POSTER: wCQ: A Fast Wait-Free Queue with Bounded Memory Usage

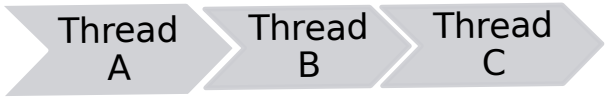
Ruslan Nikolaev *, rnikola@psu.edu, Penn State University, USA

Binoy Ravindran, binoy@vt.edu, Virginia Tech, USA

** Most of the work was done while the author was at Virginia Tech*

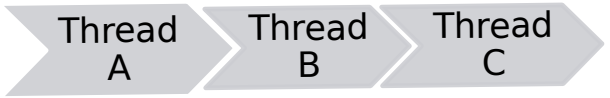
Concurrent Data Structures

- Many-core systems today require efficient access to data
  - Concurrent data structures
- Multiple threads need to ***safely*** manipulate data structures (similar to sequential data structures)
  - "nothing bad will happen"

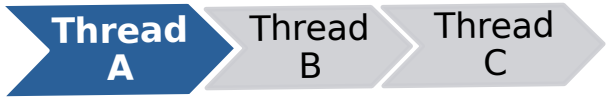


Concurrent Data Structures

- Many-core systems today require efficient access to data
  - Concurrent data structures
- Multiple threads need to ***safely*** manipulate data structures (similar to sequential data structures)
  - "nothing bad will happen"



- Concurrency also adds a ***liveness*** property, which stipulates how threads will be able to make progress
  - "something good will happen eventually"



Wait-Freedom

- **Non-blocking** data structures
  - **Lock-free** data structures require that at least ***one*** thread completes an operation after a ***finite*** number of steps
  - **Wait-free** data structures require that ***all*** threads complete ***any*** operation after a ***finite*** number of steps
- Wait-free algorithms have increasingly gained more attention due to their strongest non-blocking progress property
  - But building wait-free queues is challenging

Existing Approaches

- Kogan-Petrank's queue [PPoPP'11]
  - Wait-free but slow
- CRTurn queue [PPoPP'17]
  - Wait-free but is still slow
- Yang and Mellor-Crummey (YMC) queue [PPoPP'16]
  - Fast but has flawed memory reclamation => not truly wait-free
- LCRQ [PPoPP'13]
  - Fast and memory reclamation is correct but is only lock-free
- Scalable Circular Queue (SCQ) [DISC'19]
  - Fast but is only lock-free
- We present wait-free circular queue (**wCQ**) which extends SCQ

wCQ's Key Idea

- Memory reclamation is tough when also considering progress properties
  - ***Key insight***: avoid memory reclamation altogether
- Kogan-Petrank's ***fast-path-slow-path*** method [PPoPP'12] does not support specialized instructions such as ***fetch-and-add*** (FAA)
  - FAA scales better and is the key instruction in SCQ
  - We design our own fast-path-slow-path method for SCQ that also supports FAA
- ***Slow path:*** eventually all active threads help a thread that is stuck
  - One of these threads will eventually succeed due to the underlying SCQ's ***lock-free*** guarantees (i.e., at least one thread always succeeds)
  - All helpers must repeat exactly the same procedure as the helpee

Results

- **wCQ** is the fastest wait-free queue
  - wCQ generally outperforms YMC, for which memory usage can be unbounded
  - LCRQ can yield better performance but lacks wait-freedom
- **wCQ**'s performance is close to the SCQ algorithm

More Details

- Code is open-source and available at:
  - https://github.com/rusnikola/wfqueue
- Full paper is available as an arXiv report:
  - https://arxiv.org/abs/2201.02179

THANK YOU!