# Efficient Data Integrity Verification Using CRC Based on HDFS in Cloud Storage

Yun-Hao XIA[1], Han-Shu HONG[1], Guo-Feng LIN[1] and Zhi-Xin SUN[1,a]

[1]*School of Internet of Things, Nanjing University of Posts and Telecommunications, Nanjing, China*

**Abstract.** Data integrity verification is becoming a major challenge in cloud storage which can't be ignored. This paper proposes an optimized variant of CRC (Checker Redundancy Cyclic) verification algorithm based on HDFS to improve the efficiency of data integrity verification in cloud storage through the research of CRC checksum algorithm and data integrity verification mechanism of HDFS. A new method is formulated to establish the deformational optimization and to accelerate the algorithm by researching characteristics of generating and checking the algorithm. Moreover, this method optimizes the code to improve the computational efficiency according to data integrity verification mechanism of HDFS. A data integrity verification system based on Hadoop is designed to verify proposed method. Experimental results demonstrate that proposed HDFS based CRC algorithm was able to improve the calculation efficiency and the utilization of system resource on the whole and outperformed well compared to existing models in terms of accuracy and time.

## 1 Introduction

With the rapid development of the big data era, data volumes are exploding grow, resulting in higher storage costs simultaneously with increasing the difficulty of data storage management. Cloud storage [1] serves as a kind of "infrastructure as a service" form of expression, which can provide a high performance. low cost. easy management and virtualization storage scheme. With the development of cloud storage technology [2] and the improvement of network bandwidth and reliability, more and more users can store data in the cloud server. However, the problem of data security [3] is still a major problem in cloud storage which can't be ignored. On the one hand, the security problem caused by data leakage; on the other, the data integrity problems caused by cloud service providers themselves. Data security can be guaranteed by the data encryption technology [4], and data integrity is verified by the verification algorithm [5].

Data integrity verification [6, 7] ensures that the acquired data is consistent with the previously saved data. It uses a specific algorithm to calculate the original data and gets a calibration value. At the same time the receiver uses the same algorithm to calculate once again. If the two check value is the same, the data is complete. Data integrity verification is generally used as the underlying, background or built-in module, and the most common three data integrity verification algorithms are CRC, MD5 and SHA-1.

[a] Corresponding author: sunzx@njupt.edu.cn

At present, many companies have launched cloud storage platform and cloud services, most famous the Hadoop [8], which was greatly welcomed by the industry. Pig, ZooKeeper, HBase, Hive and HDFS [9] are derived from Hadoop. The distributed file system HDFS has its unique value and advantage, and it is the basis of system development of Yahoo, IBM and Amazon and other Large Firm system. In the distributed file system HDFS, DataNode of the file system stores the metadata for CRC verification, and the whole process of data verification ensures the integrity and security of the data in the transmission process. However, it also brings about a series of problems, such as frequent switching of JVM, system load imbalance, non-ideal speed of read and write, degradation of the performance of IO, et al, bringing great burden to data processing system.

This paper does some research in data integrity verification mechanism of HDFS, and puts forward optimization of CRC verification algorithm and verification method to improve the performance of the system. This paper proposes an optimized variant of CRC verification algorithm based on HDFS, as the underlying, background or built-in module in the cloud storage system. It avoids the loss of data caused by the potential damage by outsourcing if the enterprise or individual stores its own data into the cloud storage space of a cloud service provider. The second section of this paper will introduce the CRC-n algorithm in detail, and seek to improve the method, and analyze the operating mechanism of the distributed file system HDFS guaranteeing data integrity. The third section of this paper will exhaustively describe the optimization method based on the second section. In fourth section, effectiveness of the proposed optimization method is verified by the experimental test data of the prototype system, and the contents of the paper are summarized in the end.

## 2 Related works

Currently, there are many algorithms for verifying data integrity [10-12] in the storage system. A class of methods based on the homomorphic techniques [13] which generate and store a check element for each data block to realize unlimited time verification. However, this kind of methods have a defect in the range of data coverage in the face of large data file verification, because the calculation is related with modular arithmetic in finite fields, and it need to consume a large amount of computing resources. The other is based on the hash algorithm to determine whether the data is complete, Using MD5 [14], SHA-1[15], CRC and other one-way hash function to generate hash value to verify the consistency of the two data objects. This kind of algorithm is characterized by small computational overhead, but it usually only supports once request for verification. HDFS of Hadoop has been involved in CRC32 and MD5.

CRC-n [16] (Checker Redundancy Cyclic) is one of the most common error check codes in the field of digital data communication, mainly using linear encoding theory. In data sending end, according to m bit binary code information sequence BCISm, produce n bit check code, that is, CRC code BCISn by a certain rule (verification algorithm rules). BCISn is attached to the information sequence BCISm, and finally send out a new (m+n) binary code information sequence BCISm+n. At data receiving end, according to the rules of the algorithm between the code BCISm and CRC code BCISn, verify the integrity of file to confirm if there is an error in the transmission process. Different constants correspond to different CRC algorithms, and when this constant is 32, that is the CRC32.

CRC-n generator polynomial is

$$(Mx^n + N) \bmod G = 0$$

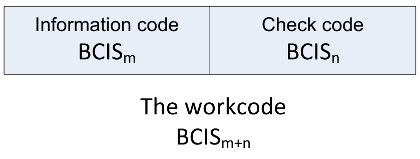

**Figure 1.** The binary code information sequence.

In the application of big data and high performance computing, the traditional CRC generation and inspection methods are faced with several problems. One problem is speed and another is data width. The size of the incoming bit stream in the execution process is different. An arbitrary number of invalid bits in the bit stream are required to be processed by the CRC calculation. The rate of CRC algorithm can be improved by hardware. However, due to the limited experimental conditions, coupled with the features that the low cost of cloud computing server to extend, we want to improve the verification efficiency by finding a variant of the CRC algorithm which seeks a suitable CRC-n algorithm and changes the number of bits checked and the size of the register during CRC calculation.

Kounavis and Berry [17] made some improvements in CRC algorithm of data integrity verification in cloud storage system. They improved CRC32 algorithm by seeking one single-bit table, and applied it in data processing platform Hadoop. The improvement was suitable for the data environment in that time and it has achieved some success. However, with the explosive growth of data and complex diversity of data types, the improvement that check 8 bits (a table) every time has been unable to meet the business needs of the current big data environment. In addition, HDFS need to switch JVM frequently in the data verification. Through the results of the experiments, the time of switching JVM accounted for 5/6 of the entire check time, bringing huge computational cost for data verification. Therefore, we need to further optimize the verification algorithm in facing with increasingly high performance computing applications demand.

Jing [18] presented a better optimization in the aspect of the method of seeking tables using CRC algorithm, and to a certain extent, it improved the efficiency of the algorithm. In this research, data integrity verification also optimized and improved CRC32 algorithm and used the method of seeking single-bit table, but the number of table increased, so that data integrity verification could verify more bits each time, hence improving the rate of data integrity verification. With the improvement of the hardware, memory expansion and the increase of bandwidth, at the same time CRC64 has a wider range of applications than CRC32 because the n of CRC-n is bigger, the algorithm is more reliable, there is room for further improvement of CRC-n data integrity verification.

Based on above-mentioned condition, this paper researches on data integrity verification technology in big data environment based on the open source cloud computing platform Hadoop, focusing on the problems existing in the optimization of the way to call checksum algorithm and the number of verified bits each time. This paper proposes a distributed file system verification model, optimizing verification algorithm and the way to call verification algorithm based on HDFS, in order to achieve the purpose of improving the system performance. This scheme not only ensures the high reliability of digital data in big data era, but also does not give the user and cloud server too heavy burden, while the process of verifying can protect privacy of user's data.

## 3 Improved CRC algorithm for data integrity verification

### 3.1 Problem description

Facing big data and high performance computing demand, there are a series of problems in data integrity verification in hadoop, such as frequent switching of JVM, system load imbalance, non-ideal speed of read and write, degradation of the performance of IO, et al,. For example, as shown in literature [13], nowadays in the HDFS, CRC32 verification algorithm was called by java.util.zip.CRC32 using Java Native Interface. Use the class java.util.zip to call the CRC32 function via the Java local interface (JNI). However, the efficiency of this method was low when it comes to calling the fine-grained file. If a file size is 100GB, the CRC32 value is only 800M, and does a CRC32 calculation each 512byte. During the process, it needs to switch the JVM frequently. The time of switching JVM will reach the entire verification time of 5/6. Thus, HDFS used the method of block storage strategy to improve the ability of system data backup and data fault tolerance, but also brought a huge computational cost.

## 3.2 Improved CRC algorithm

According to description in second section, this paper proposes a method of data integrity verification using CRC algorithm based on HDFS. In order to solve the problem of data integrity verification in current cloud storage, this paper researches system verification mechanism of hadoop, and makes efficiency optimization of the verification algorithm. Optimized method called CRC64_16_4 in the following paper can improve the verification efficiency of the distributed file system, raise system verification rate and ensure accuracy. The optimization in this paper is shown as follows: efficiency optimization variant of verification algorithm: choose more reliable CRC64 verification algorithm for reducing the impact of a collision and change the number of bits checked each time and the size of the register during CRC calculation; Optimize the code to solve a series of problems facing big data and high performance computing demand and improve the computational efficiency.

From the theoretical point of view, CRC could not complete reliably verify data integrity, because CRC polynomials are linear structures, and it is easy to reach CRC collisions by changing data. There is a more popular explanation, assuming a string with a CRC check code in the transmission, if a continuous error happens and when the number of error times reach a certain number, then almost certainly appear a collision(value is incorrect but the result of CRC is correct). However, with n of CRC-n increasing, the probability of collision will be significantly decreased. For example, CRC32 is more reliable than CRC16, and CRC64 is more reliable than CRC32.

Currently CRC32 verification algorithm used in Hadoop was calculated by checking tables. The number of tables was one and the method verified 8 bits each time by checking one table. However, this kind of method is not satisfied with computational requirements of big data nowadays. This paper will improve the verification efficiency by finding a variant of CRC algorithm which seeks a suitable CRC-n algorithm and changes the number of bits checked and the size of the register during CRC calculation. Put forward a new way to check tables: use CRC64 verification algorithm, check double-bite table, and provide four tables to check. Taking 16 as an incremental unit, 16 bit, 32 bit, 48 bit and 64 bit of verified data are calculated by XOR using CRC64 generating polynomial. Create (crc>>>16) ^T1, CRC (T4^T3^T2^T1). More bits of information field to be verified by checking fewer tables. The optimization accelerates the algorithm to guarantee and improve the verification rate. Considering the room which was used to store the tables, proposed method CRC64_16_4 would need more, but it has a very small impact on verification efficiency with storage devices nowadays, which could be seen in the following experiment. Calculation process of optimized variant of CRC-64 verification algorithm is shown in Figure 2.

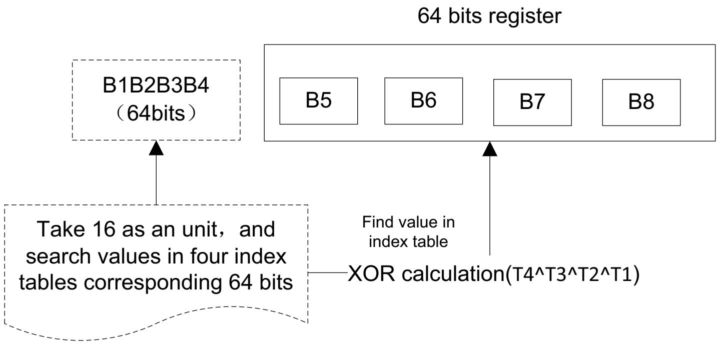
64 bits register
B1B2B3B4
（64bits）
B5
B6
B7
B8
Find value in index table
Take 16 as an unit，and search values in four index tables corresponding 64 bits
XOR calculation(T4^T3^T2^T1)

**Figure 2.** Calculation process of optimized CRC64.

In order to improve the work efficiency of verification, the quality of code must be improved combining with verification mechanism of HDFS. Therefore, quality of CRC64_16_4 code proposed in this paper was optimized to enhance the performance. Comparing with CRC algorithm called by

Sun Native and other optimized methods, proposed method used improved Java could enhance system efficiency.

## 4 System implementation and performance test

In our experiment, the data integrity verification system based on Hadoop was designed and realized. Hadoop version was 2.2.0, server configuration was apache-tomcat-7.0.42, operating system was CentOS-6.3-i386/Linux, Client was Chrome kernel browser, system architecture was B/S. The purpose of this system is to facilitate the user to manage cloud data and realize data integrity verification. The working process of system is shown in Figure 3.

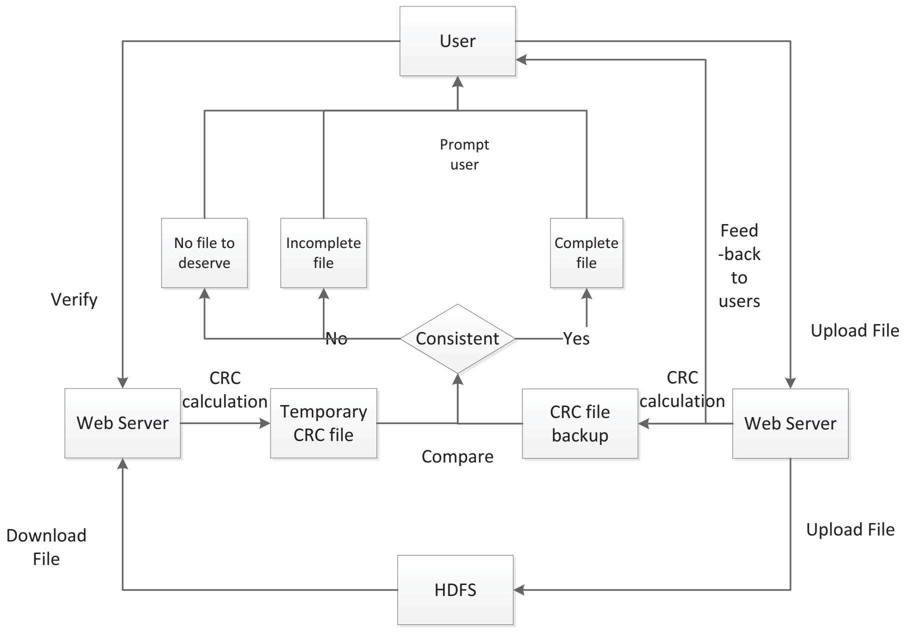

**Figure 3.** System architecture.

Possible outcomes of the system are as Table 1 shows.

**Table 1.** Possible outcomes of the system.

| Results | Instructions | System states |
|---|---|---|
| Success | File is complete | Correct verification! Whether to download? |
| | File is incomplete | Verified file is incomplete! Please download carefully!<br>Wrong location: the first 512 bytes |
| Failure | No CRC files to deserve | Could not find the corresponding CRC file on your server, and unable to identify the reliability of the file.<br>If you upload this file, please download and verify it. |

The objects of experiment were SunNative CRC32 of HDFS in the 2.2.0 version of hadoop, CRC32_8_8 checking eight single-byte tables and optimized CRC64_16_4 checking four double-byte tables. Experiments used JDK1.7, 64 bit, 3.10GHz machines for testing.

Verification process of the system is shown in Figure 4.

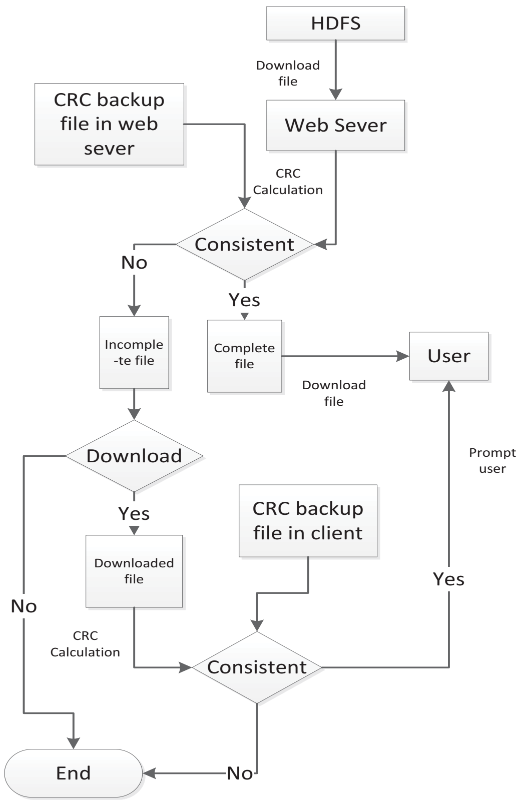

**Figure 4.** Verification process of the system.

In order to verify the correctness of the proposed method in this paper, choose a variety of different sizes of documents between 10M to 10G, and test them 2000 times in two cases that file error probability 1% and 0.5%. The results are shown in the following Table 2.

**Table 2.** Correctness of the proposed method.

| Verification scheme | File error probability | Correct check probability | Success to report error | Error check probability | Fail to report error |
|---|---|---|---|---|---|
| Native CRC32 | 1% | 99.9% | 1998 | 0.1% | 2 |
| | 0.5% | 99.8% | 1996 | 0.2% | 4 |
| CRC32_8_8 | 1% | 99.95% | 1999 | 0.05% | 1 |
| | 0.5% | 99.8% | 1996 | 0.2% | 4 |
| CRC64_16_4 | 1% | 100% | 2000 | 0 | 0 |
| | 0.5% | 99.95% | 1999 | 0.05% | 1 |

According to different chunk sizes in theory, CRC64_16_4 was faster than SunNative CRC32 and CRC32_8_8 vary between 1.2 and 8 times. But in practice, it will be affected by the network bandwidth or hardware. The performance of SunNative CRC32, CRC32_8_8 and CRC64_16_4 was compared. Experiments used JDK1.7, 64 bit, 3.10GHz machines for testing. The experimental results are shown in Figure 5.

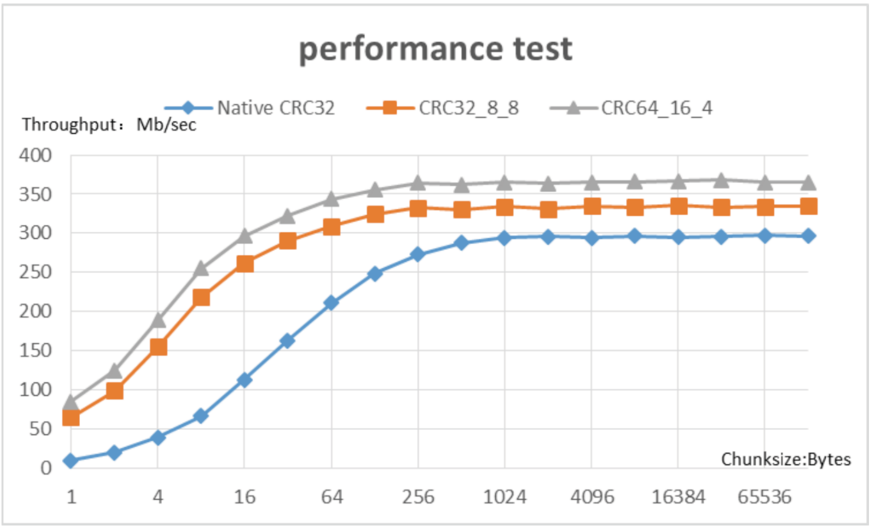

**Figure 5.** Comparison tests of three algorithms.

From the above, we can see that CRC64_16_4 has a significant effect on improving the throughput of the system. The difference between SunNative CRC32 and CRC64_16_4 was obvious, which means that SunNative wasted time in switching JVM and just verified 8 bits each time. The rate of CRC32_8_8 was limited because it verifies 64 bits each time but checking 8 tables, and there was room for improvement in the terms of the quality of code. CRC64_16_4 was optimized in two aspects, so that the experimental data show some advantages. But when chunksize reaches a certain number, the throughput will be affected by bandwidth, and it will not increase significantly.

The objects of our experiment were SunNative CRC32 of HDFS, CRC32_8_8 checking eight single-byte tables and optimized CRC64_16_4 checking four double-byte tables. Because the experimental data is greatly influenced by the hardware conditions, experimental data of proposed method fails to be compared directly with the experimental data in other literatures. But through the above experimental results, we can see the advantages of the proposed method in the theory and practical application.

## Conclusion

This paper established CRC64 deformational optimization to improve verification reliability and provided a new method for checking table to accelerate the speed of the algorithm via researching characteristics of traditional algorithm CRC. At the same time this paper improves the quality of code by researching the defects of verification mechanism in hadoop and combining the features of CRC algorithm, in order to enhance verification efficiency. Those optimizations could guarantee and improve the verification rate on the whole, and thus could achieve the goal of improving system utilization.

In this paper, the verification mechanism of the distributed file system HDFS is optimized, and the efficiency of resource utilization is improved. But the system was still not perfect, and the network bandwidth has a certain impact on the data transmission. Nowadays along with the massive amounts of data and data types of diversification, computing requirements of big data become increasingly high. Higher requirements for data storage and data integrity verification are also presented. Based on this research direction, the author will continue to research and improve the data integrity verification technology, in order to achieve more efficient, more secure purposes.

## Acknowledgement

This research is supported by the National Natural Science Foundation of China (NO.61373135, NO.61672299). The authors thank the sponsors for their support and the reviewers for helpful comments.

## References

1. Dongju Yang, Chuan Ren. VCSS: An Integration Framework for Open Cloud Storage Services [J]. Services (SERVICES), 2014 IEEE World Congress on. June 27 2014-July 2 2014, Pages: 155-160
2. Machado G.S., Bocek T., Ammann M, Stiller B. A Cloud Storage overlay to aggregate heterogeneous Cloud services [J]. Local Computer Networks (LCN), 2013 IEEE 38th Conference on. 21-24 Oct 2013, Pages: 597-605
3. Pawar C.S., Patil P.R., Chaudhari S.V. Providing security and integrity for dta stored in cloud storage [J]. Information Communication and Embedded Systems (ICICES), 2014 International Conference on. 27-28 Feb 2014, Pages: 1-5
4. Arockiam L., Monikandan S. Efficient cloud storage confidentiality to ensure data security [J]. Computer Communication and Informatics (ICCCI), 2014 International Conference on. 3-5 Jan 2014, Pages: 1-5
5. Kavuri S.K.S.V.A., Kancherla G.R., Bobba B.R. Data authentication and integrity verification techniques for trustd/untrusted cloud servers [J]. Advanced in Computing Communications and Informatics (ICACCI), 2014 International Conference on. 24-27 Sept 2014, Pages: 2590-2596
6. Data verification_Baidu Encyclopedia [Z]. 2013
7. Chang Liu, Chi Yang, et al. External integrity verification for outsourced big data in cloud and IoT : A big picture [J], Future Generation Computer System, 2015, 49, Pages : 58-67.
8. White T. Hadoop: The definitive guide [M]. O, Reilly Media, 2012
9. Shvachko K, Kuang H, Radia S, et al. The hadoop distributed file system[C]. In: 2010. Pages: 1-10
10. Zhaoguang Peng, Yu Lu, Alice Miller, et al. Formal Specification and Quantitative Analysis of a Constellation of Navigation Satellites[J], Quality and Reliability Engineering International, 2016, 32(2), Pages : 345-361.
11. Yingjie Xia, Fubiao Xia, Xuejiao Liu, et al. An Improved Privacy Preserving Construction for Data Integrity Verification in Cloud Storage [J], KSII Transactions on Internet and Information Systems, 2014, 8(10), Pages : 3607-3623
12. Yu Lu, Zhaoguang Peng, Alice A. Miller, et al. How reliable is satellite navigation for aviation ?Checking availability properties with probabilistic verification [J]. Reliability Engineering & System Safety, 2015, 144, Pages : 95-116.
13. Khatri T.S., Jethava G.B. Improving dynamic data integrity verification in cloud computing [J]. Computing Communication and Networking Technologies (ICCCNT), 2013 Fourth International Conference on. 4-6 July 2013, Pages: 1-6
14. Danyang Cao, Bingru Yang. Design and implementation for MD5-based data integrity checking system [J]. Information Management and Engineering (ICIME), 2010 the 2nd IEEE International Conference on. 16-18 April 2010, Pages: 608-611
15. Matusiewicz K, Pieprzyk J. Finding good differential patterns for attacks on SHA-1[J]. Coding and Cryptography. 2006, Pages: 164-177
16. Peterson W.W., Brown D.T. Cyclic Codes for Error Detection [J]. Proceedings of the IRE. Jan 1961, Pages: 228-235
17. Kounavis M.E., Berry F.L. A systematic approach to building high performance software-based CRC generators[C]. In: 2005, Pages: 855-862
18. Rui Jing. Research and Improvement of Data Check Strategy in Distributed File System [D]. Jilin: Jilin University, 2013.