On the Efficient Implementation of Pipelined Heaps for Network Processing

Hao Wang, Bill Lin
University of California, San Diego
Outline

- Introduction
- Pipelined Heap Structure
- Single-Cycle Operation
- Memory Management
- Conclusion
Outline

- Introduction
- Pipelined Heap Structure
- Single-Cycle Operation
- Memory Management
- Conclusion
Introduction

- Priority queues widely used in many network processing applications
  - Weighted Fair Queuing for advanced scheduling
  - Queue memory management using DRAMs
  - Network measurements

- Key design challenge: priority queues expected to operate at very high speed, e.g. 40 Gb/s and beyond
Introduction

- The conventional binary heap is widely used to implement priority queues, but requires $O(\log N)$ time complexity.

- However, too slow for 40 Gb/s and beyond.

- To address the performance requirements, a new data structure called the Pipelined Heap (P-Heap) was proposed [Bhagwan & Lin, INFOCOM 2000] with only $O(1)$ time complexity, at the expense of $O(\log N)$ pipelined stages.
In this talk, we describe two sets of improvements to the original work:

- First, we describe explicitly how we achieve single-cycle pipelined operations (which was not detailed in original work).
- Second, we describe new memory organizations to enable more efficient use of memories.
Outline

- Introduction
- Pipelined Heap Structure
- Single-Cycle Operation
- Memory Management
- Conclusion
Pipelined Heap

- Unlike conventional heap, all operations proceed top-down.
- In addition, each node in a pipelined heap has a capacity field.
Pipelined Heap

- Each pipeline stage has memory module $M_k$ that stores corresponding nodes in $k^{th}$ layer of pipelined heap.
- All operations proceed top-down thru pipelined stages.
Outline

- Introduction
- Pipelined Heap Structure
- Single-Cycle Operation
- Memory Management
- Conclusion
Single-Cycle Operation

- Use ENQUEUE operation as example

- Ideas:
  - Prepare for all possible future events
  - Drop events that are not going to happen in a later time
  - Insert new operation as frequently as possible

- Achieve twice the original speed, i.e. single-cycle per operation
ENQUEUE Operation

A' = \min\{A, A'\}

B' = \min\{B, B'\}

D' = \min\{D, D'\}

A'' = \min\{A, A'\}

B'' = \min\{B, B'\}

D'' = \min\{D, D'\}

Clock Cycles:

1. Read A and A'
2. Compare A and A'
3. Write A''

Clock Cycles:

1. Read B and C
2. Compare B and B'
3. Write B''

Clock Cycles:

1. Read D and E
2. Compare D and D'
3. Write D''
Single-Cycle Operation

- Result is that single-cycle operation can be realized for all of ENQUEUE, DEQUEUE, and ENQUEUE-DEQUEUE operations.

- Inter-operation management
  - Treat DEQUEUE operation as ENQUEUE-DEQUEUE operation that enqueues a VOID.
  - The only exception is DEQUEUE following another DEQUEUE, which results in two-cycle per operation.
Outline

- Introduction
- Pipelined Heap Structure
- Single-Cycle Operation
- Memory Management
- Conclusion
Memory Size Disparity Problem

- Because memory module $M_k$ stores corresponding nodes in $k^{th}$ layer of pipelined heap, there will be huge disparity between memory sizes of first thru last stages.
Memory Management Optimization

- We need more efficient way to “balance” the memory size requirements without interfering with pipelined operations

- We present two optimizations
  - First, we consider a “Forest-of-Heaps”
  - Next, we consider memory optimization for a single heap
"Forest-of-Heaps" Optimization

- In many applications, we may have many "logical" priority queues that share same memory buffer
- We call this a "Forest of Heaps". e.g. consider 10 heaps, each with 6-levels

For the 10 heaps together, the memory size for each of the 6 stages are 1x10, 2x10, 4x10, 8x10, 16x10, 32x10 = 10, 20, 40, 80, 160, 320, respectively

- Memory size disparity between first and last stage is 320/10 = 32x
“Forest-of-Heaps” Optimization

- Can “balance” memory requirement by alternately “inverting” half the heaps

- For 10 heaps, each with 6-levels, with optimization, memory size for each of the 6 stages are 165, 90, 60, 60, 90, 165, respectively

- Memory size disparity between largest and smallest stage is 165/60 < 3x
For single heap
- Can divide heap in half
- Then, alternately “invert” half the sub-heaps

E.g. heap with 6 stages:
- Original: 1, 2, 4, 8, 16, 32
  - Disparity = 32x
- Optimized: 21, 18, 32
  - Disparity < 2x

This idea can be combined with “forest-of-heaps” optimization
Detailed Realization

- The combination of heap with an inverted heap

- An outer control unit is needed
To implement this scheme, each memory structure needs to support potentially twice as many memory operations.

Therefore, to support one stage per cycle, each memory structure needs to support a read throughput of 8 and a write throughput of 6 per cycle.
Outline

- Introduction
- Pipelined Heap Structure
- Single-Cycle Operation
- Memory Management
- Conclusion
Conclusion

- A more aggressive pipelining structure is introduced.

- Memory management techniques aimed at resolving the disparity in storage requirements between top and bottom layers of pipelined heap are presented.

- These new proposed techniques enable a faster priority queue implementation.
Questions?

Thank You