Mathematical Biosciences and Engineering

MBE, 19 (1): 287–308.
DOI: 10.3934/mbe.2022015
Received: 23 September 2021
Accepted: 06 November 2021
Published: 12 November 2021

http://www.aimspress.com/journal/MBE

Research article

Parallel anomaly detection algorithm for cybersecurity on the high-speed train control system

Wang Zhoukai[1,*], Hei Xinhong[1,2], Ma Weigang[1], Wang Yichuan[1,2], Wang Kan[1] and Jia Qiao[1]

[1] College of Computer Science and Engineering, Xi'an University of Technology, Xi'an 710048, China
[2] Shaanxi Provincial Key Laboratory of Network Computing and Security Technology, Xi'an 710048, China

* **Correspondence:** Email: zkwang@xaut.edu.cn.

Abstract: With the rapid development of the high-speed train industry, the high-speed train control system has now been exposed to a complicated network environment full of dangers. This paper provides a speculative parallel data detection algorithm to rapidly detect the potential threats and ensure data transmission security in the railway network. At first, the structure of the high-speed train control data received by the railway control center was analyzed and divided tentatively into small chunks to eliminate the inside dependencies. Then the traditional threat detection algorithm based on deterministic finite automaton was reformed by the speculative parallel optimization so that the inline relationship's influences that affected the data detection order could be avoided. At last, the speculative parallel detection algorithm would inspect the divided data chunks on a distributed platform. With the help of both the speculative parallel technique and the distributed platform, the detection deficiency for train control data was improved significantly. The results showed that the proposed algorithm exhibited better performance and scalability when compared with the traditional, non-parallel detection method, and massive train control data could be inspected and processed promptly. Now it has been proved by practical use that the proposed algorithm was stable and reliable. Our local train control center was able to quickly detect the anomaly and make a fast response during the train control data transmission by adopting the proposed algorithm.

Keywords: cybersecurity; high-speed train; control system; parallel computing; distributed computing

1. Introduction

High-speed rails possess the advantages of fast speed, high efficiency, low energy consumption, and low pollution and play a vital backbone role in China's transportation system at present [1,2]. With the advent of the era of intelligence and the vigorous development of the 5G technique, the high-speed rail control system, which integrates new technology like big data and intelligent management, also thrives gradually [3,4]. However, now the communication environment faced by the high-speed rail control system has become more complex and complicated, and the risk of being attacked or threatened by external security has also increased over the years. Therefore, how to build up a solid defensive barrier for the high-speed rail control system and protect it from being assaulted by outside threats have become a hot research issue in both railway transit optimization and the cybersecurity field [5,6].

To guard against external malicious attacks and ensure traffic safety, China, the country with the longest high-speed railways, has proposed the China Train Control System 3 high-speed train control system (also known as the CTCS-3). In CTCS-3, an abnormal data detection algorithm based on deterministic finite automaton (DFA) is raised to proactively identify anomalous data in the data stream sent to the high-speed rail control system [7]. Nevertheless, with the fast development of China's information network, an increasing number of high-speed trains are put into operation, and the corresponding data has explosively increased [8,9]. This situation brings a problem that the traditional, serial DFA-based data detection algorithm has become more difficult to detect abnormal data and potential threats from train control data in time. Especially in Spring Festival and other vacations, the inadequate capacity to handle the newly emerging train control data has become the primary factor influencing extra trains. In the face of the rapid development of the high-speed rail train control system, many researchers are devoted to the optimization work of CTCS-3 to improve the detection efficiency of train control data and ensure the smooth and healthy operation of the high-speed rail system.

With the development of communication and information technologies, some of the researchers try to solve problems with big data and blockchain technology [10–15]. Inspired by this idea, researchers in the train control field also employ the mighty computing power of distributed computing platforms to design multi-task parallel detection algorithms and improve the efficiency of train control data detection [16,17]. Even though some progress has been made, the high coupling dependencies inside the train control data stop the detection work from performing concurrently. It causes the result that the detection efficiency for the control data is low, and the utilization of resources is not high.

In response to the above problems, this paper employed the parallel speculation technique commonly used in multi-core areas and raised a speculative parallel detection algorithm for high-speed train control data. In the beginning, the structure of the train control data is analyzed. Then the algorithm divides the train control data into small segments and eliminates internal dependencies that hide in the data. Secondly, based on the divided data, the traditional DFA-based algorithm is parallelized and transformed based on the parallel speculation technology, so that the inline control dependencies that impact the algorithm flow can be modified. Finally, the parallel algorithm is implemented on Apache Spark, the widely used distributed platform, to detect the divided data simultaneously, making full use of the mighty computing power to improve the detection efficiency of the high-speed rail train control data. The main contributions of this article are as follows.

1) The internal dependencies which impact the algorithm's parallelism are found based on the modeling analysis for the conventional data detection process.
2) The speculation technique is introduced to overcome the internal dependencies and parallelizes the

conventional data detection algorithm.

3) Implement the proposed algorithm with the speculation mechanism onto the distributed platform and improve abnormal data detection efficiency.

2. Motivation

The high-speed train control data refers to the data generated by the control system during the operation of high-speed trains and is related to the safety of high-speed train operation. For example, the high-speed train control data includes the synthetic platform signal data, the monitoring level conversion data, the driver operation data, the electronic control equipment operation data, the central control system failure data, the emergency braking command data, and so on. Unlike the equipment operation data collected by onboard train sensors, the train control data is recorded by the dedicated onboard train control system ATS (Automatic Train Supervision) and transmitted through a remote wireless network: as shown in Figure 1. In CTCS-3, the train control data is collected at set intervals. When the collection works end, the collected information is manually dumped by a laptop and then sent to the railway bureau via WLAN or wireless network GSM-R (Global System for Mobile Communications-Railway). Otherwise, the onboard computer automatically stores the collected information and transmits it through GSM-R [18].

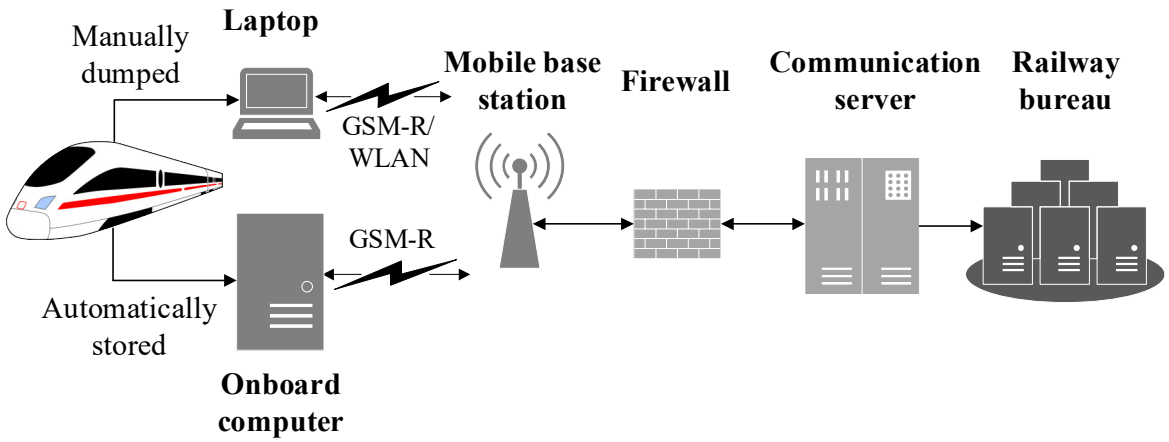


Figure 1. Schematic diagram on high-speed train control data transmission in CTCS-3.

However, due to the lack of security audits and intrusion prevention measures, GSM-R, the enclosed network system, is very vulnerable to outside security issues such as hacking, extortion, and virus attacks. The railway bureau also tends to be in danger because of the unsafe network. Even though safety methods such as SIM cloning, anti-blocking, and other IoT-based safety precautions are introduced to enhance transmission security, there is still the risk of damage or tampering [19].

Some researchers attempt to detect abnormal data with novel machine learning techniques to solve the problem and boost data detection efficiency. For instance, Belgrana and Maruno have improved network intrusion detection rates and reduced processing time with Neural Network [20,21]. In contrast, other researchers proposed a multi-stage optimized ML-based NIDS framework or fuzzy detection system that reduces computational complexity and maintains the detection performance [22,23]. However, these methods cannot be adopted by the high-speed train intrusion prevention system. Because

the safe operation of high-speed trains depends on accurate train control data, but deep learning-based methods cannot ensure 100% reliability of train control data [24].

On the contrary, CTCS-3 brings the DFA-based data detection algorithm into the communication server to eliminate the risk that impacts data transmission safety. In this way, the abnormal data can be identified, and the attacks and threats can be defended. Specifically, when the communication server receives the train control data, it will be compared with the abnormal characteristics predefined by a DFA. Then it can be known that whether there is abnormal information in the control data.

The DFA is made up of five tuples like $(S, \Sigma, \delta, s_0, F)$, where S is the set of states; $\sum$ is the alphabet, which represents the set of finite input symbols; s_0 is the initial state of the DFA; F is the set of accepting states ($F \subseteq S$); δ is the transfer function, and it means that a state in state set S calculates with a letter from the alphabet Σ, then the calculation result is another state in state set S. The DFA-based data detection process begins from the start state s_0, reading the train control data ω character by character ($\omega \in \sum^*$,* means the connection of the elements in the alphabet $\sum$) and changing its state according to the transfer function δ. Once the DFA transfers its state to the acceptance state (the state in the set F), it means that abnormality is found in the train control data ω. Otherwise, there is no abnormality after detection. The pseudo-code is shown in Table 1.

Table 1. Pseudo-code of algorithm 1.

Algorithm 1: DFA-based data detection algorithm	
Input 1: Train control data ω and its length I	
Input 2: Specific DFA that recognizes the abnormality	
Output: Whether the abnormality is found	
1	state = start_ state;
2	for i = 0 to |I-1| do
3	input char =ω[i];
4	state = δ[state][input char];
5	if state∈F then
6	return Abnormality Found;
7	end
8	end
9	return NotFound ;

It can be seen from Table 1 that the procedure of algorithm 1 consists of a series of iterations, in which the current character $\omega[i]$ and the current state are sequentially calculated, and then the state of the DFA migrated after the calculation. Assuming that the time for processing one single character is *M*, the length of the train control data is *I*, then the time and space complexity for detecting the whole train control data will be *M* * *I* and *I*, respectively. In total, the traditional algorithm is inefficient. This brings a problem that when the data volume becomes large or the state number increases, the time and space overhead will promote dramatically. Meanwhile, the efficiency of the detection algorithm will experience a significant reduction.

Most researchers use parallel computing or distributed computing technology to solve the problem and raise data detection efficiency. For example, Ding proposed a multi-step regular expression matching algorithm based on a parallel character index, which improves the matching throughput by 48.5% on average [25]. Liu proposed a multi-DFA parallel detection algorithm based on a distributed computing platform, improving the query response time by up to 43.9% [14]. Lu pointed out that the GPU platform can also implement large-scale detecting [26]. These measures focus on transplanting traditional algorithms onto new platforms and improving detection efficiency by synchronizing multiple DFAs.

However, these solutions are mainly on multiple DFAs cooperation, and the overall efficiencies are limited when the data amount grows larger. Even some parallel detection tasks with high latency would make the whole procedure time-consuming. By analyzing the underlying logic of these proposed algorithms in-depth, it can be known that the DFA-based data detection procedure proceeds iterations by iterations, but the current solutions are hard to optimize these iterative computations in parallel. When the data scale becomes large, the detection procedure becomes a long iteration chain, which is hardly shortened or parallelized by any optimization measures above. Thus the efficiency of the detection algorithm on high-speed train control data is considered low.

In summary, in order to solve the problem that the time and space overhead generated in the iterative operation affects the detection efficiency of the high-speed train control data. This paper proposes a highly effective data detection algorithm based on speculative parallel optimization techniques. Firstly, the high-speed train control data are divided into small blocks of the same size. Secondly, by introducing the speculative parallel optimization technique widely used in multi-core platforms, this paper reforms the traditional DFA-based detection algorithm with a parallel speculation mechanism to avoid the control dependencies between different iterations and make the iteration computations work in parallel. Finally, the reformed algorithm is implemented on Apache Spark, the most popular distributed computing platform with high concurrency, to detect the small divided blocks in parallel. In this way, not only can the data detection efficiency be improved, but also the computing resource utilization of the distributed computing platform can be raised simultaneously.

3. Algorithm design

Based on the description of existing problems and solutions, the speculative parallel detection algorithm for high-speed train control data is designed as follows: The algorithm is divided into three parts, namely high-speed train control data partition, speculative parallel detection, and speculation result verification. The design ideas of the algorithm are described in detail below in order.

3.1. High-speed train control data division

First and foremost, according to the typical speculative parallel optimization process, the input data should be divided into blocks and then gathered into the dataset [27]. Precisely, in the proposed algorithm, the control data should be cut into segments of the same size. Besides, to support parallel execution, every part except the first one should also be assigned an initial DFA state. In this way can all the segments be detected independently, without waiting for the state from its former part. Thereby the goal of parallel detecting the multiple data segments can be achieved.

Except for establishing the data dividing principle, the size of the division granularity also needs

to be considered. Because in the speculative parallel data detection algorithm, if the division size is too small, resulting in indirectly that the derived parallel tasks workload increases, and the number of parallel tasks is affected at the same time. Besides, the coordination between different speculative parallel tasks may be complicated and complex to be adjusted. In the worst case, fine granular data partitioning will also increase the communication frequency between parallel tasks and affect the overall performance of the proposed algorithm. On the contrary, if the granularity of the data is too coarse, the load imbalance and data skew problems in distributed computing would happen, which leads to an increase in the running time of a single parallel task and affects the overall performance of the proposed speculative parallel algorithm. Therefore, it is necessary to establish a theoretical model to reveal the relationship between data partitioning granularity and parallel algorithm time spending. Then select a suitable data partitioning granularity after careful calculation.

The model building process is as follows. Firstly, the time cost is in Eq (1). As shown in Eq (1), the time cost T of the speculative parallel algorithm consists of two parts: T_s is the time of distributing parallel tasks and dividing the high-speed train control data into equal-length data segments. But comparing with the time of parallel tasks distribution and transmission, the time for data division is so short that it can be ignored. In addition, the T_e is the time for parallel tasks' execution.

$$T = T_{\mathrm{s}} + T_{\mathrm{e}} \tag{1}$$

Assuming that the total amount of data is s, the data division granularity is n, the number of processors in the distributed cluster is k, the data transmission speed in the cluster is p. Besides, the mathematical expectation of the execution time of a single task is t, the data segmentation is s/n, and correspondingly, the number of the parallel tasks is the same as s/n. Then after bringing the relevant parameters, Eq (1) can be represented as Eq (2).

$$T = \frac{n}{p} + \frac{s/n}{k} \cdot t = \frac{n}{p} + \frac{s \cdot t}{n \cdot k} \tag{2}$$

The first part in Eq (2) is the parallel tasks' distribution time, which is subjected to the division granularity. Furthermore, the second part in Eq (2) is the time for parallel tasks' execution, and it is determined by the total data amount and every single task processing time. In the practical running environment, the high-speed train control data s, the number of the cluster's processors k, the data transmission speed p, and the mathematical expectation of a single task execution time t is known. Therefore, Eq (2) represents the relationship between the data partitioning granularity and parallel algorithm time cost, and the schematic diagram on Eq (2) is described in Figure 2.

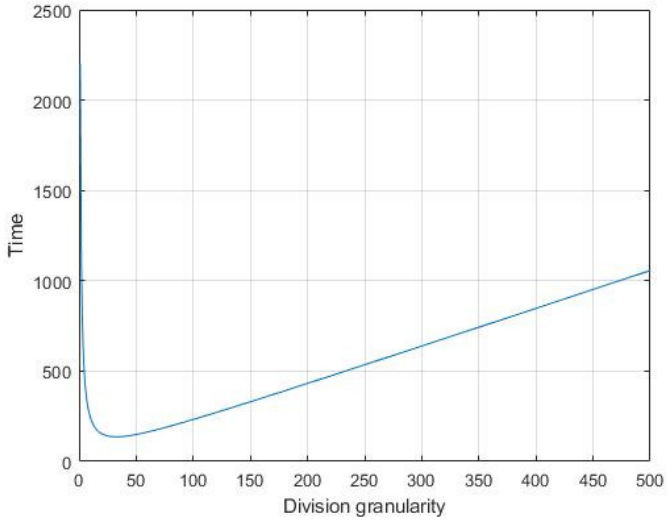


Figure 2. The diagram on data division granularity and algorithm run time.

Since the partitioning granularity and execution time cannot be negative numbers, only the first quadrant is adopted. It can be seen from Figure 2 that when the data partitioning granularity tends to 0, the number of the divided data segments tends to have positive infinity. In such circumstances, the execution time of the parallel algorithm will increase significantly. In an actual operating environment, this circumstance refers to that fine data division granularity would lead the task distribution, the message communication and the coordination of parallel tasks become complicated, and finally, the overall performance of the proposed algorithm would slow down. Instead, when the data partitioning granularity increases, the proposed algorithm's execution time increases gradually due to the rise in the running time of the single parallel task. In extreme cases, the partitioning granularity is equal to s, and the proposed algorithm degenerates to the traditional, unparallel algorithm.

Thus, the goal for setting a proper division granularity becomes to find the minimum in Eq (2). Taking the derivative of Eq (2) and setting the derivative to 0, and it is easy to know that when the division granularity meets the demand of Eq (3), the execution time of the speculative parallel algorithm is minimum, which also means that the parallel algorithm takes the shortest time.

$$n = \pm\sqrt{\frac{s \cdot t \cdot p}{k}} \tag{3}$$

3.2. Speculative parallel detection

Then the corresponding speculative parallel detection mechanism is designed to ensure that the divided data can be detected simultaneously. The proposed algorithm takes the speculative parallel optimization technique to reform the traditional data detection algorithm and make the detection procedure for large-scale data faster.

At first, it is necessary to analyze the inline relationship of the traditional DFA-based detection processes. From Algorithm 1 in Table 1, it can be seen that during the first iteration, when the first

letter of the high-speed train control data is read, based on the initial state, the algorithm starts to migrate the current DFA state with the help of the transfer function. When the state changes, the algorithm decides to continue detecting the following letters or reporting the abnormal information. In addition, it is easy to know that any iterations except the first iteration could not predict the former DFA states in advance by analyzing the relationship between different iterations. Similarly, any iterations except the first iteration could not execute until the former iterations finish. However, even though the DFA state is the critical factor that impacts the detection procedure through the analysis, for lack of the DFA state prediction method, it is still challenging to achieve the parallelization of traditional data detection procedure by a simple divide conquer strategy.

Under this circumstance, the speculative parallel optimization technique, commonly seen in the multi-core parallelism research field, is borrowed to resolve these difficulties. This technique is an explicit parallelization technology. By temporarily ignoring the dependencies between the concurrent units, threads that were initially thought not to be parallel can be tentatively executed in parallel, and the performance would be improved when such parallel execution succeeds. The speculative execution model is shown in Figure 3.

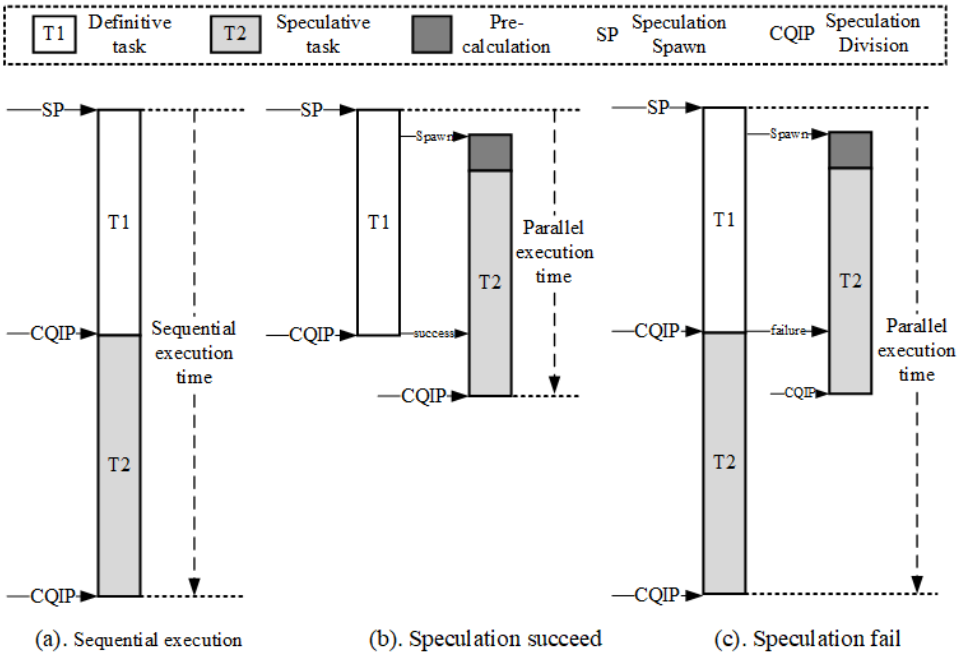


Figure 3. Schematic diagram of the speculative execution model.

As Figure 3(a) shows, T1 and T2 can only be executed serially during the sequential execution procedure due to the control dependence inside the two parts. However, as shown in Figure 3(b), the speculative parallel optimization technique can eliminate the influence of the existing control dependence with the pre-calculation method so that T2 could execute in advance instead of waiting for the result of T1. When T1 finishes, the result of T1 could validate the correctness of the pre-calculation content; thereby, the speculative result of T2 could also be validated at the same time: if the pre-calculation content is correct after validation, then the result of T2 is also correct, and the speculative execution succeeds in general; but if the validation result turns wrong, then the speculative execution on T2 fails. As shown in Figure 3(c), the speculative result will be discarded, and T2 will be

re-executed with the parameter that comes from T1.

Eliminating the dependencies between different iterations is the key to optimizing the conventional data detection algorithm based on DFA. After analyzing Algorithm 1, it can be seen that the dependencies are that the output of the previous operation is the input of the current iteration. The current iteration cannot execute independently because of the lack of the DFA state from its previous iteration. Therefore, when the division of the high-speed train control data finishes, every iteration (except the first iteration) in the data detection procedure should be assigned one DFA state so that they can perform independently. Instead of waiting for the DFA state from its former iteration, such assignment refers to the pre-calculation method in the speculative parallel optimization technique. In this way, all iterations could perform independently without relying on the results of its previous iteration, and the concurrent detection on divided segments could be achieved at the same time.

However, since the parallel iterations execute with the predicted states, the results may not be correct. So the validation procedure is established after the speculative parallel detection procedure to provide the correctness of parallel iterations. The specific step is as follows: if no abnormal data is detected after the speculative parallel detection, all data segments should be re-detected by its former iterations' result: the state calculated by speculative execution. At last, if the re-detection result is the same as the result of the speculative execution. It proves the speculative result is correct, and there is no anomaly in the control data. However, if the result conflicts, it indicates that incorrect speculation occurs in the speculative parallel detection, and the result of the re-detection process shall prevail.

For instance, *DFA(M)* could detect the abnormal data *VIRUS* like Eq (4).

$$M=\begin{pmatrix}\{0,1,2,3,4,5\},\{V,I,R,U,S,OTHERS\},\\ \delta,0,\{5\}\end{pmatrix} \tag{4}$$

Wherein the migration function δ is described in Eq (5).

$$\begin{aligned}\delta=\{&f(0,V)=1,f(1,I)=2,f(2,R)=3,\\ &f(3,U)=4,f(4,S)=5,\\ &f(\{2,3,4,5\},V)=1,\\ &f(\{2,3,4,5\},OTHERS)=1\}\end{aligned} \tag{5}$$

Moreover, the corresponding state transition diagram of δ is like Figure 4. The dotted line in Figure 4 means that the transition direction when letters except *for V*, *I*, R, *U*, *S*, is met in *DFA(M)*.

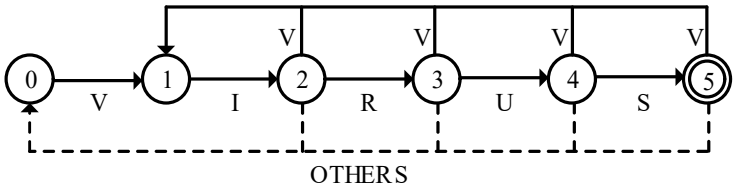


Figure 4. State transition diagram of function δ.

Suppose that a part of high-speed train control data *I=AVOIDS_VIRULENCE* needs to be checked with *DFA(M)*. According to the proposed algorithm, in the beginning, the data is separated into two

parts of equal length, wherein I_1=*AVOIDS_V* and I_2=*IRULENCE*. On this basis, the predicted state for I_2 is 0, then I_1 and I_2 can detect abnormal data simultaneously, and the specific process is in Figure 5.

I_1	A	V	O	I	D	S	_	V
s1	0							
s2		1						
s3			0					
s4				0				
s5					0			
s6						0		
s7							0	
s8								1

(a)

I_2	I	R	U	L	E	N	C	E
s1	**0**							
s2	0	**0**						
s3	0	0	**0**					
s4	0	0	0	**0**				
s5	0	0	0	0	**0**			
s6	0	0	0	0	0	**0**		
s7	0	0	0	0	0	0	**0**	
s8	0	0	0	0	0	0	0	**0**

(b)

Figure 5. Schematic diagram of speculative parallel detection.

Figure 5 shows the parallel detection process of data segments I_1 and I_2, where Figure 5(a) represents the definitive, non-speculative task based on the initial state 0. The execution result for detecting I_1 must be correct. Figure 5(b) illustrates the speculative task based on the pre-calculated state 0, and the execution result for detecting I_2 may be incorrect. In addition, the result of each state transition on the speculative task is recorded in the cache to validate the speculative result after parallel detection.

It can also be seen from Figure 5 that for the high-speed train control data I with 16 letters, the conventional algorithm would spend 16 rounds of iterations to fulfill the detection work. In contrast, only eight iterations are required (*s1* to *s8* in Figure 5) when we adopt the speculative parallel detection procedure. If the computing resources are sufficient and the division granularity is reasonable enough, the detection efficiency will be higher.

When the detection comes to an end, and abnormal data is found, the abnormality will be reported, and the proposed algorithm ends without validation. However, as shown in Figure 5, if no potential exception is found after the detection, we are not sure such results are correct. Therefore the speculative result in Figure 5(b) should be validated.

3.3. Validation on speculative results

The validation of speculative results only occurs when abnormalities are not detected after parallel detection. The specific process is as follows: the current data segment obtains the final state, which comes from the detection result on the previous data segment. Then the re-detection for the current data segment operates. Still take data segment I_2 in Figure 5(b) for example. Figure 6 illustrates the specific validation process. After the parallel detection on data segments I_1 and I_2, the validation on the speculative task begins with state 1, and state 1 is the final state when the definitive task on segment

I_1 is over. During the validation, every state that the validation process produced would be compared with the previous speculative task generated state.

As shown in Figure 6, the speculative result for data segment I_2 is stored in the cache. The validation is from *s9* to *s12*, and in total, four rounds of iterations are performed. During the validation, whenever a new state is obtained, it would be compared with the result in cache immediately: If the two states are the same, the validation procedure, as well as the whole parallel algorithm, finishes, and there is no necessity to validate the subsequent speculation results. Because if such a situation occurs, even if the verification continues, the following verification result must be consistent with the speculative result, and there are no abnormalities in high-speed train control data *I*.

I_2	I	R	U	L	E	N	C	E
Cache	0	0	0	0	0	0	0	0
s9	**2**≠0	0	0	0	0	0	0	0
s10		**3**≠0	0	0	0	0	0	0
s11			**4**≠0	0	0	0	0	0
s12				**0**=0	0	0	0	0

Figure 6. Schematic diagram of speculative parallel detection.

If the state generated by iteration during validation is different from that of the speculative result, the validation continues, and the validation process stops in two situations. The first situation is that an acceptance state appears. It demonstrates that the speculative result is proved wrong after validation, and the abnormality is recorded. The second situation is that the validation of the speculative result is over. It demonstrates that even though the speculative result is still incorrect, this fault does not affect the outcome; that is, the high-speed train control data indeed does not contain abnormal data. The efficiency of the speculative parallel algorithm is similar to the conventional, un-parallel algorithm.

In summary, the essence of the validation stage is the duplicate detection for the divided data with speculative results. Therefore, whether the additional detection affects the efficiency of the parallel algorithm is an issue worthy of discussion. Moreover, the validation for the speculative result relies on an assuming value (the last state that comes from the previous detection result), so the accuracy of the validation also needs to be considered. In response to the problems, we make the following analysis.

At first, by cleaning up the logical relations between the speculative parallel detection procedure and the validation of speculative result procedure, it can be seen that both of the two procedures are performed on the divided data concurrently, rather than detecting the data from the beginning to the end. Thus, once an acceptance state is met, it indicates that an anomaly is detected. The parallel algorithm will report the anomaly and stop immediately. In such a case, the parallel algorithm is highly effective compared with the conventional algorithm. Or a state in the validation stage is consistent with the speculative result (for example, step *s12* in Figure 6). The algorithm also finishes early, and the performance is still better than the un-parallel algorithm. In the worst case, the proposed algorithm cannot find abnormalities after the two rounds of parallel data detection. The execution time of the proposed algorithm is only equivalent to the sum of the traditional algorithm's running time and some

parallel overheads, so the detection efficiencies of both the speculative parallel algorithm and the conventional algorithm are almost the same. In total, it can be concluded that the additional detection would not affect the performance of our algorithm markedly.

Secondly, it is necessary to make an argument on the correctness of the validation procedure. The argument can be described as: for any divided data segment u_k, whether the validation for u_k based on the speculative result from its previous segments u_{k-1} is correct. The specific description of this argument and the corresponding proof is demonstrated as Theorem 1.

Theorem 1: For a specific *DFA(N)* who is responsible for detecting the abnormal data *T* in the high–speed train control data, and s_0 is the initial state of the DFA. For any continuous data segments u_{k-1},u_k,u_{k+1}, if $\varphi(s_0,u_{k-1}u_ku_{k+1})=\varphi(s_0,u_{k+1})$, then $\varphi(s_0,u_{k-1}u_ku_{k+1})=\varphi(s_0,u_ku_{k+1})$.

Proof: Assume that Theorem 1 is incorrect; it means that $\varphi(s_0,u_{k-1}u_ku_{k+1})=\varphi(s_0,u_{k+1})=s_1$, and $s_1\neq s_2$. Besides, since that *T* can be recognized by *DFA(N)*, there must be a segment *u*, which can let the result of the state transfer function $\sigma(s_1,u)$ or $\sigma(s_2,u)$ be the acceptance state. Then the following assumptions can be made based on the above conditions.

1st assumption: the result of $\sigma(s_1,u)$ is the acceptance state while the result of $\sigma(s_2,u)$ is not. In this assumption, according to the fundamental state transfer law, it is easy to know that $\sigma(s_1,u)=\sigma(\varphi(s_0,u_{k+1}),u)=\sigma(s_0,u_{k+1}u)$, so the result of $\sigma(s_0,u_{k+1}u)$ is the same as $\sigma(s_1,u)$, and it is apparent that $u_{k+1}u\in T$. Then according to the conversion principle of DFA and regular expression, $u_{k+1}u\in T\Rightarrow u_ku_{k+1}u\in T$, so the result of $\sigma(s_0,u_ku_{k+1}u)$ is also an acceptance state. Based on the deduction, $\sigma(s_0,u_ku_{k+1}u)=\sigma(\varphi(s_0,u_ku_{k+1}),u)=\sigma(s_2,u)$, and it means that the result of $\sigma(s_2,u)$ is as same as the result of $\sigma(s_1,u)$. So, contradiction happens, and the 1st assumption is invalid.

2nd assumption: the result of $\sigma(s_2,u)$ is the acceptance state while the result of $\sigma(s_1,u)$ is not. Then according to the state transfer law, $\sigma(s_2,u)=\sigma(\varphi(s_0,u_ku_{k+1}),u)=\sigma(s_0,u_ku_{k+1}u)$, so the result of $\sigma(s_0,u_ku_{k+1}u)$ is the same as $\sigma(s_2,u)$, and it is apparent that $u_ku_{k+1}u\in T$. According to the conversion principle of DFA and regular expression, $u_{k+1}u\in T\Rightarrow u_{k-1}u_ku_{k+1}u\in T$, so the result of $\sigma(s_0,u_{k-1}u_ku_{k+1}u)$ is an acceptance state, and $\sigma(s_0,u_{k-1}u_ku_{k+1}u)=\sigma(\varphi(s_0,u_{k-1}u_ku_{k+1}),u)=\sigma(s_1,u)$. It means the result of $\sigma(s_1,u)$ is as same as the result of $\sigma(s_2,u)$. So, contradiction happens, and the 2nd assumption is also invalid.

In conclusion, the two assumptions all make contradictions, $s_1 = s_2$, and Theorem 1 is tenable.

Through Theorem 1 and its proof, it can be known that even if the validation procedure is based on the speculative execution result, the outcome of the validation is still correct. To sum up, the accuracy of the validation stage can be guaranteed.

4. Algorithm implementation

The implementation platform for the proposed algorithm is Apache Spark, which is the most popular distributed computing platform in both the industrial and the scientific fields. Apache Spark with high concurrency and reliability is widely used as the general computing engine for data-intensive works in power, communications, and other areas [28–32]. According to the design philosophy of the CTCS-3 control system, Apache Spark would be deployed on the communication server. Therefore, the proposed algorithm would also work on the communication server-based Apache Spark.

Before the implementation, some parameters should be adjusted. Firstly, the task scheduling policy on Apache Spark is switched from FIAR to FIFO. In such a policy, the proposed algorithm can dynamically adjust the parallelism degree based on computation resource amount and ensure the smooth operation of every parallel task. Besides, the working threads for each task, the memory size, and other parameters should also be adjusted according to the standard tuning measure on Apache Spark. At last, the degradation of the local data latency is limited to 3000 ms so that some parallel tasks with high latency would not slow down the overall performance of the proposed algorithm.

Based on parameter adjustment, Table 2 shows the pseudo-code of the proposed algorithm.

In Table 2, algorithm 2 consists of four parts: the first part is from lines 1 to 2, whose responsibility is to calculate the data division step according to Eq (3) and initialize the state cache. The second part is from line 3 to line 5, and the main job of this part is to divide the high-speed train control data into data segments of the same length. The third part starts from line 6 and ends at line 17, and it is the speculative parallel detection part. Unlike the detection part in algorithm 1, every state that the DFA transferred should be recorded for validation later during the parallel detection part. The last part is the validation part from line 18 to line 31. In this part, the second round of data detection based on the speculative results is in parallel. In particular, line 27 and line 28 mean that if the current DFA state and the state in cache is consistent, it indicates that the speculative execution result is correct, no further validation is required, and the algorithm ends immediately.

Besides, the deployment diagram of the speculative parallel detection algorithm for high-speed train control data on the distributed computing platform Apache Spark is shown in Figure 7. Figure 7(a) shows the deployment diagram of the conventional serial data detection algorithm. It can be seen from Figure 7(a) that since the data cannot be divided in the traditional algorithm, the execution process in Figure 7(a) is restricted by the data composition order; thus, the conventional algorithm is time-consuming and unable to make full use of the computing resources. However, in Figure 7(b), by the speculative parallel optimization technique, all computing nodes in the cluster-including the main node and the worker nodes, are joining in the computation, and the high-speed train control data can be divided and conquered, which not only improves the detection efficiency of the train control data but also provide the effective use of the computing resources of the distributed

computing platform.

Table 2. Pseudo-code of algorithm 2.

Algorithm 2: Speculative parallel detection algorithm for high-speed train control data
Input 1: Train control data D1
Input 2: Specific DFA that recognizes the abnormality
Output: Position j where abnormality is found

```
1    n = ceil ( formula3, D1.size )
2    buffer [D1.size/n][n] = NULL
3    In parallel:{
4      Specblk[D1.size/n] = divide (D1, n)
5    }
6    Parallel_for (i = 0 ; i < Specblk.length ; i++):{
7      ω = Specblk[i]
8      I = Specblk[i].length
9      state = start_state
10       for j = 0 to |I-1| do:{
11          input char =ω[j]
12          state = δ[state][input char]
13          if state∈F then
14             return j
15          else buffer[i][j] = state
16          }
17   }
18   Parallel_for (i = 1 ; i < Specblk.length ; i++):{
19     ω = Specblk[i]
20     I = Specblk[i].length
21     state = buffer[ |I-1| ][i-1]
22     for j = 0 to |I-1| do:{
23        input char =ω[j]
24        state = δ[state][input char]
25        if state∈F then
26           return j
27        else if buffer[i][j] == state
28                 return NotFound
29     }
30     return NotFound
31   }
```

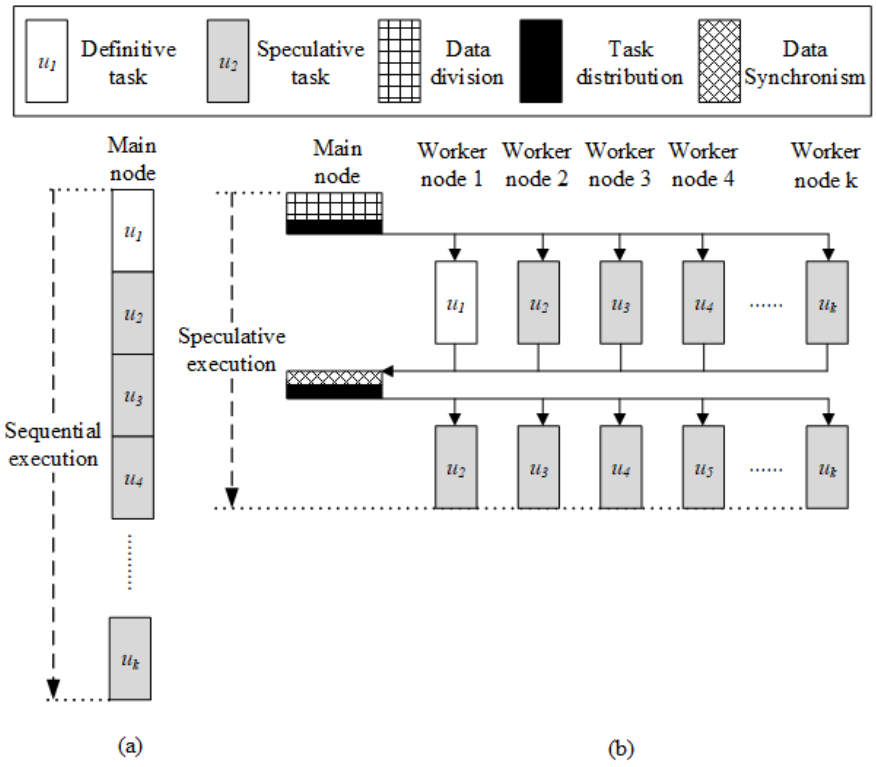


Figure 7. Deployment diagram of the speculative algorithm on Apache Spark.

Last but not least, Figure 7(b) also shows the time cost for the proposed algorithm in the worst case. If the abnormal data is found or the validation succeeds within a few iterations in the actual runtime environment, the algorithm will end earlier. In addition, data synchronization occurs after the end of the speculative parallel detection, and its purpose is to submit the cache data in each parallel task to the master node, in order to ensure the validation of the speculation result.

5. Experimental results and analysis

5.1. Experimental design

First of all, in cooperating with Shaanxi Provincial Key Laboratory of Network Computing and Security Technology, we built our distributed cluster on the high-speed EMU network control platform. The operating system is Ubuntu Server 20.04, while the distributed computing cluster is Apache Spark 2.2.0. The cluster is built up in the master-slave model with nine Lenovo Think Server TS80X servers.

The experimental scheme and specific configuration are shown in Table 3. Five groups of schemes are designed in this experiment, involving traditional serial detection algorithm and speculative parallel detection algorithm. In scheme 1, only one working node is equipped for this experiment because the traditional algorithm cannot use additional computing resources to detect train control data in parallel. Instead, other schemes in Table 3 are equipped with a different number of working nodes to test the performance of the proposed algorithms supported by the different scales of the computing resources.

Table 3. Experimental scheme and configuration.

Name	Experimental subject	Specific configuration
Scheme 1	Traditional serial algorithm	1 main node + 1 working node
Scheme 2	Speculative parallel algorithm	1 main node + 2 working nodes
Scheme 3	Speculative parallel algorithm	1 main node + 4 working nodes
Scheme 4	Speculative parallel algorithm	1 main node + 6 working nodes
Scheme 5	Speculative parallel algorithm	1 main node + 8 working nodes

Besides, with the help of Xi'an Railway Bureau, we randomly select eight pieces of high-speed train control data from all trains running from July to December 2020. Besides, all train models and the corresponding marshaling orders are included in the experiment. The specific experimental data is split into two sets according to the marshaling order; and the details are shown in Table 4.

Table 4. Information of the experimental data.

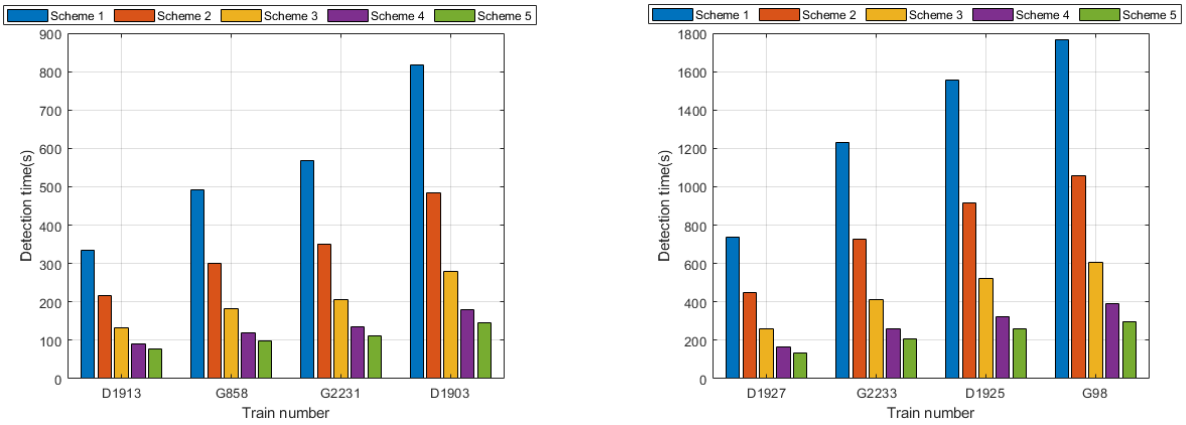
Dataset	Train number	Time	Model	Marshaling	Data size
Dataset 1	D1913	4 h 7 min	CRH380B	8	10.43 GB
Dataset 1	G858	4 h 28 min	CR400AF	8	12.05 GB
Dataset 1	G2231	5 h 31 min	CRH380B	8	14.69 GB
Dataset 1	D1903	7 h 40 min	CRH380B	8	19.79 GB
Dataset 2	D1927	3 h 32 min	CRH380B	16	18.11 GB
Dataset 2	G2233	5 h 56 min	CRH380B	16	30.19 GB
Dataset 2	D1925	7 h 7 min	CRH380B	16	38.54 GB
Dataset 2	G98	7 h 42 min	CRH380AL	16	43.27 GB

As to the detection rule, 217 regular expression rules from the famous intrusion detection system Bro are selected to construct a complex DFA with 8094 states in our experiment. In addition, the data division granularity is set 32 MB by Eq (3) during the experiment.

At last, all experimental schemes on each set of data are performed ten times to ensure the accuracy of the results, and the average value of the results on ten executions will be adopted as the experimental results. If the variance of the results on ten executions is enormous after statistics, the experiment will be repeated.

5.2. Experimental results

The experimental results on five different schemes with two sets of experimental data are in Figure 8. As Figure 8 demonstrates, the abscissa represents the experimental data, and the ordinate represents the time. Each column represents the time it takes to process a column of experimental data using a practical scheme. In addition, Figure 8(a) shows the performance of different experimental schemes for data set 1, while Figure 8(b) shows the performance of different experimental schemes when processing data set 2.



(a) (b)

Figure 8. Experimental result for datasets 1 & 2.

It can be seen from Figure 8 that compared with the traditional algorithm, the speculative parallel algorithm with multiple working nodes significantly reduces the detection time for the train control data. Meanwhile, by observing different experimental schemes in the same dataset, it can be seen that along with the increment of the computing resources, the execution efficiency of the speculative parallel algorithm grows gradually.

Moreover, by summarizing the performance of schemes 1 and 2 in Figure 8, it is easy to know that for the large-scale high-speed train control data. The average time consumption of the speculative parallel algorithm based on dual working nodes is reduced by about 35% compared with the traditional serial algorithm. Similarly, in schemes 1 and 3, schemes 1 and 4, and schemes 1 and 5 from Figure 8, it can be seen that when comparing with the traditional, un-parallel algorithm, the speculative parallel algorithm based on 4/6/8 working nodes reduces the detection time by 62, 70, and 80% respectively. In total, the experiment shows that compared with the traditional DFA-based abnormal data detection algorithm, the speculative parallel detection algorithm can significantly improve the detection efficiency for high-speed train control data.

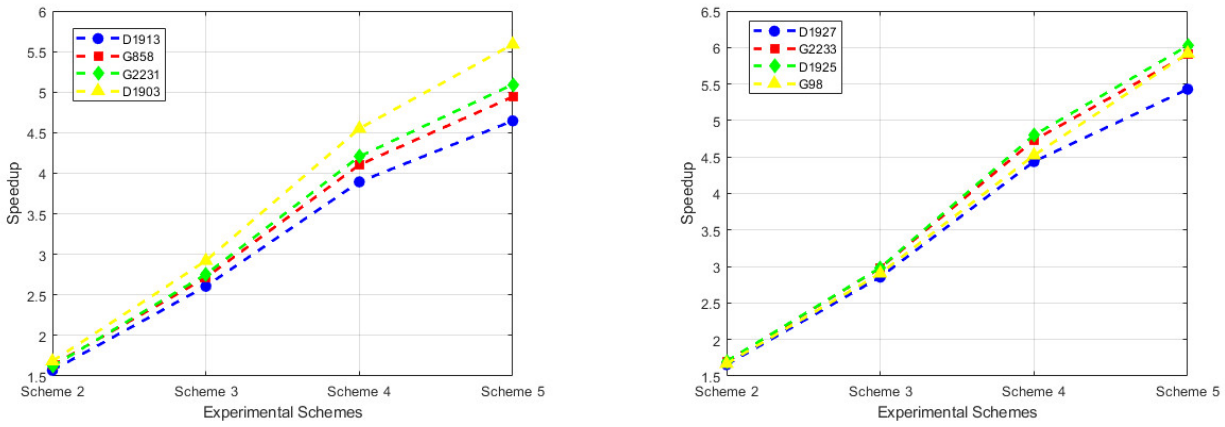
5.3. Analysis of the experimental result

The analysis of experimental results is designed to make a deep understanding of the growth law of the proposed algorithm. The speedup of each experimental scheme is calculated as Eq (6). In Eq (6), s represents the speedup value, T_1 indicates the conventional algorithm execution time, which is also known as the time cost in experimental scheme 1, and T_n represents the speculative parallel algorithm execution time.

$$s = \frac{T_1}{T_n} \tag{5}$$

After calculation, the results are collected and counted, and Figure 9 illustrates the change of the speedup curve. In this figure, the abscissa represents the experimental scheme, and the ordinate

represents the speedup value.



(a) (b)

Figure 9. Speedup change on the parallel experimental scheme.

Specifically, the speedup changes on parallel experimental schemes (Scheme 2 to Scheme 5) with data sets 1 and 2 are illustrated in Figure 9(a),(b), respectively. By observing the changing patterns on different speedups, it can be concluded that our algorithm shows good scalability. The speedup grows steadily and regularly when the computing resources keep increasing.

Besides, after calculating the slope of each speedup curve in Figure 9(a), a phenomenon can be observed that the curve slopes in Schemes 2 and 3 are large, decreasing in Schemes 4 and 5. This phenomenon happens in every speedup curve in Figure 9(a), and it means that the algorithm's scalability at the very beginning is better than the one in the end. In a real environment, it indicates that in Schemes 2 and 3, the investment of computing resources leads to a significant improvement in algorithm efficiency. However, in Schemes 4 and 5, an obvious diminishing marginal utility appears. It brings a problem that with the continuous investment of computing resources, the performance improvement of parallel algorithms is becoming increasingly limited.

The same change patterns are also witnessed in Figure 9(b), proving that this phenomenon is common in the proposed algorithm. This phenomenon is because the continuous investment of computing resources will increase network communications and data exchanges between different computing nodes in the cluster, resulting in the proposed algorithm's performance no longer maintaining the same growth rate.

In addition, comparing the curves in Figure 9 (a),(b), it can be seen that in the same experimental schemes, the curves' slopes in small size data (for instance, D1913 and G858 in dataset 1) are always greater than the curves' slopes in big size data (for example, D1925 and G98 in dataset 2). Because the increase in the amount of data will also affect the algorithm's inner processes like task scheduling, process communication, and data exchange, which in turn affect the performance of the proposed algorithm. Therefore, the focus of subsequent research work will be how to optimize the scheduling strategy of parallel units in the algorithm, reduce data exchange, and reduce communication delay. Furthermore, since the edge computing model and IoT computing devices are more flexible than the conventional centralized computing model [33–35], implementing the proposed on edge computing

equipments would also be the future work direction.

6. Conclusions

This paper proposes a parallel data detection algorithm for a high-speed rail control system based on the parallel speculation optimization technique and Apache Spark. The parallel speculation optimization technique helped overcome the complex dependencies that impact the detection order for high-speed train control data, so that the data can be detected in a parallel and speculative way. The introduction of Apache Spark provided support for large-scale concurrency. The experiment and the corresponding analysis showed that the proposed algorithm obtained better performance than the conventional data detection algorithm. Besides, the proposed algorithm also possessed outstanding scalability, and the detection efficiency could be markedly improved by increasing more computing resources. At present, with the support of Xi'an Railway Bureau, this algorithm has been implemented in the railway control center to help detect the control data generated by high-speed trains, and the overall performance was very satisfying. However, through the experiment, it was also known that there was still room for algorithm optimizations, and in-depth research on the scheduling strategy for the proposed algorithms in distributed systems would continue in the future.

Acknowledgments

This research work is supported in part by the National Joint Funds of China (U20B2050), National Key R&D Program of China (2018YFB1201500), National Natural Science Foundation of China (61801379), and Special Scientific Research Project of Shaanxi Provincial Department of Education (21JK0781).

Conflict of interests

All authors declare no conflict of interest in this paper.

References

1. Y. Chen, Non-safety-related software in the context of railway RAMS standards, in *2017 IEEE Second International Conference on Reliability Systems Engineering (ICRSE)*, (2017), 1–5. doi: 10.1109/ICRSE.2017.8030718.
2. T. Wang, Y. Quan, X. Shen, T. R. Gadekallu, W. Wang, K. Dev, A Privacy-Enhanced Retrieval Technology for the Cloud-assisted Internet of Things, *IEEE Trans. Ind. Inf.*, 2021. doi: 10.1109/TII.2021.3103547.
3. C. Feng, K. Yu, M. Aloqaily, M. Alazab, Z. Lv, S. Mumtaz, Attribute-based encryption with parallel outsourced decryption for edge intelligent IoV, *IEEE Trans. Veh. Technol.*, **69** (2020), 13784–13795. doi: 10.1109/TVT.2020.3027568.
4. C. Feng, B. Liu, Z. Guo, K. Yu, Z. Qin, K. R. Choo, Blockchain-based Cross-domain Authentication for Intelligent 5G-enabled Internet of Drones, *IEEE Internet Things J.*, 2021. doi: 10.1109/JIOT.2021.3113321.

5. L. Tan, K. Yu, F. Ming, X. Chen, G. Srivastara, Secure and resilient Artificial Intelligence of Things: a HoneyNet approach for threat detection and situational awareness, *IEEE Consum. Electron. Mag.*, 2021. doi: 10.1109/MCE.2021.3081874.
6. K. Yu, Z. Guo, Y. Shen, W. Wang, J. C. Lin, T. Sato, Secure artificial intelligence of things for implicit group recommendations, *IEEE Internet Things J.*, 2021. doi: 10.1109/JIOT.2021.3079574.
7. A. Aghdai, C. Chu, Y. Xu, D. Dai, J. Xu, J. Chao, Spotlight: Scalable transport layer load balancing for data center networks, *IEEE Trans. Cloud Comput.*, **8** (2020), 1471–1485. doi: 10.1109/tcc.2020.3024834.
8. K. Yu, M. Arifuzzaman, Z. Wen, D. Zhang, T. Sato, A key management scheme for secure communications of information centric advanced metering infrastructure in smart grid, *IEEE Trans. Instrum. Meas.*, **64** (2015), 2072–2085. doi: 10.1109/TIM.2015.2444238.
9. L. Zhang, M. Peng, W. Wang, Z. Jin, Y. Su, H. Chen, Secure and efficient data storage and sharing scheme for blockchain-based mobile-edge computing, *Trans. Emerging Telecommun. Technol.*, (2021), 4315. doi: 10.1002/ett.4315.
10. W. Wang, H. Xu, M. Alazab, T. R. Gadekallu, Z. Han, C. Su, Blockchain-based reliable and efficient certificateless signature for IIoT devices, *IEEE Trans. Ind. Inf.*, 2021. doi: 10.1109/tii.2021.3084753.
11. L. Tan, K. Yu, N. Shi, C. Yang, W. Wei, H. Lu, Towards secure and privacy-preserving data sharing for covid-19 medical records: A blockchain-empowered approach, *IEEE Trans. Network Sci. Eng.*, 2021. doi: 10.1109/TNSE.2021.3101842.
12. H. Xiong, C. Jin, M. Alazab, K. Yeh, H. Wang, T. R. Gadekallu, et al., On the design of blockchain-based ECDSA with fault-tolerant batch verication protocol for blockchain-enabled IoMT, *IEEE J. Biomed. Health Inf.*, 2021. doi: 10.1109/JBHI.2021.3112693.
13. J. Song, Z. Han, W. Wang, J. Chen, Y. Liu, A new secure arrangement for privacy-preserving data collection, *Comput. Stand. Interfaces*, (2021), 103582. doi: 10.1016/j.csi.2021.103582.
14. Z. Liu, A. K. Nath, X. Ding, H. Fu, M. Khan, W. Yu, Multivariate modeling and two-level scheduling of analytic queries, *Parallel Comput.*, **85** (2019), 66–78. doi: 10.1016/j.parco.2019.01.006.
15. K. Yu, L. Tan, L. Lin, X. Cheng, Z. Yi, T. Sato, Deep-learning-empowered breast cancer auxiliary diagnosis for 5GB remote E-health, *IEEE Wireless Commun.*, **28** (2021), 54–61. doi: 10.1109/MWC.001.2000374.
16. W. Wu, S. Xu, Intrusion detection based on dynamic gemini population DE-K-mediods clustering on hadoop platform, *Int. J. Pattern Recognit. Artif. Intell.*, **35** (2021), 2150001. doi: 10.1142/S0218001421500014.
17. Y. Gong, L. Zhang, R. Liu, K. Yu, G. Srivastava, Nonlinear MIMO for industrial Internet of Things in cyber-physical systems, *IEEE Trans. Ind. Inf.*, **17** (2020), 5533–5541. doi: 10.1109/TII.2020.3024631.
18. Y. Song, Y. Wen, D. Zhang, J. Zhang, Fast prediction model of coupling coefficient between pantograph arcing and GSM-R antenna, *IEEE Trans. Veh. Technol.*, **69** (2020), 11612–11618. doi: 10.1109/TVT.2020.3015057.
19. N. M. Balamurugan, S. Mohan, M. Adimoolam, A. John, T. R. Gadekallu, W. Wang, DOA tracking for seamless connectivity in beamformed IoT-based drones, *Comput. Stand. Interfaces*, **79** (2022), 103564. doi: 10.1016/j.csi.2021.103564.

20. F. Z. Beana, N. Benamrane, M. A. Hamaida, A. M. Chaabani, A. T. Ahmed, Network Intrusion Detection System using neural network and condensed nearest neighbors with selection of NSL-KDD influencing features, in *2020 IEEE International Conf. Internet Things Intell. Syst.(IoTaIS)*, Bali, (2021), 23–29. doi: 10.1109/IoTaIS50849.2021.9359689.
21. M. D. Mauro, G. Galatro, A. Liotta, Experimental review of neural-based approaches for network intrusion management, *IEEE Trans. Network Serv. Manage.*, 2020. doi: 10.1109/TNSM.2020.3024225.
22. M. Injadat, A. Moubayed, A. B. Nassif, A. Shami, Multi-stage optimized machine learning framework for network intrusion detection, *IEEE Trans. Network Serv. Manage.*, 2020. doi: 10.1109/TNSM.2020.3014929.
23. Z. Guo, K. Yu, A. Jolfaei, A. K. Bashir, A. O. Almagrabi, N. Kumar, A fuzzy detection system for rumors through explainable adaptive learning, *IEEE Trans. Fuzzy Syst.*, 2021. doi: 10.1109/TFUZZ.2021.3052109.
24. Z. Guo, K. Yu, Y. Li, G. Srivastava, J. C. Lin, Deep learning-embedded social internet of things for ambiguity-aware social recommendations, *IEEE Trans. Network Sci. Eng.*, 2021. doi: 10.1109/TNSE.2021.3049262.
25. L. Ding, K. Huang, D. Zhang, Multi-stride regular expression matching using parallel character index, *J. Comput. Res. Dev.*, **52** (2015), 681–690. doi: 10.7544/issn1000-1239.2015.20131255.
26. S. Lu, Z. Zuo, L. Wang, Progress in parallelization of static program analysis, *J. Software*, **31** (2020), 1241–1254. doi: 10.13328/j.cnki.jos.005950.
27. R. Yang, C. Hu, X. Sun, P. Garraghan, T. Wo, Z. Wen, et al., Performance-aware speculative resource oversubscription for large-scale clusters, *IEEE Trans. Parallel Distrib. Syst.*, **31** (2020), 1499–1571. doi: 10.1109/TPDS.2020.2970013.
28. H. Dong, H. Zhu, Y. Li, Y. Lv, S. Gao, Q. Zhang, et al., Parallel intelligent systems for integrated high-speed railway operation control and dynamic scheduling, *IEEE trans. Cybern.*, **48** (2018), 3381–3389. doi: 10.1109/TCYB.2018.2852772.
29. H. Li, K. Yu, B. Liu, C. Feng, Z. Qin, G. Srivastava, An efficient ciphertext-policy weighted attribute-based encryption for the Internet of health things, *IEEE J. Biomed. Health Inf.*, 2021. doi: 10.1109/JBHI.2021.3075995.
30. C. Feng, B. Liu, K. Yu, S. K. Goudos, S. Wan, Blockchain-empowered decentralized cross-domain federated learning for 5G-enabled UAVs, *IEEE Trans. Ind. Inf.*, 2021. doi: 10.1109/TII.2021.3116132.
31. L. Zhen, Y. Zhang, K. Yu, N. Kumar, A. Barnawi, Y. Xie, Early collision detection for massive random access in satellite-based internet of things, *IEEE Trans. Veh. Technol.*, 2021. doi: 10.1109/TVT.2021.3076015.
32. W. Shang, J. Chen, H. Bi, Y. Sui, Y. Chen, H. Yu, Impacts of covid-19 pandemic on user behaviors and environmental benefits of bike sharing: A big-data analysis, *Appl. Energy*, **285** (2021), 116429. doi: 10.1016/j.apenergy.2020.116429.
33. Y. Sun, J. Liu, K. Yu, M. Alazab, K. Lin, PMRSS: Privacy-preserving medical record searching scheme for intelligent diagnosis in IoT healthcare, *IEEE Trans. Ind. Inf.*, 2021. doi: 10.1109/TII.2021.3070544.
34. L. Liu, J. Feng, Q. Pei, C. Chen, Y. Ming, B. Shang, et al., Blockchain-enabled secure data sharing scheme in mobile-edge computing: An asynchronous advantage actor-critic learning approach, *IEEE Internet Things J.*, **8** (2020), 2342–2353. doi: 10.1109/JIOT.2020.3048345.

35. J. Feng, F. R. Yu, Q. Pei, X. Chu, J. Du, L. Zhu, Cooperative computation offloading and resource allocation for blockchain-enabled mobile-edge computing: A deep reinforcement learning approach, *IEEE Internet Things J.*, **7** (2019), 6214–6228. doi: 10.1109/jiot.2019.2961707.

AIMS Press