A Brief Review of "Hoard: A Scalable Memory Allocator for Multithreaded Applications"

Yi Luo

1 Summary

Traditional serial memory allocators do not scale well for multi-threaded programs because they typically use a single lock to protect the heap, which serializes memory operations and also introduces contention. Besides, most existing concurrent allocators have multiple-heap allocation but cannot solve a number of terrible problems such as false sharing and blowup. All of these disadvantages make such allocators the bottleneck of most current parallel and multi-threaded applications. This paper proposes a scalable memory allocator, named Hoard, to overcome deficiencies of these existing allocators.
Hoard has a novel heap organization which combines one global heap and per-processor heaps with a reasonable discipline. It mainly solves two problems which are false sharing and blowup. [1]

2 Merits

The strategy of Hoard to overcome false sharing is a very good one. The basic fact is that false sharing can never be completely avoided, therefore, a good solution is to reduce the possibility. In parallel applications, false sharing is a terrible problem which may force programs to do a large amount of unnecessary work. Allocators can cause false sharing of heap objects by dividing cache lines into a number of small objects that different processors write.
Hoard uses a multiple-heap structure and superblocks to avoid false sharing. When multiple threads request memory at the same time, their requests can be satisfied from different superblocks because a superblock is always owned by exactly one heap. But Hoard does not completely avoid false sharing since it may move superblocks between two heaps so that they may still share cache lines. This paper claims that superblock transfer is an infrequent event so after all Hoard indeed reduces the opportunity for false sharing.

3 Drawbacks

This paper benchmarks a number of popular allocators and reveals that Hoard provides a fast, highly scalable allocator to improve the performance of multi-threaded applications. Its experiments are detailed and convincing and their results also display good solutions of Hoard to overcome problems such as false sharing and blowup. However, there is no discussion on the latency of different allocators which I believe should be a very important evaluation aspect.

4 Personal Thoughts

In the description of their hashing mechanism which maps thread id's to per-processor heaps, authors actually use extra heaps to handle collisions in some platforms/operating systems. I am curious about whether there is a way to achieve the same goal without the using of extra heaps.

References

[1] Robert D. Blumofe Paul R. Wilson Emery D. Berger, Kathryn S. McKinley. Hoard: A Scalable Memory Allocator for Multithreaded Applications. *ASPLOS*, 2000.