The Contention Avoiding Concurrent Priority Queue

Kjell Winblad
Konstantinos Sagonas

Department of Information Technology
Uppsala University, Sweden

Sept 30, 2016

Background: Concurrent Priority Queues

- Operations
  - Insert
  - DeleteMin
- Important for Parallel Applications
  - Scheduling
  - Discrete Event Simulation
  - Best First Search Algorithms
    - E.g. Dijkstra's Single Source Shortest Paths

UPPSALA UNIVERSITET

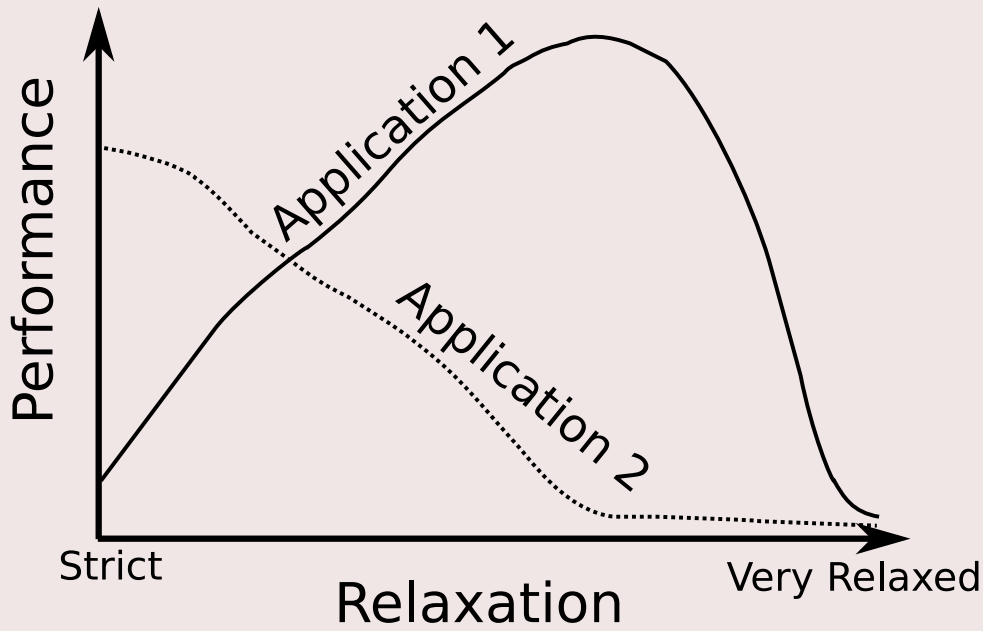
Current Concurrent Priority Queues

- Strict Semantics
  - DeleteMin always returns the smallest item
- Relaxed Semantics
  - DeleteMin may return an item which is not the smallest

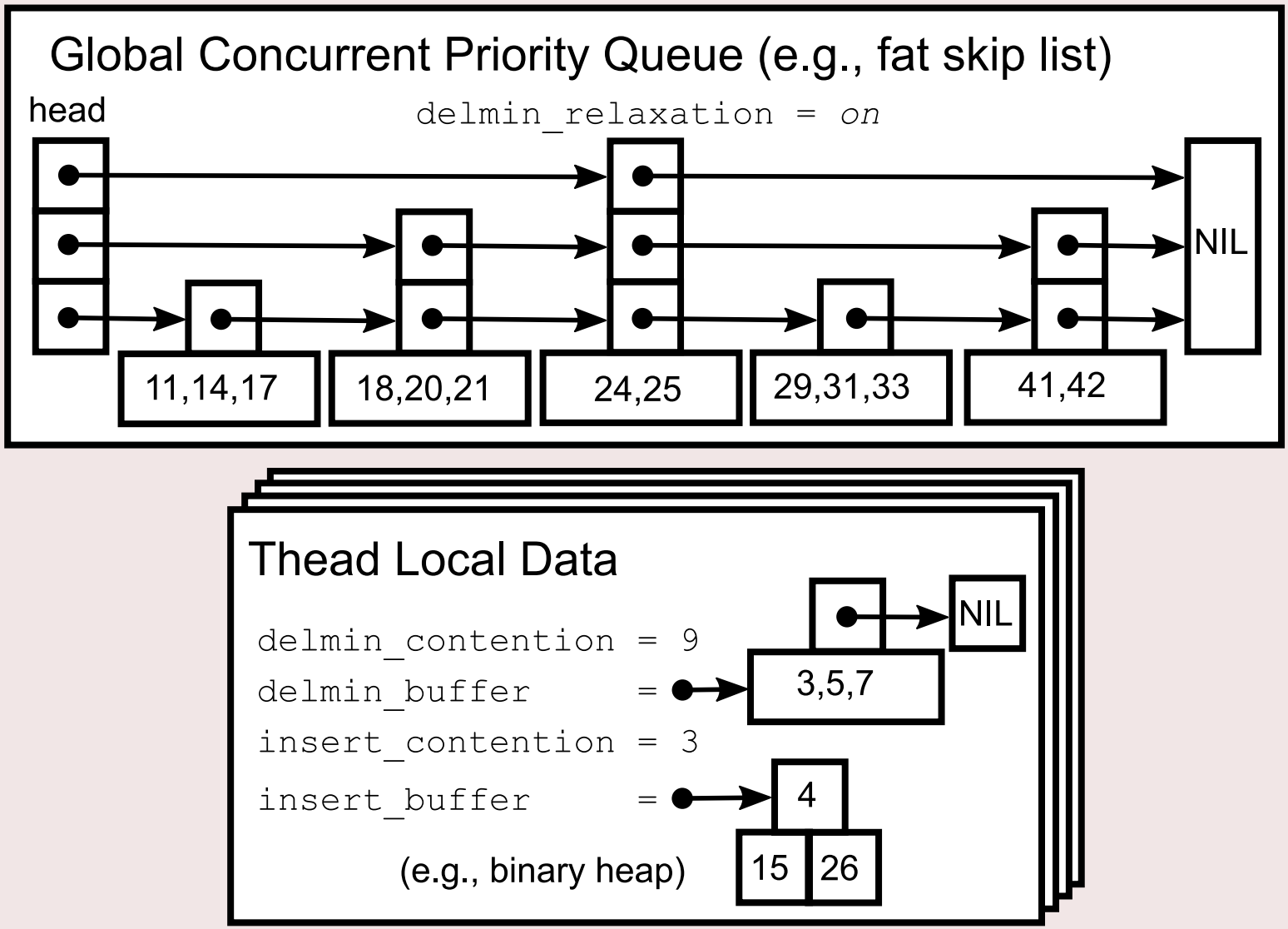
Our Contribution

- Contention Avoiding Concurrent Priority Queue (CA-PQ)
  - Adaptive semantics
    - Strict semantics when contention is low
    - Relaxed semantics when contention is high
  - Avoids shared memory accesses

Strict vs Relaxed

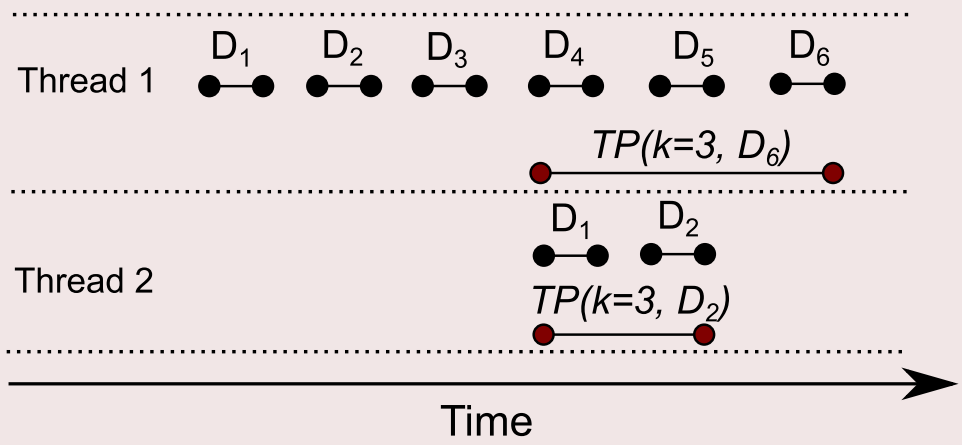


The Structure of CA-PQ



Properties

Semantic Guarantees



See paper for more details

UPPSALA
UNIVERSITET

Related Work

- SprayList (Alistarh *et al.*, PPoPP'2015)
- *k*-LSM (Wimmer *et al.*, PPoPP'2015)
- MultiQueue (Rihani *et al.*, SPAA 2015)

UPPSALA
UNIVERSITET

Comparison with Related Work

- CA-PQ doesn't access shared data in every DeleteMin call
  - \+ Less traffic in the memory system
  - − Precision of DeleteMin
- CA-PQ only activates relaxed semantics under contention
  - \+ Works well both when contention is high and low
  - \+ Data structure automatically fine tunes to the application

Evaluation

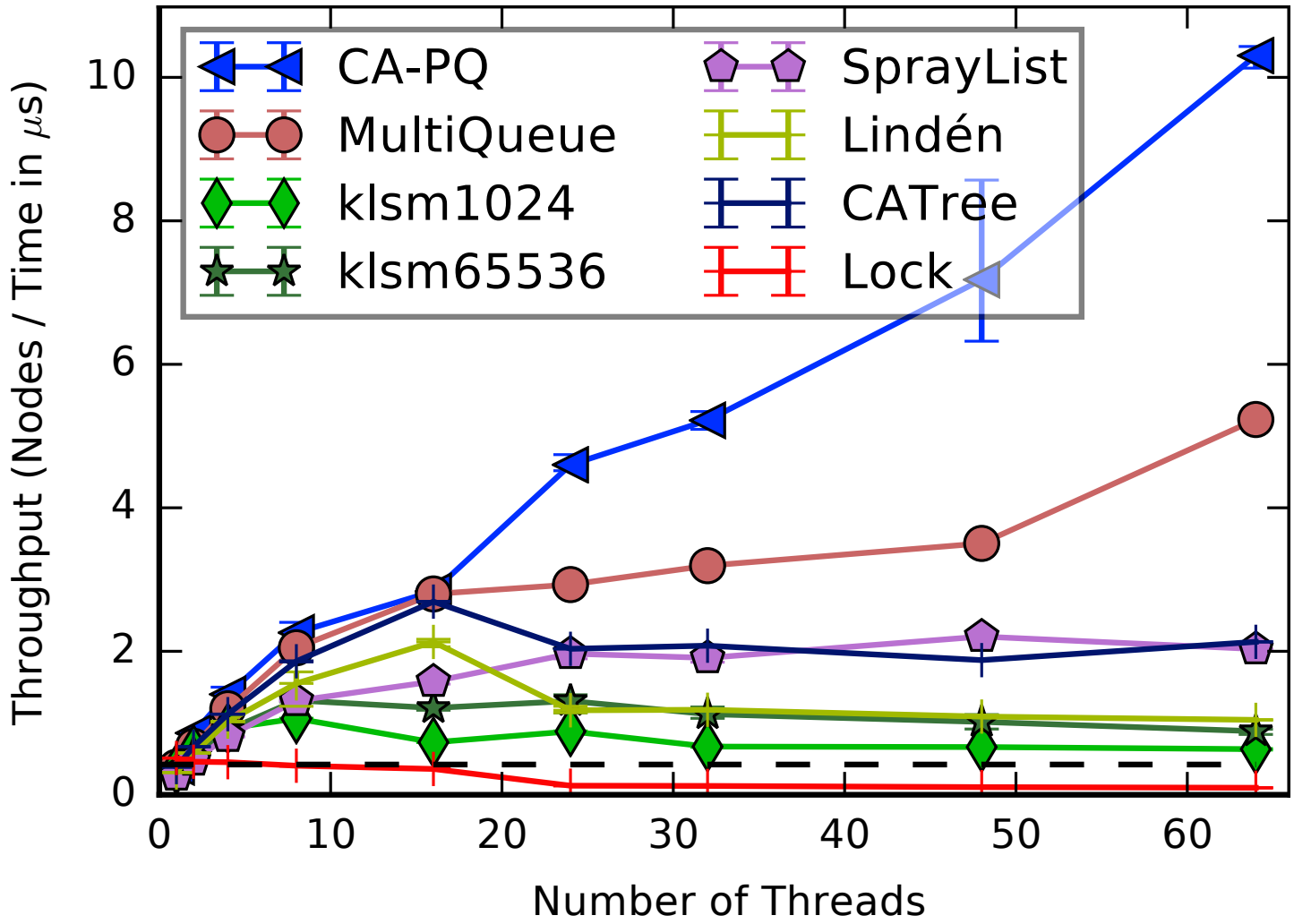
Benchmark Application

- Parallel version of Dijkstra's single source shortest path (SSSP) algorithm
- More wasted work when the item returned by DeleteMin is further from the actual minimum

Machine

- Four Intel(R) Xeon(R) E5-4650 CPUs (2.70GHz, turbo boost turned off)
- **Total of 64 logical cores**

Comparison to Related Data Structures (LiveJournal)



UPPSALA
UNIVERSITET

Evaluation of different CA-PQ variants

- DeleteMin contention avoidance is beneficial in **all** instances
- Insert contention avoidance is beneficial in **some** instances
- The dynamic CA-PQ variant always close to the best variant

What to take home?

The Contention Avoiding Concurrent Priority Queue

- Adaptively changes semantics based on detected contention
- Avoids shared memory accesses and contention by buffering items
- Outperforms other concurrent priority queues on SSSP