A Unifying Approach to the Exploitation of File Semantics in Distributed File Systems

Philipp Hahn

System Software and Distributed Systems group
Department for Computing Science
Carl von Ossietzky University Oldenburg
Germany

December 15, 2005

Observations

- Many (specialized) distributed file systems exist
- Few distributed file systems are wildly used
  - General DFS for common case
  - Handling all files the same
  - Optimized for average performance
- But some files are “more equal” than others
  - Need for special casing arises
  - Useful for optimizations
  - Requirements might change over time

Different dimensions

- Concurrency: writer vs. readers / writers
- Locking: mandatory / advisory
- Availability vs. consistency: work must go on
- Fault mode: error / alternative version / block
- Caching: write-back vs. write-through
- Access frequency: read-once vs. many times
- Access pattern: random access vs. sequentially
- Versions: whole BLOB vs. delta changeset
- Content: BLOB vs. structured data
- Encryption: on disk / on network
- Compression: on disk / on network

Benefits

- Stacking functionality as needed:
  - Bypass locking for backup software
  - Weaker consistency in disconnected mode
  - Do not replicate temporary files
  - Different replication placement strategies
  - Expensive concurrency strategies only when needed
  - Cache needed files, collect multiple updates
  - Prefetch sequentially accessed files
  - Do not encrypt / compress again
  - Different storage/versioning for BLOB/structured data

Basic Idea

- Framework for distributed file systems
- Pluggable modules:
  - consistency strategies
  - caching strategies
  - fault mode
  - transparent compression
  - transparent encryption
  - file versioning
  - security models
- Configuration:
  - users know best, else fall back to default
  - limits by administrator
  - self-tuning by system: watch history