Future Generation Computer Systems 89 (2018) 58–67

Contents lists available at ScienceDirect

Future Generation Computer Systems

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

A gray-box performance model for Apache Spark

Zemin Chao, Shengfei Shi *, Hong Gao, Jizhou Luo, Hongzhi Wang

School of Computer Science and Technology, Harbin Institute of Technology, China

HIGHLIGHTS

- We propose a gray-box Spark performance model and achieve improvements in accuracy.
- We considered interaction between cluster resources and configurations.
- We decompose the performance prediction into several sub problems.

ARTICLE INFO

Article history:
Received 15 October 2017
Received in revised form 31 March 2018
Accepted 18 June 2018
Available online 21 June 2018

Keywords:
Gray-box method
Apache Spark
Performance prediction model
Machine learning

ABSTRACT

Apache Spark is a powerful open source data processing platform. It is getting more and more popular with the growing need of processing massive amounts of data. A performance prediction model not only helps administrators to have a better understanding of system behavior, but also is useful in performance tuning. However, considering the complex application processing mechanism of Spark, it is not an easy job to model the relationship between system performance and configuration settings.

In this paper, we present a gray-box performance model for Spark applications based on machine learning algorithms. Given a specific Spark application, the size of its input data and some key system parameters, this performance model is able to forecast its execution time according to history information. To achieve better accuracy, our model takes basic hardware information and the resource allocation strategy of Spark into consideration.

In our experiments, result shows our gray-box model is better than typical black-box approaches in most of the cases. We consider this model is helpful for further researches on Apache Spark.

1. Introduction

We are in an era of industrial informatization and there is a growing demand for handling huge amount of data. Apache Spark [1] is a widely used open source data processing platform. Unlike the MapReduce based programming framework of Hadoop [2], Spark leverages distributed memory on cluster via an efficient abstraction model called resilient distributed dataset (RDD) and performs far better than Hadoop on iterative workloads if memory is sufficient [1,3].

Spark has nearly 200 configuration parameters, some of which have a significant impact on system behavior. These parameters can be adjusted to improve the performance or meet specific requirements. This mechanism allows users to tune the parameters according to deployment environment or application features. Therefore, an application performance model is important for giving advices on proper system configurations.

The authors in [4] leverage classification algorithms to predict the relative performance improvements and guide the performance tuning process. However, classification neither tells the differences within the same class, nor gives out a prediction of execution time. Moreover, a fixed classification algorithm does not take full advantage of the features of Spark and may not be suitable for all Spark applications.

We consider there are more characteristics to exploit, and therefore in this paper, a novel gray-box method is proposed. We decompose the performance (time consumption) prediction into several sub problems, each sub problem corresponds to the performance of one stage in that application. A typical stage in Spark only carries out a few operations and is simple enough to be predicted with fine precision, then the overall time consumption of an application can be estimated according to that of stages.

Our model automatically chooses the best regression algorithm to handle each sub problem during the training period. Cluster hardware information and the resource allocation mechanism of

* Corresponding author.
E-mail addresses: 17S003056@stu.hit.edu.cn (Z. Chao), shengfei@hit.edu.cn (S. Shi), honggao@hit.edu.cn (H. Gao), luojizhou@hit.edu.cn (J. Luo), wangzh@hit.edu.cn (H. Wang).

https://doi.org/10.1016/j.future.2018.06.032
0167-739X/

Spark are taken into our consideration for better adaptability. By analyzing the interaction between hardware resources and a few system parameters, we estimate the available resources and degree of parallelism for a Spark application. In addition, a probability based refinement approach is proposed to further improve its accuracy.

As a characteristic of machine learning algorithms, a relatively balanced parameter distribution is required for training data. Thus, there is a need for initial training data collection. History information has to be collected for each application before our performance model works. But in practice, the process of generating historical data does not have to be carried out intentionally. What need to be done is to run applications with different configuration when the user needs to invoke it. Once sufficient historical information is gathered, this model could forecast the performance with system configurations and the size of input data.

Our model divides the performance prediction issue into several sub problems and leverages the Spark's strategy of launching executors in standalone mode. Although this model involves black-box machine-learning algorithms, we regard our model as a gray-box approach.

In experiments, we conclude the accuracy of our gray-box performance model is better than that of black-box approaches, which proves it is worthwhile to dig for characteristics of Spark and combine them with traditional machine-learning approaches. It is the first attempt to combine the resource allocation strategy with traditional machine learning methods to predict the performance of Spark application with system parameters.

The rest of this paper is organized as follows: In Section 2, we introduce the related work. Brief preliminaries about Spark are given in Section 3. Details of machine-learning based performance model are introduced in Section 4. Finally, in Sections 5 and 6 we present the experiments and our conclusion.

2. Related work

Performance prediction is not a novel subject for Hadoop, studies have been carried out with white-box method several years ago [5,6]. However, although Spark essentially handles the same problem with Hadoop, it is more flexible in programming and more complex in task scheduling. That makes cost based model or white-box methods hard to be implemented.

There is not much research on performance model of Spark, some efforts around Spark focus on providing high level programming model by warping Spark API [7], or to optimize the calculation in specific fields [8]. In [9], a Trial-and-Error method is brought about to tune the performance via configuring parameters, which illustrates some relationship between performance and the system parameters. In [10] the impact of system parallelism is studied.

A stage aware model based on polynomial formulas is proposed in [11]. In [12], the authors decomposed the run time of applications into stages. These two studies assume system configuration to be fixed and make prediction on how the execution time changes with the size of input data or cluster. Another stage aware collaborative filtering based approach is proposed to predict the upper and lower bounds of runtime [13], but the estimated interval does not cover all of the cases.

Machine learning approaches are also used in terms of performance prediction and system tuning. A probabilistic Bayesian approach is brought up to find optimal parameters for applications [14]. Another representative work is [4], which address performance prediction with classification algorithms in tuning process. However, its performance model does not take full advantage of hardware information or the features of Spark, and there is still room for improvements.

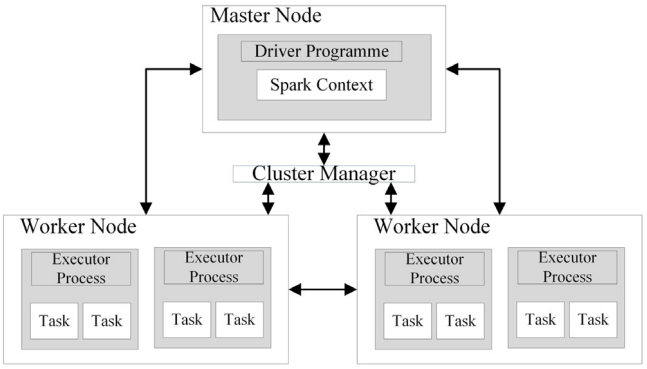

Fig. 1. The typical structure of a Spark cluster.

3. Preliminaries

In this part, we briefly introduce some details of Apache Spark, which will be helpful to understand the following sections.

As shown in Fig. 1, normally a Spark cluster consists of a master node as well as multiple worker nodes. A driver process runs on the master node, which deals with the program submitted by users, and is also responsible for requiring hardware resources from the cluster resource manager. One or more executor processes run on the worker nodes. Each executor has its own exclusive hardware resources, and handles tasks with multi threads.

Spark introduced the concept of resilient distributed datasets (RDD) [15], which is a user friendly and powerful computational abstraction. RDD adopts a mechanism called lineage to record the calculations, and the real execution will not be performed until the result is required, which provides a fault tolerance mechanism for Spark.

After submitted to Spark cluster, an application is divided into one or more jobs, and each job is divided into a few stages according to the dependency relationship between the RDDs. Tasks in the same stage have no data dependence between each other, thus they can be handled in parallel by executors.

4. A gray-box performance model

In this section, our gray-box performance model and how we leverage the characteristics of Spark are presented in details.

4.1. Overview of performance model

As proposed in [6], the performance of a specific application can be described as:

$$Pref = F(p, d, r, c) \tag{1}$$

In Eq. (1), *Pref* stands for the execution time of application p; d is the input data of that application; c represents system configurations; r is the hardware resource of cluster, and would be fixed in this expression once a specific cluster is given. When deployed in industry, a Spark cluster spends most of its time on a few fixed applications. These applications run many times on different input data with its implementation unchanged, which provide sufficient history information to train our model. For example, on wind energy farms, there are thousands of sensors collecting real-time data, and the applications which check if the generator goes well or forecast the wind direction run once in a while. For stability reasons, their implementation rarely changes.

In this scenario, we set program p to be fixed, and study the impact of configurations c and the size of input data d on the cluster