**Fast Thread-safe Priority Queue**

*Yong Wan, Hualong Zhu, Zhihao Zhang*

Abstract

This report indicates fast thread-safe priority queues using in Java. Although Java provides PriorityBlockingQueue which is a kind of priority queues for the implementation of thread-safe concurrency, it cannot meet the requirements of fast multithreaded processing. In this report, we achieve a comparison of LockFreePriorityQueue, Pipelined PriorityQueue, and PriorityBlocking -Queue. This report also expounds the basic knowledge used in these priority queues, implementation details, the methodology for implementation. The testing and evaluation of our implementation are illustrated. For future works, we will do more research for different priority queues which can do parallel computing and test their actual efficiency.

1. Introduction

Priority queues are playing an important role in algorithm design, which is provided by Java SDK. Meanwhile, priority queues are logically implemented using a heap structure (complete binary tree), which is physically implemented using a dynamic array. Priority queues use key-value pairs to implement priority operations since the priority is represented by the key. To insert or delete elements in a priority queue, it supports insert and delete operations, which are all based on the key (priority) of the element.

Java also provides a priority queue called PriorityBlockingQueue to implement the thread safe parallelization. PriorityBlockingQueue is an unbounded concurrent security priority queue based on the priority heap. However, when using PriorityBlockingQueue, other concurrent operations cannot make any progress while the access to the shared resource is blocked by the lock. PriorityBlockingQueue can lead to deadlock and priority inversion.

Consequently, we explored two alternative fast thread-safe priority queues: LockFree -PriorityQueue and Pipelined PriorityQueue.

Unlike PriorityBlockingQueue which uses a global lock to keep data safe when reading and writing, instead Pipelined PriorityQueue uses a double-level lock to implement a pipelined queue, in which only a part of data in the queue would be locked when r/w operation occurs.

In our project, we will analyze and implement both LockFreePriorityQueue and Pipelined PriorityQueue. After that, these three queues including PriorityBlockingQueue will be compared together on the performance.

1. Basic knowledge

In this section, the basic knowledge is introduced including binary heap, SkipList and lock, which is used in our project.

1. **Binary Heap**

The binary heap is a special kind of heap, which is a complete binary tree or an approximate complete binary tree. Furthermore, it satisfies the structure and sequence of the tree. The tree mechanism characteristic is the structure which the complete binary tree should have. The heap order is that the key values of the parent node always remain in a fixed order in relation to the key value of any child node. Also, the left subtree and right subtree of each node is a binary heap. It has two manifestations: the smallest heap (as shown in Figure 2.1) and the largest heap (as shown in Figure 2.2).

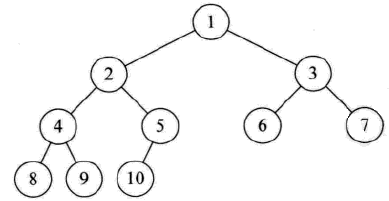
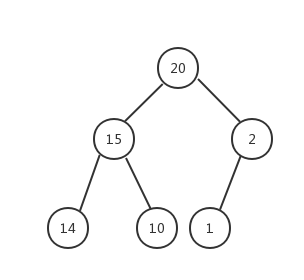
Figure 2.1 The smallest heap tree 

Figure 2.2 The largest heap tree

As shown in the figures, the smallest heap follows the principle that the parent node’s key value is always greater than or equal to the key value of any one of the child node(s). Also, the largest heap is the opposite.

1. **Skip-list**

Skip-list is a widely used data structure which supports fast searching. According to William’s works, a skip list saves elements in a hierarchical list in an orderly fashion[3].

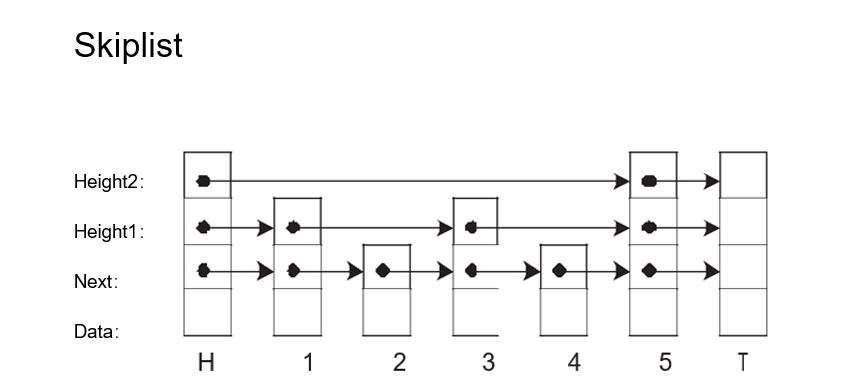
I. The Data Structure Of Skip-list

Skip list uses randomization and has a probabilistic time complexity of O(log N) where N is the maximum number of elements in the list[1]. The data structure based on a list which contains ordered data, then add shortcuts which randomly distributed for improving the whole search efficiency on this struct. In general, the level of shortcuts (the maximum height) of the data structure is log N. The highest priority node are located first after node head in the list.

II. An Example Of Skip-list

As shown in Figure 2.3, there are three levels including Height 2, Height 1 and Next. While H means Head and T means Tail, these two nodes are fixed in every skip list, other nodes are normal data nodes. the height 2 level only has Node H and 5 and T, and then the height 1 level has node H,1 then 3 then 5 and T. The lowest level “Next” goes through the all nodes.

For example, when we do a search for a node equals 4, At first we start from the Highest level which is Height 2, we compare the first node after H which is 5 and it is great than 4. so ,we go to the lower level which is Height1, then we still compare the first node after H which is 1, because 1 is less than 4, so we go to the next node three which is also less than 4, so we keep going to his next node 5, which is great than 4 again, so we should go back node three and then lower down the level which goes to level（next), then we get node 4.

Figure 2.3 The structure of a skip list

III. How SkipList works On Multi-Threads

Three of the standard atomic synchronization primitives, which names Test-And-Set (TAS), Fetch-And-Add (FAA) and Compare-And-Swap (CAS), is implemented on the current popular computer system. skiplist inserts a node or deletes a node by using these three atomic methods.

Especially, for Java, we should use AtomicMarkableReference. An AtomicMarkable -Reference maintains an object reference along with a mark bit, that can be updated atomically. The main methods of AtomicMarkableReference include compareAndSet() ,set(), isMarked() etc.

1. **Pipelining**

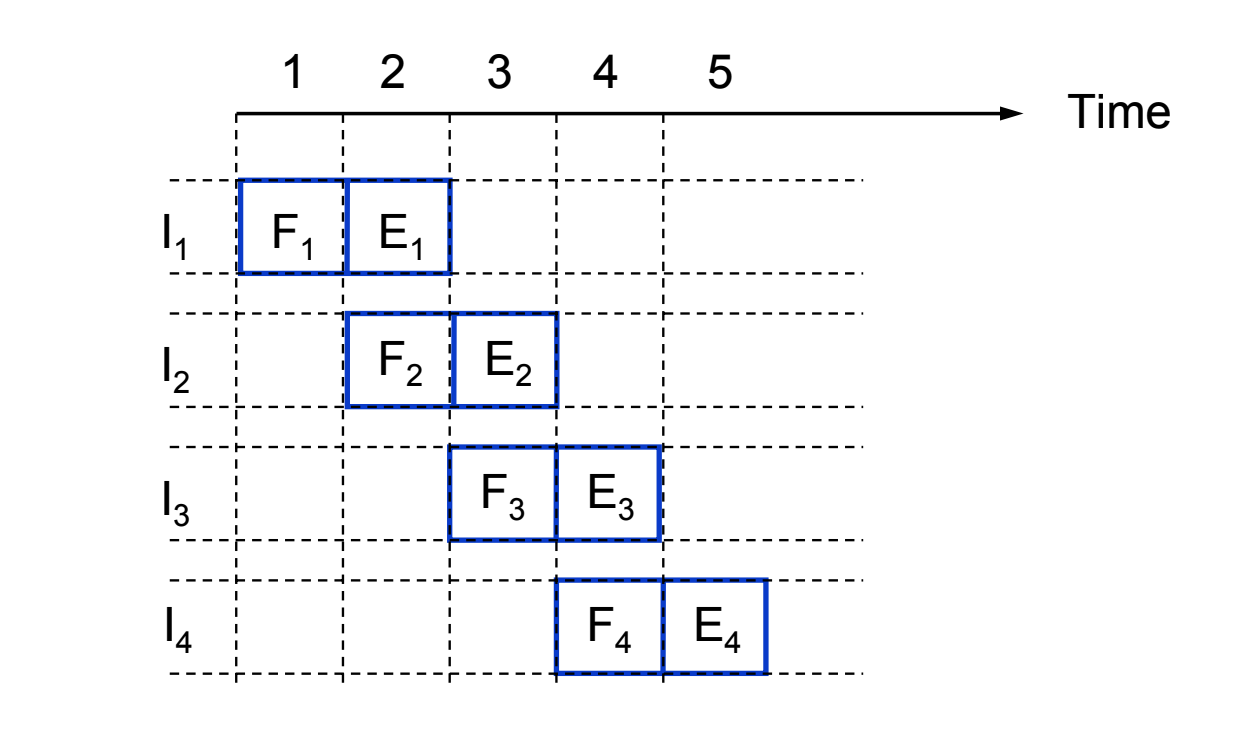


Figure 2.4 The basic idea of instruction pipelining

Pipelining is a particularly efficient way of organizing concurrent activity in a computer system.

In the field of computing, a pipeline, which is also known as data pipeline, is a set of data processing elements connected in series. The elements of a pipeline are often executed in parallel or in time-sliced fashion.

1. Implementation
2. Methodology
3. **Insert Node By LockfreePriorityQueue**

The main algorithm steps, for inserting a new node at a rodam position in our skip list has two steps:

(I) From the highest level node of heat, comparing the priority, if the number of priority less than next node, lowing the level, compare again until the lowest level to find a right position to add a new node.

(II) From the lowest level of to-be-previous nodes of the new node,update their next pointers atomically ,also include the new node’s next pointer.

1. **Delete Node By LockfreePriorityQueue**

The main steps of the algorithm for deleting a node at a random position has four steps:

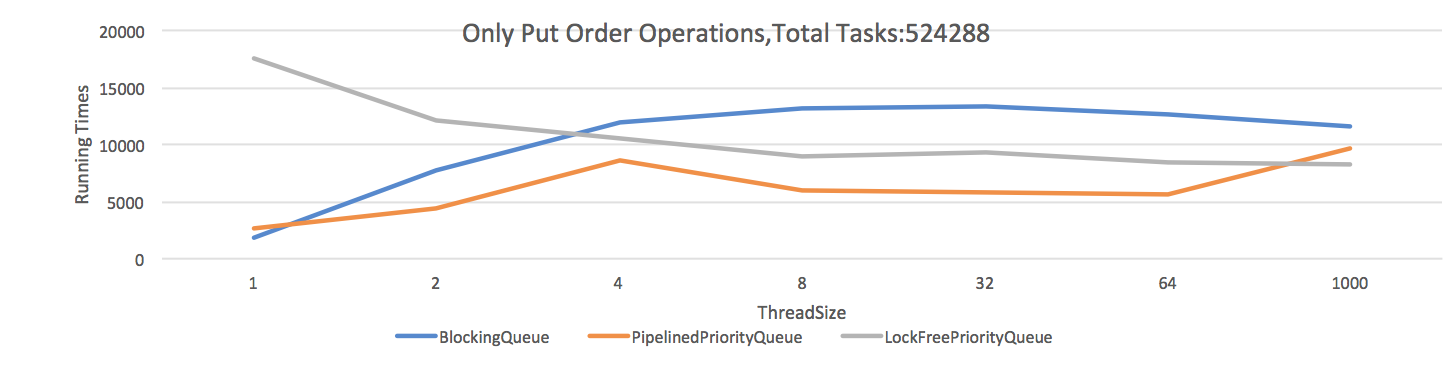
The first is to find the appropriate position for deleting, the method likes the first step of inserting a node.

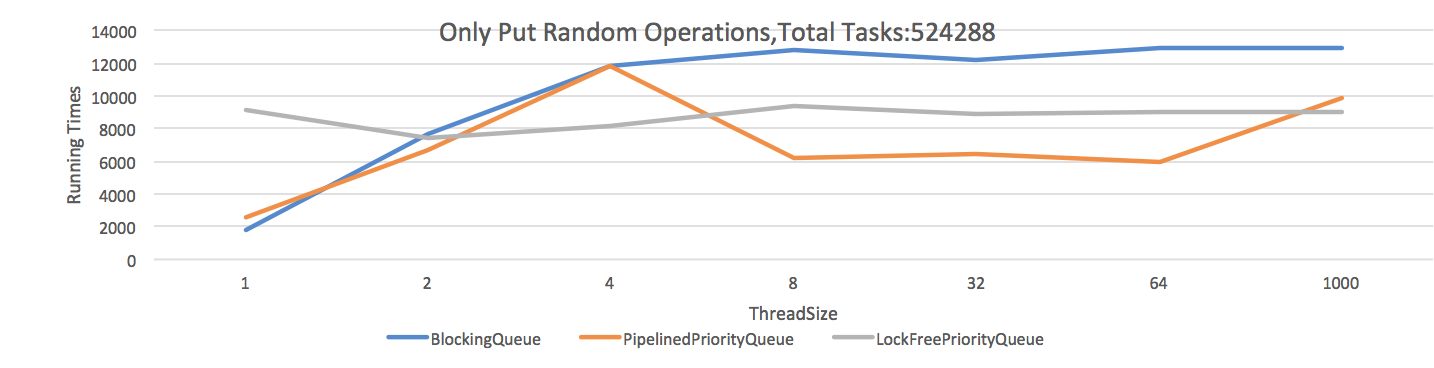
Then the to-be-deleted node should be written a deletion indication to avoid being used by another thread.

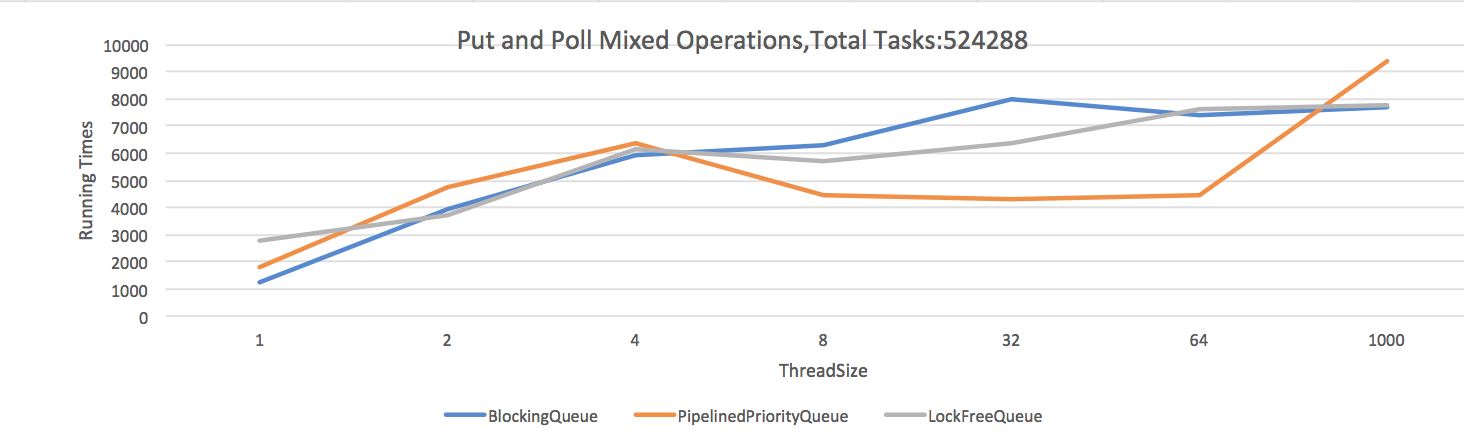
After that, the deletion marks also should be set on the next pointers of the to-be-deleted node, from the lowest level to the highest level which the node hold.

The last step has two part. The first part, in the lowest level, atomically update the next pointers of the previous nodes of the to-be-deleted node; after then, unlike the previous steps, it should update atomically the next pointers from the most topmost level of its parent to the second lowest level.

1. Testing and Evaluation







1. Future work

According to what we discussed in previous sections,

1. Conclusion

To sum up,

PriorityBlockingQueue uses a lock to blocking related operations like put and poll.

Pipelined PriorityQueue only locks two levels including the level of current node and the next level.

LockfreePriorityQueue doesn't use a lock, it updates data by atomic methods.

References

[1] Sundell, H., & Tsigas, P. (2005). Fast and lock-free concurrent priority queues for multi-thread systems. Journal of Parallel and Distributed Computing, 65(5), 609-627.

[2] Bhagwan, R., & Lin, B. (2000). Fast and scalable priority queue architecture for high-speed network switches. In INFOCOM 2000. Nineteenth Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings. IEEE (Vol. 2, pp. 538-547). IEEE.

[3] Pugh W. (1989) Skip lists: A probabilistic alternative to balanced trees. In: Dehne F., Sack J.R., Santoro N. (eds) Algorithms and Data Structures. WADS 1989. Lecture Notes in Computer Science, vol 382. Springer, Berlin, Heidelberg

Contribution Table