Concurrent Distributed Dynamic Sized Worker Queue

Yunjia Wang (yunjiaw@andrew.cmu.edu)
Jiaan Dai (jiaand@andrew.cmu.edu)

Nov 18, 2019

URL: `https://flowersh0026.github.io/618-final/`

Schedule and Current Progress

Week	Task	Progress
11/04 – 11/10	Understand the data structure of lock-free queue using CAS	Done
	Understand the data structure of lock-free queue using RTM	Done
11/11 – 11/17	Implement the lock-free queue using CAS	In progress
	Implement the lock-free queue using RTM	Done
	Finish checkpoint report	Done
11/18 – 11/24	Implement the coarse-/fine-grained locked queues (Yunjia)	In progress
	Implement the lock-free queue using CAS (cont'd) (Yunjia)	In progress
	Verify correctness of all concurrent queues (Jiaan)	
	Explore the implementations of distributed versions (Both)	
11/25 – 12/01	Implement the distributed versions (Both)	
	Optimize lock-free concurrent queues (Both)	
12/02 – 12/08	Benchmark and performance analysis (Both)	
	Finish final report (Both)	
	Finish poster (Both)	

Work Completed

We almost finished the implementations of the lock-free queue using CAS and the lock-free queue using transactional memory. For the lock-free queue using transactional memory, we verified that it works well under high contention concurrent environment. For the lock-free queue using CAS, we still need some times to debug some corner cases in the concurrent environment. We also set up the microbenchmark environment with some benchmark cases to measure the performance of these concurrent queues.

Goals and Deliverables

Goals

We think that our progress aligns well with our schedule. Most of the stuffs planed for the first two weeks have been finished, and we also have some stuffs ready which are planned for the following weeks. We believe we can achieve the ‘’plan to achieve” goal (two versions of lock-free queues). For the ‘’hope to achieve” goal (distributed versions), we may evaluate it after we finish our exploration in the week of Nov 18. Our new list of goals is listed below.

Plan to achieve

- Coarse-grained locked queue
- Fine-grained locked queue
- Lock-free queue using CAS instructions
- Lock-free queue using transactional memory
- Benchmark
- Performance analysis

Hope to achieve

- Distributed version of coarse-grained locked queue and fine-grained lock queue
- Distributed version of lock-free queues

Fallback Plan

- Skip the lock-free queue using transactional memory (already avoided)

Deliverables

We will show the speedup graphs and the comparisons between different versions of concurrent queues we implement. To demonstrate we did a good job, we will provide efficient implementations of lock-free concurrent queues as well as the performance analysis to show that when we should use a lock-free version to achieve higher efficiency.

Our project is a system project, so the main goal is to implement a better concurrent queue using lock-free techniques. However, because of the nature between the optimistic and pessimistic concurrency control, there may be some scenarios that it is inappropriate to adopt a lock-free queue. Therefore, we will also provide an analysis about it.

Preliminary Results

No, though we have some infrastructure of ourselves to evaluate our code during the development, we don’t have any well summarized results to show at this point.

Challenges

Multi-threaded debugging is tough, and sometimes it is hard to accurately capture the race conditions. Compared with the lock-free versions, locked versions are much simpler. For the distributed versions, we still need some times to identify the potential difficulties.