An XGB-based runtime prediction algorithm for cloud workflow tasks

Jingwei HUANG [1], Huifang LI [1], Yizhu WANG [1], Lingguo CUI [1]

[1] School of Automation, Beijing Institute of Technology, Beijing 100081, China
E-mail: 931648295@qq.com; huifang@bit.edu.cn; wangyizhu1206cuc@163.com; cuilingguo@bit.edu.cn

Abstract. Cloud Computing and workflows provide a good deployment and execution environment for scientific applications, but effective scientific workflow management depends on the estimation of task execution time to a great extent. However, due to the diversity of task inputs, the heterogeneity of resources and the high dynamics of cloud environments, as the basis of task scheduling and resource allocation in cloud computing, the task runtime estimation still faces great challenges. In this paper, we propose an XGB (Extreme Gradient Boosting)-based workflow task runtime prediction method to mine the relationship between task runtime and its static and dynamic influencing factors through feature analysis to finally result in a prediction model for online use. To verify the effectiveness of our algorithm, a series of experiments are conducted over three real scientific workflow applications, and the experimental results show that the proposed method is superior to the existing algorithms in task runtime prediction accuracy.

Keywords: Cloud Computing; Workflows; Execution Time Prediction; Feature Analysis; XGB

1. Introduction

The cloud computing provides scientists with great advantages for hosting and deploying their scientific applications in a fast and cost-effective way, like elastic resource scaling, rapid and on-demand resource provisioning as well as pay-per-use pricing scheme [1]. These benefits drive scientists to migrate their large-scale applications to the cloud and use workflow to manage its execution [2]. These scientific applications are usually data-intensive or resource-intensive, involve a large number of computing tasks, and can be described as Directed Acyclic Graphs (DAGs), where nodes represent tasks, and directed edges denote inter-task data or control dependencies. Since scientific workflow applications are often time-consuming and resource-intensive, the key to manage them efficiently is to optimize its execution time and resource usage so as to reduce their renting cost for the cloud. These goals can be achieved by optimization algorithms, such as scheduling or resource provisioning, i.e., selecting the most suitable resource for each workflow task respectively to minimize the makespan and execution cost for the whole workflow. But scheduling or resource provisioning technologies need related information about the runtime of workflow tasks to match an appropriate and best set of resources for all the workflow tasks. Hence, estimating runtime for workflow tasks is essential and critical, particularly in the dynamic and cost-centric cloud environment [3]. Accurate runtime estimation for workflow tasks, can enhance the enforceability of scheduling solutions and effectiveness of scheduling algorithms, on the other hand, can improve the utilization rate of cloud resources [4], and eventually increase the customer Quality of Service (QoS) as well as the competitiveness of cloud data centers or Cloud Service Providers (CSPs) [5].

However, most of the previous work is based on traditional methods, and more or less ignores the dynamic characteristics of the cloud, such as statistical description [6], similarity analysis [7] and distribution function [8]. For regression and classification problems, machine learning (ML) based methods are considered to be the most advanced and potential solution. It learns the relationship between a set of inputs and their related outputs through an in-depth observation of the characteristics of corresponding data. In order to capture more attributes that may affect the execution time of cloud workflow tasks, a machine learning method is adopted to provide a better solution by characterizing the dynamic changes in tasks, other static factors and cloud performance. There are three methods to apply machine learning methods to the runtime prediction of workflow tasks in the cloud: (1) time series, (2) feature analysis, and (3) the combination of time series and feature analysis.

For the time series related methods, the execution time of workflow tasks is predicted based on the time series records of previous tasks and their influencing factors. Chen J et al. [9] developed a runtime prediction method, in which some potential periodic patterns were identified from a large amount of time series data and the attenuation factors were introduced to control the influence of different time periods. Yan G et al. [10] found that the short-term contributes greater to prediction than the long-term for time series data, so they selected the neighborhood of time series through a local trend to predict the runtime for workflow tasks. Tran N et al. [11] proposed a multivariable runtime prediction method by using fuzzy technology, which considers the dynamic changes of tasks, cloud resource performance and other time factors comprehensively.

However, in a cloud like IaaS, due to the heterogeneity brought by resource provision and the performance differences resulting from resource configurations, for the same workflow task, different resource allocation may lead to different execution times. Predicting task execution time only based on time series is difficult to provide enough accurate execution time prediction for workflow tasks in the cloud. To capture the attributes of tasks, applications and resources that affect the change of task execution time, the second type of time prediction based on feature analysis is developed. This category requires workflow application properties, such as input data and application parameters, as well as specific details of cloud resources and cloud environment, such as the CPU, memory and bandwidth. Silva R F D et al. [12] estimated task execution time by analyzing the impact of fine-grained information such as I/O, memory and CPU on task execution time for cloud workflows. Pham T P et al. [13] proposed a two-stage prediction method which achieved better results than the prior single-stage ones, by taking pre-runtime parameters, such as provenance and resource features obtained from benchmark, and the fine-grained information defined as runtime parameters into account.

Although these feature analysis methods consider the complexity from the diversity of tasks and the heterogeneity of cloud resources, they cannot grasp the dynamic characteristics of the cloud computing environment, such as network bandwidth, time change and other factors affecting task execution time. This is not applicable to real-time scenes that need to process a large amount of data and deal with the performance changes of cloud resources at the same time.

To predict task execution time, we consider its influencing factors comprehensively in cloud workflows from 3 aspects, i.e., virtual resources, and static & dynamic attributes related to physical resources, then construct an XGB-based prediction model by combining the runtime and pre-runtime parameters together to realize more precision prediction.

2. Proposed Method

In this section, we first introduce the cloud workflow in Section 2.1, then briefly introduce the XGB algorithm in Section 2.2, and finally introduce our time prediction model based on XGB in Section 2.3.

2.1. Cloud workflows

With the increasingly mature and widespread application of cloud computing, more and more scientists execute their complex scientific computing processes in the cloud, that is, scientific applications or cloud workflows. A workflow application can be modeled as a task-on-node DAG $W=(T,E)$, in which $T=\{t_1,t_2,\cdots,t_n\}$ is the set of tasks and $E=\{(t_i,t_j)\,|\,t_i\in T,t_j\in T,\ i<j\}$ is the set of edges. Each edge (t_i , t_j) denotes the dependencies between task t_i and its successor t_j in the workflow. Hence, task t_j can be executed only after task t_i has completed execution and the associated data or file resulting from t_i have been transferred to t_j . Usually, the definition and execution of a workflow are accomplished by the workflow management system, and we assume that each task can be processed on one or several rented resources, e.g., virtual machines.

The complexity of tasks and the heterogeneity of cloud resources may lead to different execution times. Please note that workflow tasks differ in their instruction lengths, inputs and outputs, while virtual and physical resources have different configurations, such as disk, memory and CPU. In addition to the static (pre-run) factors available before task execution, runtime variables (such as network speed and disk I / O) that can only be obtained during task execution also affect task calculation time. The combination of pre-run and runtime attributes can obviously provide better execution evaluations and enhance the prediction accuracy.

As mentioned above, the complexity of tasks and the dynamic characteristics of cloud influence task execution time. Some attributes can be obtained before execution, while the others can only be collected during execution. Tasks are continuously monitored in cloud data centers, so that their resource consumptions can be recorded, and the usage of different resources such as CPU, memory, and I/O are captured. These time-series records are stored in a monitoring database, which need to be updated and are later used to estimate task execution time. And the considered runtime attributes are detailed in Table. 1.

Table. 1 The considered runtime parameters

Type	Runtime parameter and description
Workflow Task	Start time: start moment of task execution. End time: end moment of task execution. Tcpu_avg: average CPU used by the task. Tcpu_max: max CPU used by the task. Tmem_avg: average memory used by the task. Tmem_max: max memory used by the task.
Virtual Resource	Vcpu_util: CPU utilization of virtual computational resource. Vmem_util: memory utilization of virtual computational resource. Vmem_gps: normalized memory bandwidth of virtual computational resource. Vnet_in: normarlized in coming network traffic virtual computational resource. Vnet_out: normarlized out going network traffic of virtual computational resource. Vdisk_io: disk IO of virtual computational resource.
Physical Resource	Mcpu_util: CPU utilization of physical resource. Mmem_util: memory utilization of physical resource. Mmem_gps: normalized memory bandwidth of physical resource. Mnet_in: normarlized in coming network traffic of physical resource. Mnet_out: normarlized out going network traffic of physical resource. Mdisk_io: disk IO of physical resource.

The pre-runtime attributes are available before workflow execution, which describes workflow tasks and their execution environments. In particular, every cloud resource is companied with its own ID, memory size, CPU's cores and other attributes, and every task also has its own name and type. For a specific cloud resource or workflow task, its pre-runtime information is relatively fixed and uniquely adhere to itself. The considered pre-runtime parameters and their details are listed in Table. 2.

Table. 2 The considered pre-runtime parameters

Type	Runtime parameter and description
Workflow Task	T_name: name of a cloud workflow task. T_type: type of a cloud workflow task. Tplan_cpu: the needed CPU number of a cloud workflow task. Tpan_mem: the needed memory size of a cloud workflow task.
Virtual Resource	V_id: ID of virtual computational resource. Vcpu_request: CPU request of virtual computational resource. Vcpu_limit: CPU limit of virtual computational resource. Vmem_request: memory request of virtual computational resource.
Physical Resource	M_id: ID of physical resource. Mcpu_num: CPU number of physical resource. Mmem_size: memory size of physical resource. Mdisk_size: disk size of physical resource.

2.2. Extreme gradient boosting(XGB)

Recently, XGB outperforms in feature analysis prediction problem, especially in many machine learning competitions, such as those held by Kaggle. Comparing other tree models, XGB contains penalty term in its objective function, so that it can penalizes the complexity of the model to obtain better performance, its regularized objective is calculated as formula (1):

$$L=\sum_i l\left(\hat{y}_i, y_i\right)+\sum_k \Omega\left(f_k\right) \tag{1}$$

where i represents the i-th sample; l is a differentiable convex loss function that measures the difference between the prediction $\hat{y}_i$ and the target y_i; the second term Ω is the tree structure complexity, and another name is penalty term; f_k represents the structure of the k-th classification and regression tree. The gain after split, which is used to evaluate the split candidates, is given by formula (2):

$$G_{split}=\frac{1}{2}\left[\frac{\left(\sum_{i\in I_L} g_i\right)^2}{\sum_{i\in I_L} h_i+\lambda}+\frac{\left(\sum_{i\in I_R} g_i\right)^2}{\sum_{i\in I_R} h_i+\lambda}-\frac{\left(\sum_{i\in I} g_i\right)^2}{\sum_{i\in I} h_i+\lambda}\right]-\gamma \tag{2}$$

where I_L and I_R are the sample sets of left and right nodes after the split, and $I=I_L\cup I_R$; g_i and h_i are first and second order gradient statistics on the loss function; γ is the penalty term, penalizing the adding new leaves; λ is the regular term coefficient.

The purpose of the XGB algorithm is to construct a bunch of classification and regression trees. That is, at each step of the algorithm, the point with the maximum gain which is greater than the threshold, is selected to be a splitting point, and then the splitting point is split to obtain a new tree. After the training is completed, in order to predict the score of a sample, the algorithm computes each tree according to the characteristics of the sample to get a score, and adds all scores to get the predicted value of the sample.

2.3. The execution time prediction model

In this section, we introduce the execution time prediction model. Its structure and workflow are shown in Fig. 1.

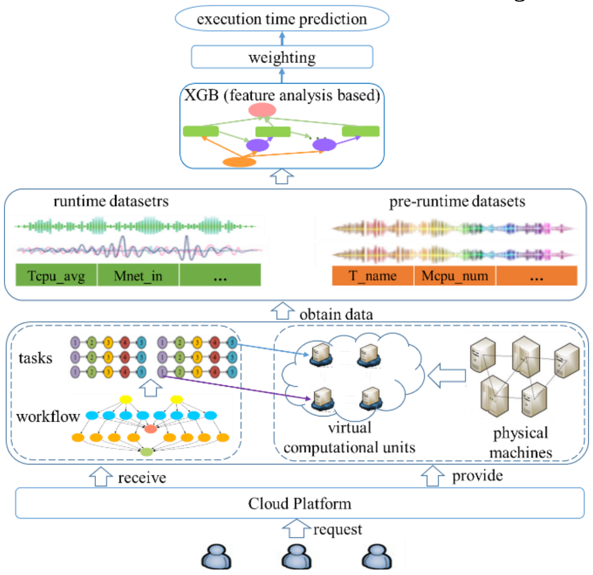


Fig. 1 The procedure of our task execution time prediction model

Among them, users initially request resources for their workflow on the cloud platform, and then provide different computing resources to meet their needs. For the arriving workflow, collect the pre-run parameters of its task and candidate computing resources, and record the corresponding run-time parameters and task execution time. Once the historical data of pre-run and runtime parameters are obtained, we input them into XGB to predict the execution time of workflow tasks through feature analysis. Because the execution time prediction based on feature analysis can fully reflect the influence of pre-run and run-time parameters, we get a well-trained execution time prediction model and apply it into real-time scenarios to predict task execution time for users. In addition, the implementation details of our method are also outlined in the WTSFA algorithm, and its pseudo code corresponding to the above process is as follows:

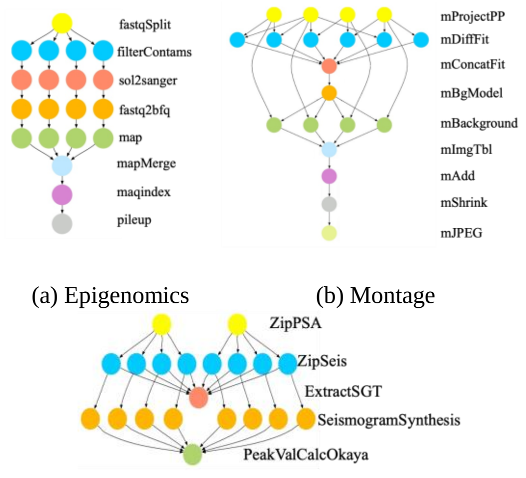
```
Algorithm: WTSFA Approach for Runtime Prediction
  Input: $t_i$ , A task of cloud workflow, $1\le i\le n$;
     $v_i$ , A virtual resource where task $t_i$ is executed;
     $m_i$ , A physical resource where virtual resource $v_i$ is virtualized;
  Output: The predicted task execution time set $P\{\ \}$ , in which $p_i$ is runtime time of $t_i$ hosted by $v_i$ in $m_i$ ;
1.    $D\{\ \}\leftarrow\varnothing, S\{\ \}\leftarrow\varnothing, P\{\ \}\leftarrow\varnothing$;
2.    Sort all tasks $t_i$ in ascending order according to their execution start time, and get the sorting table $T$ ;
3.    while there are unreached tasks do
         4.   for each arriving $t_i\in T$ do
         5.      $S\{\ \}\leftarrow$ collect pre-runtime parameters
         6.      $D\{\ \}\leftarrow$ record runtime parameters
         7.      Import $D\{\ \}$ and $S\{\ \}$ to XGB to predict the execution time, and put the result to $P\{\ \}$
         8.   end for
9.    end while
         10. return $P\{\ \}$
```

When cloud workflow task t_i is reached, the reached moment is recorded as sub_i. First of all, we extract the pre-runtime parameters at sub_i and the runtime parameters monitored before sub_i of task t_i, virtual resource v_i and physical resource m_i. Secondly, the runtime and pre-runtime parameter set are fed to XGB to calculate the task execution time, so that the complexity and diversity of tasks, the heterogeneity of cloud resources and the impact from the dynamic characteristics of cloud environments can be fully considered.

3. Experimental Design and Result Analysis

3.1. Data description

The Pegasus workflow management system is employed to execute workflow in the cloud, so that the monitoring data, like the pre-runtime and runtime parameters as well as the corresponding execution time can be collected. Our experimental data is generated according to the Extensive Markup Language (XML) files of workflows in WorkflowSim. Three popular scientific workflows are considered: Epigenomics, Montage and CyberShake. In general, there are 12000 tasks for each workflow, which belong to different types with different number of tasks and execution times. The number of tasks for each type and the value range of its execution times are listed in the following tables, corresponding to their structures and XMLs in WorkflowSim. The structures of three workflows with small sizes are shown in Fig. 2, and detailed information of the generated data is summarized in Tables. 3-5.



(a) Epigenomics (b) Montage

(c) CyberShake

Fig. 2 The considered workflows

Table. 3 The information of generated tasks in the Epigenomics

Task name	Value(s)	Task ID	Number
fastqSplit_chr21	(20, 100)	1	500
filterContams_chr21	(0, 5)	2	2500
sol2sanger_chr21	(0, 1.75)	3	2500
fastq2bfq_chr21	(0, 3)	4	2500
map_chr21	(1000, 27500)	5	2500
mapMerge_chr21	(10, 25)	6	500
maqindex_chr21	(0, 0.06)	7	500
pileup_chr21	(1500, 5000)	8	500

Table. 4 The information of generated tasks in the CyberShake

Task name	Value(s)	Task ID	Number
ZipPSA	(0, 4)	1	400
ZipSeis	(0, 10)	2	400
ExtractSGT	(60, 200)	3	800
SeismogramSynthesis	(20, 70)	4	5200
PeakValCalcOkaya	(0.6, 1.8)	5	5200

Table. 5 The information of generated tasks in the Montage

Task name	Value(s)	Task ID	Number
mProjectPP	(11, 15)	1	2400
mDiffFit	(9, 13)	2	4320
mConcatFit	(0, 60)	3	480
mBgModel	(0, 100)	4	480
mBackground	(9.5, 12)	5	2400
mImgTbl	(0, 70)	6	480
mAdd	(0, 100)	7	480
mShrink	(0, 30)	8	480
mJPEG	(0, 3.5)	9	480

As shown in the above tables, the value range of tasks' execution time in each workflow can be classified into three groups, which are referred to the small, medium and large respectively. The small group includes tasks whose execution time's upper bound is lower than 5, like sol2sanger_chr21 and maqindex_chr21 in Epigenomics workflows, ZipPSA and PeakValCalcOkaya in CyberShake workflows, and mJPEG in Montage workflow. The medium group reflects the tasks whose execution time is less than 60, such as mapMerge_chr21 in Epigenomics workflows, ZipSeis in CyberShake workflows, and mProjectPP in Montage workflows. The rest tasks belong to the large group, for example, map_chr21 in Epigenomics workflows, ExtractSGT in CyberShake workflows, and mBgModel in Montage workflows. Different groups of tasks may have different impact on the performance of execution time prediction methods, which will be examined in experiments.

3.2. Experiment configuration

In this section, the experiment configuration will be illustrated, including the experiment environment and the parameters of all the evaluated methods. Our internal cloud infrastructure is composed of a cloud, with three different physical servers, as listed in Table. 6. In order to guarantee a fair evaluation, we use similar virtual resource types for data generation, referring to those provided in the commercial cloud, as described in Table. 7.

Table. 6 The information of the physical servers

Sever	CPU	Memory	Disk
1	Intel Xeon E5-2637 V3 3.5 GHz 4C*2	64G DDR4*12	1T SAS 3.5 7.2K*2
2	Intel Xeon E5-2620 V4 2.1GHz 8C *2	128G DDR4*12	2T SAS 3.5 7.2K*6
3	Intel Xeon E5-2650 V4 2.2GHz 12C*2	128G DDR4*12	4T SAS 3.5 7.2K*2

Table. 7 The generated virtual resources

Virtual resource ID	The reference	CPU_{vm}	Mem_{vm}
1	t2.small (EC2)	1	2
2	t2.medium (EC2)	2	4
3	t2.xlarge (EC2)	4	16
4	t2.2xlarge (EC2)	8	32

To provide a fair comparison among different hardware of cloud providers, we choose the same operating systems CentOS 7 (64-bit) for virtual resources, and the type1, type2, type3 as well as type4 of virtual resources correspond to t2.small, t2.medium, m4.xlarge and m4.2xlarge in Amazon EC2. Extensive experiments are conducted on Python 3.6.1, Anaconda 3 (64 bits) to evaluate the proposed task execution time prediction approach.

We choose regression methods solely based on pre-runtime or runtime parameters in this paper. It is difficult to compare our work with other research efforts, such as Chen J [9] and Yan G [10], since some of them need access to different applications and detailed hardware information, which are not the case for some of the considered workflow applications in this work. Secondly, most of these methods require more detailed hardware information, which is an unrealistic assumption for public clouds.

Related work often uses three main regression algorithms: linear regression, neural networks, and regression trees. In this work, we use XGB. As mentioned earlier, XGB has been proved very successful in feature analysis and prediction. More specifically, we study a total of five different regression algorithms: bagging ridge (BR), random forest (RF) and AdaBoost (ADA). These methods are implemented by using the sklearn library. Table. 8 lists their adjustment parameters.

3.3. Experimental results and analysis

3.3.1. Root mean squared logarithmic error

We present and discuss all the experimental results to verify the performance of our proposed approach in this work. Specially, instead of using the mean value of the obtained results of all tasks in the workflow, extra experiments for the whole workflows are also conducted, of which the compositions refer to XML files in WorkflowSim. And 10-fold cross-validation is used, meanwhile every experiment is repeated 10 times.

Table. 8 Parameters of the adopted Regression Algorithms

Algorithm	Parameters
BR	alpha = 10, base_estimator = ridge, n_estimators = 50
RF	n_estimators = 100, max_features =7
ADA	n_estimators = 100
XGB	max_depth =9, n_estimators=100, learning_rate =0.08, subsample=0.85, reg_lambda=0.19, feval = root mean squared logarithmic error

Assuming m is the number of output values, i.e., the number of the task, the Root Mean Square Logarithm Error (*RMSLE*) is recorded and can be calculated as follows:

$$RMSLE = \sum_{i=1}^{m}\left(\hat{y}_i - y_i\right)^2 \qquad (3)$$

Once the runtime parameters are updated, the feature analysis can be conducted. Here, XGB is exploited and compared with RF, BR and ADA. All these ML-based methods are ensemble ones, which perform well in feature analysis-based prediction problems. Their *RMSLE* values are compared in Figs. 3-5 and the corresponding analysis is followed.

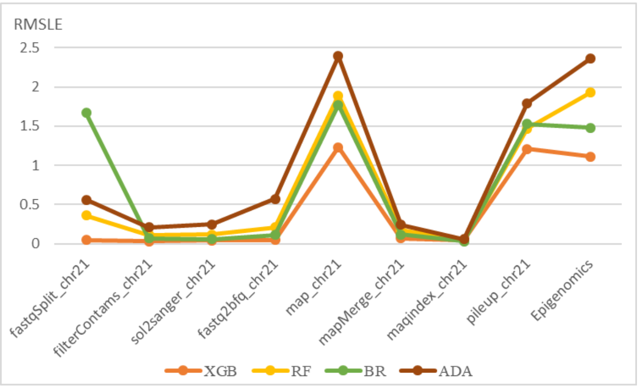


Fig. 3 *RMSLE* comparisons over the subtasks in Epigenomics workflows

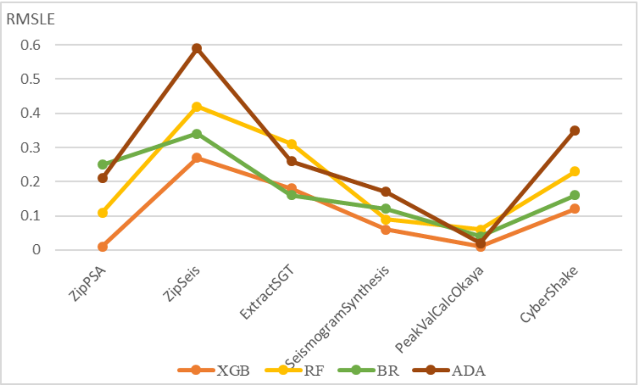


Fig. 4 *RMSLE* comparisons over the subtasks in CyberShake workflows

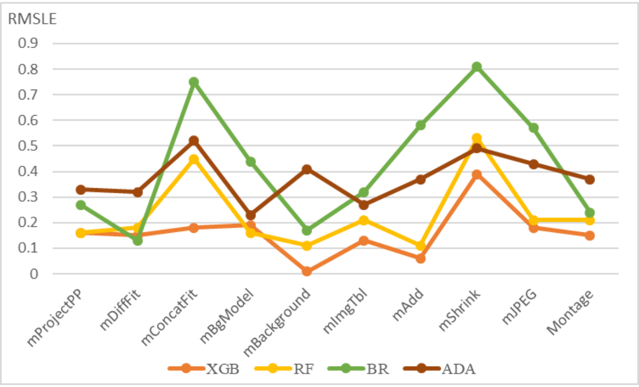


Fig. 5 *RMSLE* comparisons over the subtasks in Montage workflows

From these figures, it is found that the *RMSLE* curves of XGB are always at the bottom, indicating the superiority of XGB over the other three methods. The possible reason is as follows. When comparing with RF and ADA, pruning naturally occurs during the leaf node splitting process of XGB. And it is well known that, pruning helps trees improve the generalization ability. For BR, its weakness is strongly presented in the predictions of Montage, since its base learner is Ridge, which may fail when there is a high collinearity between characteristic variables, such as the functional relationship between variables X1 and X2. In a word, XGB performs well resulting from its pruning operations.

3.3.2. Elapsed time

Additionally, elapsed time of different methods are also compared, the same with the previous prediction part, as shown in Figs. 6-8.

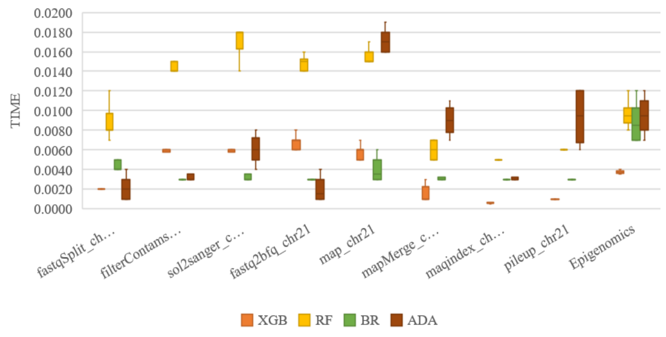


Fig. 6 Elapsed time comparisons over the subtasks in Epigenomics workflows

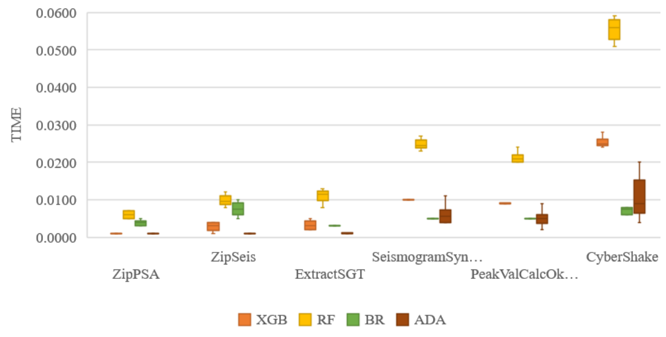


Fig. 7 Elapsed time comparisons over the subtasks in CyberShake workflows

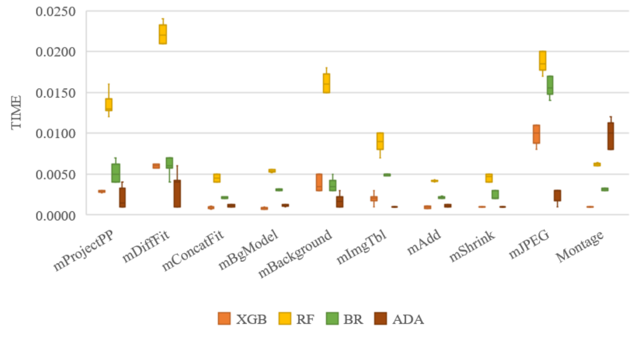


Fig. 8 Elapsed time comparisons over the subtasks in Montage workflows

Focusing on the elapsed time comparisons, the time of XGB is not the least one in all cases, but it always maintains a relatively low level. Comparing with other ensemble methods, the elapsed time of XGB changes much less, illustrating that XGB is stable in computation, and on the other hand, XGB’s parallel computing operations help it keeps an average computational cost. Combining with the previous *RMSLE* analysis, we can find that, XGB obtains the best prediction result without increasing time overhead. These results demonstrate the effectiveness of XGB in tasks execution time prediction for cloud workflows, which is based on feature analysis.

4. Conclusions and Future Work

The task execution time estimation has a significant impact on workflow scheduling and resource allocation in the cloud. In this paper, aiming to predict the task execution time for workflows, we propose an XGB-based runtime prediction method for cloud workflow tasks. First, we analyze the influencing factors, like CPU, Memory and bandwidth, and pre-process these data to construct our datasets. Then, we establish a feature analysis-based model by using XGB to realize task runtime prediction. Experimental results show that the proposed algorithm is superior to the existing peers in the accuracy of task runtime prediction, and *RMSLE* and running time. Especially in *RMSLE*, compared with other algorithms, XGB's leaf node splitting process prunes naturally, which improves the generalization ability of the tree. Therefore, it has higher computation efficiency.

As future work, we plan to study how the relationship between workflow tasks (such as data transmission and dependencies), and the tasks’ category attributes affect their execution time. For improving the performance of runtime prediction for workflow tasks, more latest and effective machine learning methods will be further studied and leveraged. In addition, we intend to apply this prediction method to workflow scheduling for improving the operability of scheduling solutions in the real application scenarios.

Acknowledgements

This work is supported in part by the National Natural Science Foundation of China under Grant No. 61836001; and in part by the National Key Research and Development Program of China under Grant No. 2018YFB1003700.

References

[1] Sahni J, Vidyarthi D. A Cost-Effective Deadline-Constrained Dynamic Scheduling Algorithm for Scientific Workflows in a Cloud Environment. IEEE Transactions on Cloud Computing, 2018, 6(1): 2-17.
[2] Deelman E, Gannon D, Shields M, et al. Workflows and e-Science: An overview of workflow system features and capabilities. Future Generation Computer Systems, 2009, 25(5): 528-540.
[3] Rodriguez M A, Buyya R. Deadline Based Resource Provisioning and Scheduling Algorithm for Scientific Workflows on Clouds. IEEE Transactions on Cloud Computing, 2014, 2(2): 222-235.
[4] Wang X, Cao B, Hou C, et al. Scheduling Budget Constrained Cloud Workflows with Particle Swarm Optimization. IEEE Conference on Collaboration and Internet Computing. IEEE, 2016: 219-226.
[5] Abrishami S S, Naghibzadeh M M, Epema D D. Deadline-constrained workflow scheduling algorithms for Infrastructure as a Service Clouds. Future Generation Computer Systems, 2013, 29(1): 158-169.
[6] Chirkinab A M, Kovalchuka S V. Towards Better Workflow Execution Time Estimation. International Conference on Future Information Engineering, 2014: 216-223.
[7] Pietri I, Juve G, Deelman E, et al. A Performance Model to Estimate Execution Time of Scientific Workflows on the Cloud. Workshop on Workflows in Support of Large-scale Science. IEEE, 2014: 11-19.
[8] Chirkin A M, Belloum A S Z, Kovalchuk S V, et al. Execution Time Estimation for Workflow Scheduling. IEEE 9th Workshop on Workflows in Support of Large-Scale Science. 2014: 1-10.
[9] Chen J, Li K, Rong H, et al. A Periodicity-based Parallel Time Series Prediction Algorithm in Cloud Computing Environments. Information Sciences, 2018: 1-32.
[10] Yan G, Jia S, Ding J, et al. A Time Series Forecasting based on Cloud Model Similarity Measurement. Soft Computing, 2018, 1-12.
[11] Tran N, Nguyen T, Nguyen B M. A Multivariate Fuzzy Time Series Resource Forecast Model for Clouds using LSTM and Data Correlation Analysis. International Conference on Knowledge-Based and Intelligent Information & Engineering Systems. 2018, 636-645.
[12] Silva R F D, Juve G, Rynge M, et al. Online Task Resource Consumption Prediction for Scientific Workflows. Parallel Processing Letters, 2015, 25(3): 2-26.
[13] Pham T P, Durillo J J, Fahringer T. Predicting Workflow Task Execution Time in the Cloud using A Two-Stage Machine Learning Approach. IEEE Transactions on Cloud Computing, 2017, 1-13.