Skyline-enabled Storage System

H. Howie Huang and Nan Zhang

George Washington University

I. Introduction

This is a vision statement for a novel idea of enabling *skyline* operators over a large-scale file system, in order to support efficient and automated management over large files systems. As file systems grows rapidly, so does the complexity of the underlying storage systems, which often contains a variety of hardware with distinct performance profiles - e.g., hard disks, Flash memory based devices, tapes, etc. Even for a particular type of hardware, e.g., Flash memory based devices, different devices may use different interfaces (PCI, SATA, etc.), employ different technologies (single-level cell/SLC, multi-level cell/MLC), and in turn perform differently on key measures such as latency, bandwidth, IOPS, etc. We believe that, to enable automated management in this environment, a key prerequisite is the ability to process *top-k queries* over the file system in an efficient manner. In general, each top-k query defines a unique scoring (preference) function over all files in the system, and is supposed to retrieve the k files[1] which have the highest scores. For example, if one is looking for files to be migrated from disk storage to tape archive, the corresponding top-k query should "prefer" files with larger sizes and are accessed less frequently. As such, it may define the scoring function (for files) as a weighted combination of the file size and the inverse of its access frequency. Similarly, give a top-k query answer with scoring function being the access frequency and a WHERE clause requiring the retrieved files to be read-only, one may find the desired files to be moved to a PCI-based flash storage for faster access.

A key challenge for processing top-k queries, however, is that, given the heterogeneity of storage devices in a large-scale system, one may have to support top-k queries with a wide variety of scoring functions (which are determined by the performance profiles of the corresponding storage devices). It is clearly inefficient to scan through all files for processing each top-k query. On the other hand, simply building an index for each file-meta-data attribute used by the scoring functions cannot solve the problem either, as many scoring functions involve more than one attributes of file meta data. For example, the above-mentioned top-k query for tape archive involves both size and access frequency of files in the system.

To support the efficient processing of top-k queries, this work-in-progress report proposes a novel paradigm of *Skyline-enabled Storage System* (S^3). The key idea here is to maintain a list of *skyline* files in the system that can support the efficient processing of *all* top-k queries, regardless of what the scoring function (i.e., the corresponding storage device performance profile) might be. To understand the definition of a skyline file, consider a set S of files, $S = \{f_1, f_2, ..., f_n\}$, a *skyline* operator will include a file $f_i \in S$ if and only if it is not *dominated* by any other file $f_j \in S$ - i.e., f_i has a higher "preference" than f_j on at least one meta-data attribute specified by the skyline operator - e.g., size, create/modify/access time, access frequency, permission, etc. While the preferential (partial) order are obvious for certain attributes - e.g., a larger file is in general more interesting to storage management than a smaller file; a read-only or write-only file is more interesting than a read-write file because the former two permissions suggest a well-disciplined access pattern of the file - there are certain attributes for which more than one preferential order is required. One can see from the above-mentioned examples that while least-frequently-accessed files are more interesting for tape archives, most-frequently-accessed files are more interesting for Flash memory based devices. For these attributes, we need to maintain *multiple* skyline operators to accommodate one meaningful order with each operator.

Fig. 1 presents an example of a 2-dimensional skyline on file size (larger is more interesting) and access frequency (larger is more interesting), where the skyline consists of four files, l, k, f, and e, for which there exists no file that is better in both dimensions. For example, file e is better than file d because file e is bigger and accessed more frequently.

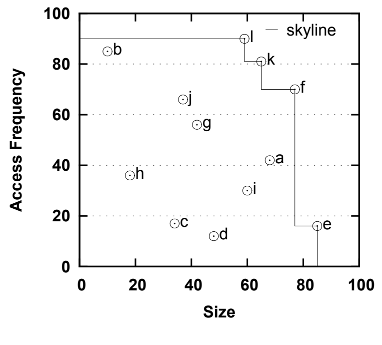

Fig. 1. Skyline operator on file size and access frequency

Borrowing the idea of the skyline operator [2], [4] from the database community, we extend the concept to automated management of large file and storage systems, and in this work, we use file management over heterogeneous storage devices as an example. Note that S^3 is orthogonal to existing research [3], [1], [5] in the area of storage QoS management, which can potentially be improved by the skyline-based techniques.

[1]Note that here the specification of k is loosely defined - e.g., instead of asking for the k files with the highest scores, one may ask for the highest-scored files to fill a given space, say 2GB.

II. Approach

Fig. 2 shows the S^3 architecture, where a database is used to maintain the metadata of the system, and the sensors and actuators are utilized to perform management tasks. As the sensors take periodical measurements, the actuators, following a set of predefined rules, will instrument the system to carry out the actions. The main idea behind S^3 is that a good set of candidate components returned by the skyline operator will enable better precision and greater impact, leading to more efficient and effective storage management. To reuse the previous example, in a two-level heterogeneous storage system that consists of multiple solid-state drives (SSDs) and a large number of hard drives, the actuators may distribute the skyline files to different locations - a simple method is to move files b and h on the SSDs, and file c and d on the hard drives. A more sophisticated strategy can be devised with the top-k skyline files. The technical challenges for designing and developing S^3 are 1) to identify the dimensions of interest, given a management task; 2) to quickly determine, based on the rules, the skyline query to be processed; and 3) to process the query quickly, preferably online without reading the whole metadata database.

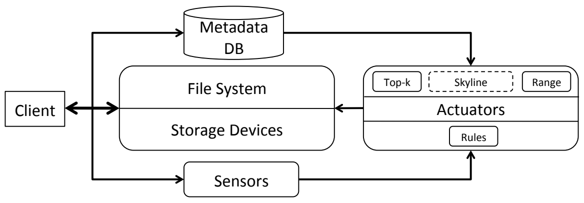

Fig. 2. Skyline-enable storage system architecture

III. Status

We are in the process of developing the prototype system. For evaluation, we plan to utilize several public file system traces, e.g., NTFS, NFS, and Plan 9 file system.

References

[1] Anderson, E., Hobbs, M., Keeton, K., Spence, S., Uysal, M., and Veitch, A. Hippodrome: Running circles around storage administration. In *Proceedings of the 1st USENIX Conference on File and Storage Technologies* (Berkeley, CA, USA, 2002), FAST '02, USENIX Association.

[2] Börzsönyi, S., Kossmann, D., and Stocker, K. The skyline operator. In *Proceedings of the 17th International Conference on Data Engineering* (Washington, DC, USA, 2001), IEEE Computer Society, pp. 421–430.

[3] Ganger, G. R., Strunk, J. D., and Klosterman, A. J. Self-* storage: Brick-based storage with automated administration. Tech. rep., Technical Report CMU-CS-03-178, Carnegie Mellon University, 2003.

[4] Papadias, D., Tao, Y., Fu, G., and Seeger, B. Progressive skyline computation in database systems. *ACM Trans. Database Syst. 30* (March 2005), 41–82.

[5] Yin, L. Automatic action advisor for storage system performance management.