Data & Knowledge Engineering 102 (2016) 42–56

Contents lists available at ScienceDirect

Data & Knowledge Engineering

journal homepage: www.elsevier.com/locate/datak

CAT: A Cost-Aware Translator for SQL-query workflow to MapReduce jobflow

Aibo Song[a,b], Zhiang Wu[c], Xu Ma[a,b], Junzhou Luo[a,*]

[a] School of Computer Science and Engineering, Southeast University, Nanjing, China
[b] Key Laboratory of Computer Network and Information Integration, (Southeast University), Ministry of Education, Nanjing, China
[c] Jiangsu Provincial Key Laboratory of E-Business, Nanjing University of Finance and Economics, Nanjing, China

ARTICLE INFO

Article history:
Received 13 July 2014
Received in revised form 17 October 2015
Accepted 28 December 2015
Available online 19 January 2016

Keywords:
MapReduce
SQL-to-MapReduce
Intra-SQL correlations
Cost estimation model
Hadoop
Query

ABSTRACT

MapReduce is undoubtedly the most popular framework for large-scale processing and analysis of vast data sets in clusters of machines. To facilitate the easier use of MapReduce, SQL-like declarative languages and SQL-to-MapReduce translators have attracted increasing attentions recently. The SQL-to-MapReduce translator can automatically generate the MapReduce jobflow for each SQL query submitted by users, which significantly simplifies the interfacing between users and systems. Although a plethora of translators have been developed, the auto-generated MapReduce programs still suffered from extremely inefficiency. In this paper, we attempt to address this challenge by developing a novel Cost-Aware Translator (CAT). CAT has two notable features. First, it defines two intra-SQL correlations: Generalized Job Flow Correlation (GJFC) and Input Correlation (IC), based on which a set of looser merging rules are introduced. Thus, both Top-Down (TD) and Bottom-Up (BU) merging strategies are proposed and integrated into CAT simultaneously. Second, it adopts a cost estimation model for MapReduce jobflows to guide the selection of a more efficient MapReduce jobflows auto-generated by TD and BU merging strategies. Finally, comparative experiments on `TPC-H` benchmark demonstrate the effectiveness and scalability of CAT.

1. Introduction

Recent years have seen an astounding growth of enterprises that have requirements to collect, store, and analyze enormous data sets to acquire helpful information for decision making. These requirements step over a wide spectrum of application domains, ranging from e-business and search engine to social networking. With necessary components available, e.g., networks, computational nodes and storage systems, large-scale distributed clusters can be conventionally and quickly built for many classes of data-intensive applications [1]. MapReduce has long been a widely-used distributed data processing framework in a cluster, due to many of its conspicuous merits such as flexibility, scalability, efficiency, and fault tolerance [2]. Then, various platforms implementing MapReduce have emerged, such as Google's MapReduce [3], Hadoop [4], Microsoft's Dryad [5], and Amazon's EC2 Elastic [6]. Among them, the open-source Hadoop gains the particular interests in practice.

The fundamental idea of MapReduce is to simplify the parallel processing using a distributed computing platform that offers only two interfaces: `map` and `reduce`. Programmers implement their own `map` and `reduce` functions. This low-level hand coding

* Corresponding author. Tel.: +86 25 52091010.
E-mail addresses: absong@seu.edu.cn (A. Song), zawuster@gmail.com (Z. Wu), 220111421@seu.edu.cn (X. Ma), jluo@seu.edu.cn (J. Luo).

http://dx.doi.org/10.1016/j.datak.2015.12.004
0169-023X/

provides considerable flexibility, but it extremely increases the complexity for developing an application and also for debugging a program [2,7]. To facilitate the easier use of MapReduce, the high-level language abstraction has attracted much more attentions recently. It aims to provide SQL-like declarative languages and their translators for automatic optimization. With these high-level languages, programmers only need to submit a simple logic command to the platform, just like submitting a SQL query to traditional database systems, and thus the MapReduce programs will be auto-generated. Especially to deserve to be mentioned, the SQL-like languages are more suitable for transactional applications (e.g., ad-hoc queries) rather than analytical applications [8].

A number of SQL-like declarative languages and translators have been presented in recent years. Most of them are from the industry circle, such as Hive from Facebook [9], Pig Latin from Yahoo! [10], Jaql from IBM [11], Tenzing from Google [12], and SCOPE from Microsoft [13]. However, the auto-generated MapReduce programs for many queries are often extremely inefficient compared to hand-optimized programs by experienced programmers, which incurs a series of research efforts towards designing more efficient SQL-to-MapReduce translators [7,14]. The existing translators can fall into two categories, one is for single SQL query and the other is for a batch of SQL queries. The former attempts to merge correlated operations in the operator tree of a SQL query to obtain optimized MapReduce jobflow. Our work falls into this category. Translators in the former category also include Hive [9], SCOPE [13], Tenzing [12] and YSmart [7]. Yet, the latter merges MapReduce jobs coming from different queries but without merging correlated operations in a query. Translators in this category contain Pig Latin [10] and MRShare [14].

As is well known, a SQL query is often represented as an operator tree by query optimization in database [15]. An operator might have various correlations with each other, such as sharing the input data or holding intra-job dependencies. The task of the SQL-to-MapReduce translator targets at mapping each operator to a MapReduce job, encapsulated by the `map` and `reduce` functions. Thus, a MapReduce jobflow corresponding to the original operator tree will be auto-generated. Without merging any correlated operators (e.g., Hive [9]), the MapReduce jobflow will be equivalent to the operator tree, which is obviously very inefficient. It is therefore necessary to consider intra-SQL correlations in complex queries to obtain a more simplified MapReduce jobflows. Along this line, YSmart [7] improves Hive by defining several intra-SQL correlations and merging correlated operators in the Bottom-Up traversal order. However, there are still some intractable problems in current translators. Firstly, the intra-SQL correlations and the derived merging rules in YSmart are too strict, which leaves out some correlated operators that can be merged together. Secondly, given an operator tree, the Bottom-Up traversal order is unlikely to obtain the optimal jobflow, yet other traversal orders (e.g., Top-Down) exist. Also, it is not evident to decide which jobflow is more efficient. A typical example will be given in Section 3.2. Last but not the least, any translator developed on Hive usually contains a great deal of redundant codes, which largely reduces the efficiency in practice. Motivated by these, a novel Cost-Aware Translator (CAT) for SQL-to-MapReduce translation is proposed in this paper. The main contributions are summarized in three-fold as follows:

1. We extend the definition of the intra-SQL correlation and propose several looser merging rules, so that correlated operators can be richly merged.
2. We present two merging strategies with Top-Down (TD) and Bottom-Up (BU) traversal order respectively, and integrate them into CAT simultaneously.
3. We propose a cost estimation model for MapReduce jobflows to guide the selection of a better MapReduce jobflows auto-generated by TD and BU merging strategies.
4. We build CAT, an independently designed SQL-to-MapReduce translator. Compared with Hive and YSmart, CAT can automatically generate cleaner MapReduce codes for complex SQL queries.

The remainder of this paper is organized as follows. In Section 2, we overview the CAT framework. In Section 3, we present merging rules based on two kinds of job correlations, and thus describe the algorithmic details of TD and BU merging strategies. In Section 4, we introduce the cost estimation model and the computational details of cost reduction during job merging. Experimental results will be given in Section 5. We present the related work in Section 6, and finally conclude this paper in Section 7.

2. Framework of CAT

In the realm of database, the query optimization [15], which targets at translating the SQL query into physical operators, is often guided by the cost estimation. With an accurate cost estimation model, the query optimization is only as good as its cost estimates. In light of this, a feasible way for improving the performance of SQL-to-MapReduce translators is to take the cost estimation model into consideration on the basis of intra-SQL correlations. Note that the cost estimation here is very different from that of query optimization, that is, it should substantially cover the resources exhausted by the MapReduce framework. In this paper, we aim to develop a novel Cost-Aware Translator (CAT) for SQL-to-MapReduce translation, which adopts the cost estimation model to guide the selection of different correlated job merging strategies (e.g., Top-Down and Bottom-Up). Fig. 1 illustrates the three main phases of CAT:

- Phase *i* aims to transform the algebraic representation of a SQL query into an optimized operator tree. Examples of typical operators are select, join, and aggregation. The simplest way to think of operators is as pieces of codes that are used