Tacked Link List - An Improved Linked List for Advance Resource Reservation

Li-bing Wu[1,2], Jing Fan[1], Lei Nie[1,2], and Bing-yi Liu[1]

[1] School of Computer, Wuhan University, Wuhan 430072, China
[2] State Key Laboratory of Software Engineering, Wuhan University, Wuhan, China
wu@whu.edu.cn

Abstract. Since advance resource reservation is a widely used mechanism in distributed systems and high-performance networks, the optimization of its performance has been greatly concerned. And the performance of the data structure plays an important role for the overall performance of the advance resource reservation. In this paper, the authors figured out the disadvantages in the existing data structures used in advance resource reservation and proposed an improved data structure called 'tacked list', to overcome these disadvantages. To demonstrate the performance of this improved data structure, the authors made mathematical analysis to explore the tradeoff between performance and cost. At last, the result of the simulation experiments show that the improved data structure can highly improve the performance of the whole reservation system at the starting up phase and still have a relatively good performance at the stable phase.

1 Model

Admittedly, the linked list could solve the problem that the query operation needs to traverse the data too many times in advance resource reservation. Based on the linked list, the indexed link list could improve the locating operation by introduce the index array and it has better performance than other data structures in advance resource reservation which has already been proved in previous article [3].

However, the index linked list still has some disadvantages. At the start phase, the index linked list needs to traverse backwards on each index to find the position of a certain value if the index has no value before. To solve this problem, we try to eliminate traversing from previous index point by adding a dummy node in the tacked link. The topology of these two structures is shown in figure 1.

2 Performance Analysis

The tacked list utilizes the index array to locate the target node, so the overall performance of tacked list is closely related to the size of the index array. Although a large index size can significantly promote the locating speed, it also increases the difficulty of maintenance at the same time. We can figure out that the cost of each query operation match with formula 1.

$$Cost = C + D_{node} + \frac{2TR_{m1}}{t} + \frac{2d\left(R_{m2} + W_{m}\right)}{t} + \frac{tC_{release}}{2T} \quad (1)$$

R_{m1} means the average time cost on traversal operation of each index node for locating the start point. R_{m2} and W_m mean the cost on traversing and modifying the value node. The $Cost_{release}$ means the time of the memory release operation. T means the interval between two index and t means the interval between two requests. d means the average duration of the requests.

We can minimum the cost of each query operation by set an optimal interval between two index nodes.

3 Simulation

The following experiments runs on the Intel(R) Pentium(R) CPU G2020 @ 2.90GHz dual-core CPU with 4GB memory. And the Operation System is Windows 7 Service Pack 1 64bit. The test programs is written by C++ and compiled with MinGW in eclipse. Since the main point of this paper is concentrated on the performance of the data structure, all of the reservation requests are pre-generate requests. And the reservation system will read them from the memory immediately to ignore the network influence. The other details will be described before each experiment.

The difference between tacked list and indexed list are analyzed in this section. the experiment contain the start-up performance test and the stable performance test. From the experiment, we notice that the mainly improvement of the tacked list is the performance during the start-up phase.

In the start-up performance test, we will record the processing time of the first 20 requests for these two data structures with different index size. From the analysis in the previous part, the processing time has no correlation with the time limited by reservation system, so the maximum of the reservation time is set to a relatively large value, 1048576 (220). In this way, we can make the parameter T to change precisely with the size of the index array. Furthermore, in order to get an accurate processing time, the system generated 20000 random requests at first and these requests were divided into 1000 groups. And the system also initialized 1000 instances of each data structure in advance. Then the system inserted the requests of each group into each instance of the data structure. After all the 1000 groups have been processed, the system will record the total processing time. The interval between two requests and the duration of every request are 4 time units. The results are shown in figure 2.

Since there is no backward search operation for the index nodes in the tacked list, the processing time is relatively stable along with index size growth. However in the indexed list, the processing time in the start-up phase increases linearly with the size growth of the index array.

In the stable performance test, the system generated 100 thousands requests at the beginning. To eliminate the influence of the start-up phase, the data structure will be fully filled before recording the processing time. To fill the data structure, the system will read a part of these requests until the time of the reservation system pass the reservation limit. After the data structure is fully filled, the system will process all of the 100 thousands requests again and record the processing time. The interval between two requests and the duration of them are the same as in the start-up performance test. The result is shown in figure 3.

Since the operations of these two data structures are highly similar in the stable phase, their performance with different index size is also highly similar. These are reflected in figure 3 and their average processing time are 297.3ms for tacked list and 297.0ms for indexed list.

After the comparison with the indexed list, the final test is the system capacity test. In this test we try to find the system capacity of the tacked list. In order to make the results more intuitive, there will be some comparison tests among indexed list, tacked list, time slot array and RRB+ tree. There are else some other data structures using for advance resource reservation, but they has already been researched comparatively [3].

In this test, the system will generate 1 million requests in advance. The duration range of these requests is from 30 seconds to 1800 seconds. The reservation time limit is 432000 seconds (5 days) after the received time. To make every request be accepted, the quantity of resource in every request is set to 1 unit and the max resource quantity is set to UINT_MAX. And the size of the index array is set to a relatively high value, 108000, to fit the high-traffic situation. During the test, the system will run two threads at the same time, one for reservation and one for recording.

From figure 4, we can find that the performance of the time slot array was very stable, because the only influential factor is the average duration of all requests. On the other hand, the performances of the other three data structures gradually decreased over time. This is because these data structures have very simple structures after the initialization. But with the requests filled in, their structures will become complex and the performances decrease. This will continue until it achieves the balance point that the old nodes become failed at the same speed of new nodes inserted.

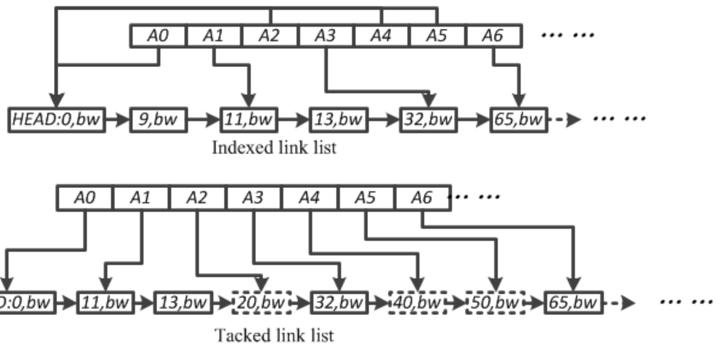

Fig. 1. Topology of indexed list and tacked list

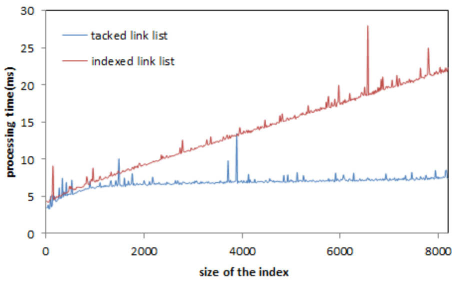

Fig. 2. Start-up performance

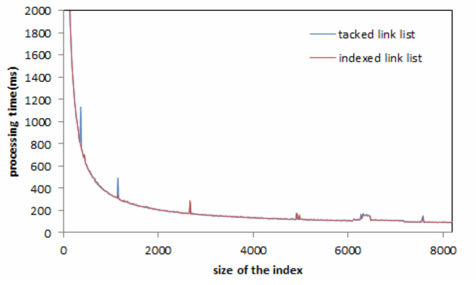

Fig. 3. Stable performance

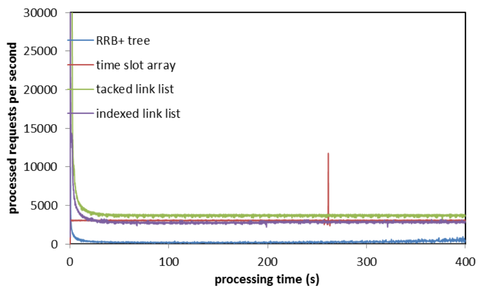

Fig. 4. System capacity test

4 Conclusion

After all of these analyses and experiment, we find that the tacked list has a much better performance at the start-up phase than indexed list. And in the stable phase the performance of tacked list is similar to the indexed list.

Acknowledgments. This work is supported by National Science Foundation of China (No. 61170017, 61272112), Science & Technology Plan of Wuhan city (No. 2013010501010146.) and the Fundamental Research Funds for the Central University (No. 2014211020202.)

References

1. Burchard, L.-O.: Analysis of data structures for admission control of advance reservation requests. IEEE Transactions on Knowledge and Data Engineering 17(3), 413–424 (2005)
2. Wu, L., Yu, T., He, Y., Li, F.: Index linked list suited for resource reservation. Journal of Wut (Information & Management Engineering) 33(6), 904–908 (2011) (in Chinese)
3. Yu, T.: Research of Data Structures and Algorithms on the Reservation of Grid Resource. M.Sc. Thesis. Wuhan University, China (2012)