International Journal of Engineering, Economics and Management
ISSN: 2319-7927, Volume 3, Issue 1

A New Progressive Analytics to Reduce Cost in Relational Domain

AnkitaV.Torawane AmitkumarManekar

Abstract — **Progressive analytics is the term which means to reduce the cost of data analytics in relational domain. Data scientists extract the samples of increasing data size (progressive samples) using sampling strategies for exploratory querying. This samplings strategies provides them with user-control, repeatable semantics, and result provenance. Thus these type of solutions results in tedious workflows. Existing query processing systems gives results, but they do not offer the above benefits for complex ad-hoc queries. So a new progressive analytics system is proposed which is based on a progress model called Prism which allows users to communicate progressive samples to the system; also allows efficient and deterministic query processing over samples; and provides repeatable semantics and provenance to data scientists. This system is also applicable for atemporal relational queries using an unmodified temporal streaming engine which re-interpret temporal event fields to denote progress. Now!, architecture is build based on prism model for progressive data-parallel computation framework. Now! Architecture works with "progress-aware reducers", also works with streaming engines to support progressive SQL over big data. By using the application Prism model with MR applied to another computational model where the results are represented in the graphical form and also the cost of the relational domain will be reduce**

Key Words- **Progressive analytics, approximate query processing (AQP), Map reduce online**

I. INTRODUCTION

Approximate query processing (AQP) is proposed by the database community systems which are DBO [14] which performs progressive analytics. Progressive analytics produce results of analytics queries based on partial data and the results are refined as more data is received. Sufficient accuracy or query incorrectness is observed in the results of progressive analytics.

Map reduce online(MRO)[12] adds pipelining to MR but MRO reports the progress metrics that does not eliminate the problems related to map reduce.

The lack of system support results in tedious and error prone work that precludes the reuse of work across progressive samples.The system is needed which allows user to communicate with progressive samples to the system and allow efficient and deterministic query processing samples. The key idea is for users to encode their chosen progressive sampling strategy into the data by augmenting tuples with explicit progress intervals (PIs). PIs denote logical points where tuples enter and exit the computation, and explicitly assign tuples to progressive samples. PIs offer flexibility for encoding sampling strategies and ordering for results, including arbitrarily overlapping sample sequences.

A new progress model is introduce into an existing relational engine appears challenging. A progressive in-memory relational engine based on Prism can be realized immediately using an unmodified temporal streaming engine, to denote the progress reuse of temporal fields are carefully done. Tuples from successive progressive samples get incrementally processed when possible, giving a significant performance benefit. The temporal engine is unaware that it is processing atemporal relational queries; the queries can simply re-interpret its temporal fields to denote progress points. While it may appear that in-memory queries can be memory intensive since the final answer is computed over the entire dataset, Prism allows us to exploit sort orders and foreign key dependencies in the input data and queries to reduce memory usage significantly.

AQP is generalized by prism progress semantics which are compatible with queries for which prior AQP techniques with statistical guarantees apply, and thus don't require user involvement. These techniques simply correspond to different PI assignment policies for input data. Variants of ripple join [18] are different PI assignments for a temporal symmetric-hash-join, with intervals computed as part of the query. Thus, Prism is orthogonal and can leverage this rich area of prior work, while adding the benefits of repeatable and deterministic semantics. Prism gives progressive results form of determinism and control the final results.

The Prism model is particularly suitable for progressive analytics on big data in the Cloud, since queries in this setting are complex, and memory and CPU intensive. Scalable distributed frameworks such as MR are not pipelined, making them unsuitable for progressive analytics. MRO[12] adds pipelining, but does not offer the semantic underpinnings of progress necessary to achieve the desirable features.

This problem is addressed by designing and building a new framework called Now! For progressive analytics. Now! runs on Windows Azure; it propagates progress based on the Prism model inside the framework. Now! generalizes the popular data-parallel MR model and supports progress-aware reducers that understand explicit progress in the data. Now! can work with a temporal engine ie. StreamInsight [13] as a progress-aware reducer to enable scaled-out progressive relational (SQL) query support in the Cloud.

Now! provides a substantial reduction in processing time, memory and CPU usage as compared to current schemes; performance is significantly enhanced by exploiting sort orders and using our memory-only processing mode.

II. LITERATURE SURVEY

Online aggregation is proposed by Hellerstein et al. [20], where the focus was on grouped aggregation with robust confidence intervals based on random sampling. This was again extended to handle join queries using the ripple join [19] operators. Laptev et al. [7] proposed MR jobs to compute iteratively on increasing data samples until a desired approximation goal is achieved. BlinkDB [1] constructs a large number of multi-dimensional samples online using a particular sampling technique ie stratified sampling and choose samples based on a user-specified budget. Instead of taking systems responsibility for query accuracy (e.g., as sampling techniques) which may not be possible in general is followed with a different approach, which involves the query writer's specification of progress semantics. A query processor using Prism model support a variety of user-defined progressive sampling schemes; that helps the assignment of PIs in a semantically appropriate manner.

Map-Reduce Online (MRO) [12] supports progressive output by adding pipelining to MR. The result of snapshots are produced by reducers, each annotated with a rough progress estimate based on averaging progress scores from different map tasks.Progress in MRO is an operational and non-deterministic metric that cannot be controlled by users or used to correlate progress to query accuracy or to specific input samples. MRO sorts subsets of data by key and redundant computations as reducers repeat aggregations over increasing subsets [5].

Li et al. [8] propose scalable one pass analytics (SOPA), where he replace sort-merge in MR with a hash based grouping mechanism inside the framework. The focus of the paper is on progressive queries, with a goal of establishing and propagating explicit progress in the platform. Like SOPA, the framework sorting is eliminated to leave it to the reducer to process progress-sync ordered data. Stream engines use efficient hash-based grouping, which allows to realize similar performance gains as SOPA inside our reducers.

Progressive results for atemporal queries over atemporal online data, and show that new progress model can in fact be realized by leveraging and re-interpreting the notion of time used by temporal SPEs. Now! is an MR-style distributed framework for progressive queries; marked in different from distributed SPEs [15] as it leverages the explicit notion of progress to build a batched-sequential data-parallel framework that does not target real-time data or low-latency queries. The progress batched files are used for the movement of data which allows Now! to transfer costs across reducer per-tuple computation cost. Now! architecture is designed for cloud setting along with the lines of MR with extended map and also to reduce APIs.

Dremel [11] and PowerDrill [2] are distributed system for interactive analysis of read-only large columnar datasets. Spark [3] provides in-memory data structures to persist intermediate results in memory, and is used to interactively query big data sets or get medium-latency results on real-time data [34]. These engines have a different goal they provide full results to queries in-memory data in milliseconds, for which they use careful techniques such as columnar in-memory data organization for the (smaller) subset of data that needs such interactivity. The generic interactivity is processed over large datasets, in terms of meaningful results on progressive samples and refining results. Based on results, users can choose to potentially end (or possibly refine) computations sufficient accuracy or query incorrectness.

In online aggregation [9], a database system processes a user's aggregation query in an online fashion. During processing, the system gives the user an estimate of the final query result.The paper focus on the idea of online aggregation which can be built into a MapReduce system for large-scale data processing.

P.Upadhyaya [10] address the problem of making online, parallel query plans fault-tolerant: i.e., provide intra-query fault-tolerance without blocking. The approach is developed that not only achieves this goal but also use different fault-tolerance techniques at different operators within a query plan. Each operator use a different fault- tolerance strategy that leads to a space of fault-tolerance plans amenable to cost-based optimization.A cost-based fault-tolerance optimizer selects the best strategy for each operator in a query plan in a manner that minimizes the expected processing time with failures for the entire query. The prototype parallel query-processing engine approach is also implemented.There is no single best fault-tolerance strategy for all query plans, and the optimizer correctly identifes winning fault-tolerance confugurations.

C.Jermaine [14] describes query processing in the DBO database system.The other database systems are designed for ad-hoc, analytic processing, DBO is able to compute the exact answer to queries over a large relational database in a scalable fashion.DBO constantly maintain a guess as to the final answer to an aggregate query throughout execution, along with statistically meaningful bounds for the guess's accuracy which are designed for analytic processing. As DBO gathers more and more information, the guess gets more and more accurate, until it is 100% accurate as the query is completed. This allows users to stop the execution at any time that they are happy with the query accuracy, and encourages exploratory data analysis.

B.Chandramouli [6] focuses the concept of "Big Data" in map-reduce (M-R) clusters is fundamentally temporal. Display advertising uses Behavioural Targeting (BT) to select ads for users based on prior searches, page views, etc.

Previous work on BT has focused on techniques that scale well for offline data using M-R. There limitations for BT-style applications that deal with temporal data many queries are temporal and not easily expressible in M-R, and moreover, the set-oriented nature of M-R frontends such as SCOPE is not suitable for temporal processing; and commercial systems mature, they may need to analyze and react to real-time data feeds since a high turnaround time can result in missed opportunities, but it is difficult for current solutions to operate over real-time streams.

J.Dean and S.Ghemwat [17] describes the concept of MapReduce which are used as a programming model and are associated with implementation for processing and generating large data sets. A map function is specified by the user that processes a key/value pair to generate a set of intermediate key/value pairs, and a reduce function is used to merge all the intermediate values associated with the same intermediate key. Many real world tasks are expressible in this model. Programs written in this functional style are automatically parallelized and executed on a large clusters. The run-time system takes care of the details of partitioning the input data, scheduling the program's execution across a set of machines, handling machine failures, and managing the required inter-machine communication. This allows programmers without any experience with parallel and distributed systems to easily utilize the resources of a large distributed system. The implementation of MapReduce runs on a large cluster of commodity machines and is highly scalable. A typical MapReduce computation processes many terabytes of data on thousands of machines.

N.Laptev [7] focused on Analytical applications have massive data sets which can satisfy their time and resource constraints. The methods and tools for the computation of accurate results are currently not supported in MapReduce-oriented systems although these are intended for `big data'. A non-parametric extension of Hadoop allows the incremental computation of results for arbitrary workflows is eastablished along with reliable online estimates of the degree of accuracy achieved so far in the computation. These estimates are based on a technique called bootstrapping which has been employed in statistics and can be applied to arbitrary functions and data distributions. They describe Early Accurate Result Library (EARL) for Hadoop that is designed to minimize the changes required to the MapReduce framework. Various tests of EARL of Hadoop are presented to characterize the frequent situations where EARL can provide major speed-ups over the current version of Hadoop.

Borealis[15] is a second-generation distributed stream processing engine that is being developed at Brandeis University,which describes the basic design and functionality of Borealis.In real-world applications, the need for dynamically revising query results and modifying query specifications are motivated.The paper focuses on how Borealis addresses these challenges through an innovative set of features, including revision records, time travel, and control lines. It also present a highly flexible and scalable QoS-based optimization model that operates across server and sensor networks and a new fault-tolerance model with flexible consistency availability trade-offs.

M.Hammad focus on concept NILE [16] or streamInsight [13] which can modify a database engine to add PI support to all operators in the engine. The idea is to leverage a stream processing engine (SPE) as the progressive query processor. The semantics underlying a temporal SPE such as NILE can be leveraged to denote progress, with the added benefit of incremental processing across samples when possible. Nile extends the query processor engine of an object-relational database management system to support data streams.

P.Haas and J.Hellerstein[18,19] present join algorithms, called ripple joins, for online processing of multi-table aggregation queries in a relational database management system. Such queries arise naturally in interactive exploratory decision-support applications. Online join algorithms are designed to minimize the time to completion of the query. Whereas ripple joins are designed to minimize the time until an precise estimate of the query result is available, measured by the length of a confidence interval. Ripple joins are adaptive, adjusting their behavior during processing in accordance with the statistical properties of the data. Ripple joins also permit the user to dynamically trade off the two key performance factors of online aggregation.The time between successive updates of the running aggregate, and the amount by which the confidence-interval length decreases at each update. How ripple joins can be implemented in an existing dbms using iterators are shown, and an overview of the methods used to compute confidence intervals and to adaptively optimize the ripple join parameters.

CONCLUSION

A new progress model called Prism is proposed which allows users to communicate progressive samples to the system; also allows efficient and deterministic query processing over samples; and provides repeatable semantics and provenance to the data scientists.By using the Prism model with map reduce applied to another computational model where the results are represented in the graphical form and also the cost of the relational domain will be reduce

REFERENCES

[1] S. Agarwal et al. Blinkdb: Queries with bounded errors and bounded response times on very large data. In EuroSys, 2013.
[2] A. Hall, O. Bachmann, R. B¨ussow, S. G˘anceanu, and M. Nunkesser. Processing a trillion cells per mouse click. PVLDB July 2012.
[3] M. Zaharia et al. Resilient distributed datasets: a fault-tolerant abstraction for in-memory cluster computing. NSDI'12.

[4] M. Zaharia et al. Discretized streams: An efficient and fault-tolerant model for stream processing on large clusters. In HotCloud, 2012
[5] B. Chandramouli et al. Scalable progressive analytics on big data in the cloud. Technical report, MSR.2012
[6] B. Chandramouli et al. Temporal analytics on big data for web advertising. In ICDE, 2012.

[7] N. Laptev, K. Zeng, and C. Zaniolo. Early accurate results for advanced analytics on mapreduce. PVLDB 2012.
[8] B. Li, E. Mazur, Y. Diao, A. McGregor, and P. J. Shenoy. A platform for scalable one-pass analytics using mapreduce. In SIGMOD 2011
[9] N. Pansare, V. R. Borkar, C. Jermaine, and T. Condie. Online aggregation for large mapreduce jobs. PVLDB, 2011.
[10] P. Upadhyaya, Y. Kwon, and M. Balazinska. A latency and fault-tolerance optimizer for online parallel query plans. In SIGMOD, 2011
[11] S. Melnik et al. Dremel: interactive analysis of web-scale datasets. PVLDB 2010
[12] T. Condie, N. Conway, P. Alvaro, J. M. Hellerstein, K.Elmeleegy, and R. Sears. Mapreduce online. In NSDI, 2010.
[13] M. Ali et al. Microsoft CEP Server and Online Behavioral Targeting. 2009.
[14] C. Jermaine, S. Arumugam, A. Pol, and A. Dobra. Scalable approximate query processing with the dbo engine. SIGMOD '07.
[15] D. Abadi et al. The design of the Borealis stream processing engine.2005.
[16] M. Hammad et al. Nile: A query processing engine for data streams.2004.
[17] J. Dean and S. Ghemawat. Mapreduce: simplified data processing on large clusters. OSDI'04.
[18] P. J. Haas and J. M. Hellerstein. Ripple joins for online aggregation.In SIGMOD 1999.
[19] P. J. Haas and J. M. Hellerstein. Join algorithms for online aggregation. In IBM Research Report RJ 10126, 1998.
[20] J. M. Hellerstein, P. J. Haas, and H. J. Wang. Online aggregation. In SIGMOD, 1997.