SECapacity: A Secure Capacity Scheduler in YARN

Chuntao Dong[1,2], Qingni Shen[1,2(✉)], Lijing Cheng[1,2], Yahui Yang[1,2], and Zhonghai Wu[1,2]

[1] School of Software and Microelectronics, Peking University, Beijing, China
{chuntaodong, morleycheng}@pku.edu.cn,
{qingnishen, yhyang, wuzh}@ss.pku.edu.cn
[2] MoE Key Lab of Network and Software Assurance,
Peking University, Beijing, China

Abstract. In this paper, aiming to the requirement that isolation of user's job and data security, we deeply analyze the mainstream computing framework Hadoop YARN, and start with the core module of YARN - resource scheduler. Using the existing label-based scheduling policy, we design and implement a SECapacity scheduler. Our main work including: First, according to the principle of least privilege, we propose a user-classification based scheduling policy, which divided users to several levels based on their attributes, then restrict which nodes could be used by this user according to the user level. Second, we design and implement a SECapacity scheduler to implement user-classification based scheduling. Third, we verify and analyze the effectiveness and efficiency of SECapacity scheduler, the results shows that SECapacity scheduler can ensure 100% isolation of users at different levels, and the performance overhead is about 6.95%.

Keywords: Big data platform · Hadoop · User-classification based scheduling · SECapacity scheduler

1 Introduction

Big data has rapidly developed into a hotspot, and big data analysis has been widely applied. In order to make good use of big data, academia and industry has proposed numerous techniques, such as Hadoop, Spark, Storm, and Graphlab etc., Hadoop has quickly become a major platform, which includes many technique as an ecosystem, such as MapReduce [2], YARN [8] and HDFS etc. With the development of big data applications, the security and privacy of big data have received sufficient attention. Due to the characteristics of big data and its complexity of processing, the security issues of big data are very complex and difficult to handle. The built-in security mechanism of Hadoop can only limit internal users to use resource legitimately. To ensure security and privacy of big data platform, academia and industry have proposed plenty of solutions. We will introduce these solutions in the related work of Sect. 2.

In this paper, first we analyze the security characteristics of big data platform and the related work. Then, we describe the threat scenario. Third, we refer to the principle

K.-Y. Lam et al. (Eds.): ICICS 2016, LNCS 9977, pp. 184–194, 2016.
DOI: 10.1007/978-3-319-50011-9_15

of least privilege [3] and proposed SECapacity scheduler to enhance the isolation of tasks [1]. Fourth, we conduct a massive experiments to test performance and security of our SECapacity scheduler. Finally, we summarize our work and clarify the value of our work, illustrate the limitation of our work, and introduce our research in the future.

In this paper, **our main contributions** are summarized as following points:

(1) We proposed a user-classification based scheduling refer to the principle of least privilege, which can enhance isolation of user's job and protect data security.
(2) Based on user-classification based scheduling, we design and implement the SECapacity scheduler based on Capacity scheduler.
(3) We test the performance and analyze the security of SECapacity scheduler by conduct a series of experiments.

The rest of this paper is organized as follows. We introduce the background in Sect. 2, and describe the threat scenario in Sect. 3. In Sect. 4, we present the design details of SECapacity, and introduce the implementation details of SECapacity in Sect. 5. We then evaluate and analyze experimental results in Sect. 6. We conclude the paper in Sect. 7.

2 Background

This section provides the background information about security analysis of Hadoop, related work and the mechanism of Capacity scheduler.

2.1 Security Analysis

As a distributed computing platform, big data platform has its own advantages, as well as disadvantages in security. So we analyze it from the pros and cons.

Big data platform has native characteristic of security and robustness: (1) The distribution makes the platform more robustness. The entire system wouldn't be impact, when adversary attacks a single node or a small number of nodes; (2) In the big data environment, the value of a few number of data become less. It's not significant that malicious users just get the data on a single node; (3) Big data platform is constituted by thousands of storage and computing nodes, except for central node, the common nodes has no difference, it's not easy to find the valuable attack point among the nodes.

Big data platform also brings convenience to the attacker, so it's difficult to defend the threat: (1) The big data platform is consisted of thousands of storage and computing nodes, which makes it very difficult to detect and deal with intrusion timely. (2) The software on the big data platform uses the master - slave structure, it's hard to remove the feature of centralization. As long as there is such a pivot or management center node like ResourceManager or ApplicationMaster, it's easy to be attacked. (3) During data processing, because of the low value of single node, the attacker will target to steal data processing results to improve the attack efficiency.

2.2 Related Work

There are many security and privacy issues in the big data platform now. The academic have proposed many solutions to solve these issues. We will introduce several solutions in the part.

Indrajit Roy etc. present Airavat [7], a MapReduce-based system which provides strong security and privacy guarantees for distributed computations on sensitive data. Airavat is a novel integration of mandatory access control and differential privacy. Tien Tuan Anh Dinh etc. develop a solution which avoids using the expensive ORAM construction and ensure privacy preserving computation in MapReduce, and implement their solution in a system called M^2R [4]. Olga Ohrimenko etc. analyze the security of intermediate traffic between mappers and reducers, and they describe two provably-secure, practical solutions [6]. In the paper [9], the author present SecureMR, a practical service integrity assurance framework for MapReduce, which provide a set of practical security mechanisms that prevent replay and Denial of Service attacks, preserve the simplicity, applicability and scalability of MapReduce.

We summarize the current research work and find that there is few research focus on the security of data processing results. Therefore, we focus on the data leakage issue during the big data processing. The current solutions mainly focus on static data security and access security, but ignore the security and isolation of tasks. We detailedly describe the threat scenario and steal scheme in our previous work [5]. According to the security features of the big data platform, we select a systematic risk management strategy to enhance the security of the platform.

2.3 Capacity Scheduler

In this paper, we add the security scheduling strategy in Capacity scheduler of Hadoop to enhance the security. The Capacity scheduler is based on the ratio of memory using, but rarely consider of security issues. The main design and development of our work is using label-based-scheduling strategy [1].

Label-based Scheduling provides a method match the shared cluster resources on nodes to a queue. The node, which has the same label with the queue, can be used by this queue. Using label-based scheduling, an administrator can control exactly which nodes are chosen to run jobs submitted by different users and groups. It's useful for data locality and multi-tenancy use cases.

3 Threat Scenario

Because we have described the threat model that steal user's processing results and implement the solution in our previous work [5], we will only describe the threat scenario in this section.

Threat Model. We assume adversary is regarded as trusted but have malicious intentions. He tries to steal sensitive results from other user that he is not allowed to access. The threat model is depicted in Fig. 1. There are three types of entities:

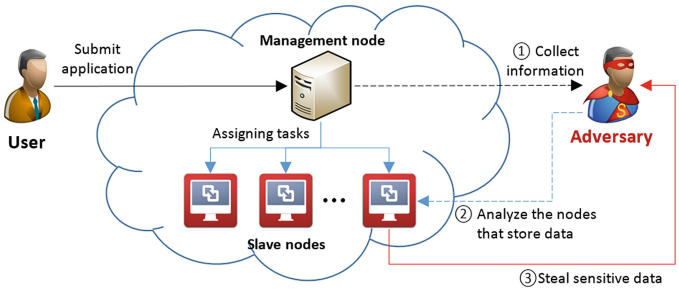

Fig. 1. The model of detecting the node storing the results

User: These entities have data to be stored in the cluster and interact with the Hadoop Cluster to manage their data and submit applications on the cluster.

Adversary: The adversary intends to steal the processed results of other users. He collects information and analyzes the nodes that store results.

Hadoop Cluster: The Hadoop cluster provides resources and services for users.

Based on the threat model, we proposed and implement the attack scheme [5]. The key of our scheme is confirm the node that storing the processed data. In general, the malicious want to running task on the node that the target user's job is running. In other words, lacking of job isolation mechanism lead to security problem in big data platform.

In this paper, we assume that the malicious user can control a few of nodes if they can deploy tasks on these nodes. Based on assume, the isolation mechanism of Hadoop is not enough to guarantee the security of the platform, we need to proposed a new solution at the platform level. So we design SECapacity Scheduler as a systematic risk management strategy to enhance the security of the platform.

4 SECapacity Scheduler

In this section, we introduce the details of the SECapacity Scheduler. First, we take an overview of the SECapacity. Then, we introduce the user-classification based scheduling strategy (UCBS). Third, we introduce the label based scheduling. At last, we proposed two scheme of UCBS.

4.1 Overview

According to the threat analysis, we knew that tasks of users and malicious users running on same node is not safe. We wish to separate the users from the malicious users, and isolate execution of their tasks. The scheduler limit the resource usage of each user, but don't limit the number of used nodes, user can apply for computing resources on any node. It is very easy to be utilized by malicious users. In the process of designing SECapacity scheduler, we take the least privilege as design principle.

The principle of Least Privilege [3] requires that in a particular abstraction layer of a computing environment, every module must be able to access only the information and resources that are necessary for its legitimate purpose. The principle means giving a user account only those privileges which are essential to that user's work.

In this paper, the principle is extended to design secure scheduling strategy. The scheduler should ensure the resource quantity. What's more, the scheduler should limit the scope of resource usage. By limiting the scope of resource usage, we limit influence scope of potential malicious users and improve the safety of the platform.

4.2 User-Classification Based Scheduling

According to the design principle, we face two problems: which user is a malicious user, and SECapacity scheduler take user as resource division unit will seriously affect the performance, so we proposed UCBS.

A. ***User classification***

We need to identify potential malicious users among all the users. In order to describe our secure scheduling, we provide a reference user classification scheme. Our user classification scheme rating user according to attributes, and then divide all users into several levels on the basis of classification criteria. The classification criteria is determined by the administrator.

In our scheme, every user has four attributes, include user privilege, user resource quota, registration time and safety rating. The scores of these four attributes are represented by symbols U, R, T, and S, and rating criteria are shown in Table 1. According to rating criteria and the formula (1), we can give each user a score. Where a, b, c and d are parameters, we simplify all the parameters to 1.

$$G = a*U + b*R + c*T + d*S \tag{1}$$

If we take user as resource division unit will seriously affect the scheduling performance. So we propose a classification scheme to divide all users into several levels. The classification criteria is determined by the administrator. For example, assuming there are five users User1, User2, User3, User4 and User5 in the cluster, and the score

Table 1. User rating criteria

User property	User attribute classification and score evaluation criteria
User privilege	Super administrator = 50, Administrator = 10, Ordinary User = 1
User resource quota	User resource percentage of cluster resource
Registration time	Score of user registration time = [Log_2T], T is the number of days
Safety rating based on behavior of users	Malicious behavior (such as tracking the other tasks) minus 10 score

of five users are: 21, 20, 31, 35 and 56. The classification criteria: 0–24 is level 1, 25–49 is level 2, and 50–75 is level 3.

B. ***Resource classification***

According to the principle of least privilege, we limit the scope of nodes that user used. But we also need to ensure the "authority" of users, the "authority" is the resource quota of users. In the last part, we divide all users into several levels. We need to recalculate the total resource of users at the same level. Assuming that the number of nodes in the cluster is n, and the memory resource of each node is x, and there are m users $U_1, U_2 \ldots U_m$. The resource quota of m users is $X_1, X_2 \ldots X_m$. By user rating and user classification, m user are divided into K level $L_1, L_2 \ldots L_K$. The resource quota of all the users in level L_k ($0 \le k \le K$) is recorded as G_k, we use the formula (2) to calculate G_k.

$$Gk = \sum Xi, Ui \in Lk \tag{2}$$

Through the calculation, we can confirm the resource quota of K users are: G_1, $G_2 \ldots G_K$, then we use the formula (3) to calculate Nk, the number of nodes belong to level L_k ($0 \le k \le K$).

$$Nk = [Gi * n] \tag{3}$$

C. ***Label based scheduling***

After divided all the users into several levels, we need to schedule the tasks to the corresponding nodes, we use label to achieve our scheduling goals. For convenience, we assume that every user monopolize a leaf queue. By configuring labels of queues and nodes, we schedule the tasks in the queue to the nodes that has the same labels with the queue. We proposed two isolation scheme as follows:

Node label setting policy: according to $N_k (0 \le k \le K)$, choose N_k nodes to set label k, and ensure the resource localization as much as possible.

4.3 The Scheme of Isolation Scheduling

According to the principle of least privilege, we proposed two schemes of isolation scheduling: complete isolation scheduling and range control scheduling.

Complete isolation scheduling: the jobs in the queue that has the label k can only use the nodes that has the same label with the queue.

Queue label setting policy: the queue that the user U_i belongs to should be set label k, $U_i \in L_k$, $0 \le k \le K$. We take the example that mentioned in the last part as example, its configuration scheme is as shown in the Fig. 2.

Range control scheduling: the jobs in the queue that has the label k can use the nodes that has the same label k and less than k.

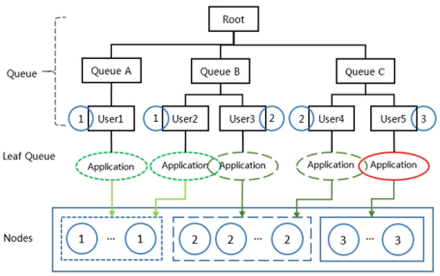

Fig. 2. Complete isolation scheduling scheme

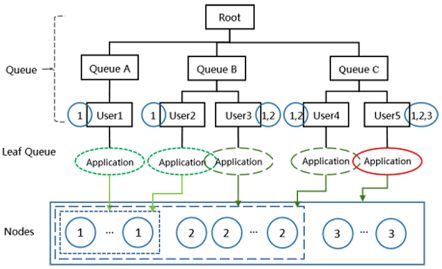

Fig. 3. Range control scheduling scheme

Queue label setting policy: the queue that the user U_i belongs to should be set label 1, 2… k, $U_i \in L_k$, $0 \le k \le K$. We take the example that mentioned in the last part as ex-ample, its configuration scheme is as shown in the Fig. 3.

The detailed scheduling rules: users can use resources of lower levels users, but user can't use resource of higher level users. Users should firstly use resource that belong to their own level, we achieve this through starting resource preemption strategy.

5 Implementation

In this section we introduce the design and implementation of SECapacity scheduler. First we introduce the structure of SECapacity scheduler, then demonstrate the function and realization of each module in detail.

5.1 The Structure of SECapacity Scheduler

SECapacity scheduler is based on the Capacity scheduler of Hadoop 2.6.0, and using the label based scheduling. We implement the complete isolation scheduling in SECapacity scheduler. The structure of SECapacity scheduler is as shown in Fig. 4.

5.2 Function of Each Module

In this part, we will detailedly describe the function and implement of each module in the SECapacity Scheduler.

(1) **Queue and Node label management module:** rates and classify the user to generate the label configuration scheme. The module includes five components as follows:

User level management component rates all users and generates user classification scheme according to user's attributes. It has two function: initialization and updating.

Queue label management component is based on the user classification scheme to generate queue label configuration scheme.

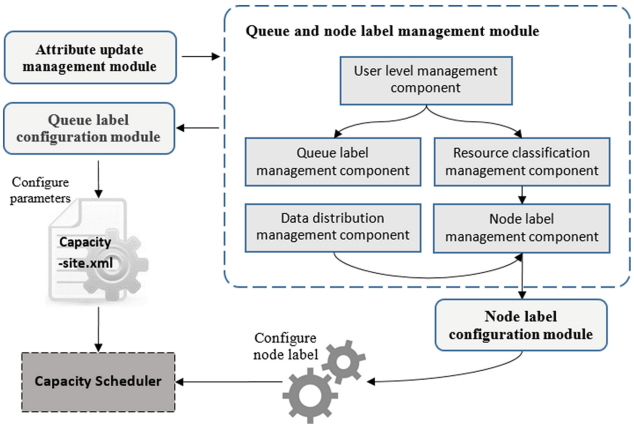

Fig. 4. The architecture of SECapacity scheduler

Resource classification management component calculates the total resource quota of each level's users according to the user classification scheme and the resource quota.

Data distribution management component uses the information of data blocks distribution to calculate the number of data blocks on each node.

Node label management component generates node label configuration scheme according to the nodes' number of each level and the distribution of all users' data block.

(2) **Attribute update management module** monitors user's attributes, and the changing of users and nodes in the cluster. According to these changes, it calls Queue & Node label management module to modify the classification of user, and generates a new queue and node label configuration scheme.

(3) **Queue label configuration module** uses the queue label configuration scheme to set the related configuration file. This module can automatically configure capacity-site.xml. This module mainly modifies the following parameters:
   (1) *yarn.scheduler.capacity.<queue-path>.accessible-node-labels*: decide which labels can be used by the "*queue-path*".
   (2) *yarn.scheduler.capacity.<queue-path>.accessible-node-labels.<label>.capacity*: decide the available proportion of resources on the nodes labeled "*label*" which can be used by user in the queue named "*queue-path*".
   (3) *yarn.scheduler.capacity.<queue-path>.accessible-node-labels.<label>.mamimum-capacity*: decide the upper bound of the resource labeled "*label*" which can be used by users in the queue named "*queue-path*".

(4) **Node label configuration module** configures each node's label according to node label configuration scheme. This module uses node label configuration scheme to generate and run the configuration script.

6 Evaluation

In this section, we evaluate the performances of SECapacity Scheduler. We describe the experiment environment and scenarios, then conducted a series of experiments to test two schemes of isolation scheduling, and analyze the experimental results at last.

6.1 Experiment Scenario

Experiment Environment. The Hadoop cluster including 19 nodes, a master that deployment ResourceManager and NameNode, and 18 nodes that deployed DateNode and NodeManager. Every node using the local 64-bit Centos operation system with an Intel Core 7 processor running at 3.4 GHz, 4096 MB of RAM, and run Hadoop 2.6.0. We set up the size of every data block is 128 MB. The input files is 4.8 GB, and each block has 3 duplicates. To get the best performance in experiment, we assume there are three users and the quota of them is 33%, 33% and 34%. These three users are divided into three levels, and they execute the same job with the same size input files.

Setup. To test SECapacity scheduler, we need to configure the hadoop cluster. The configuration process includes the following four steps:

Step 1: add system level label. According to the experimental scenario, we need to add three labels one, two and three. Execute command: *rmadmin -addToClusterNodeLabels one, two, three.*
Step 2: label all nodes. The command is: *rmadmin -replaceLabelsOnNode yarn "nodeId = label".* In our experimental scenario, we configure six node with a same label.
Step 3: configure the label recovery. The label information will be saved to the HDFS after the YARN restart.
Step 4: according to the scheme of isolation scheduling, configure the capacity-scheduler.xml.

6.2 Performance Analysis

We mainly choose two **evaluation standard**: resource localization rate and runtime. The resource localization rate is the rate of map task running at the node that storing the input files. Dividing the cluster into several parts will reduce the resource localization rate and increase the runtime of job.

At first, we run 3 wordcount jobs that consists 39 map tasks and 1 reduce tasks to test the Capacity scheduler. In order to ensure the accuracy of experimental results, we run 20 experiments and calculate the average. A part of experiment results as shown in Table 2. Then, we also run 3 wordcount jobs that consists 39 map tasks and 1 reduce tasks in three levels to test the SECapacity scheduler. The nodes' number of each level is 6. A part of experiment results as shown in Table 3.

Table 2. The experiment results of capacity scheduler

Job	The first run		The second run		The third run	
	Runtime	Localization rate	Runtime	Localization rate	Runtime	Localization rate
Job 1	376	22/40	367	27/39	365	27/41
Job 4	384	23/41	336	30/41	359	31/41
Job 7	330	30/42	327	32/41	349	27/40

Table 3. The experiment results of SECapacity scheduler

Job	The first run		The second run		The third run	
	Runtime	Localization rate	Runtime	Localization rate	Runtime	Localization rate
Job 1	348	20/40	356	22/40	417	29/41
Job 4	361	22/40	374	18/40	430	27/42
Job 7	331	21/41	321	20/40	449	22/40

We summarize and analyze the results of Capacity scheduler and SECapacity scheduler in Table 4. By comparing the results, we find that the localization rate decline from 61.78%–71.65% to 49.32%–53.74%. The runtime rise from 336 s–367 s to 360 s–396 s, and rise 6.95% at average. According to the above results, we can get the conclusion that the runtime is associated with localization rate. We should improve the data distribution management component to increase the localization rate.

Table 4. Capacity scheduler vs. SECapacity scheduler

Job	Capacity scheduler		SECapacity scheduler		
	Runtime (s)	Localization rate (%)	The level of user that submit job	Runtime (s)	Localization rate (%)
Job 1	367	62.76	level 1	381	51.25
Job 2	372	61.78	level 1	369	53.74
Job 3	363	63.14	level 1	390	50.13
Job 4	358	70.12	level 2	394	52.50
Job 5	352	71.02	level 2	388	53.17
Job 6	364	69.35	level 2	396	50.37
Job 7	339	70.71	level 3	363	50.91
Job 8	344	70.13	level 3	376	49.32
Job 9	336	71.65	level 3	360	51.67

6.3 Security Analysis

If the level of a malicious user is higher than or equal to the level of users, users have the risk of information leakage, because a malicious user can apply for nodes that belongs to the other users. The SECapacity can only ensure safety and isolation of users at different levels, but can't protect users at the same level. So the SECapacity

scheduler relay on the accuracy of user classification scheme. In the future work, we will ensure the safety and isolation of users at same levels.

7 Conclusion

In this paper, we propose a UCBS according to the principle of least privilege, and implement the scheduling strategy in SECapacity scheduler to enhance isolation of different level's jobs. Through the experiments, we verify the effectiveness and performance of SECapacity scheduler. The performance cost of our scheme is about 6.95%, and we don't need to modify Hadoop source directly. However, the effect of the SECapacity scheduler is highly dependent on the user classification scheme. We need to further improve our user classification scheme. Another problem is that SECapacity scheduler is poor at defend APT attacks, a malicious user can improve their permission through long-term hidden. We also need to solve the problem that how to defend against APT attack in the future work.

Acknowledgments. This work is supported by the National High Technology Research and Development Program ("863" Program) of China under Grant No. 2015AA016009, the National Natural Science Foundation of China under Grant No. 61232005, 61672062, and the Science and Technology Program of Shen Zhen, China under Grant No. JSGG20140516162852628.

References

1. Apache hadoop. http://hadoop.apache.org
2. Dean, J., Ghemawat, S.: MapReduce: simplified data processing on large clusters. In: Conference on Symposium on Operating Systems Design & Implementation, vol. 51, pp. 107–113. USENIX Association (2004)
3. Denning, P.J.: Fault tolerant operating systems. ACM Comput. Surv. **8**(4), 359–389 (1976)
4. Dinh, T.T.A., Saxena, P., Chang, E.C., et al.: M^2R: enabling stronger privacy in mapreduce computation (2015)
5. Dong, C., Shen, Q., Li, W., Yang, Y., Wu, Z., Wan, X.: Eavesdropper: a framework for detecting the location of the processed result in hadoop. In: Qing, S., Okamoto, E., Kim, K., Liu, D. (eds.) ICICS 2015. LNCS, vol. 9543, pp. 458–466. Springer, Heidelberg (2016). doi:10.1007/978-3-319-29814-6_39
6. Ohrimenko, O., Costa, M., Fournet, C., et al.: Observing and preventing leakage in MapReduce. In: ACM SIGSAC Conference, pp. 1570–1581 (2015)
7. Roy, I., Setty, S.T.V., Kilzer, A., et al.: Airavat: security and privacy for MapReduce. In: Usenix Symposium on Networked Systems Design and Implementation, NSDI 2010, San Jose, pp. 297–312 (2010)
8. Vavilapalli, V.K., Murthy, A.C., Douglas, C., et al.: Apache hadoop YARN: yet another resource negotiator. In: Symposium on Cloud Computing, pp. 1–16 (2013)
9. Wei, W., Du, J., Yu, T., et al.: SecureMR: a service integrity assurance framework for MapReduce. In: Computer Security Applications Conference, pp. 73–82. IEEE (2009)