A Dynamic Hashing Method for Storage Optimization of Spacecraft Verification Database

Hongjing Cheng(✉) and Yanfang Fan

Beijing Institute of Spacecraft System Engineering, Beijing 100094, China

Abstract. In order to facilitate the repetition, deduction and playback of the problems in the test process, the corresponding requirements for the retention of test data are put forward in the current test process of spacecraft data management software. In order to deal with the operation based on data itself and the higher level of extraction and analysis based on data, the use of database to store data has become the highest performance and the most convenient way. Due to the large amount of test data, using a single table storage method can cause the time of data query to increase rapidly until it is unacceptable. However, several commonly used table partitioning methods also have their own defects which make it difficult to apply in this scenario. In this paper, a dynamic hashing method (DSH algorithm) is proposed to partition the data according to the characteristics of generating and using the test data of spacecraft data management. The data is hashed through partitioning process feedback to ensure the bias balance of queue hashing. The dynamic partitioning method ensures that the partition fragmentation is controllable and the number of partitions is reduced. It improves the speed of data storage and query, and improves the efficiency of storage space. So that in the same hardware platform, running a larger and more complex tube software testing work provides a stable and reliable solution.

Keywords: Data management test · Database partition · Dynamic hash

1 The Introduction

As the brain of the spacecraft, the spacecraft data management subsystem is responsible for the data organization, processing, transfer and transmission within and between the spacecraft and the satellite-ground link. The quality and life of the spacecraft are closely related to the stability and reliability of the tube subsystem. Because of its importance and complexity, the development and testing of the subsystem software are paid more attention to. Spacecraft data management software testing database provides basic data for software testing application tools to ensure the accuracy of data time sequence, content and correlation.

The amount of data produced in the testing process of several tubes of software is very large, up to tens of megabytes per second according to different models. The insertion and search speed of these data are greatly affected by the huge amount of data. In the process of using data, the application software directly faces the user, so the

Q. Guo et al. (Eds.): WiSATS 2021, LNICST 410, pp. 166–176, 2022.
https://doi.org/10.1007/978-3-030-93398-2_16

speed of reading, organizing and displaying data is required to be high. In order to improve the reading speed of data, it is necessary to set up an effective index and partition the data. The index design of the database will not be further explored in this article, but the index itself should be considered in the establishment of the index on the impact of insert speed.

It can be seen from the test data that adding a reasonable number of indexes can obviously improve the speed of data retrieval (see Fig. 1). However, if the index is not added properly, the data insertion speed may be slowed down (see Fig. 2). When the index design is complete, the partition is created with the primary index. The significance of data partitioning lies in that all the data that need to be written is stored in the specified storage area according to certain rules, which makes different data be written into different disk pre-divided space in the physical storage of the database, which can further improve the speed of data reading.

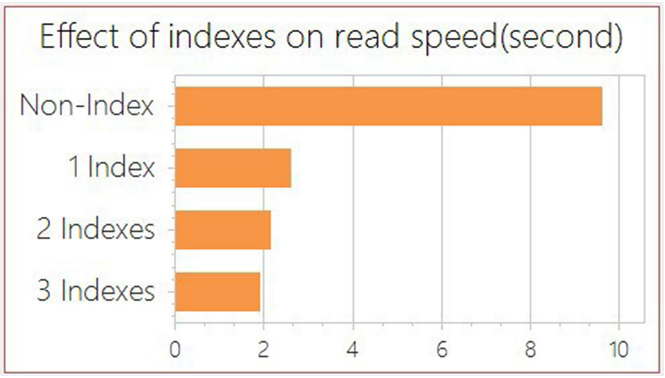

Fig. 1. Effect of indexes on read speed (mysql 5.7, 1000 items were retrieved from the table with 1,200,000 records)

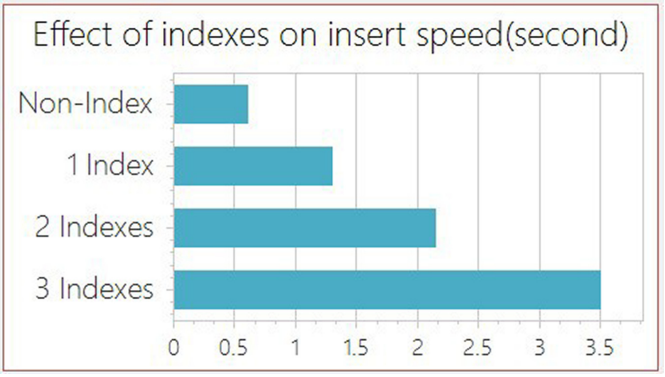

Fig. 2. Effect of indexes on insert speed (mysql 5.7, 1000 items were inserted to the table with 1,200,000 records)

There are obvious criteria for partition creation: the more partitions there are, the more expensive it is to manage and maintain. The greater the difference in the amount of data stored within a partition, the less meaningful the data partition is. Therefore, in the design of the test database of the spacecraft data management software, it is necessary to choose an algorithm that can reduce the partition fragmentation as far as possible and divide the storage evenly to complete this task.

2 Selection of Common Algorithms

Hash Algorithm, can map binary data of any length to a shorter binary string, and it is difficult for different data to map to the same Hash value. It can be understood as a spatial mapping function, mapping from a large value space to a very small value space. This mapping method from large to small ensures that the mapping process is not reversible. Therefore, hash algorithm is often used in encryption algorithm.

Hashing algorithm is a series of algorithm requirements, including the basic requirements of input sensitivity, irreversible, conflict avoidance. Any algorithm that satisfies these requirements can be called a hash algorithm. The existing MD5, SHA1, SHA2, Siphash etc. on the market can all meet the demand of data hashing.

For example, MD5 code can convert any data into 256-bit source code value. Assuming that 50,000 e-books need to be placed into 100 classified bookshelves according to their contents, MD5 code can be calculated for the binary files of each book, and the value obtained is the bookshelf number by dividing the result by 100 to get the residual value.

In the design of spacecraft data management test database, the first level is divided according to the parameter code, and the input source is used as the partition basis. In different models, the number of parameters is on the order of thousands to tens of thousands, and it is not possible to create separate partitions for each parameter. Therefore, after the use of the hashing algorithm, it is still necessary to take the remainder again to restrict all data classification within a certain range.

Among the current hash algorithms, the algorithms that can convert arbitrary data into fixed-length source code are MD5, SHA and other algorithms. These algorithms can fulfill the requirements of hashing algorithm well, and meet the requirements of input sensitivity, irreversibility and conflict avoidance. However, in database design, the need for conflict avoidance is not strong, and the requirement for partition balance is high. After the traditional hash algorithm is used to merge the remainder, under the condition that the input is completely random, the partition equilibrium of itself can be almost evenly distributed. However, the spacecraft test database storage does not simply depend on the number of categories stored, but also depends on the amount of data in the category. At this point, you can only attempt to redistribute the data by increasing the number of partitions.

3 Dynamic Hash DSH Algorithm

Spacecrafts test data has such a feature that some parameters are updated very frequently, while some parameters are updated at a low speed, and the speed difference may be more than one hundred times. If you do not weight parameter data volumes and hash only by parameter names, you may end up with several large data volumes of parameters being partitioned in the same partition.

Therefore, the first step of DSH algorithm is to ensure the uniformity of the hash algorithm under the condition of permitting collisions.

3.1 The First Round of Uniform Hashing of DSH Algorithm

Create a hash map space in memory, the length of which is determined by the computer based on the hardware environment, and can be assumed to be 100 and the number of mapped Spaces is 5.

After receiving the new parameter, judge whether there is already a mapping. If there is, enter the original mapping, and add one to the mapping space data count. If not, select the queue with the least data count in the mapped space (see Fig. 3).

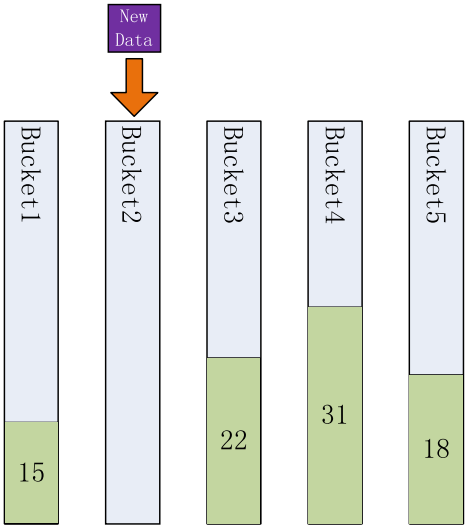

Fig. 3. Insert new data into a Bucket with least data

When all mapped Spaces are non-zero, such as a minimum of 15, the data is written and the memory space is cleared (see Fig. 4).

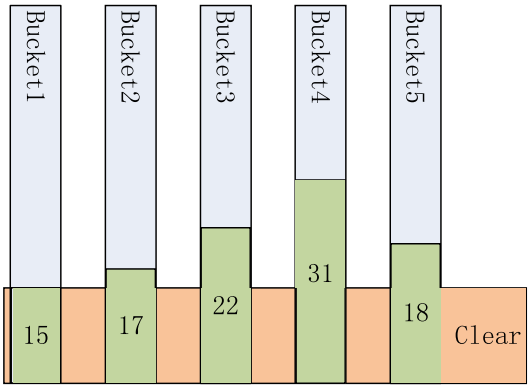

Fig. 4. Remove Buckets' fixed data when all Buckets have a certain amount of data.

In some special cases, the single-mapped space becomes full before the overall memory cleanup threshold is reached due to the explosion of data volume in the single-mapped space. At this time, the memory space will be cleaned up to the maximum value of the current whole space, and the cleaned data will be written into the database (see Fig. 5).

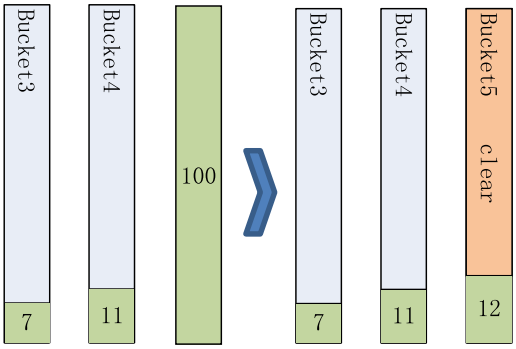

Fig. 5. Clear the special bucket and keep the bucket with most data count remain.

With the use of DSH, the stacked data moves from a single partition to each partition, ensuring partition balance.

On the basis of evenly divided data, the storage mode still has the possibility of optimization. Due to the uncertainty of the number of partition Settings, individual parameters with small data volume may occupy individual partitions.

Using DSH algorithm, the statistical data in the process of data generation is used to merge and store the data with small amount and low frequency, while the data with large amount is stored in a proportional exclusive partition. The number of partitions is small and the uniformity of data partition can be guaranteed.

3.2 DSH Algorithm Sub-round Dynamic Partitioning

The system calculates the partitioning results of the existing data once according to the default partition statistical unit time (90s). A total of L rounds of statistics were conducted. until

The system calculates the partitioning results of the existing data once according to the default partition statistical unit time (90s). A total of L rounds of statistics were conducted. until

$$\sum_{r=0} f(r) - f(l) < k \tag{1}$$

This indicates that statistics have converged. On this basis, partition calculation is carried out. Let the partition number of the current test model design be s, where I, j, k ∈ S. If any I, j, k exists, the following relation can be satisfied:

$$n_i + n_j < n_k \tag{2}$$

Then partition I and partition j can be combined into a partition, creating a mapping relationship such that all

$$x \in n_j \rightarrow x \in n_i \tag{3}$$

And $s = s - 1$; Delete n_j partition and retraverse s so that it does not exist

$$n_i + n_j < n_k \tag{4}$$

At this point, the number of partitions is compressed to a minimum, and the partition uniformity is guaranteed. According to the measured data, when the initial S is set as m, the value of S after multiple cycles is at

$$\overline{4} \leq s \leq \overline{2} \tag{5}$$

Therefore, in the initial setting of the algorithm, when the user's expected partition value is set to s, the algorithm will use 2*s partition number to start iteration in the first round of partitioning, and the final partition result may be greater than or less than s. Specific data values are affected by the largest number of parameter values.

After the real-time test of more than 17,000 parameters with a total of 400MB data, the partitioning results of DSH algorithm and MD5 residual algorithm are shown in the figure below:

It can be seen that when the number of partitions is 32 and 64, the result of partitioning is that some partitions are too large due to insufficient hashing (see Fig. 6 and Fig. 7). When the number of partitions increases to 128, the largest data block is partitioned, which also leads to a large number of empty partitions, wasting system resources and increasing management costs (see Fig. 8).

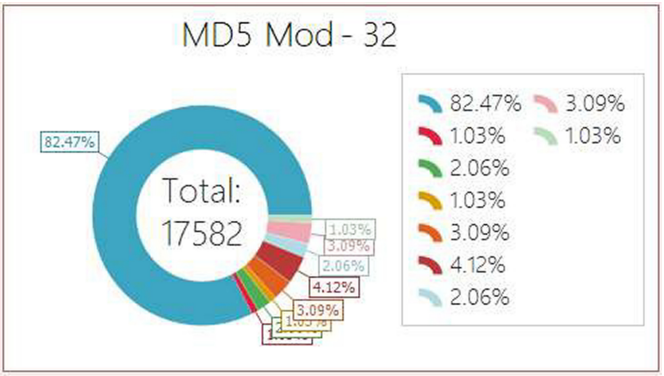

Fig. 6. 32 partitions for MD5 mod algorithm. The biggest partition own 82% data size in all 17582 records.

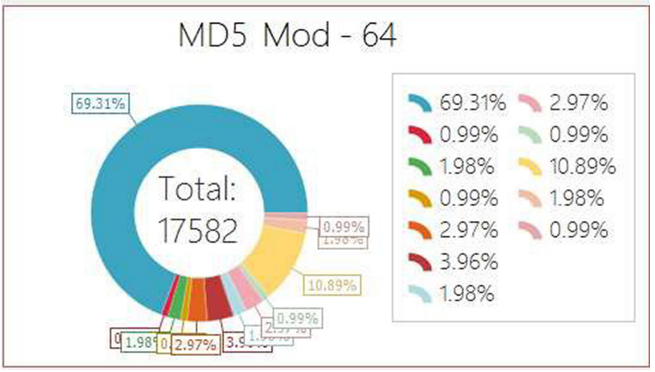

Fig. 7. 64 partitions for MD5 mod algorithm. The biggest partition own 69% data size in all 17582 records.

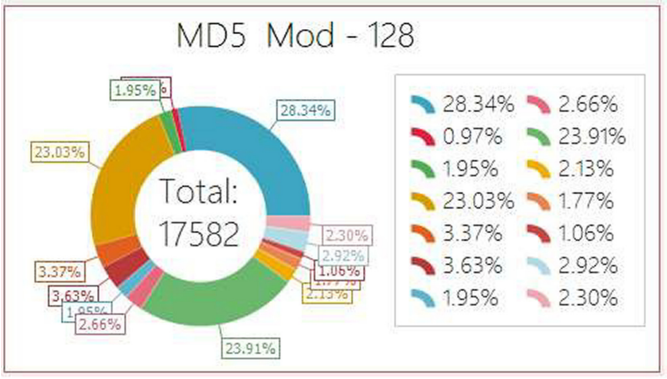

Fig. 8. 128 partitions for MD5 mod algorithm. The biggest partition own 28% data size in all 17582 records. And many partitions are not assigned data.

After the use of DSH algorithm, the data partition has a good result:

When DSH algorithm is used, it can be seen that when the number of partitions is 4, 9 and 14, good partition results can be obtained. The appropriate number of partitions can be specified according to the number of parameters of the specific model without considering the influence of the selection of the number of partitions on the final partition effect (see Fig. 9, Fig. 10 and Fig. 11).

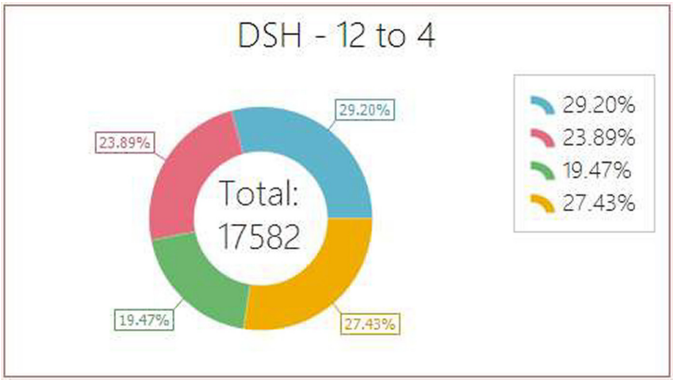

Fig. 9. The initial designation of 12 partitions is reduced to 4 partitions after calculation by DSH algorithm, and the largest partition is 10% larger than the smallest partition.

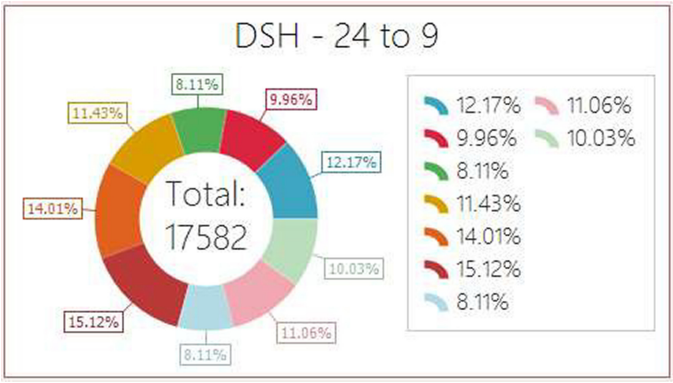

Fig. 10. The initial designation of 24 partitions is reduced to 9 partitions after calculation by DSH algorithm, and the largest partition is 7% larger than the smallest partition.

After the use of DSH algorithm, additional computing resources are needed due to the addition of dynamic partition algorithm, so the time spent in the data insertion process is recorded. It can be seen:

Inserts are fastest when the data is not partitioned or indexed at all, because there is no sorting of the database data.

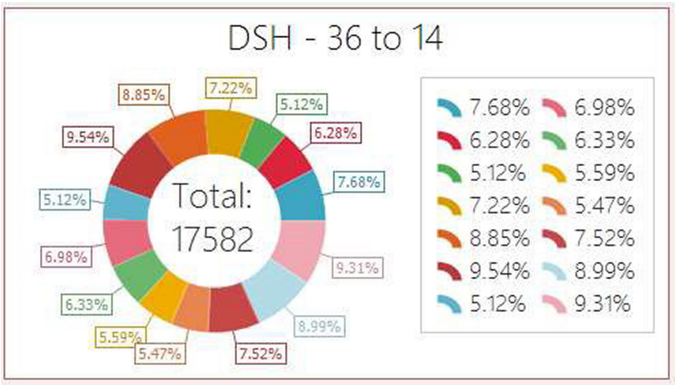

Fig. 11. The initial designation of 36 partitions is reduced to 14 partitions after calculation by DSH algorithm, and the largest partition is 7% larger than the smallest partition.

When DSH algorithm and other hash algorithms are used, the difference in insertion speed is negligible, indicating that the computing power consumed by the algorithm itself is not a decisive factor in the database update scenario and does not affect the insertion efficiency (see Fig. 12).

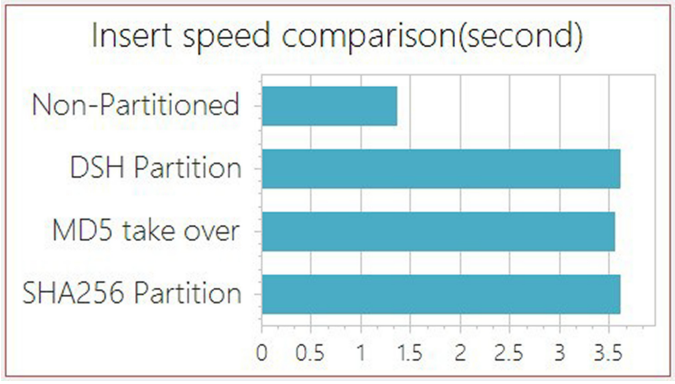

Fig. 12. In the algorithm with partition, the insertion speed difference is less than 1%, which is negligible.

In order to verify the data stored by DSH algorithm, read the 15-h test data after querying the data, and read the 1000 records before the specified parameter code:

It can be seen that the DSH algorithm is close to the MD5 residual algorithm with better partition in time, but it still has an advantage of about 15% (see Fig. 14). The speed of DSH algorithm is about one times higher than that of MD5 redundancy algorithm with poor partition. Data without adding any index partitions consumes a lot of time resources (see Fig. 13).

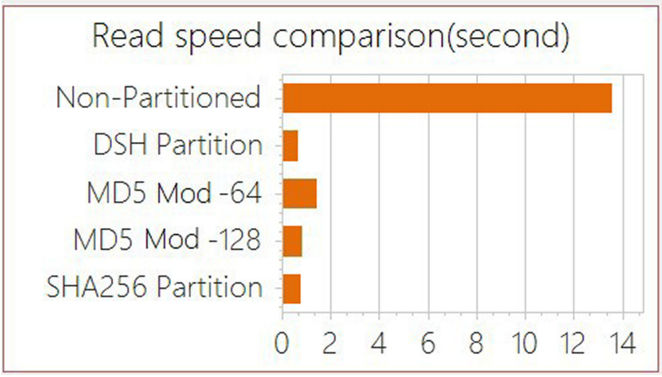

Fig. 13. Data reads are unacceptably slow without any partitioning.

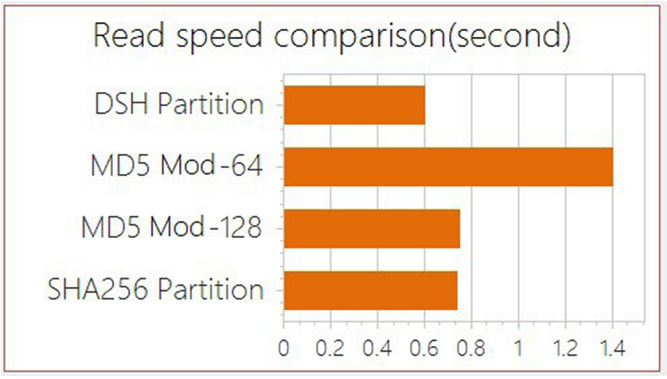

Fig. 14. DSH reads 15% faster than the already over-partitioned MD5 mod algorithm, and reduces a lot of administrative overhead.

4 Conclusion

In this paper, a dynamic hashing method is proposed to partition the test data before warehousing for spacecraft data management software, which has many test data, high requirement of storage and reading speed, and poor performance of the current storage algorithm. DSH algorithm can dynamically count the data of the same parameters in the stored procedure and make uniform hashing and partition adjustment based on it, which can effectively improve the rationality of data partition, reduce the cost of system management and avoid partition failure. The partition algorithm studied in this paper can not only be used in the storage and processing of spacecraft test data, but also be extended to all data processing scenarios requiring uniform data partition. In the future, this algorithm should be verified and promoted in more environments to ensure its effectiveness.

References

1. Aumasson, J.-P., Bernstein, D.J.: SipHash: a fast short-input PRF (2012)
2. Wang, Z., Zhang, H., Qin, Z., Meng, Q.: Fast attack algorithm on the MD5 hash function (2006)
3. Liang, J., Lai, X.: Improved collision attack on hash function MD5 (2007)
4. Teh, J.S., Samsudin, A., Akhavan, A.: Parallel chaotic hash function based on the shuffle-exchange network. Nonlinear Dynamics (2015)
5. Kim, D.-C., Hong, D., Lee, J.-K., Kim, W.-H., Kwon, D.: LSH: a new fast secure hash function family. In: Lee, J., Kim, J. (eds.) ICISC 2014. LNCS, vol. 8949, pp. 286–313. Springer, Cham (2015). https://doi.org/10.1007/978-3-319-15943-0_18