Why?
  - Full-text search more than a few attribute/value pairs per file
  - Inverted files most efficient structure for large text collections

Conclusion

- File systems
  - More flexible in their functionality than article repositories
  - More reliable and amenable to stats gathering than the web
  - More efficient in fine-granularity operations than RDBs
- Full-text search on file systems
  - Useful for different applications and system services
  - Should be designed from scratch, free from inherent drawbacks of solutions from other environments