Parameter Optimization on Spark for Particulate Matter Estimation

Zhenyu Yu

School of Computer and Information Technology

Northeast Petroleum University

Daqing, China

yuzy@stu.nepu.edu.cn

Zhibao Wang

School of Computer and Information Technology

Northeast Petroleum University

Daqing, China

wangzhibao@nepu.edu.cn

Lu Bai

School of Computing

Ulster University

Belfast, UK

l.bai@ulster.ac.uk

Liangfu Chen1,2

1 State Key Laboratory of Remote Sensing Science

Aerospace Information Research Institute of Chinese

Academy of Sciences Chinese Academy of Sciences;

2 University of Chinese Academy of Sciences

Beijing, China

chenlf@radi.ac.cn

Jinhua Tao1,2

1 State Key Laboratory of Remote Sensing Science

Aerospace Information Research Institute of Chinese

Academy of Sciences Chinese Academy of Sciences;

2 University of Chinese Academy of Sciences,

Beijing, China

taojh@radi.ac.cn

ABSTRACT

With the rapid growth of remote sensing satellites, the volume of remote sensing data has been continuously increasing, which makes it necessary to utilize the big data platform for the rapid practical application of remote sensing inversion algorithms. This paper proposes an atmospheric remote sensing inversion processing method based on Spark. As a popular large-scale data processing framework, the memory-based iterable calculation model of Spark makes it suitable for the application of atmospheric remote sensing inversion. In this paper, we use the Spark computing framework to calculate the average value of the particulate matter in China over the past 10 years and the running time is much faster than the traditional single-node method. Furthermore, how Spark configuration parameters affect the performance of the task is explored. Different regression models in XGBoost are used to evaluate the performance of the parameters obtained by the parameter optimization algorithm in order to find the Spark optimal configuration parameters that meet the requirements.

CCS CONCEPTS

• Applied computing • Physical sciences and engineering • Earth and atmospheric sciences

**This is the author's version of the work. It is posted here for your personal use. Not for redistribution. The definitive Version of Record was published in https://dl.acm.org/doi/10.1145/3456389.3456406**

Keywords

Particulate matter estimation, Spark, Parameter optimization, Performance prediction

ACM Reference format:

Zhenyu Yu, Zhibao Wang, Lu Bai, Liangfu Chen and Jinhua Tao. 2021. Parameter Optimization on Spark for Particulate Matter Estimation. In *Proceedings of 2021 Workshop on Algorithm and Big Data (WABD 2021)*. *ACM, Fuzhou, China, 5 pages*. https://doi.org/10.1145/1234567890

1. Introduction

In recent years, with the continuous launch of artificial satellites and the rapid development of sensor technology, the amount of remote sensing image data has shown explosive growth. The Sentinel-5P satellite generates approximately 3TB data and the Landsat8 satellite also generates about 3TB OLI/TIRS Level-2 data monthly over China. Such an explosive growth of remote sensing images poses a huge challenge to both academia and industry [1]. With the rapid development of big data technology today, the emergence of Hadoop and Spark [2] has enabled us to quickly process large-scale data. Spark's memory-based computing model and multi-language programming interface can improve the integration efficiency of the program and make more reasonable use of resources.

Spark is a widely-used open source big data computing framework [2], and it is also a memory-based iterable big data processing framework. This feature of Spark is theoretically compatible with the need for complicated computing process for remote sensing inversion and computing tasks involved the calculation of a large amount of data. Since the introduction of the big data framework, performance optimization has been widely used. There are mainly two types of models for the performance improvement of MapReduce jobs in Hadoop [3, 4] - white box model and black box model for optimization. The white box model mainly focuses on the operating principle of the system. The black box model treats the complex structure of the system as an unknowable black box, and only focuses on the relationship between parameters and job operation performance. In the previous studies, the white box model was used as the research object to improve the running performance of Spark [5]. However, due to the high coupling between the white box model and the system, it has the disadvantages of high construction cost and low flexibility.

Therefore, this paper focuses on the use of black box model to explore the relationship between the configuration parameters the Spark uses for the application task and the performance of the entire application task. Recently, there are some related works around the optimal inversion calculation and improvement of configuration parameters for the large-scale data volume on Spark. Petridis et al. proposed a trial and error method for performance tuning through configuration parameters [6], which explained the relationship between Spark's operating performance and configuration parameters. However, there are more than 100 configurable parameters when the Spark application is submitted; this trial and error method requires a certain amount of manual experience and time cost. Li et al. [7] studied the impact of parameter parallelism on the running performance of Spark tasks. Wang et al. [8] tried to solve the problem of Spark configuration parameter optimization with a classification method, using the default configuration parameters as the benchmark value, and classify the different values of the configuration parameters according to the improvement of job performance. However, the prediction accuracy of the classification model on the running performance of Spark tasks needs to be improved.

To address the above challenges, based on literature [6–9] and experiments we select important configuration parameters that have a greater impact on job operation performance from more than 100 configuration parameters. In this work, a Spark-based PM10 inversion calculation data set is built based on XGboost regression algorithm and a Spark task performance prediction model is established. We then evaluate the model by using the several heuristic algorithms in parameter configuration in order to find the most suitable heuristic search algorithm.

1. Methods
	1. Spark Performance Prediction Model

According to the particulate matter estimation inversion algorithm and the Spark task performance prediction requirements, the meta-model for the parameter optimization is defined as a quadruple, as shown in equation (1).

$P=F(p,s,t,h)$ (1)

In which, P represents performance, which refers to the task running time in this paper; p is a set of parameters; s is the amount of data to be processed; t represents the task itself, and h represents the available resources in the cluster. The experiments in this paper are all running on a cluster with a single task, so the resources preemption in multitasking is not considered. We mainly focus on the influence of data volumes and configuration parameter p on performance P when other parameters are fixed. Figure 1 shows the architecture of the parameter optimization method.

****

**Figure 1: Architecture of parameter optimization for Spark model**

* 1. Parameters to Be Optimized

When submitting a task to a Spark cluster, there are more than 100 optional configuration parameters. Therefore it is unrealistic to optimize all of the parameters. Also in practical applications there is no need to set so many parameters for a task and default values are used for most of the parameters. This work refers to the literature [6–8] to select several parameters that have a large impact on the experimental results. The selected parameters are as follows:

* Spark.executor.number: Specifies the number of initial executors.
* Spark.executor.memory: Specifies the total amount of memory allocated to the driver process, which affects the resources that can be used by the job.
* Spark.executor.cores: Specifies the number of CPU cores that each executor can use, which affects the resources that can be used by the job.
* Spark.driver.cores: Specifies the number of CPU cores allocated to the driver process, which affects the resources available for the job.
* Spark.driver.memory: Specifies the total amount of memory allocated to the driver process, which affects the resources that can be used by the job.
* Spark.default.parallelism: Specifies the default number of tasks for each stage, which affects the degree of parallelism of the job.
	1. Establishment and Evaluation of Performance Model

In this paper, the regression model in XGBoost [9] is used to establish a task performance prediction model running on Spark. Spark configuration parameters and data volume are used as independent variables, and the running time of the Spark task is used as a dependent variable. MAE, R-Squared, RMSE are used to evaluate the results.

XGBoost is an open-source machine learning project developed by Chen et al. [10]. It efficiently implements the Gradient Boosting Decision Tree (GBDT) algorithm and has made many improvements in the algorithm and engineering. XGBoost is an improvement of the gradient boosting algorithm and uses the 2nd order derivative as an approximation. In addition, a regularization term is applied to the loss function. The objective function during training consists of two parts, the first part is the loss of the gradient boosting algorithm, and the second part is the regularization term. The loss function is defined as equation (2) [10]:

$L(∅)=\sum\_{i=1}^{n}l\left(y\_{i}^{'}-y\_{i}\right)+\sum\_{k}^{}Ω(f\_{k})$ (2)

Where n is the number of training function samples, and l is the loss of a single sample assuming it is a convex function. $y\_{i}^{'} $is the predicted value of the training sample by the model, and $y\_{i}$ is the true label value of the training sample. The regularization term defines the complexity of the model as in equation (3) [10]:

$Ω\left(f\right)=γT+\frac{1}{2}λ\left‖ω\right‖$ (3)

In which, γ and λ are manually set parameters, ω is the vector formed by the values of all leaf nodes of the decision tree, and T is the number of leaf nodes.

* 1. Parameter Optimization

The process of solving the optimal configuration parameters is to search the solution space. In order to improve the search efficiency, this paper uses the Genetic Algorithm (GA) [11] combined with the parameter performance prediction model discussed in section above to optimize the Spark parameters. Genetic algorithm is an optimal solution search algorithm that simulates biological evolution. Its special internal mechanism can ensure that the approximate optimal solution is found in a limited solution space. Its optimization method has a positive effect to avoid falling into local optimum [11]. In this paper, the performance of the atmospheric remote sensing inversion calculation task is related to the configuration parameters and the amount of data. For each configuration parameter to be optimized as a chromosome gene, the individual in the genetic algorithm is a vector which is composed of the parameters to be optimized. The specific optimization process is shown in Figure 2. Spark default configuration parameters are used as the input of the Spark performance prediction model trained in Section 2.1 to obtain task running time. Then this set of default configuration parameters and task running time are as the input of the genetic algorithm, and the performance model is used as the fitness function to judge the pros and cons of each set of parameters. Crossover and mutation is implemented on the configuration parameters to obtain a new set of parameter configurations. The new set of parameter configurations is passed into the Spark performance prediction model to predict its performance (e.g. task running time). If the performance reaches the required value, the optimization process will stop and the configuration parameters will be saved. Otherwise the previous process will continue until the search meets the conditions for configuration parameters. The specific process of the algorithm is as follows:

1. Initial population generation: The initial population is randomly generated and set to P. P represents a single individual in the population, that is, the chromosome formed by encoding the Spark configuration parameters to facilitate subsequent operations (such as crossover and mutation). In this paper, the genes in the chromosome are each configuration parameter, and then each chromosome is composed of a combination of 12 parameters.
2. Calculate the fitness of an individual: According to the Spark performance model as the fitness function F of the genetic algorithm, the model is used to calculate the fitness of the chromosomes in the configuration parameter population.
3. Selection operation: The roulette selection method is used to select configuration parameter populations and select k configuration parameter chromosomes to continue genetic operations.
4. Crossover and mutation operations: Perform genetic operations on the selected chromosomes, including single point crossover and uniform mutation, and then put the new configuration parameter chromosomes into the original population to generate a new population Pnew.
5. Population update: replace the initial population P with the new population Pnew obtained after crossover and mutation.
6. Decision making of termination conditions: When the specified number of iterations t is reached, output the configuration parameters of the closest target performance; otherwise, go to step 3.

****

**Figure 2: Parameter optimization process**

****

**Figure 3: Model training and selection process**

1. Experiments and Results Analysis

In the experiments, we use a cluster consists of 1 master node and 4 slave nodes. The master node serves as the Master in the Spark cluster, and the slave node serves as the Worker. The operating system is Centos7.3, and the hardware configuration of the master node is Pentium(R) Dual-Core CPU E6700 @ 3.20GHz, 4G memory, the hardware configuration of the slave node is Intel(R) Core(TM) i5-7500 CPU @ 3.40GHz 16G memory. The master node does not participate in the calculation. The experimental data used is the aerosol optical thickness AOD (Aerosol Optical Depth) data in the MYD04\_3K [12] data of the past 10 years obtained from the NASA website and stored on HDFS after preprocessing. The estimated value of particulate matter is calculated using the inversion algorithm. The selected region covers entire China, with a total of nearly 48,000 rows of data. First, the daily data is merged into a TIF file and then is converted into pixel values and written into a text file which is then transferred to HDFS. Each file size is about 45M. In total 3650 files are selected for experimental processing. Compared with the traditional single-node method, the Spark-based method shows significantly higher operating efficiency.

1. Evaluation of Performance Prediction Models

To evaluate the performance of the prediction models, this paper uses the tested PM10 inversion algorithm and preprocessed remote sensing data with randomly generates multiple sets of configuration parameters. The experimental steps for evaluating the three regression methods including Linear, Gamma, and Tweedie in XGBoost are shown in Figure 3.

In the data preprocessing stage, the experimental data is divided into a training set and a test set. Three regression algorithms are used to train the training set to obtain the corresponding models. Then predicted values are gained by applying the trained models on the test set. MAE, R-Squared and RMSE are used to evaluate the performance of the model. The experimental results are shown in Table 1 for comparison.

**Table 1: Evaluation of three regression models in XGBoost**

|  |  |  |  |
| --- | --- | --- | --- |
| Percentage of training set (%) | Linear | Gamma | Tweedie |
| MAE | R2 | RMSE | MAE | R2 | RMSE | MAE | R2 | RMSE |
| 70 | 7.664 | 0.887 | 9.619 | 7.421 | 0.894 | 9.308 | 7.968 | 0.881 | 9.859 |
| 80 | 7.333 | 0.893 | 9.13 | 7.125 | 0.897 | 8.950 | 7.59 | 0.881 | 9.604 |
| 90 | 7.458 | 0.880 | 9.316 | 7.000 | 0.896 | 8.678 | 7.486 | 0.882 | 9.264 |

It can be seen from the table that Linear regression is not stable when increasing the training volume, and the three evaluation values fluctuate. For Gamma regression, when the training set increases, the MAE and RMSE both decrease, and R -Squared fluctuates. Compared with the other two models, Tweedie regression is more stable.

1. Performance Optimization Effect

The running time of using the optimization parameters obtained in Section 3.1 are compared with that of using Spark default configuration parameters in the same environment, as shown in Figure 4.



**Figure 4: Running time of the Spark tasks using different configuration parameters (optimal configuration parameters and Spark default configuration parameters)**

It can be seen that the task using the optimized configuration parameters takes much less time than the same task using the Spark default parameters. The result shows that the automatic optimization algorithm can effectively shorten the execution time of Spark tasks.

1. Conclusion

Based on the XGBoost regression model and genetic algorithm, this paper proposes a method that can automatically optimize the configuration parameters of Spark tasks. The use of optimized configuration parameters shows a significant reduction in task running time compared with using the default parameters on tasks for large-scale data. Moreover, the parameter optimization process consumes a shorter time. This proves the method proposed in this paper can effectively reduce the running time of Spark tasks and improve operating efficiency.

ACKNOWLEDGMENTS

This work was supported in part by TUOHAI special project 2020 from Bohai Rim Energy Research Institute of Northeast Petroleum University under Grant HBHZX202002 and project of Excellent and Middle-aged Scientific Research Innovation Team of Northeast Petroleum University under Grant KYCXTD201903.

REFERENCES

[1] A. Plaza, J.A. Benediktsson, J.W. Boardman, J. Brazile, L. Bruzzone, G. Camps-Valls, J. Chanussot, M. Fauvel, P. Gamba, A. Gualtieri, M. Marconcini, J.C. Tilton and G. Trianni. 2009. Recent advances in techniques for hyperspectral image processing. *Remote Sensing of Environment.* (2009). https://doi.org/10.1016/j.rse.2007.07.028

[2] M. Zaharia, M. Chowdhury, M.J. Franklin, S. Shenker and I. Stoica. 2010. Spark: Cluster computing with working sets. In *2nd USENIX Workshop on Hot Topics in Cloud Computing, HotCloud 2010*.

[3] H. Herodotou, H. Lim, G. Luo, N. Borisov, L. Dong, F.B. Cetin and S. Babu. 2011. Starfish: A self-tuning system for big data analytics. In *CIDR 2011 - 5th Biennial Conference on Innovative Data Systems Research, Conference Proceedings*.

[4] N. Yigitbasi, T. L. Willke, G. Liao and D. Epema. Towards machine learning-based auto-tuning of MapReduce. In *Proceedings - IEEE Computer Society's Annual International Symposium on Modeling, Analysis, and Simulation of Computer and Telecommunications Systems, MASCOTS*. https://doi.org/10.1109/MASCOTS.2013.9.

[5] H. Zhang, Z. Liu and L. Wang. 2018. Tuning performance of spark programs. In *Proceedings - 2018 IEEE International Conference on Cloud Engineering, IC2E 2018, Institute of Electrical and Electronics Engineers Inc.* pp. 282-285. https://doi.org/10.1109/IC2E.2018.00057

[6] P. Petridis, A. Gounaris and J. Torres. 2017. Spark parameter tuning via trial-and-error. In *Advances in Intelligent Systems and Computing.* Springer Verlag, 2017, pp 226-237. https://doi.org/10.1007/978-3-319-47898-2\_24

[7] M. Li, J. Tan, Y. Wang, L. Zhang and V. Salapura. 2015. SPARKBENCH: A comprehensive benchmarking suite for in memory data analytic platform spark, In *Proceedings of the 12th ACM International Conference on Computing Frontiers, CF 2015*. https://doi.org/10.1145/2742854.2747283

[8] G. Wang, J. Xu and B. He. 2017. A novel method for tuning configuration parameters of spark based on machine learning. In *Proceedings - 18th IEEE International Conference on High Performance Computing and Communications, 14th IEEE International Conference on Smart City and 2nd IEEE International Conference on Data Science and Systems, HPCC/SmartCity/DSS 2016*. https://doi.org/10.1109/HPCC-SmartCity-DSS.2016.0088.

[9] Z. Li and Z. Liu. 2019. Feature selection algorithm based on XGBoost. *Tongxin Xuebao/Journal on Communications.* 40 (2019), 101-108. https://doi.org/10.11959/j.issn.1000-436x.2019154

[10] T. Chen and C. Guestrin. 2016. XGBoost: A scalable tree boosting system. In *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining.* 2016. https://doi.org/10.1145/2939672.2939785

[11] D. Whitley. 1994. A genetic algorithm tutorial. *Statistics and Computing.* (1994). https://doi.org/10.1007/BF00175354

[12] MODIS. (n.d.). https://ladsweb.modaps.eosdis.nasa.gov/missions-and-measurements/products/MYD04\_3K/