Server Implementations of HTTP/2 Priority

Kazu Yamamoto
@kazu_yamamoto
History

- **h2o (in C)**
  - Kazuho Oku
  - Array of Queue (external)
  - Enqueue O(1), dequeue O(1), delete O(1)
  - Deficit and delete information is managed outside

- **nghttp2 (in C)**
  - Tatsuhiro Tsujikawa
  - Binary Heap (external)
  - Enqueue O(log N), dequeue O(log N), delete O(log N)
  - Deficit and delete information is managed outside

- **Warp (in Haskell)**
  - Kazu Yamamoto
  - Random Skew Heap
  - Enqueue O(log N), dequeue O(log N), delete O(N log N)
  - No deficit and delete information
  - Now using PSQ (Priority Search Queue)
  - Enqueue O(log N), dequeue O(log N), delete O(log N)
Today’s topic

- Flat priority queue only
- Nested priority queue can be build over flat ones
Background

- Using weight as priority of max heap
  - it’s not fair

- Example
  - A for weight 10
  - B for weight 5
  - C for weight 1

- Result sequence
  - A(10), A(9), A(8), A(7), A(6), A(5), B(5), A(4), B(4), ...
Random Skew Heap

- Selecting a frame based on a random value
  - 1 - 10 for A
  - 11 - 15 for B
  - 16 for C

- To implement $O(\log N)$ operations, skew heap is used
Random Skew Heap

Pros
- No additional information

Cons
- It is hard for me to prove fairness
- It is difficult to write test cases
- Pseudo random generators are slow for this purpose
- delete is $O(N \log N)$
Weighted Fair Queueing

- Inverted weight with min heap
  - New: \( \text{deficit} = \text{min_deficit_in_heap} + \frac{\text{constant}}{\text{weight}} \)
  - Exist: \( \text{deficit} = \text{last_deficit} + \frac{\text{constant}}{\text{weight}} \)

- Deficit examples (constant is 65536)
  - A for weight 10, deficit = 6553
  - B for weight 5, deficit = 13107
  - C for weight 1, deficit = 65536

- Result sequence
  - A (6553)
  - A (13106)
  - B (13107)
  - A (19659)
  - A (26212)
  - B (26214)
  - ...
Weighted Fair Queueing

- Pros
  - Fairness is proved already though I don’t understand
  - It’s easy to write test cases
  - All operations could be $O(\log N)$

- Cons
  - Need to memorize deficit for each entry
  - Deficit could be overflowed (but it is unlikely)
Min Heap

- **Binary heap**
  - Many people knows
  - Perfect balance in arrays
  - $O(\log N)$ for enqueue, dequeue and delete
  - The array must be glow if the concurrency is increased

- **Okasaki heap**
  - Immutable data
  - $O(\log N)$ for enqueue and dequeue
  - $O(N)$ for delete

- **Priority search queue**
  - Immutable data
  - Blend of search tree and heap
  - $O(\log N)$ for enqueue, dequeue and delete
Array of Queue

- Emulating heap with an array of queues
  - Behavior is a little bit different
- Deficit and offset
  - Exist: deficit = \((last\_deficit + constant / weight) \mod constant2\)
  - Exist: offset = \((last\_deficit + constant / weight) / constant2\)
- An element is queued according to its offset
  - "Find first bit set" in \(O(1)\) can be used to find a non empty queue
Array of Queue

- **Pros**
  - It’s easy to write test cases
  - All operations could be O(1)
  - Deficit is not overflowed

- **Cons**
  - Implementation is a little bit complicated
Comparison

- 13 implementations
  - Random Skew Heap <- old Warp
  - Okasaki Heap (internal)
  - Okasaki Heap (external)
  - Priority Search Queue (internal) <- new Warp
  - Priority Search Queue (external)
  - Binary Heap (internal)
  - Binary Heap (external) <- nghttp2
  - Binary Heap STM (Software Transactional Memory) (internal)
  - Binary Heap STM (external)
  - Array of Queue (internal)
  - Array of Queue (external) <- h2o
  - Array of Queue STM (internal)
  - Array of Queue STM (external)

- Information managed internally or externally
  - Deficit
  - Deletion hints

- "internal" means abstract data type
Benchmark on enqueue & dequeue

- Repeating 10000 enqueue & dequeue with 100 streams
Benchmark of delete

- Deleting 100 streams
Conclusion

- Binary Heap would be the first choice for most programming language
  - nghttp2

- Array of Queue would be the next choice if you are not satisfied with the performance
  - h2o

- Priority Search Queue is recommended for highly concurrent programming language
  - Warp