COLA: A Cloud-Based System for Online Aggregation

Yantao Gan, Xiaofeng Meng, Yingjie Shi

School of Information, Renmin University of China

Beijing 100872, China

{ganyantao19901018,xfmeng,shiyingjie}@ruc.edu.cn

***Abstract*—Online aggregation is a promising solution to achieving fast early responses for interactive ad-hoc queries that compute aggregates on massive data. To process large datasets on large-scale computing clusters, MapReduce has been introduced as a popular paradigm into many data analysis applications. However, typical MapReduce implementations are not well-suited to analytic tasks, since they are geared towards batch processing. With the increasing popularity of ad-hoc analytic query processing over enormous datasets, processing aggregate queries using MapReduce in an online fashion is therefore an emerging important application need.**

We present a MapReduce-based online aggregation system called COLA, which provides progressive approximate aggregate answers for both single table and multiple joined tables. COLA provides an online aggregation execution engine with novel sampling techniques to support incremental and continuous computing of aggregation, and minimize the waiting time before an acceptably precise estimate is available. In addition, user-friendly SQL queries are supported in COLA. Furthermore, COLA can implicitly convert non-OLA jobs into online version so that users don't have to write any special-purpose code to make estimates.

I. Introduction

Business intelligence (BI) is a fast growing multi-billion dollar market [9]. It has been identified as an increasingly high priority technology to filter vast and growing amounts of information to reach insights and decisions in the digitized world. As an important category in interactive exploratory decision-support applications, ad-hoc queries are expected to respond quickly for effective user interaction, which is increasingly challenging due to the increasing volumes of data. A very compelling proposition is online aggregation (OLA) [1] that enables running estimates with the progressively refined confidence intervals.

The size of data sets being collected and analyzed is growing exponentially [10] in various application domains, including business data processing, web graph and social network analysis, and bioinformatics research. In big data analytics, MapReduce [6] has emerged as probably the most popular paradigm for parallel processing, and it already has a great impact on data management research. Unfortunately, the MapReduce model is primarily designed for batch processing of queries on large datasets, which forces users to wait a long period of time without feedback before the whole query processing finishes. Because of the increasing popularity of ad-hoc analytic query processing over enormous datasets, processing aggregation using MapReduce in an online fashion is of tremendous importance.

While OLA has been studied for some time in the context of RDBMS [1], [2] and later extended to peer-to-peer systems [3], it is somewhat discouraging to note that OLA has not been widely adopted in real applications. We believe that the cloud computing environment will make OLA commercially significant. Cloud platforms make MapReduce an attractive proposition for small organizations that need process large datasets, but lack sufficient computing resources to throw at the problem. Elastic MapReduce, for example, is a hosted platform on the Amazon cloud where users can provision Hadoop clusters instantly to perform data-intensive tasks. Since cloud platforms are typically pay-as-you-go, the end-user will more likely choose OLA to save money by terminating the computation prematurely once sufficient accuracy has been obtained. As a motivating example, assuming that we performed a MapReduce job on an Amazon EC2 cluster, the total running time was about 95 hours, and more than $2500 flowed into the cloud. But if a satisfactory estimate can be obtained when the query progress reaches less than 10%, thus over $2000 can be saved. It stands to reason that MapReduce-based OLA is much more attractive. In addition, it becomes much more feasible to build OLA into a MapReduce system than expensive database given the availability of free open-source implementations such as Hadoop and Hyracks. Furthermore, MapReduce jobs are typically composed of many subtasks executing on distributed nodes, OLA could help increase opportunities for parallelism, improve resource utilization and reduce response time by processing parallel subtasks in online mode. Lastly, in an OLA system, the data flow between subtasks is pipelined and the estimate result is refined continuously, which can alleviate or eliminate the performance bottleneck due to the slowest nodes.

We are not the first to look at OLA based on MapReduce, Reference [6] and [4] are the closest to our work, they try to adapt OLA to the MapReduce programming framework. A modified version of the Hadoop MapReduce framework called Hadoop Online Prototype (HOP) can be found in [6], which allows intermediate data to be pipelined between operators. HOP can provide the original snapshots of the MapReduce jobs at data dependent intervals, and it supports OLA by scaling up the snapshots with the job progress but without any confidence bounds involved. When new snapshot

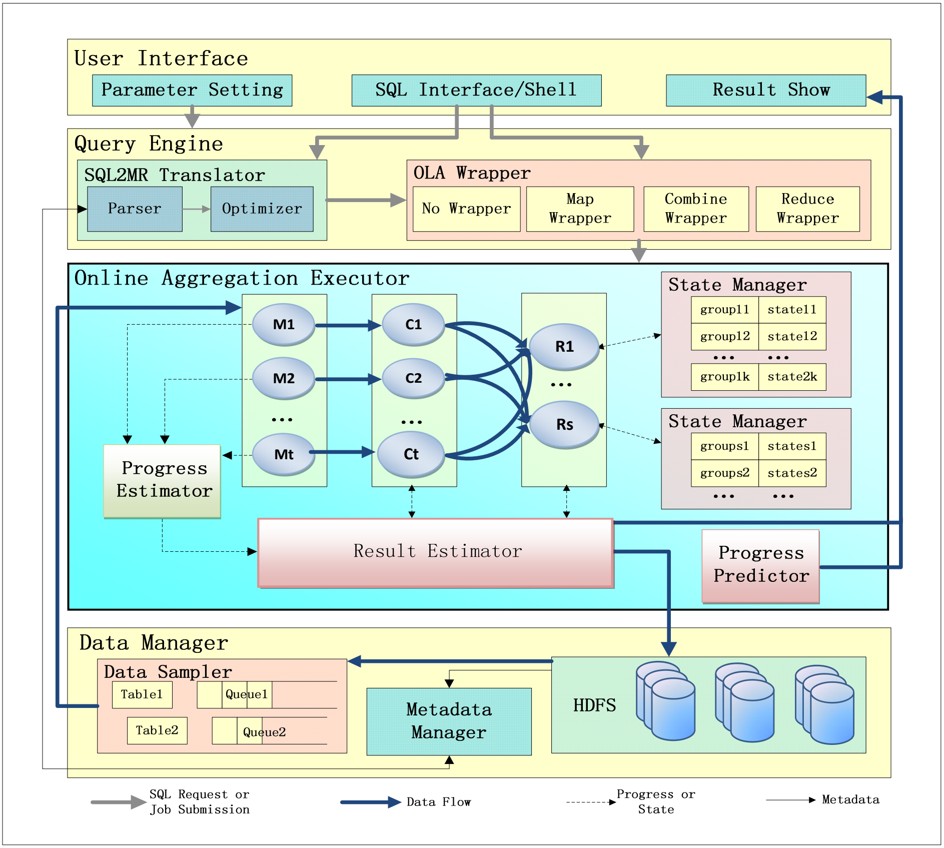


Fig. 1. COLA system architecture

is required, HOP simply re-runs the entire intermediate data, which is grossly inefficient. Reference [4] implements OLA over MapReduce based on Bayesian framework. The authors consider the correlation between the aggregate value and processing time of each block, so they take into account the scheduling time and processing time of each block as observed data during the estimate processing. This paper has focused on single-table aggregate query involving one MapReduce job, which needs to conform to specific programming interface. COLA [7] is built on top of HOP and our contributions beyond above systems are four-fold: (1) COLA is the first work to our knowledge that considers online processing of join aggregation over MapReduce. (2) COLA ensures representative early estimates that are amenable to statistical guarantees by supporting block-level sampling for single-table aggregation and two-phase stratified sampling for multi-table aggregation respectively. (3) COLA implements an incremental processing algorithm that enables both fast early responses and short end-to-end query time. (4) COLA is easier to use with a friendly interface that supports SQL queries and also non-OLA MapReduce programs.

II. COLA Architecture and Implementation

Figure 1 illustrates the overall COLA architecture, the system comprises four modules: User Interface, Query Engine, Online Aggregation Executor and Data Manager. Users can submit queries through SQL or command-line interface and monitor running estimates via the User Interface. The Query Engine serves as a translator that transforms SQL queries into MapReduce jobs and converts non-OLA jobs to online mode. The Online Aggregation Executor fetches uniform-random samples from the Data Manager continuously, processes the samples through MapReduce jobs in online fashion and reports the estimates back to the client.

A. User Interface

COLA provides interactive and flexible interfaces, users can issue SQL query request through SQL interface or submit MapReduce program via shell interface. In addition, the graphical interface can facilitate observing the query progress and online estimates with associated confidence intervals during the query processing, COLA also indicates the estimate of residual completion time and amount saved so far.

B. Query Engine

The Query Engine is responsible for compiling the SQL query into directed acyclic graph of MapReduce jobs, and translating the non-OLA jobs to online version. Hence users can submit batch-oriented MapReduce programs and don't need to have the knowledge of the estimate computation.

OLA Wrapper: The MapReduce execution plan returned by SQL2MR Translator or user-supplied MapReduce program is typically batch-oriented. The OLA Wrapper implicitly adds estimation algorithms into the map, combine and reduce functions to produce approximate results and confidence bounds, thus converts batch jobs to OLA implementations.

C. Online Aggregation Executor

The Online Aggregation Executor is the key module of COLA to perform our online query processing algorithm over MapReduce. It is called to process the sample data, produce approximate answers with their associated confidence intervals and progressively refine the answers. In addition, the module makes predictions about the residual completion time, and also estimates amount saved so far.

Result Estimator: The Result Estimator applies CLT (central limited theorem) to computing estimate results and associated confidence intervals, which is implemented through combiner and reducer of the aggregate_job. Specifically, combiner is responsible for partial aggregation and pre-estimation, and the reduce function is invoked to produce total aggregate values and make estimates once a new snapshot is needed. Moving some work to map-side combiners can substantially reduce the burden of reducers and network traffic, and help increase the opportunities to improve resource utilization since mappers typically far exceed reducers in number. During the query processing, the Result Estimator is supplied with current job progress from the Progress Estimator, which monitors the progress of task execution to compute the job progress score for a snapshot.

State Manager: Online aggregation needs to provide "early returns" and HOP system supports that by allowing the intermediate results of all operators to be pipelined during the query processing. But it simply re-processes all intermediate data when new output from precursor operators arrives, which is grossly inefficient, increasing result latency and squandering hardware resources and energy. Alternatively, we re-use prior result to incrementally produce estimates, so that COLA can achieve fast early responses and short end-to-end query time, therefore COLA are comparable to conventional Hadoop in case users want the query run to completion. To do so, COLA dynamically maintains a local State Manager for every reducer to manage the state of each group. Here "dynamically" means we only save the latest states and update them once a new snapshot has been taken. As the state information needs to be served fast, the State Manager holds all the states in the memory.

Progress Predictor: The SQL queries submitted by users will be transformed into MapReduce DAGs. Based on that, the Progress Predictor models every map/reduce task with its duration and failure probability, and transforms the query procedure into a stochastic PERT (Project Estimate and Review Technique) network. Then the component computes the most critical path of the PERT network, which can represent the execution of the whole query. Our paper [8] has the detailed discussion about how to make an estimate of the query progress according to the critical path.

D. Data Manager

The Data Manager makes use of HDFS to store and manage data. It stores metadata such as mappings between tables and HDFS directories in Metadata Manager, that can be used to do query optimization and compilation in SQL2MR Translator. In addition, in order to make efficient estimates, we design block-level random sampling for single-table online aggregation and two-phase stratified sampling for multi-table aggregation. The goal of both sampling mechanisms is to improve the accuracy of approximate answers and speed up the convergence of confidence intervals.

Data Sampler: We design a block-level sampler for single-table OLA involving only one MapReduce job. The sampler retrieves data directly from the source data. After the input table is split to fixed-size blocks, all the blocks are uniformly shuffled and added to the scheduling queue. Once a map task is scheduled, the queue serves up the head-of-line block. Different from single-table OLA, aggregation on joins involves multiple MapReduce jobs, namely join_job and aggregate_job. For this kind of aggregation, we propose a two-phase stratified sampling mechanism involving both aforementioned sampler and a stratified one. We define the tuples of each table sent to each reducer of join_job as the sampling stratum. The first phase is used to estimate the stratum distribution of the population. It's just like the above-mentioned sampling, the only difference lies in that multiple scheduling queues are involved here, one for each table, and the blocks are scheduled from queues in turn. The second-phase sampler allows reducers perform sampling from mappers' output during shuffle phase of join_job, and produces distributed stratified random samples from multiple tables respectively. You can find more details in our paper [8].

III. Demonstration Scenario

In the demonstration we show the online processing of aggregate queries over Wikipedia traffic statistics dataset [11], which contains 7 months of hourly page view statistics for more than 2.5 million Wikipedia articles. This constitutes 320G of compressed data, we replicate it to about 1TB for our demonstration. An example of the Wikipedia traffic statistics data is presented in Table I, each line has 4 fields: language, pagename, pageviews, pagesize. We will use a 20-node computing cluster to demonstrate following scenarios.

TABLE I
WIKIPEDIA TRAFFIC STATISTICS DATA

language	pagename	pageviews	pagesize
en	Barack_Obama	997	123091092
en	Barack_Obama%27s_ first_100_days	8	850127
en	Barack_Obama,_Jr	1	144103

A. Scenario 1

Demonstrate the online processing of single-table aggregation. Assuming that a user wants to count the total number of page views for each specified language and each hour of the day, and he/she issues the following SQL query via the SQL interface:

SELECT *language*, SUM (*pageviews*)
FROM TABLE *wiki_log*
WHERE *language* IN (‘*en*’,‘*ja*’,’*de*’,’*es*’,’*fr*’,’*it*’,’*pl*’,’*ru*’)
GROUP BY *language*.

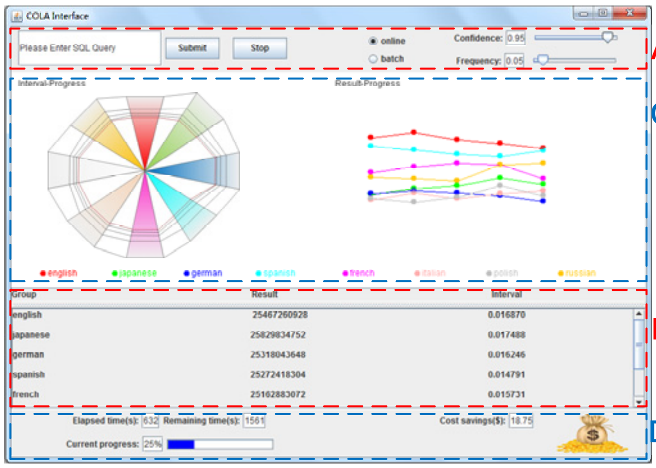
We will demonstrate that COLA can deliver reasonable precise online estimates within a time period two orders of magnitude shorter than that used to produce exact answers. We will also compare COLA's running time against HOP to demonstrate that all the operations extended into HOP incur little costs.

B. Scenario 2

Demonstrate the online processing of aggregation over equal-join results of two tables. According to the traffic data of Wikipedia, we construct two tables: hit_log and page_size. Each line of table hit_log contains 3 fields: pagename, language and pageviews, while table page_size contains the page size information with 3 fields: pagename, language, pagesize. Assuming following SQL query is implemented as a MapReduce program:

SELECT *page_size.language*, SUM (*pageviews*)
FROM TABLE *hit_log*, *page_size*
WHERE *hit_log.pagename=page_size.pagename*
AND *hit_log.language=page_size.language*
AND *page_size.pagesize*>=5000
GROUP BY *page_size.language*.

A snapshot of COLA interface is shown in Figure 2, which consists of four parts.



[✓ online] batch
Confidence: 0.95
Frequency: 0.05

Group	Result	Interval
english	25467260928	0.016870
japanese	25829834752	0.017488
german	25318043648	0.016246
spanish	25272418304	0.014791
french	25162883072	0.015731

Elapsed time(s): 632 Remaining time(s): 1561
Current progress: 25%
Cost savings($): 18.75

Fig. 2. User Interface

Part A provides the interface for users to submit SQL query and set configuration parameters. There are two processing modes available in COLA: batch-processing and online aggregation. When OLA mode is chosen, the confidence and frequency need to be specified prior to submitting. If the frequency is set to 0.05, it means once 5% input is consumed, the Result Estimator would be called to produce new estimates. Part B displays the latest estimate results with associated confidence bounds, one line for each output group. In our demonstration, the aggregate query is with a GROUP BY clause and more than one group is displayed. Part C shows the estimate variation with time. We use colour to distinguish groups. In the left image, the confidence interval of each group is depicted with a colourful triangle, the height of a triangle represents the width of associated confidence interval. Admittedly, the confidence bounds become tighter over time, so the height of all triangles gets smaller with every update. We connect the bottom edges of all triangles to form an irregular polygon, which shrinks with the query processing, thus figuratively indicates the refinement of the confidence bounds. The right image shows the variation curve of the approximate answers. Part D presents the current progress, elapsed time and remaining time, and the amount saved. During aggregation processing, COLA refreshes the graphical interface according to the specified frequency. This interface can facilitate observing the query processing and online estimates, and also help users to decide when to stop.

IV. Conclusions

In this paper, we propose a MapReduce-based online aggregation system named COLA, which was designed and implemented to provide progressive approximate aggregate answers for both single table and multiple joined tables. COLA can produce acceptable approximate answers within two orders magnitude shorter time compared to getting the accurate results, which makes it possible to save huge computing cost from the pay-as-you-go cost model in the context of cloud computing.

Acknowledgment

This research was partially supported by the grants from the Natural Science Foundation of China (No.61070055, 91024032, 91124001); the Fundamental Research Funds for the Central Universities, and the Research Funds of Renmin University (No.11XNL010); the National High Technology Research and Development Program of China (863 Program) (No. 2012AA010701).

References

[1] J. M. Hellerstein, P. J. Haas, and H. J. Wang, "Online aggregation," in *Proc. SIGMOD*, 1997, p. 171-182.
[2] P. J. Haas, J. M. Hellerstein, "Ripple joins for online aggregation," in *Proc. SIGMOD*, 1999, p. 287-298.
[3] S. Wu, S. X. Jiang, B. C. Ooi, and K. Tan, "Distributed online aggregation," *PVLDB*, vol. 2, pp. 443–454, 2009.
[4] N. Pansare, V. Borkar, C. Jermaine, and T. Condie, "Online aggregation for large MapReduce jobs," *PVLDB*, vol. 4, pp. 1135–1145, 2011.
[5] T. Condie, N. Conway, P. Alvaro, and J. M. Hellerstein, "Online aggregation and continuous query support in MapReduce," in *Proc. SIGMOD*, 2010, p. 1115-1118.
[6] J. Dean, S. Ghemawat, "MapReduce: Simplified data processing on large clusters," in *Proc. OSDI*, 2004, p. 137-150.
[7] Y. J. Shi, X. F. Meng, F. S. Wang and Y. T. Gan, "You can stop early with COLA: Online processing of aggregate queries in the cloud," in *Proc. CIKM*, 2012, p. 1223-1232.
[8] Y. J. Shi, X. F. Meng, and B. B. Liu, "Halt or continue: estimating progress of queries in the cloud," in *Proc. DASFAA*, 2011, p. 169-184.
[9] Gartner, Inc. (2011) Market share: Business intelligence platform software, worldwide. [Online]. Available: http://www.gartner.com/it/page.jsp?id=1971516
[10] IDC. (2011) IDC Digital Universe Study. [Online]. Available: http://www.emc.com/collateral/demos/microsites/emc-digital-universe-2011/index.htm
[11] P. K. Skomoroch (2009) WikiPedia page traffic statistics. [Online]. Available:http://aws.amazon.com/datasets/2596?_encoding=UTF8&jiveRedirect=1