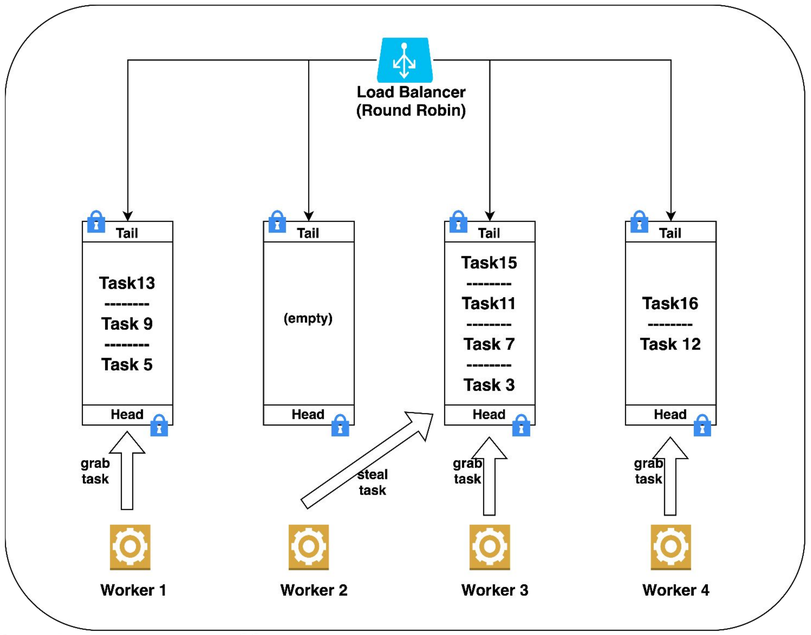
Zorro: A Distributed Thread-Pool with Work Stealing

Yukun Jiang(yukunj) & Leo Guo(jiongtig)

Abstract

In this project, we incrementally designed and implemented a distributed threadpool with work stealing policy called **Zorro** in C++17, which mimics **Cilk** to a great extent. We benchmarked different workload on PSC supermachine and achieved great performance in reducing **contention** and **workload imbalance**.



Zorro Distributed Thread Pool

API set

- **void Begin()**
- **void Exit()**
- **void WaitUntilFinish()**
- **void Submit(Task t)**

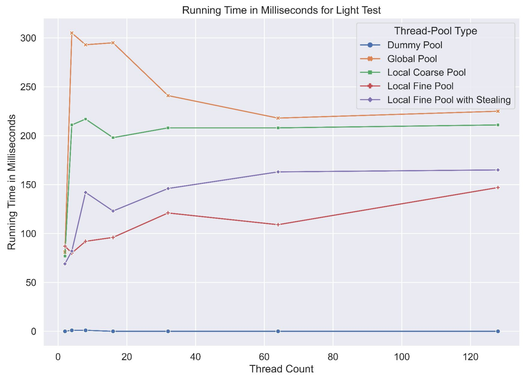
Design Tradeoff

Traditional single work queue threadpool typically suffers from a high contention on the queue entry, while distributed work queue sees workload imbalance problem. We strives to tackle both problems compatibly.

Implemented Frameworks

- Dummy Pool: serial baseline
- Global Pool: a single global work queue with mutex lock
- Local Coarse Pool: thread-local work queue with mutex lock
- Local Fine Pool: thread-local work queue with fine-grained lock
- Local Fine Pool with Stealing: Local Fine Pool + unidirectional stealing work from neighbors when idle.

Contention Benchmark



By implementing a local work queue and a fine-grained locking queue with separate locks on pop and push, we were able to successfully reduce the problem of lock contention that occurs when workers try to submit or grab tasks from the thread pool.

Work-stealing would slightly increase the contention, as each worker would try to grab task from peers.

Benchmark on Different Workloads