XQueue: Extreme Fine-grained Concurrent Lock-less Queue

Poornima Nookala[1], Peter Dinda[2], Kyle Hale[3], Ioan Raicu[4]

[1,3,4]Illinois Institute of Technology {pnookala@hawk.iit.edu, khale@cs.iit.edu, iraicu@cs.iit.edu}, [2]Northwestern University {pdinda@northwestern.edu}

Overview

A single general purpose shared memory machine today may have hundreds of hardware threads available. Operations that may run in tens of cycles on a single core using a single thread can take upwards of millions of cycles when multiple threads are competing for shared resources. The concurrent multiple producer, multiple consumer (MPMC) queue is a critical building block in numerous systems such as the task scheduler of a runtime system (or operating system) that supports modern parallel programming models. We present the design, implementation, and evaluation of XQueue, a novel lock-less concurrent queuing system with relaxed ordering semantics that is geared to realizing scalability to hundreds of concurrent threads. XQueue implements the MPMC interface using multiple queues in order to reduce contention, applies simple deterministic load balancing techniques, and eliminates all use of locks and atomic operations with a lock-less design. Experimental results show that XQueue can deliver concurrent operations with 110 to 400 cycle latency at scales up to 384 hardware threads.

Concurrent queues are terrible!

- Analyzed latency and throughput of mutex, spinlock, semaphore and atomic fetch-and-add for an increment operation on Mystic testbed.
- Mystic covers latest many-core architectures from Intel, AMD, IBM and ARM with processors such as Haswell, Broadwell, Skylake, Phi, Opteron, Ryzen, Threadripper, Epyc, Power9, and ThunderX
- **Latency** of a single atomic increment on a **Skylake system with 192-cores** and 384 hardware threads when running on all threads concurrently is **33592 cycles** whereas on **Intel Xeon Phi Knights Landing** with 64-cores and 256 hardware threads, latency reaches **3868 cycles**.
- Latency of enqueue/dequeue operation on **SPSC queue** takes **30 to 70 cycles** depending on the architecture and clock frequency.
- **Average throughput** of enqueue/dequeue operations on **SPSC queue** reaches **270 million operations per second** on **Intel Skylake 192-core** machine.
- Latency of **MPMC queue** can reach up to millions of cycles under high contention and **throughput can drop up to as low as 300,000 operations per second**.
- These results provide enough motivation to investigate methods to exploit full concurrency on many-core architectures while not compromising on the lowest latency that can be achieved.

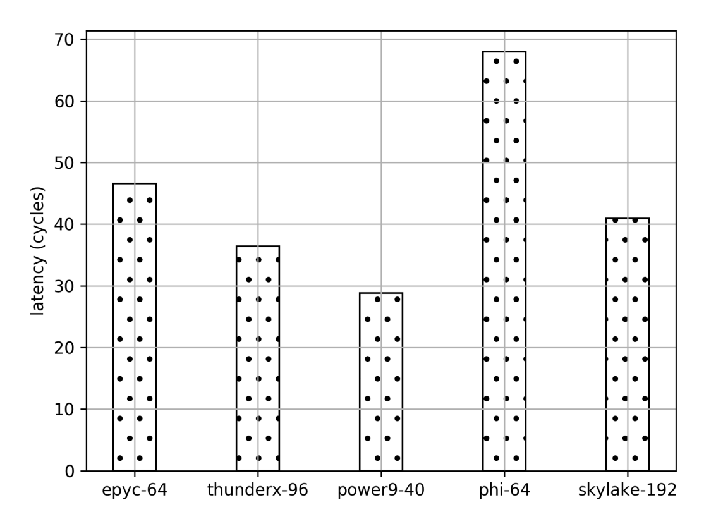

Figure 1: SPSC Queue Latency

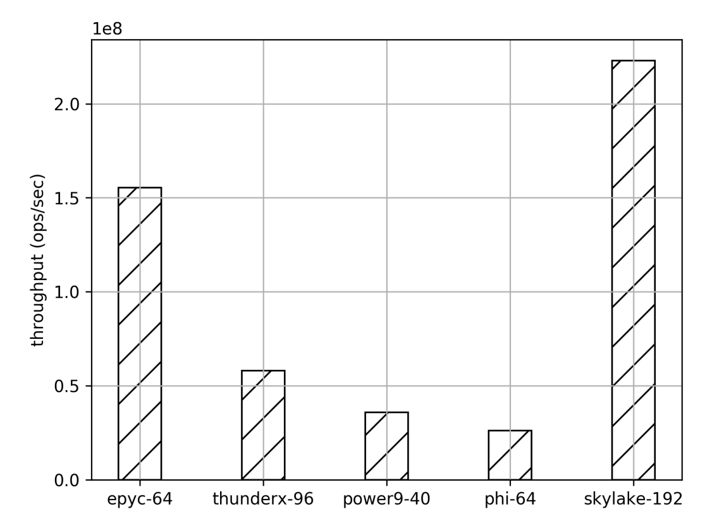

Figure 3: SPSC Queue Throughput

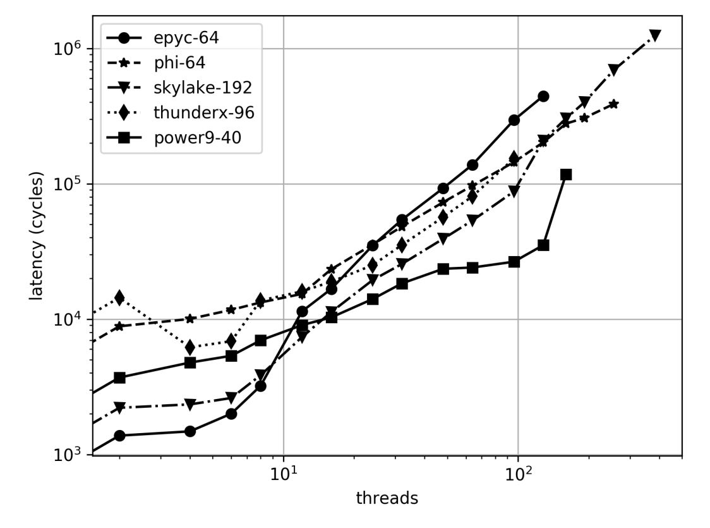

Figure 2: MPMC Queue Latency

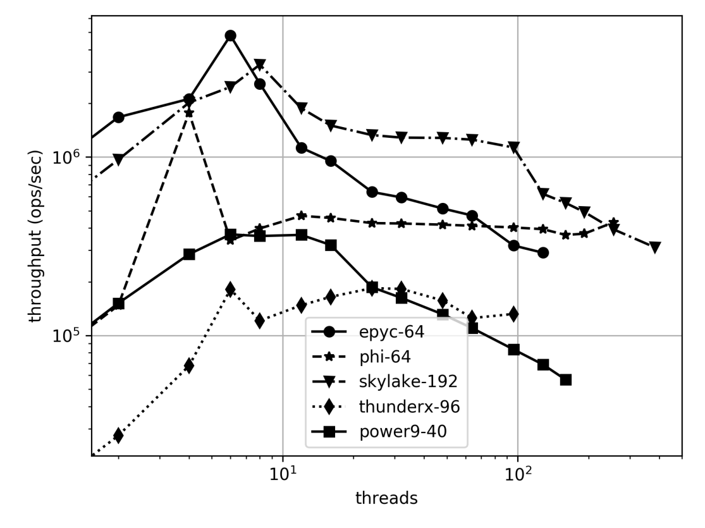

Figure 4: MPMC Queue Throughput

Problem Statement

Having scalable and fast queue data structures under high concurrency is a critical missing link towards the realization of efficient parallel runtimes on many-core architectures. *It is essential to analyze and optimize basic data structures that form the barebones of parallel runtime systems so they do not become the bottleneck for performance.*

XQueue - Design and Implementation

- Core idea of **XQueue** is to have **two queues per core**, one being the master and other being the auxiliary queue.
- There is one master queue and one auxiliary queue on each core with one producer thread per queue and a common consumer thread for both queues.
- XQueue is implemented **without using any types of locks or atomic operations or barriers**. Current implementation uses B-queue [1] which is a lock-free concurrent SPSC queue.
- XQueue employs **load balancing** techniques where items can be added to auxiliary queues **depending on the topology** defined (round-robin in Figure 5).

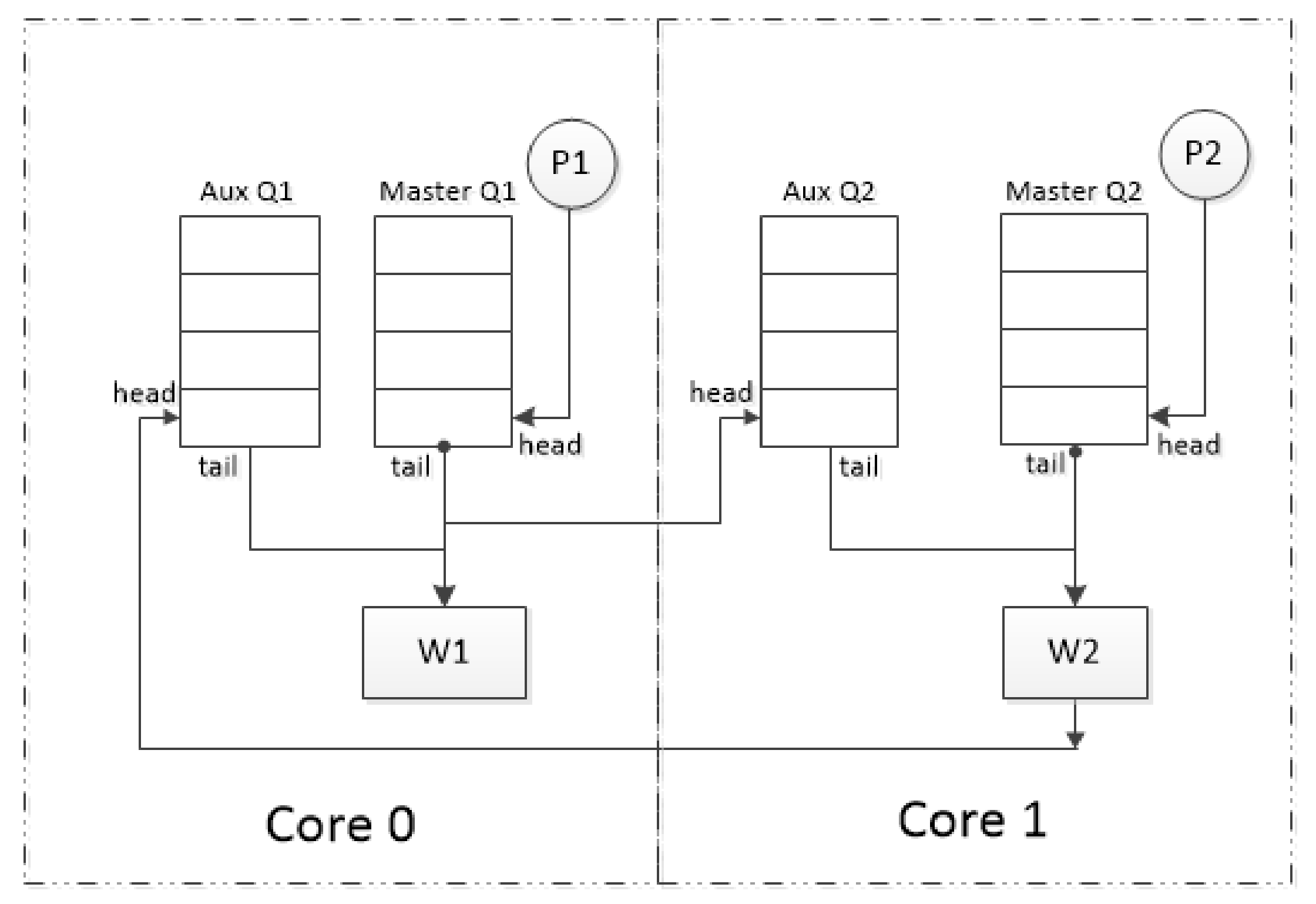

Figure 5: XQueue Architecture

Microbenchmarks

- For evaluation purposes, we implemented two versions of XQueue, one with a lock-less SPSC queue and another one with mutex-based MPMC queue.
- The **throughput** achieved on **skylake-192** with **XQueue with lock-less queue** is **5 billion operations per second**. For **XQueue using lock-based queue**, the average throughput achieved is **135 million operations per second**.
- **Latency** of queue operations on **XQueue using lock-less queue is 110 to 400 CPU cycles** on average on all the different architectures **with 384 threads** of execution.

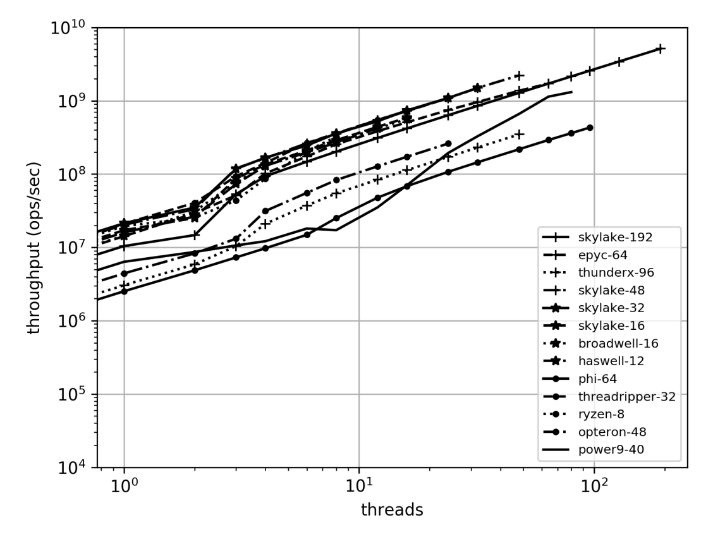

Figure 6: XQueue Throughput using lock-less queue

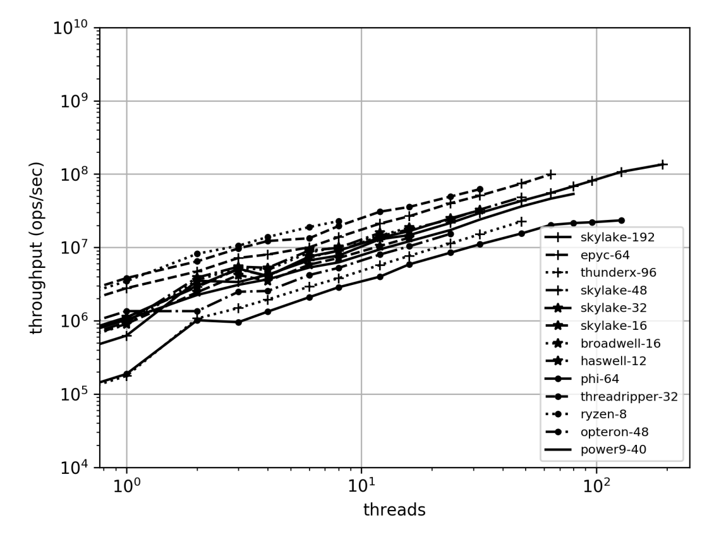

Figure 7: XQueue Throughput using lock-based queue

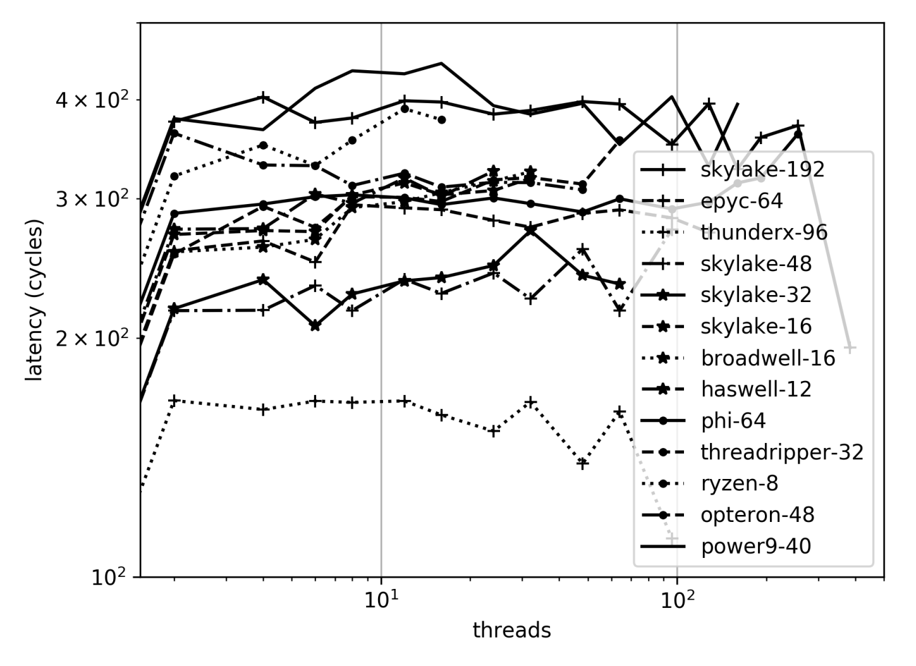

Figure 8: Latency of Xqueue using lock-less queue

latency <400 cycles with 384 threads

Conclusion

- XQueue is an extremely scalable lock-less con-current MPMC out of order queue that can scale up to hundreds of threads of execution.
- Future Work: XTask, a light-weight parallel runtime system which can achieve low latency and high throughput at extreme scale.

[1] *Junchang Wang, Kai Zhang, Xinan Tang, and Bei Hua. B-queue: Efficient and practical queuing for fast core-to-core communication.* International Journal of Parallel Programming, *41(1):137–159, Feb 2013.*