Just-In-Time Analytics on Large File Systems

H. Howie Huang, Nan Zhang, Wei Wang — George Washington University
Gautam Das — University of Texas at Arlington
Alexander S. Szalay — Johns Hopkins University

FAST 2011
February 17, 2011

Outline

- Introduction
- Aggregate Query Processing
- Evaluation
- Related Work
- Conclusion

Motivations

- Large file systems are common
- Users are interested in performing *Just-In-Time* analytics
  - Must be completed within a short amount of time
  - Has no prior knowledge of file system being analyzed
- Border patrol
  - E.g., check a traveler's laptop for pirated movies and software ISO

Data Analytics

- Aggregate Query
  - E.g., "What is the total size of various types of documents?"
  - SELECT SUM(file.size) FROM filesystem
    WHERE file.extension IN { 'txt', 'doc'};
    // AVG and COUNT are also in this category

- Top-k Query
  - E.g., "Which are the 100 largest files that belong to John?"
  - SELECT TOP 100 files FROM filesystem
    WHERE file.owner = 'John' ORDER BY file.size DESC;

Current Approaches

- Scan file system for each query
  - E.g., find command in Linux
  - Inefficient
  - Growing gap between storage performance and capacity

- Utilize pre-built indexes that are regularly updated
  - E.g., Google Desktop and Beagle
  - Undesirable when the metadata indexes are not available
  - The queries are scarcely needed

At a Glance

- Tradeoff between query accuracy and cost
  - Provide approximate (i.e., statistically accurate) answers that reside close from the precise answer
- Glance, a just-in-time query processing system
  - Produce answers based on a small number of samples (files or folders)
- File system agnostic
  - Works seamlessly with the tree structure of the system
  - Can be applied instantly over any new file system
- Remove the need of disk crawling and index building
  - Without a priori knowledge or pre-processing of file system

Glance Architecture

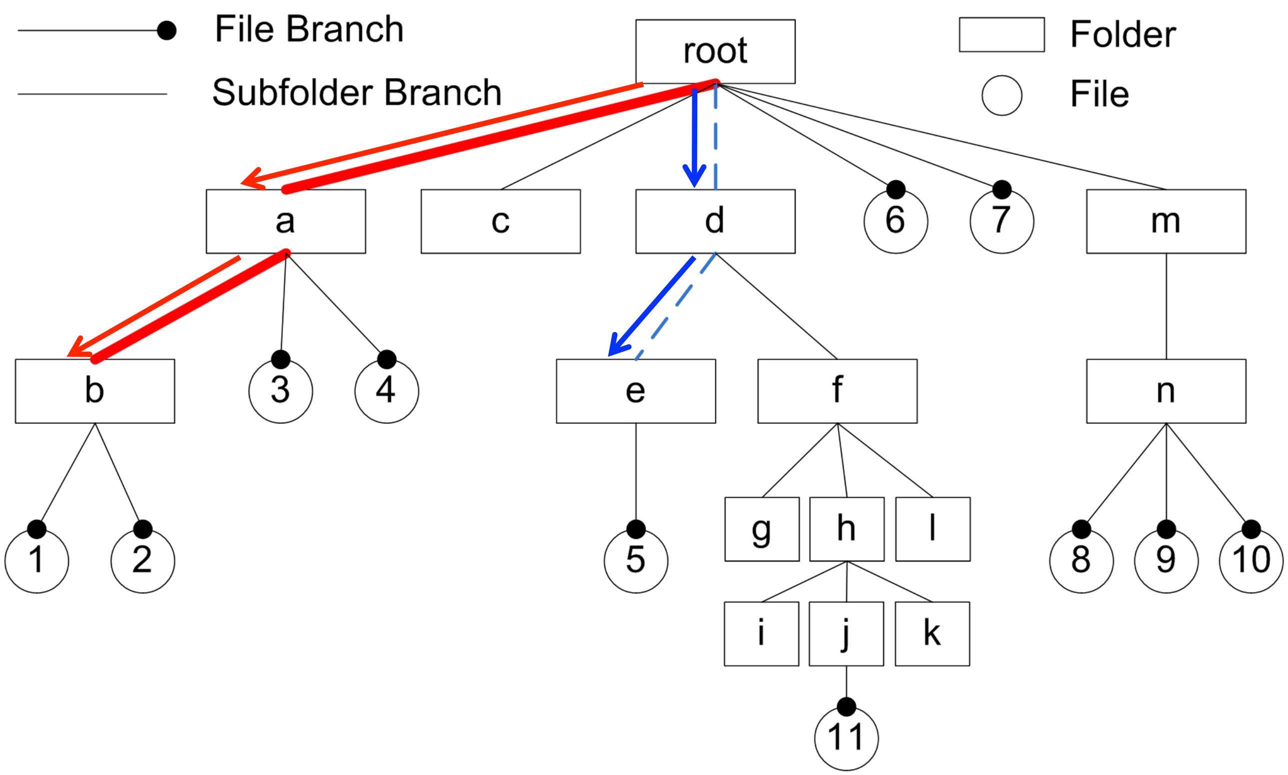
- Consists of two algorithms
- FS_Agg for approximate processing of aggregate queries
  - FS_Agg_Basic: a random descent technique for unbiased aggregate estimations
  - Two enhancements to reduce the error and performance overhead
- FS_TopK for approximate processing of top-k queries
  - A pruning-based technique

FS_Agg_Basic - Random Descent

Estimate the COUNT of all files in the system

$$\tilde{n} = \sum_{i=0}^{h} \left(f_i \cdot \prod_{j=0}^{i-1} s_j \right)$$

f_i: number of files
s_j: number of subfolders



- Solid

$\langle f0, f1, f2 \rangle = \langle 2,2,2 \rangle$ and $\langle s0,s1,s2 \rangle = \langle 4,1,0 \rangle$ Estimation of 2 + 8 + 8 = 18

- Dotted

$\langle f0,f1,f2 \rangle = \langle 2,0,1 \rangle$ and $\langle s0,s1,s2 \rangle = \langle 4,2,0 \rangle$ Estimation of 2 + 0 + 8 = 10

Unbiased Estimation

- The estimation produced by each random descent process is completely unbiased

- The expected value of the estimation is exactly equal to the total number of files in the system

$$E\left(\frac{|v_{i-1}|}{p(v_{i-1})}\right) = \sum_{v_{i-1}} \left(p(v_{i-1}) \cdot \frac{|v_{i-1}|}{p(v_{i-1})}\right) = F_i.$$

  – $|v_{i-1}|$: the number of (i-level) files in the folder v_{i-1}
  – p(v-1): the probability for v_{i-1} to be reached in the random descent

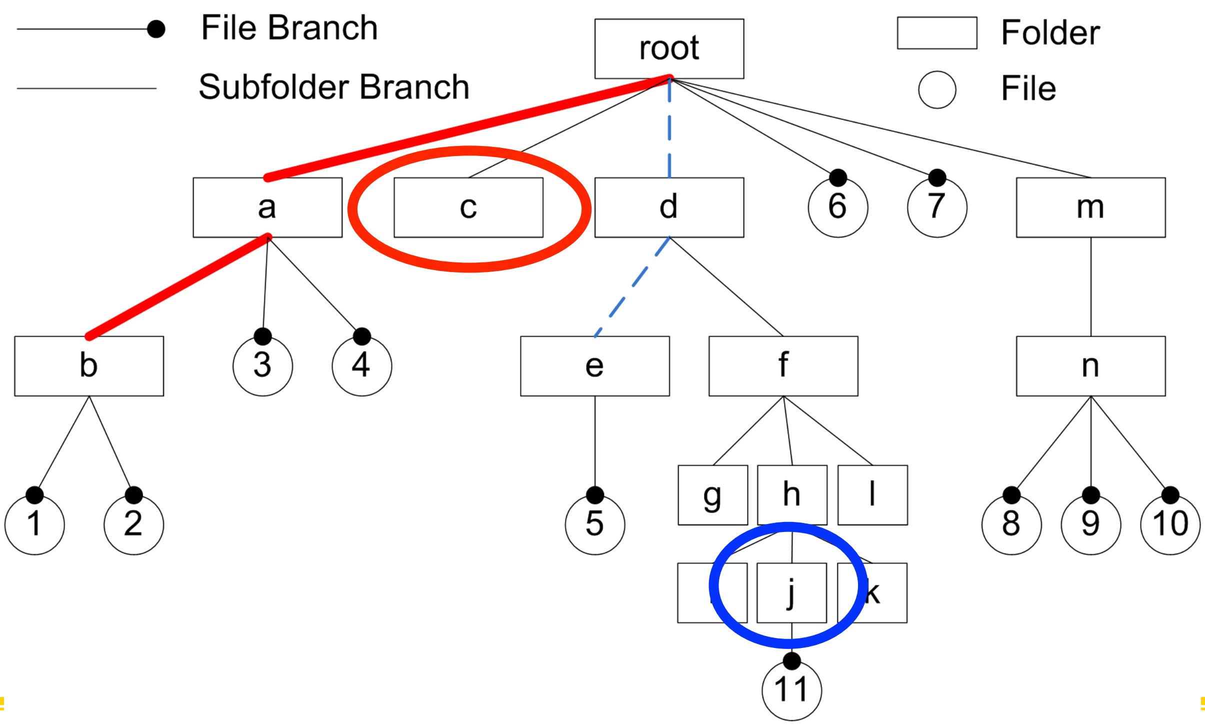
$$p(v_{i-1}) = \prod_{j=0}^{i-2} \frac{1}{s_j(v_{i-1})}$$

Processing of Aggregate Queries

- SUM: similar to COUNT, but set f_i as the SUM of a meta attribute over all files
- AVG: compute as SUM/COUNT
  - Such an estimation is no longer unbiased
- Selection conditions: only evaluate f_i over the files that satisfy the conditions

Disadvantages of Basic Algorithm

- Two types of folders may lead to extremely high estimation variance
  - High-level leaf-folders, i.e., "shallow" folders with no subfolders
  - Deep-level folders which reside at much lower levels



FS_Agg Improvements

- High-level crawling for level *i* and above
  - Eliminate the negative impact of high-level leaf-folders on estimation variance
- Breath-first descent instead of depth-first
  - At any level of the tree, randomly selects a set of folders to access at the next level
  - Significantly increase the selection probability for a deep folder

Evaluation Setup

- A prototype in C code for Linux/ext3
  - FS_Agg has three parameters
    - *h* - the number of (highest) levels for crawling
    - P_{sel} - the selection probability
    - S_{min}- the minimum number of selections
    - p_{sel} and s_{min} determine how many subfolders to be selected
  - FS_TopK has a parameter
    - γ - the (estimation) enlargement ratio

- Hardware
  - Intel Core 2 Duo processor, 4GB RAM, and 1TB Samsung 7200RPM hard drive

- Report the average of five runs

Test File Systems

- Windows file systems from Microsoft trace
  - m100K (largest with <100K), m1M, m10M (largest in the trace)
  - m100M (largest 33 systems), and m1B
- Plan 9 (Unix-like) systems from Bell Lab (~2M files)
- NFS from Harvard trace (2.3M files)
- Synthetic file systems generated by Impressions
  - E.g., i10K, i100K, i1M

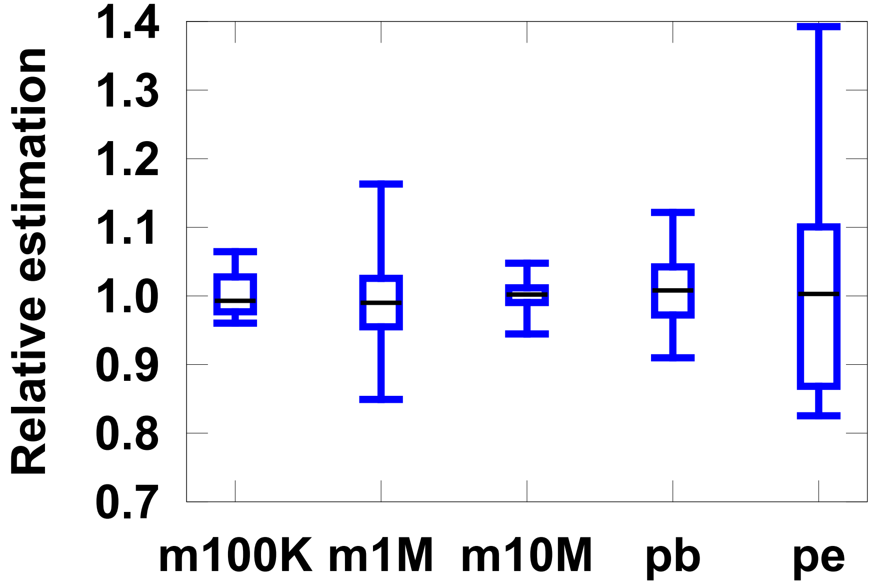
- Welcome large real-world file systems

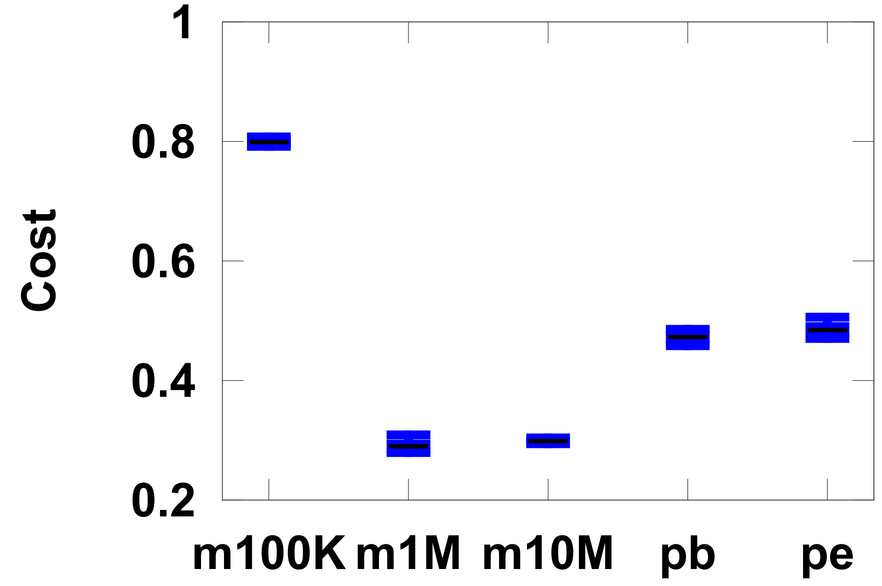
Metrics

- Query accuracy
  - For aggregate queries, the relative error of the approximate answer *apx* compared with the precise one *ans* - i.e., $|apx - ans| / |ans|$
  - For top-k queries, the percentage of items that are common in the approximate and precise top-k lists

- Query efficiency
  - Query time, i.e., the runtime of query processing
  - Query cost, i.e., the ratio of the number of directories visited by Glance to that of crawling the file system

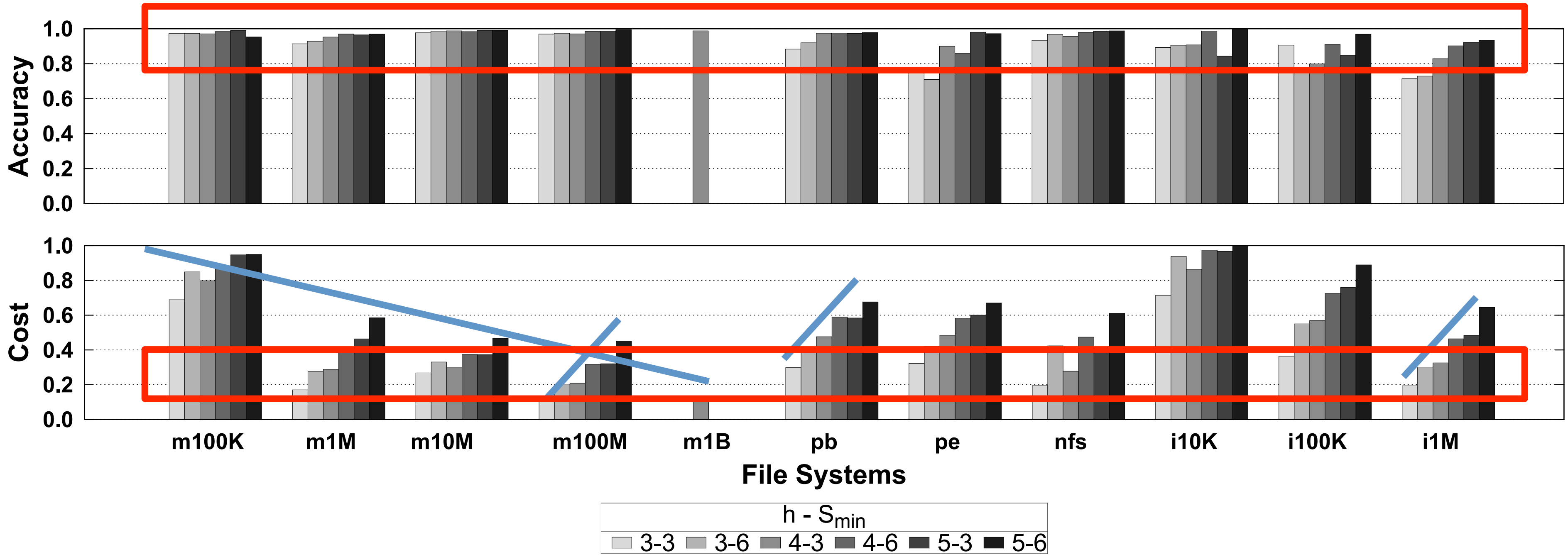
Aggregate Queries





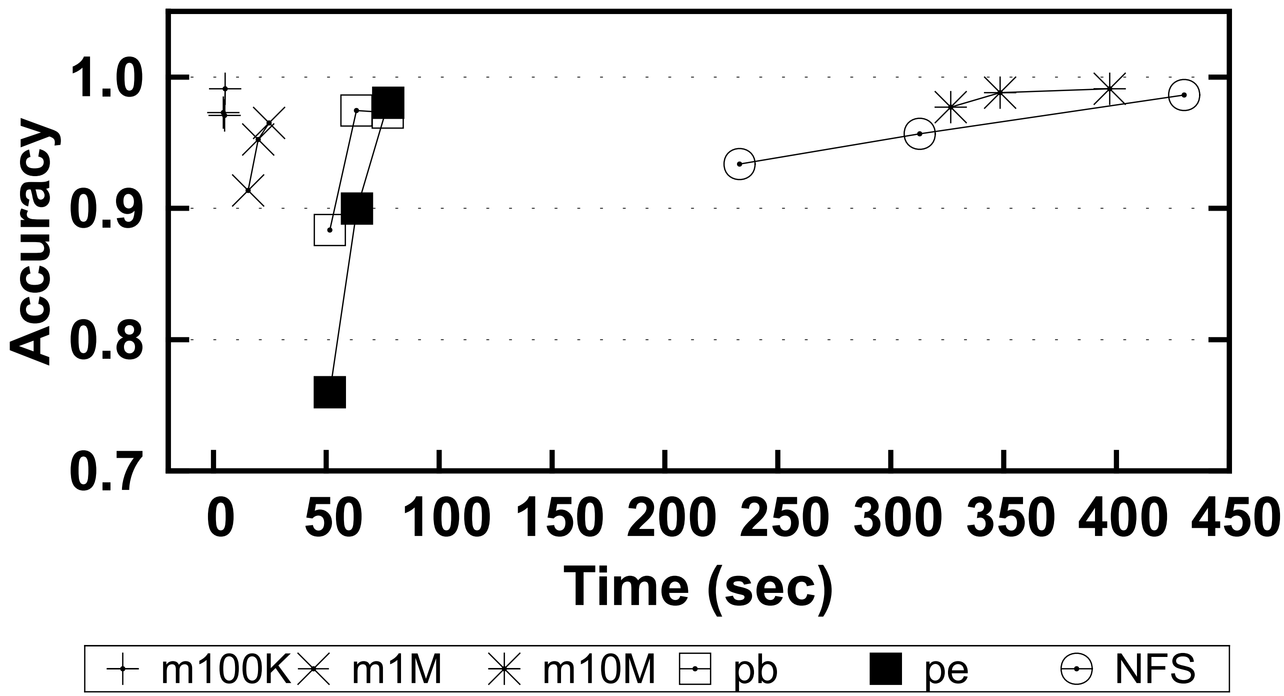
- Glance consistently generates accurate query answers
  - E.g., for m10M, sampling 30% of directories produces an answer with 2% average error

Aggregate Queries - Accuracy and Costs



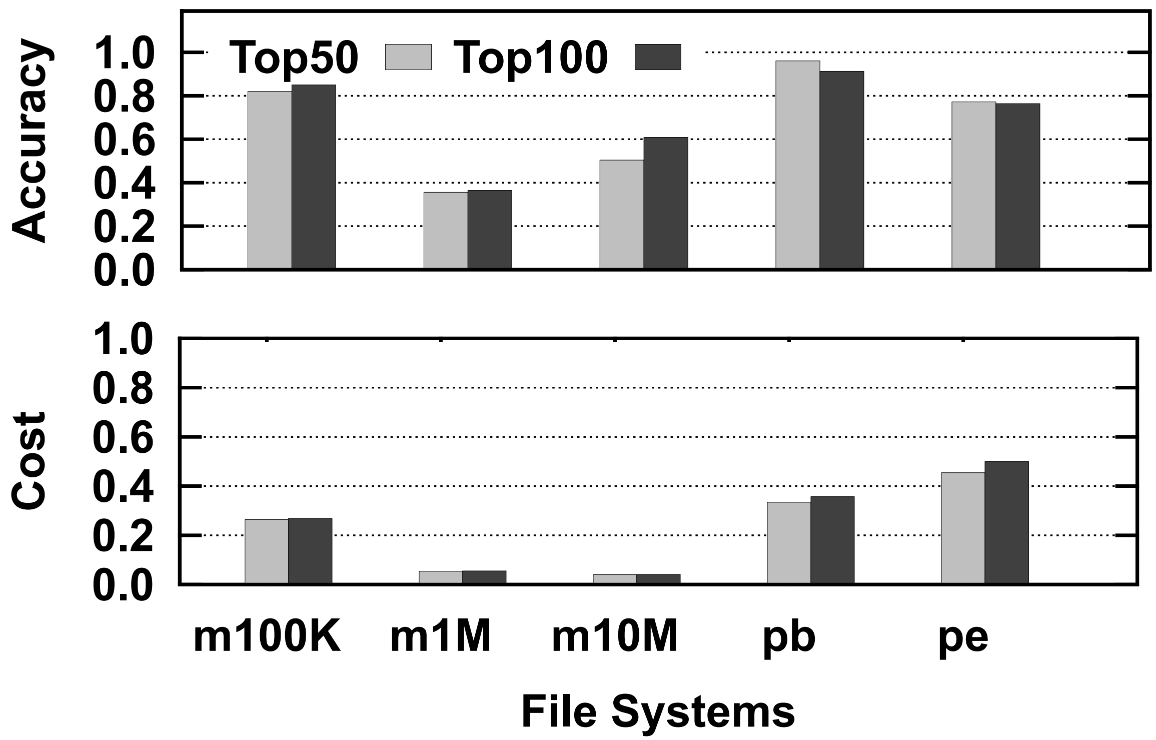
- For all file systems, Glance produces the answers with <10% relative error
- The performance of Glance is independent of the type of the file system
  - Achieves over 90% accuracy for NFS, Plan 9, and NTFS (m10M to m1B)
  - The cost ranges from less than 12% of crawling for large systems with 1B files and 80% for the small 100K system
- The algorithm scales well to large file systems

Aggregate Query Runtimes



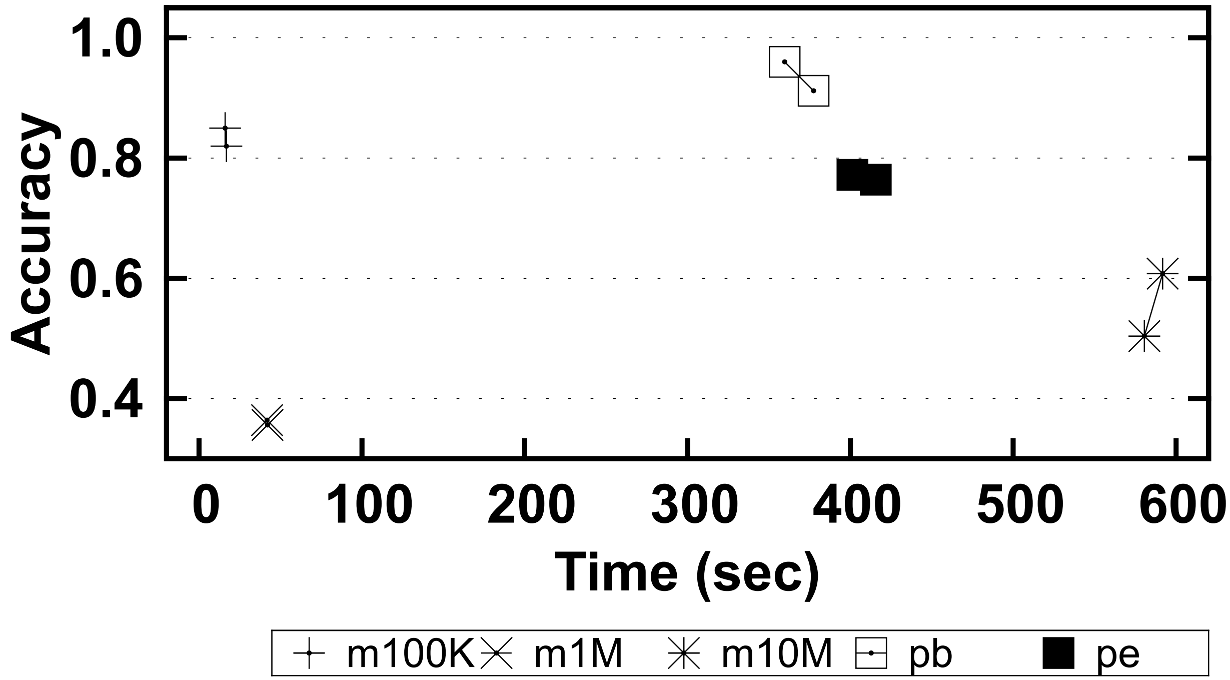
- For different values of h from 3 to 5, query runs slightly longer but the accuracy improves
- The absolute runtime depends heavily on the size of the file system
  - A few seconds for m100K, several minutes for nfs (2.3M files), and 1.2 hours for m100M (not shown in the figure)

Top-k Queries on File Size



- For all but one case (m1M), Glance is capable of locating at least 50% of all top-k files (for pb, more than 95% are located)
- The cost is as little as 4% of crawling (for m10M)

Top-k Query Runtime



- The runtime is correlated to the size of the file system
  - The first point of each line stands for top-50 and the second for top-100
  - The queries take only a few seconds for small file systems, and up to ten minutes for large systems (e.g., m10M)

Related Work

- Metadata query on file systems
  - Spyglass [Leung et al 2009]
  - SmartStore [Hua et al 2009]
  - Utilize multi-dimensional structures (e.g., K-D trees and R-trees) to build indexes upon subtree partitions or semantic groups

- Database sampling and query processing
  - Random sampling [Cochran 1977]
  - Sampling of hidden web databases [Dasgupta et al 2010]

Future Directions

- Glance is not yet an any-time algorithm and cannot be stopped in the middle of the execution
  – Be predictive about the run-time and self-adjust the work flow based on the real-time requirements

- Currently employs a ”static” strategy over file systems and queries
  – Leverage the results from the previous queries to significantly expedite the future ones
  – Utilize the semantic knowledge of a file system

Summary

- Just-in-time analytics over a large-scale file system through its tree- or DAG-like structure
- A random descent technique to produce unbiased estimations for SUM and COUNT queries and accurate estimations for other aggregate queries
- A pruning-based technique for the approximate processing of top-k queries
- A comprehensive set of experiments that demonstrate the effectiveness of our approach over real-world file systems

OCI, CISE