Big Data Provenance: Challenges and Implications for Benchmarking

Boris Glavic

Illinois Institute of Technology
10 W 31st Street, Chicago, IL 60615, USA
glavic@iit.edu

Abstract. Data Provenance is information about the origin and creation process of data. Such information is useful for debugging data and transformations, auditing, evaluating the quality of and trust in data, modelling authenticity, and implementing access control for derived data. Provenance has been studied by the database, workflow, and distributed systems communities, but provenance for Big Data - let us call it *Big Provenance* - is a largely unexplored field. This paper reviews existing approaches for large-scale distributed provenance and discusses potential challenges for Big Data benchmarks that aim to incorporate provenance data/management. Furthermore, we will examine how Big Data benchmarking could benefit from different types of provenance information. We argue that provenance can be used for identifying and analyzing performance bottlenecks, to compute performance metrics, and to test an application's ability to exploit commonalities in data (provenance). In this discussion we assume the (unlikely) existence of *GOD*, an efficient and generic Big Provenance management solution. We are well aware that such a system is currently far from being reality. However, the main purpose of this paper is to analyze the impact of data provenance on Big Data benchmarking and not to propose new solutions for Big Provenance management.

Keywords: Big Data, Benchmarking, Data Provenance

1 Data Provenance for Big Data

Provenance information explains the creation process and origin of data by recording which transformations were responsible in creating a certain piece of data (a so-called *data item*) and from which data items a given data item is derived from. We refer to the first type as *transformation* provenance and the second type as *data provenance*. Additional meta-data such as the execution environment of a transformation (the operating system, library versions, the node that executed a transformation, ...) is sometimes also considered as provenance. A standard approach to classify provenance information is granularity. *Coarse-grained* provenance handles transformations as black-boxes: provenance records which data items were used as the inputs and outputs of a given transformation.

Usually this information is represented in a graph structure by linking data items or collections to the transformations that produced or consumed them. *Fine-grained* provenance provides insights about the data-flow inside a transformation, i.e., it exposes the processing logic of a transformation by modelling which parts of the inputs were necessary/sufficient/important in deriving a specific output data item. Often provenance information is modelled as *annotations* on the data. Using this approach, provenance management amounts to efficiently storing these annotations, and *propagating* and modifying them according to the behaviour of transformations. This technique is used by relational provenance management systems such as Orchestra [6], DBNotes [2], or Perm [4] to name a few. Provenance has found applications in debugging data (e.g., to trace back an erroneous data item to the sources from which it was derived), trust (e.g., by combining trust scores for the data in a data item's provenance), probabilistic data (the probability of a query result can be computed from the probabilities of the data items in its provenance [12, 8]), and security (e.g., enforce access-control to a query result based on the access-control policies of items in its provenance).

Providing a complete overview of provenance research is beyond the scope of this paper. Here we only present a few approaches that are related to Big Data research or interesting to the discussion in the remainder of this paper. Provenance research from the database community is largely focused on fine-grained provenance, but has mostly ignored distributed aspects of provenance management. Coarse-grained provenance has been studied intensively by the workflow community. Even though scientific workflow systems tend to be distributed, many approaches for tracking provenance still rely on centralized storage.

Recently, Ikeda et al. [7] introduced an approach for tracking the provenance of workflows modelled as MapReduce jobs. The authors introduce a general fine-grained model for the provenance of map and reduce functions. Provenance is stored in HDFS by annotating each key-value pair with its provenance (appended to the value).[1] The approach provides wrappers for the map and reduce functions that call the user-provided versions of these functions. These wrappers strip off the provenance information from the value before passing it to the user function and attach provenance to the output based on the input's provenance and the semantics of the mapper and reducer functions. Another approach for MapReduce Provenance adapts database provenance techniques to compute the provenance of workflows expressed in a subset of the Pig language [1] corresponding to relational algebra. Similarly, the approach from [14] adapts a database provenance model for a distributed datalog engine.

Malik et al. [9] present an approach for recording provenance in a distributed environment. Provenance is captured at the granularity of processes and file versions by intercepting system calls to detect dependencies between processes, files, and network connections. Each node stores the parts of a provenance graph corresponding to its local processing and maintains links to the provenance graphs of other nodes. To support querying of the distributively stored provenance

[1] To be precise, there is an additional indirection in storing the provenance of a reducer output. See [7] for details.

graph across node boundaries, the nodes exchange summaries of their provenance graphs in the form of bloom filters. Muniswamy-Reddy et al. [10] introduce protocols for collecting provenance in a cloud environment. Each node runs *PASS* (provenance aware storage system), a system that collects provenance at the file level by intercepting system calls. Provenance is stored using cloud storage services like S3 and SimpleDB. One major concern in this work is how to guarantee that provenance and data is coupled consistently when the underlying storage services only provide eventual consistency. Seltzer et al. [11] apply the PASS approach to extend the Xen Hypervisor to collect provenance information by monitoring the system calls of a virtual machine.

In summary, existing approaches address some aspects of Big Provenance such as distributed storage, low-level operating system provenance, or fine-grained provenance for Big Data languages that are can be mapped to relational query languages (for which provenance is well-understood). However, Big Provenance still remains a challenging problem for the following reasons:

- Big data is often characterized as highly heterogeneous and users expect to be able to run ad-hoc analytics without having to define extensive types of meta-data like, e.g., a schema. This makes it hard to define a common structure to model the provenance of such data sets - especially for fine-grained provenance. For example, if we do not know how data entries are organized in a file, we cannot reference individual entries from the file in the provenance.
- Big Data analytics systems tend to make the distribution of data and processing transparent to provide simpler programming models. This enables analysts with little knowledge about distributed systems to run large scale analytics. However, if the purpose of collecting provenance is to analyze the performance of a Big Data analytics system, then we would like to include information about data and processing locations in the provenance of a data item. For instance, this type of information could be used to check whether a data item was shipped to a large number of distinct locations during processing.
- A data item may have been produced by transformations that are executed using different Big Data analytics and storage solutions. The provenance of such a data item will reference data and transformations from each system that was used to create the data item. Since shipping all data items and process information in the provenance of a data item together with the data item will result in prohibitively large amounts of information to be transferred between systems, a query solution for Big Provenance has to interact with more than one system and understand several storage formats to be able to evaluate queries over provenance information.

2 Implications for Benchmarking

A comprehensive benchmark should precisely define data sets, the workload, and performance measures. Acting upon the results of a benchmark to improve the

performance of a system usually requires additional monitoring and profiling to identify the causes of poor performance. In the following we discuss the impact of provenance on each of these aspects. Provenance is a compelling use-case to explore in benchmarks workloads and data sets, because it has some unique properties that distinguishes it from other potential benchmarking workloads. Besides from being an interesting and challenging use-case for workload design, Big Provenance could also be used as a supporting technology for benchmarks. For sake of the discussion we assume the existence of *GOD*, a hypothetical system that tracks whatever provenance we image for any type of distributed Big Data processing and storage. Furthermore, *GOD* allows efficient query access to provenance, solving all our querying needs. Assuming the existence of *GOD* will allow us to focus on the impact of Big Provenance on benchmarking without being distracted by the question whether the postulated Big Provenance functionality can actually be realized by current systems.

Using Provenance to Measure Exploitation of Data Commonality Provenance can be orders of magnitude larger than the data for which provenance is collected. This is not surprising, because provenance models the relationship between inputs and outputs of a transformation and, thus, can be quadratic in the number of inputs and outputs. This problem is aggravated for fine-grained provenance (e.g., when tracking the provenance of each data entry in a file instead of handling the file as a single large data item) or when each data item is the result of a sequence or DAG of transformations. However, the provenance information of two data items often overlaps to a large extend [3]. Thus, a benchmark that includes workloads running over provenance data can be used to test how well a system exploits commonality in the data it is processing and producing.

Using Provenance to Generate Large Datasets and Workloads It is relatively easy to generate large data sets and computationally expensive workloads by collecting the provenance of a small set of transformations at fine granularity, because of the potential size of provenance relative to the size of the data it is describing. For example, consider a build process for a piece of software using a parallel invocation of `make`. During the build temporary files are created and deleted as the result of compilation. The build process executes tests on the compiled software which results in additional files being created and destroyed. Assume we execute the build using an approach that collects provenance for files by intercepting system calls (e.g., [11]). The resulting provenance graph will be large. Similarly, consider a workload that applies an image processing algorithm to an input file. We could use a provenance approach that instruments the program to record data dependencies as provenance information [13]. The results would be provenance at pixel granularity and for individual variable assignments of the program. Thus, the amount of recorded provenance would be gigantic. These examples demonstrate that by computing the provenance of relatively small and simple workloads we can generate large benchmark datasets. In addition, provenance collection could be directly used as a benchmark workload.

Provenance-based Performance Measures One major goal for Big Data systems is robustness of performance and scalability [5]. Provenance can be used

to record execution times and data movement at a very fine granularity. This information could be used to compute measures for individual jobs in a workload and to compute new performance metrics using provenance information. For example, assume a system performs reasonably well on a complex benchmark workload, but actually one job was taking up most of the available resources while most of the jobs performed better than expected. The poor performance is hidden in the overall well performance, but may become problematic if we change the input size of the poor-performing job. Assume we use *GOD* to record the execution times for all tasks of a job and record the movement of data items between nodes as provenance. Benchmarks could exploit such information to define new data-centric performance measures that take robustness of performance into account. For example, the benchmark could require the execution of several workloads with overlapping sets of jobs and define the deviation of execution times and data movements of a job over all workload executions as a performance measure.

Provenance for Monitoring and Profiling Big Data benchmarks should consist of complex and diverse workloads. However, complex workloads complicate understanding the performance characteristics of a benchmarked system. Provenance could be used to complement monitoring solutions for Big Data systems. Assume we again use *GOD* to record resource utilization of transformations and location changes of data items as the provenance of a data item. This information constitues data-centric profiling information. As the data-centric equivalent of profiling execution times of functions in, e.g., a Java program, we could compute the amount of resources that were spend on producing a data item from this type of provenance information. Coupling of data with performance measurements of the transformations that created it enables novel types of profiling analysis. For example, to identify redundant computations, we simply have to check whether the provenance of two data items contains the same data item produced by different transformations (possibly executed at different locations). Furthermore, if an intermediate result is not in the fine-grained data provenance of any final result of a task, then it was unnecessary to produce this intermediate result at all.

3 Conclusions

This paper discusses challenges for Big Provenance and outlines several strategies for how provenance could be used in benchmarking Big Data systems. Furthermore, we argue that provenance information may aide developers in identifying bottlenecks in the performance, scalability, and robustness of their systems. Provenance can be used for computing fine-grained data-centric performance metrics, for measuring if a system is able to exploit data commonalities, and for profiling systems. However, such applications rely on further research to develop models for Big Provenance and making provenance management scale to Big Data dimensions.

4 Bio

Boris Glavic is an assistant professor in computer science at Illinois Institute of Technology (IIT). He received his Ph.D. from the University of Zurich, Switzerland in 2010. Before joining IIT in 2012, he held a Post doctoral fellowship at the University of Toronto working at Prof. Miller's group. Boris's research is focused on database systems, in particular, data provenance and its application in information integration. Other topics he explored in his research are data stream processing, declarative request scheduling and concurrency control, data mining, and applications of data management technologies in interdisciplinary contexts. He enjoys to do systems research based on sound theoretical foundations. One of his main contributions is Perm, a relational provenance management system that uses query rewrite techniques to generate provenance information for SQL queries.

References

1. Amsterdamer, Y., Davidson, S., Deutch, D., Milo, T., Stoyanovich, J., Tannen, V.: Putting Lipstick on Pig: Enabling Database-style Workflow Provenance. PVLDB 5(4), 346–357 (2011)
2. Bhagwat, D., Chiticariu, L., Tan, W.C., Vijayvargiya, G.: An Annotation Management System for Relational Databases. VLDB Journal 14(4), 373–396 (2005)
3. Chapman, A., Jagadish, H.V., Ramanan, P.: Efficient Provenance Storage. In: SIGMOD. pp. 993–1006 (2008)
4. Glavic, B., Alonso, G.: Perm: Processing Provenance and Data on the same Data Model through Query Rewriting. In: ICDE. pp. 174–185 (2009)
5. Graefe, G.: Benchmarking robust performance. In: 1st Workshop on Big Data Benchmarking (WBDB) (2012)
6. Green, T., Karvounarakis, G., Tannen, Z.: Provenance in ORCHESTRA. IEEE Data Eng. Bull. 33(3), 9–16 (2010)
7. Ikeda, R., Park, H., Widom, J.: Provenance for generalized map and reduce workflows. In: CIDR. pp. 273–283 (2011)
8. Karvounarakis, G., Green, T.: Semiring-annotated data: Queries and provenance. SIGMOD Record 41(3), 5–14 (2012)
9. Malik, T., Nistor, L., Gehani, A.: Tracking and sketching distributed data provenance. In: eScience. pp. 190–197 (2010)
10. Muniswamy-Reddy, K., Macko, P., Seltzer, M.: Provenance for the cloud. In: FAST. pp. 197–210 (2010)
11. Seltzer, M., Macko, P., Chiarini, M.: Collecting provenance via the xen hypervisor. In: TaPP (2011)
12. Widom, J.: Trio: A System for Managing Data, Uncertainty, and Lineage. Managing and Mining Uncertain Data pp. 1–35 (2008)
13. Zhang, M., Zhang, X., Zhang, X., Prabhakar, S.: Tracing Lineage beyond Relational Operators. In: VLDB. pp. 1116–1127 (2007)
14. Zhou, W., Mapara, S., Ren, Y., Li, Y., Haeberlen, A., Ives, Z., Loo, B., Sherr, M.: Distributed time-aware provenance. PVLDB 6(2) (2012)