Layout-Aware Exhaustive Search

Aravindan Raghuveer, David H.C. Du

Introduction

- Exhaustive Search
  - Examine all objects in a storage system.
  - Expensive Operation

- Why Exhaustive Search ?
  - Fuzzy Queries:
    - Semantic gap in image, video → hard to annotate
    - Content-based (Query-by-Example)
    - Demonstrated in the Diamond project at Intel/CMU
  - Index Creation:
    - Not effective: Curse of dimensionality
    - Too expensive
    - Not always possible: Fuzzy queries

A "necessary evil" feature on all filesystems.

Technology Trends and Exhaustive Search

- Bits per unit area increasing rapidly
- I/O Bandwidth lagging behind
- Effect on exhaustive search:
  - 1 day to sequentially read 10TB*
  - 5 months with 8KB chunk random access !!
- Filesystem level exhaustive search: Recursive exploration of directories.
- With aged, fragmented filesystems:
  - At the disk: an Exhaustive search will look more like random access than sequential.

* Dr. Jim Gray's keynote from FAST'05:

Filesystem Applications and Exhaustive Search

- Exhaustive Search : Long running, I/O intensive task.
- Other filesystem applications running concurrently.
- Concurrent execution of both:
  - Performance Isolation:
    - Impact on response time of other applications should be minimal.
    - Impact on efficiency of exhaustive search should be as low as possible.

What this work is about ?

- A fresh look at Exhaustive Search
- As a first class service provided by the storage system.
- Close-to-sequential performance always
- Concurrent execution with other filesystem apps.
  - Without compromising extensively on response time and efficiency

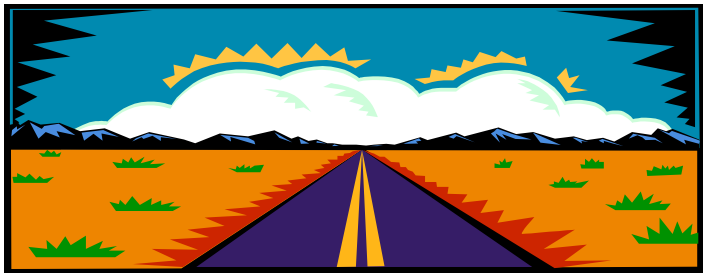
An Overview of proposed approach

- Layout aware:
  - Search order not based on logical filesystem view but physical on-disk organization.
  - As close to sequential performance as possible.
- Suspend-and-resume
  - On a real-time request to disk:
    - *Suspend* exhaustive search.
    - *Service* real-time request.
    - *Resume* exhaustive search.
  - Modify search order based on current disk head position.

Ingredients in the Solution

- Architecture:
  - Where to embed functionality: filesystem or smart object based disk ?
- Layout-Aware Search:
  - Planning the search ?
  - Metadata handling and placement?
    - Where are object extents located
    - List of objects already scanned
- Suspend-Resume:
  - Maintaining search progress metadata to avoid re-scanning [suspend]
  - Computing new search plan [resume]

Current Status



- Layout-Awareness:
  - 2 modes of layout-aware search.
  - Pre-planned and adhoc.
    - Pre-planned used when the disk stores a small number of objects.
    - Adhoc mode used when the disk is almost full.
    - Pre-planned and adhoc can be used at finer granularities (example: different modes on different areas of the disk)
  - Suspend-Resume:
    - Suspend: Search Metadata is distributed over the disk, close to the data.
    - Resume: Based on the remaining number of objects we either shift to the pre-planned or adhoc mode.