Search-Oriented Interfaces for Distributed Metadata Indices

Sasha Ames, Ethan Miller, Carlos Maltzahn

Storage Systems Research Center
University of California - Santa Cruz

Motivation

- Users need effective means to locate their files
- Increase in scale of today's file systems makes sole use of directories for organization unmanageable
  - **What does 50 billion files mean?**
    - Millions of directories
    - Long path names
    - Large fan-out, too many files in each
  - **Examples:**
    - Scientific applications
    - Enterprise data
    - Archives
    - Server log data
  - **Users need to share and find others files within an organization**
- Users have already use search tools for file systems
  - Domain specific or not part of file system
  - Some limited to ranked list, limited organization
  - Interfaces have room for improvement

Our Solution

- Indexing of metadata: links and attributes - to enable search capabilities in file system
- Hybrid of flat and graph-based namespace
- Paths defined through query language that includes
  - Link traversals
  - Query terms based on attributes and links
    - names and directories not required
  - View specifications
- Queries in paths may directly lookup files or produce "virtual directories"
  - Hybrid of boolean and vector based retrieval
  - Listings can be ordered by score from ranking functions
- Nameless file creating: may specify attributes or allow system to assign automatically via transducers
- Faceted file listing - any attribute may be used for names
- Places storage for links, attributes and indexing under the file system interface
  - Should guarantee consistency, ensuring all users and applications have the same views
- Support for location-based search using links
- Support for faceted search interfaces on top of file system
- Provide for comprehensive transition for users and applications
  - Includes full POSIX support
- "Binding" of queries provides temporarily stable paths for applications

Queries for Files

Example Syntax: `/!operator[param1]+facet1=value1;^facet2=value2;-facet3=value3a>value3b/query2`

Old: `dist_md/doc/papers/ames-fast07/ames-fast08.tex`

New:

```
/!match:project=dist_md;category=doc>papers;author=ames;venue=fast;
      year=2007;format=tex

vi (-autocreate -bindpath) /project=Distributed_Metadata
Automatically creates a new file with a tag in the project
facet.  The application may continue to use the same path
expression to reopen the same file.

ls /!match:+keyword=file systems;+category:publications|
 !listfacets[_]author;year
```

Returns:

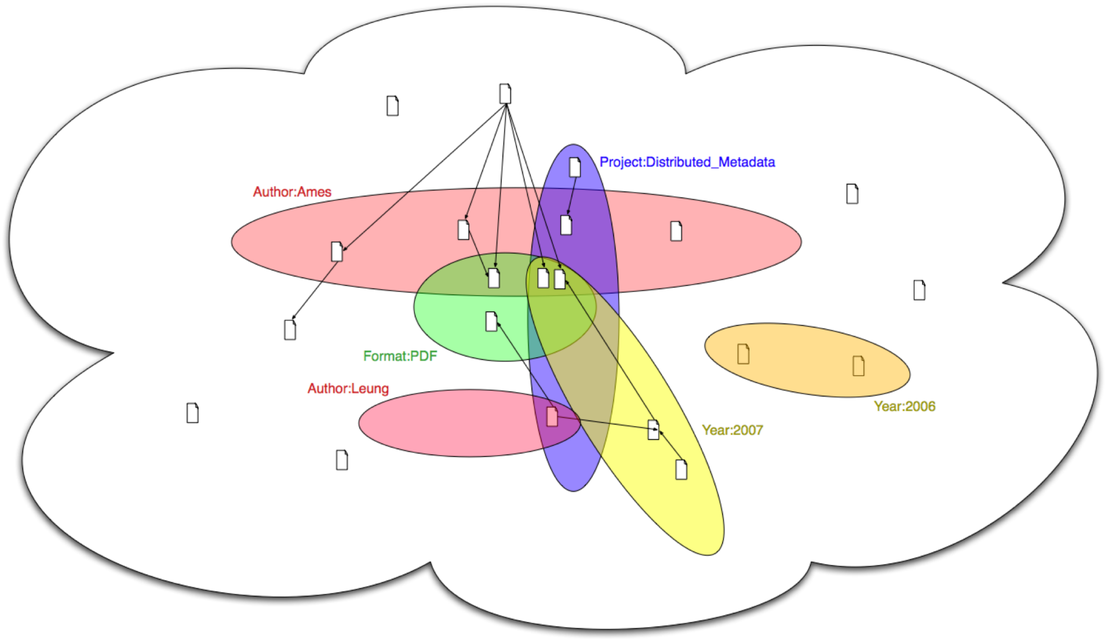
```
Ames_2005
Ames_2006
Leung_2007
Miller_2006
```

Refinement:

```
ls /!match:+keyword=file systems;+category=publications;
 +author=Miller;+year=2006|!listfacets:booktitle
```

Returns:

```
OSDI
StorageSS
```



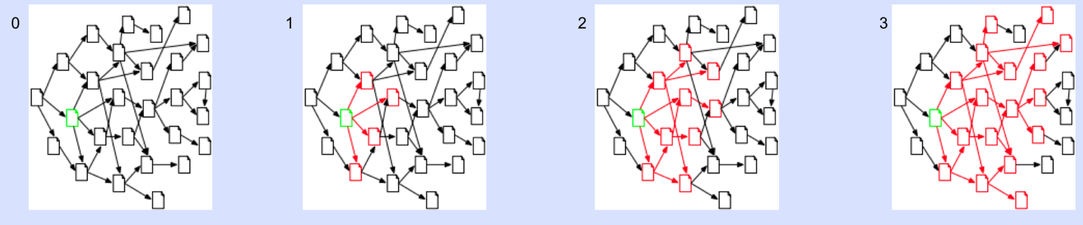
Location Based Search

- Queries use link distance to limit scope of results
- Requires a point of origin (file or directory)

Other uses of links for search

- Link traversals [Linking Files System]
- Matching attributes on ancestor and descendant files
- Improved ranking [PageRank, Connections]

Link Distance



Metadata Categorization

Type \ Source	Automatic: transduced	Manual: user input
Intrinsic	Content based: Full Text File specific structured data Context based: user, time Systems state and settings	Tagging Naming
External	Content: Citations, Include directives, Hyperlinks, Cut and Paste Context: Temporal Locality [Connections], Causality, Provenance [PASS]	User created relational links Placement within directories

Interface Characteristics for Facets

	Hierarchical	Flat
Matching Values	Browsing Trees	Pull down lists
Ranges of Values	Alphabetic browsing A B C D E F G … AB AC AD AG AI …	Numeric lists 1-100 101-200 201-300 301-400 401-458

Status and Open Issues

- Prototype Implementation
  - Use of FUSE for file system interface
  - Use existing library to build indices, such as Lucene or Indri
- Versioning with auto-create files - tracking version.
- Ranking
  - Choosing ranking techniques
  - Weighting files and metadata for better ranks
- Pathname Stability
  - At what granularity of caller should pathnames be bound?
- What experiments to run?
  - Latency benchmarks
  - Storage statistics
  - Anecdotal evaluation
  - User trials
  - Precision/Recall of search results
- Scalable index implementation and integration with CEPH