Odyssey: A Multi-Store System for Evolutionary Analytics

Hakan Hacıgümüş
Jagan Sankaranarayanan
Junichi Tatemura

NEC Laboratories America

Jeff LeFevre
Neoklis Polyzotis

University of California, Santa Cruz

1. INTRODUCTION

We present a data analytics system, *Odyssey*, that is being developed at NEC Labs in collaboration with NEC's commercial business units and our academic collaborators. The design principles of the system are based on the business requirements identified through extensive surveys and communications with the practitioners and customers. Most notable high-level requirements are:

1) The analytics system should be able to effectively use both structured and unstructured data sources. **2)** Business requirements are not captured in a single simple metric but combination of metrics, such as value of data, performance, monetary costs, and they are dynamic. The system should manage data by observing constantly changing metric values. **3)** Time-to-insight is very important, so the system should enable immediate exploratory querying of data without heavy prerequisite processes. **4)** The system should efficiently support both ad-hoc queries and application workloads. **5)** Very often there are already data analytics solutions / products in place (such as a traditional data warehouse), hence the system should be able to incorporate the existing settings.

The data analysis is performed in a rapidly changing way and includes the new role of data scientist who is tasked with finding the benefit in big data, which come from disparate sources. The trend of collecting ever-growing data with unknown and unproven benefits, and the nature of the exploratory queries that are posed on these datasets represent an emerging type of data analysis. The fluid nature of this analysis is that the analyst may start by posing simple questions on the data but then evolve towards more sophisticated reasoning as well as apply sophisticated techniques. The evolutionary nature of the investigation is due to the analyst who may not initially be able to express her goals well, thus modifying her workflow slightly and iteratively refining it until achieving the intent. The evolutionary process may also require incorporating more data sources into the analysis to obtain richer and more confident answers. We call this iterative process by which an analyst finds benefits in the data as *evolutionary analytics* [4]. We envision a platform to support such data explorations by enabling users to 1) perform increasingly sophisticated analysis as they gain better understanding of the data (*Query Evolution*), 2) automatically benefit from analysis performed by other analysts(*User Evolution*) , and 3) easily add new data sources for their analysis (*Data Evolution*).

Submission to VLDB 2013 Industrial Track Vol. 6, No. 11

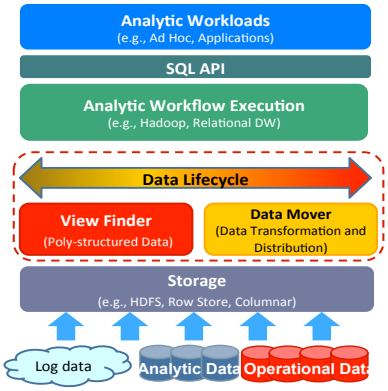

Figure 1: Odyssey System Architecture

We have concluded that there is no single data analytics platform that would satisfy all of these requirements efficiently and effectively. Therefore, Odyssey system employs a ***multi-store approach*** to store and query data. The system exploits the strengths of individual stores to serve diverse – in the sense of varying business requirements – analytic workloads in a differentiated manner. While using multiple stores in a unified manner is an opportunity, it represents significant technical challenges, which Odyssey system aims at addressing.

2. SYSTEM ARCHITECTURE

Figure 1 shows the overall architecture of the envisioned Odyssey system. Odyssey adopts the following principles for data processing: 1) optimization for whole workloads, instead of per-query/workflow optimization, 2) end-to-end management of data lifecycle, and 3) effective use of opportunistic materialized views for significant performance gains. We will explain these points below.

The system takes data from various data sources including unstructured data (e.g., logs), structured data (e.g., operational data from transactional systems), and other sources of analytics data (e.g., legacy data warehouse data). Currently, the system uses two execution engines, namely; Hadoop and

the Relational DW. The future phases of the system development plan to include additional execution engines, such as a columnar in-memory store. The application workloads interact with the system through the SQL-like API.

Naturally, employing multiple stores requires making the decisions on where the data should be stored and processed with the changing business requirements and the benefits of the data. Hence, at the core (shown in dashed lines), the system aims at answering the following key questions: *Which* data sets should be moved?, *Where* the data sets should be moved to?, and *When* the data sets should be moved?

The **View Finder** component opportunistically identifies beneficial views for workload optimization by considering views in both HDFS and DW. In that sense the View Finder processes poly-structured data. The **Data Mover** component is responsible for efficiently moving data between the stores (shown as dotted arrows in Figure 2). This process includes necessary data transformations (such as type conversions, file formatting) and distributions. The Data Mover and the View Finder components represent a fundamental process called *Data Lifecycle Management (DLM)* in the system. DLM is essentially defined as creating/changing the representation (view) and the placement of data sets in an optimized manner based on the changing business metrics and priorities, such as the benefit of the data, performance requirements, and monetary costs.

The system does not make any assumptions about the location of data. However, in our experience most of the new exploratory data are first loaded into the Hadoop store, and DW typically has legacy data. Initially, splitting the query processing is based on the current data availability in each store.

3. QUERY PROCESSING ON MULTI STORES

The split query processing is depicted in Figure 2. The system first analyzes the query and identifies which fragments of the query can be executed in each store. Then, the query optimization process starts. The system has two modes of optimization 1) single query optimization and 2) workload optimization. Here we first describe the single query optimization.

The system uses the cost based optimizer to decide when it is most beneficial to move the query processing and data from one store to another (e.g., from Hadoop to DW). The optimization considers the query execution costs on both stores and also data transfer/load costs.

Our observation is that the data transfer and load is always the most dominant cost in the breakdown of the multi-store query processing times. Therefore the cost based optimizer usually chooses to perform the most of the processing in Hadoop and move to DW at a very late stage when the transfer data size becomes sufficiently small. This observation is consistent with [1]. However, for most of the workloads, this last stage movement does not provide substantial benefits as vast majority of the data sets in Hadoop remain large for the significant part of the execution, which is also observed and reported in other real life applications [1]. Therefore, although it is useful, single query optimization over multiple stores showed very limited optimization benefits in our application cases. This limitation is the motivation behind the *workload optimization*.

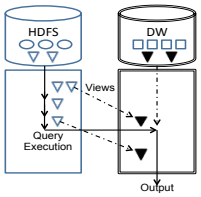

Figure 2: Multi Store Query Execution with Views

Opportunistic Optimization: As stated, unless the data size is considerably small, transferring data – hence the query execution – between the stores is not meaningful. We observed that even the parallel load performance to the commercial DW system is disproportionally slower compared to load times into the HDFS cluster. However, exploiting materialized views (MVs) could be helpful because of their customizable size. Nonetheless, views are justified when they are used repeatedly. Therefore, Odyssey considers the optimization for multiple queries that define the workloads with complete visibility to all views in all stores.

MVs are very common in traditional DWs. In Hadoop, each MR job involves the materialization of intermediate results (the output of mappers and the input/output of reducers) for the failure recovery purposes. These materialized results are the artifacts of query execution, which are generated automatically as a by-product of query processing. Therefore, we call them *opportunistic views*. More details of our discussion on opportunistic views can be found in [3]. Thus, we propose using the opportunistic views to rewrite the queries in the system for improved performance.

For optimization, the system generates an "annotated" query execution plan (Fig. 2). The annotations are the inclusion of the opportunistic views (shown as triangles) in the query plan. Then, the optimizer examines possible *transfer points* (shown as solid arrows) to move the query execution between the stores and chooses the optimal plan.

Workload Optimization: The optimization process described above still considers a query-at-a-time. The main principle behind the workload optimization is that the system uses previously defined opportunistic views in all stores for workload optimization. The system continuously monitors the workloads and performs an online analysis to identify, which views would deliver the most benefit for the overall workload. This process is analogous the online view and index selection problems. As a result, even some queries might be under optimized, the total benefit for the overall workload over the time is much greater. We present some preliminary results of the optimizations in [4].

Acknowledgment: We thank NEC's product and business teams for their generous support and contributions.

4. REFERENCES

[1] S. Agarwal, , S. Kandula, N. Bruno, M.-C. Wu, I. Stoica, and J. Zhou. Reoptimizing data parallel computing. In *NSDI*, 2012.

[2] H. Hacıgümüş, J. Tatemura, W.-P. Hsiung, H. J. Moon, O. Po, A. Sawires, Y. Chi, and H. Jafarpour. CloudDB: One size fits all revived. In *IEEE SERVICES*, 2010.

[3] J. LeFevre, J. Sankaranarayanan, H. Hacigümüş, J. Tatemura, and N. Polyzotis. Exploiting opportunistic physical design in large-scale data analytics. *CoRR*, abs/1303.6609, 2013.

[4] J. LeFevre, J. Sankaranarayanan, H. Hacigümüş, J. Tatemura, and N. Polyzotis. Towards a workload for evolutionary analytics. In *SIGMOD, DanaC Workshop*, 2013.