Patheon: Exascale File System Search for Scientific Computing

Joseph L. Naps, Mohamed F. Mokbel, and David H. C. Du

Department of Computer Science and Engineering
University of Minnesota, Minneapolis, MN, USA
{naps, mokbel, du}@cs.umn.edu

SSDBM 2011

Motivation

- At present, high performance computing generates petabytes of data, and billions of files.
- These files are stored in multi-tiered parallel file systems, under a hierarchical organizational structure.
- Current navigational methods for these hierarchies involve complicated file names coupled with use of POSIX commands such as *ls*, *grep*, and *find*.
- Currently, this organizational method is barely sufficient for data at current scales. This problem will only compound as storage system push to exascale and beyond.

Problem

- The high performance computing community needs a solution for searching files better than file names and POSIX commands.
- Current file organization methods are very deeply integrated.
- What is needed is a method for efficient file search within the existing storage environment.
- We propose Pantheon, a file system querying tool for large scale storage systems.
- Instead of using a DBMS as a makeshift metadata server, Pantheon pushes aspects of a DBMS, i.e. indexing and query optimization, directly into the file system.

Challenges

- Such a search tool is targeted at all scientists, not simply computer scientists. It must be easily accessible.
- Impact on the underlying system must be minimized. Checkpoint dumps already tax storage systems greatly, we don't want to make this worse.
- Storage systems are multi-tiered, typically consisting of fast platter drives, and slower tape drives. All of these storage media must be considered.
- Most proposed solutions for exascale storage system add additional tiers, typically flash, to the storage hierarchies.

Assumptions for Initial System

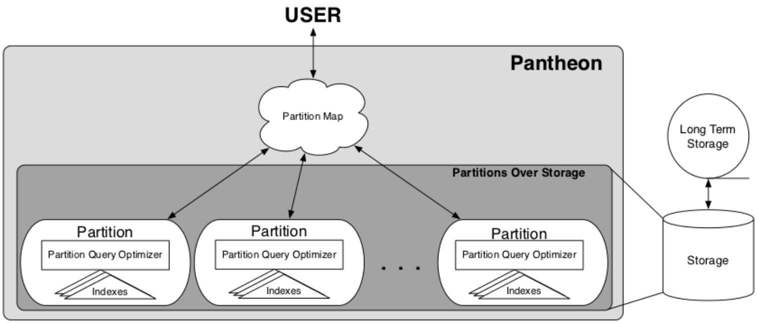
- To create ease of use, Panthoen is implemented using the File System in User Space(FUSE) library. This allows tight integration with the POSIX API.
- Indexing is not done in real time. Indexes are done in batches, typically during off times. This can be tweaked.
- There are only two tiers in the storage hierarchy, namely platter drives and tape drives.
- We are currently concerned with search over metadata.

Architecture

Pantheon integrates the following into the file system:

- Hierarchical Partitioning
- Indexing (B$^+$-Trees)
- Selectivity Based Query Optimization

Architecture



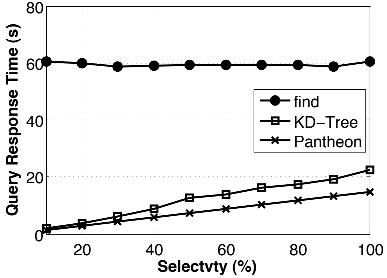
Experiments

- Non-distributed experiments done on a single computer running ext4 file system
- Distributed experiemts done on a NFS cluster

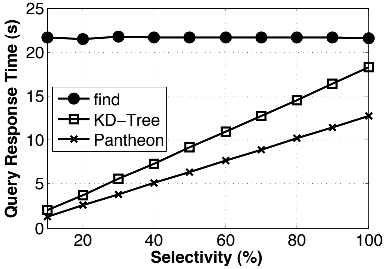
Experiments

Selectivity vs Query Response Time

Non-Distributed

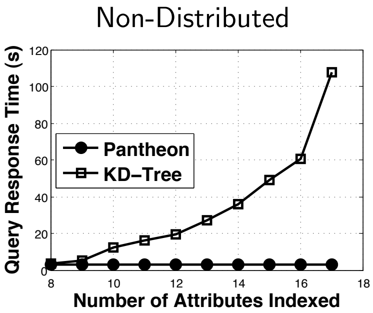


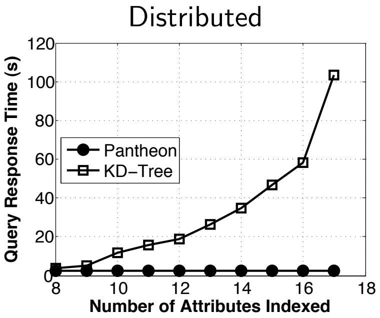
Distributed



Experiments

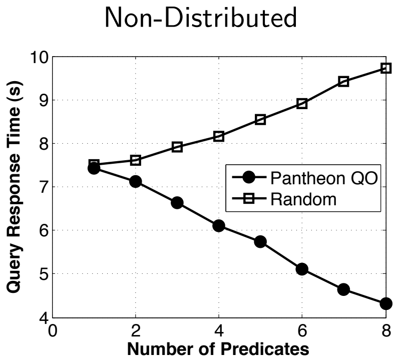
Number of Attributes vs Query Response Time

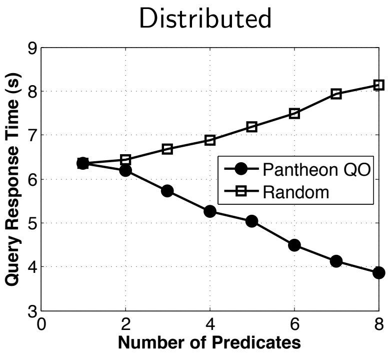




Experiments

Number of Query Predicates vs Query Response Time





Conclusion

- As storage system and their associated file systems continue to scale out, the need for organizational tools will continue to grow.
- Pantheon combines aspects of database management systems, such as indexing and query optimization, to allow for efficient search within a parallel file system environment.
- By implementing Panthoen in FUSE, this allows for scientists to quickly search for files with minimal impact on the underlying system.
- Currently, we are in the process of integrating this prototype discussed into the systems at Los Alamos National Laboratory.

Questions?

Thank You