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 - which we refer to as *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 a system's ability to exploit commonalities in data and processing.

Keywords: Big Data, Benchmarking, Data Provenance

1 Introduction

Provenance for Big Data applications is a relatively new topic that has not received much attention so far. A recent community white paper [5] on the challenges and opportunities of Big Data has identified provenance tracking as a major requirement for Big Data applications. Thus, provenance should be included in benchmarks targeting Big Data. We first give a brief introduction to provenance and review the current state-of-the-art of provenance for Big Data systems and applications. Afterwards, we discuss the implications of provenance for benchmarking. In particular, we try to answer the following questions: How to generate workloads with provenance aspects? What are the differences between provenance workloads and the workloads currently used in benchmarking? Finally, we argue that provenance information can be used as a supporting technology for Big Data benchmarking (for data generation and to allow new types of measurements) and profiling (enable data-centric monitoring).

2 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. 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: it records which data items are 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. For example, consider a transformation that counts the frequency of words in a collection of documents and outputs pairs of words and their count. If we consider documents as atomic units of data, then a coarse-grained approach would consider all input documents as the provenance of one output pair (w, c). In contrast, the fine-grained provenance of a pair (w, c) would only consist of the documents containing the word w.

Provenance has found applications in debugging data (e.g., to trace an erroneous data item back 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 [13, 8]), and security (e.g., enforce access-control to a query result based on access-control policies for items in its provenance [11]). All these use-cases translate to the Big Data domain. Even more, we argue that provenance is critical for applications with typical Big Data characteristics (*volume*, *velocity*, and *variety*)[1]. A standard approach to deal with the velocity (and to a lesser degree also the variety) aspect of Big Data is to apply data cleaning and integration steps in a pay-as-you-go fashion. This has the advantage of increasing the timeliness of data, but in comparison with the traditional ETL approach of data warehousing comes at the cost of less precise and less well-documented metadata and data transformations. Without provenance information, it is impossible for a user to understand the relevance of data, to estimate its quality, and to investigate unexpected or erroneous results. Big Data systems that automatically and transparently keep track of provenance would enable pay-as-you-go analytics that do not suffer from this loss of important metadata. Furthermore, provenance can be used to define meaningful access control policies for heavily processed and heterogenous data. For instance, a user

[1] To be more precise, for state-of-the-art implementations of such applications.

could be granted access to analysis results if they are based on data she owns (have data that she owns in their provenance).

3 State-of-the-art

Having motivated the need for Big provenance, we now present a brief overview of provenance research related to Big Data and highly scalable systems. Since providing a complete overview of provenance research for distributed systems is beyond the scope of this paper, we only present a few approaches that are related to Big Data research or relevant for the discussion. Provenance research from the database community has been largely focused on fine-grained provenance, but has mostly ignored distributed provenance tracking. 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).[2] 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 original user function and attach provenance to the output based on the input's provenance and the semantics of the mapper and reducer functions. The HadoopProv system [2] modifies Hadoop to achieve a similar effect. Another approach for MapReduce provenance adapts database provenance techniques to compute the provenance of workflows expressed in a subset of the Pig language [3] corresponding to relational algebra. Similarly, the approach from [15] adapts a database provenance model for a distributed datalog engine.

While most workflow systems support distributed execution of workflows, provenance techniques for these systems are mainly coarse-grained (with a few noticeable exceptions) and rely on centralized storage and processing for provenance. 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 parts of a provenance graph corresponding to its local processing and maintains links to the provenance graphs of other nodes. To support queries over the provenance 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. [12] apply the PASS approach

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

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 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 (*variety*) 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 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.

4 Provenance as a Benchmark Workload

As mentioned before, provenance is of immense importance in the Big Data context. Thus, benchmarks for Big Data systems should include provenance workloads such as tracking provenance during the execution of a regular workload or querying pre-generated provenance data. In principle, there are two options for integrating provenance into benchmark workloads. First, existing provenance systems could be used as data generators for a benchmark and the actual workload would consist of queries over this provenance data. Second, tracking provenance could be part of the workload itself. Given the lack of Big Provenance systems discussed in Section 2, the first approach seems to be more realistic in the short term. However, in contrast to the second approach, it does not test

the ability of Big Data systems to deal with provenance information. Before discussing these two options in more depth, we first discuss how provenance workloads differ from "regular" workloads and how these differences influence what aspects of a system will be stressed by a provenance workload.

4.1 Provenance vs. Standard Workloads

Typical analytics over large datasets produce outputs that are significant smaller than the input data set (e.g., clustering, outlier detection, or aggregation). Provenance, however, can be orders of magnitude larger than the data for which provenance is collected. Provenance models the relationship between inputs and outputs of a transformation and, thus, even in its simplest form, can be quadratic in the number of inputs and outputs. This increase of size 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. Furthermore, the provenance information of two data items often overlaps to a large extend [4]. A benchmark that includes workloads running over provenance data stresses a system's capability to exploit commonality in the data (e.g., compression) and to avoid unnecessary shipping of data.

4.2 Pregenerated Provenance Workloads

Because of the potential size of provenance relative to the size of the data it is describing, it is possible to generate large data sets and computationally expensive workloads by collecting the provenance of a small set of transformations at fine granularity. This property could be exploited to generate data at the scale required for benchmarking a Big Data system. A common problem with benchmark data sets of such size is that it is unfeasible to distribute full datasets effectively over the internet (limitation of network bandwidth). Hence, a Big Data benchmark should include a data generator that allows users of the benchmark to generate the data sets locally. Generating detailed provenance for a small real-world input workload using an existing provenance system is one option to realize such a data generator. In contrast to other types of data generators, this approach has the advantage that it can be bootstrapped using a small input dataset as shown in the example below.

Example 1. Consider a build process for a piece of software using the `make` build tool. 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., [12]). 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 [14]. This would produce provenance at

pixel granularity and for individual variable assignments of the image processing 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.

4.3 Provenance Tracking as Part of the Workload

Alternatively, provenance collection could be directly used as a benchmark workload. The advantage of this approach is that it measures the ability of Big Data systems to deal with provenance efficiently. However, given the current state of the art discussed in Section 4.1, a benchmark with such a workload would prevent most Big Data systems from being benchmarked. Even for systems for which provenance tracking has been realized (e.g., Hadoop) we may not want to use provenance support until its impact has been understood sufficiently well and the systems have been optimized to a reasonable extend. A solution that allows for a smoother transition is to design a workload in such a way that available provenance information could be exploited to improve performance, but is not strictly necessary to execute the workload.

Example 2. Assume a workload that requires the benchmarked system to count the appearances of words in a collection of documents (e.g., word-count for wikipedia articles from the *PUMA* benchmark [1]) and retrieve simple provenance (the original documents in which the words occur) for a small, randomly selected subset of words. A Big Data system with provenance support could use stored provenance to execute the second part of the workload efficiently while a system without provenance support could fall back to the brute force method of searching for the specific word in all documents.

In summary, the main arguments for adding provenance to Big Data benchmark workloads are:

- Provenance has been recognized as an important functionality for Big Data [5]. Thus, it is natural to expect a benchmark to test a system's capability to deal with provenance.
- Provenance workloads stress-test the ability of a system to exploit commonalities in data and processing which is essential for Big Data systems. Including provenance in a workload will allows us to generate benchmarks that target this specific aspect.

We have discussed two options for integrating provenance in benchmark workloads:

- Run an existing provenance system to pre-generate a provenance workload. Using this approach we can generate provenance benchmarks for Big Data systems without provenance support. Furthermore, the sheer size of provenance information can be exploited to (1) generate large data sets from existing small real-world workloads and (2) develop concise benchmark specifications that can be shipped and expanded to full-sized workloads locally.

– Use provenance tracking as part of workload, i.e., the benchmarked system is required to track provenance. This method would test the ability of a system to efficiently track and query provenance, but requires broad adaptation of provenance techniques for Big Data to be feasible (unless, as explained above, provenance support is made optional).

5 Data-Centric Performance Measures

Besides from being an interesting and challenging use-case for workload design, Big Provenance could also be used as a supporting technology for benchmarks. A major goal for Big Data systems is robustness of performance and scalability [6]. Provenance can be used to provide a fine-granular, data-centric view on execution times and data movement by propagating this information based on data-flow. For example, we could measure the execution times of each invocation of a mapper in a MapReduce system and attach this information as provenance to the outputs of the mapper. The individual execution times are then aggregated at the reducer and combined with the reducer's execution time. This type of provenance can be used to compute measures for individual jobs in a workload and to compute new performance metrics using provenance information.

Example 3. Assume a system performs reasonably well on a complex workload. However, 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. We could record the execution times for all tasks of a job and the movement of data items between nodes as provenance. Based on this information we can identify jobs that use a large amount of resources relative to the size of data they consume or produce.

Note that in the example above the data-centric, provenance-based view on performance measurements is substantial for computing the measure. Benchmarks could exploit such information to define new data-centric measures for robustness of performance. 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 measure of robustness.

6 Monitoring and Profiling

Acting upon the results of a benchmark to improve the performance of a system usually requires additional monitoring and profiling to identify and understand the causes of poor performance. Big Data benchmarks should consist of complex and diverse workloads. However, understanding why a system performs good or poor over a complex workload is hard. Provenance could be used to complement monitoring solutions for Big Data systems.

Assume we record resource utilization of transformations and location changes of data items as the provenance of a data item. We could compute the amount of resources that were spend on producing a data item from this type of provenance information. Note that this is the data-centric equivalent to profiling execution times of functions in, e.g., a Java program. Coupling data with performance measurements for the transformations that created it enables novel types of profiling. For example, to identify redundant computations, we simply have to check whether the provenance of the final outputs of a transformation contains a data item multiple times (possibly produced by different transformations at different locations). This information can be used to automatically detect potential optimizations (e.g., it may be cheaper to ship the data item than to reproduce it). Furthermore, if an intermediate result is not in the fine-grained provenance of any final result of a task, then it was unnecessary to produce this intermediate result at all.

7 Conclusions

This paper discusses the importance of and challenges for Big Provenance for benchmarking. In addition to sketching the advantages and issues of generating Big Data provenance workloads, we argue that provenance may also be used to aide developers in identifying bottlenecks in the performance, scalability, and robustness of their systems. Provenance can be used for 1) computing fine-grained, data-centric performance metrics, 2) for measuring if a system is able to exploit data commonalities, and 3) for profiling systems.

References

1. Ahmad, F., Lee, S., Thottethodi, M., Vijaykumar, T.: PUMA: Purdue MapReduce Benchmarks Suite. Tech. Rep. TR-ECE-12-11, Purdue University (2012)
2. Akoush, S., Sohan, R., Hopper, A.: HadoopProv: Towards Provenance As A First Class Citizen In MapReduce. TaPP (2013)
3. 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)
4. Chapman, A., Jagadish, H.V., Ramanan, P.: Efficient Provenance Storage. In: SIGMOD. pp. 993–1006 (2008)
5. Divyakant, A., Bertino, E., Davidson, S., Franklin, M., Halevy, A., Han, J., Jagadish, H.V., Madden, S., Papakonstantinou, Y., Ramakrishnan, R., Ross, K., Shahabi, C., Vaithyanathan, S., Widom, J.: Challenges and opportunities with big data (2012)
6. Graefe, G.: Benchmarking robust performance. In: 1st Workshop on Big Data Benchmarking (WBDB) (2012)
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. Park, J., Nguyen, D., Sandhu, R.: A provenance-based access control model. In: PST. pp. 137–144 (2012)
12. Seltzer, M., Macko, P., Chiarini, M.: Collecting Provenance via the Xen Hypervisor. In: TaPP (2011)
13. Widom, J.: Trio: A System for Managing Data, Uncertainty, and Lineage. Managing and Mining Uncertain Data pp. 1–35 (2008)
14. Zhang, M., Zhang, X., Zhang, X., Prabhakar, S.: Tracing Lineage beyond Relational Operators. In: VLDB. pp. 1116–1127 (2007)
15. Zhou, W., Mapara, S., Ren, Y., Li, Y., Haeberlen, A., Ives, Z., Loo, B., Sherr, M.: Distributed time-aware provenance. PVLDB 6(2), 49–60 (2012)