BMW Tree: Large-scale, High-throughput and Modular PIFO Implementation Using Balanced Multi-way Sorting Tree

(SIGCOMM '23)
Ruyi Yao¹, Zhiyu Zhang¹, Gaojian Fang¹, Peixuan Gao³,
Sen Liu¹, Yibo Fan¹, Yang Xu²¹², H. Jonathan Chao³

¹Fudan University
²Peng Cheng Laboratory
³New York University
Packet Scheduling

We need a programmable packet scheduler.

- Minimizing Flow Completion Time
  - SJF, SRPT
- Bandwidth Allocation Fairness
  - PGPS, WFQ, DRR, SFQ
- Minimizing Tail Packet Delay
  - LSTF, FIFO+
- Deterministic Packet Delay
  - TAS, CQF, ATS
- Future Scheduling Goals

To Wire
Push-In First-Out (PIFO) Model

Rank Store (SRAM)

Flow Scheduler

1

push

0 0 0 1 1 2

pop

Decreasing ranks

Rank Computation

9 6 4 3 1
8 4 2
8 7 5 3
9 7 5 3 1
9 7 5 3 1
6 4 3 2
4
Push-In First-Out (PIFO) Model

WFQ
- \( f = \text{flow(pkt)} \)
- \( \text{pkt.start} = \max(f.\text{finish}, \text{virt}\_\text{time}) \)
- \( f.\text{finish} = \text{pkt.start} + \frac{\text{pkt.len}}{f.\text{weight}} \)
- \( \text{pkt.rank} = \text{pkt.start} \)

Rank Store (SRAM)
- Decreasing ranks

Flow Scheduler
- push
- pop

Example:
- Push: 4
- Pop: Decreasing ranks
- Flow Scheduler:
  - 1
  - 2 1 1 0 0 0
Push-In First-Out (PIFO) Model

LSTF
- pkt.slack = (D - t - e')
- pkt.rank = pkt.slack + pkt.arrival_time

A packet scheduler works as a priority queue according to ranks.
Desirable Properties of a Packet Scheduler

- Large Scalability
- High Throughput

- Time budget for scheduling is short
  - 30 ns for MTU pkt at 400Gbps
  - ns-precision needed in TSN

We need a **large-scale and high-throughput programmable packet scheduler.**
Existing Programmable Schedulers

- **Heap-based Implementation**: pHeap\(^1\), pipelined Heap\(^2\)
  - Use SRAM to store packets' metadata, high scalability
  - Low throughput, Complex hardware

- **Register-only Implementation**: sequencer\(^3\), PIFO\(^4\), APQ\(^5\)
  - High throughput
  - Low Scalability

- **FIFO-based Approximation**: AFQ\(^6\), EFQ\(^7\), PCQ\(^8\), Gearbox\(^9\), AIFO\(^10\)
  - Simple data structure
  - Ranking accuracy sensitive to rank patterns

---

[1] INFOCOM’00  
[2] ToN’07  
[3] SIGCOMM’92  
[5] VLSI’18  
[6] NSDI’18  
[7] ICDCS’20  
[8] NSDI’20  
[9] NSDI’22  
[10] SIGCOMM’21
Our Solution

BMW-Tree = Balanced Multi-way Sorting Tree

Core: 1 data structure + 2 hardware designs

- High throughput
- Large Scale
- Accurate Ranking
Data Structure

BMW-Tree = **Balanced Multi-way Sorting Tree**

- Each node contains up to $M$ unsorted elements in an $M$ order tree
- Heap property is satisfied
Data Structure

BMW-Tree = **Balanced Multi-way Sorting Tree**

**Operation:** push 17

- Push the element into the **leftmost least loaded** sub-tree
- Counter + 1
Data Structure

BMW-Tree = Balanced Multi-way Sorting Tree

Operation: push 29
Data Structure

BMW-Tree = Balanced Multi-way Sorting Tree

Operation: push 29
Data Structure

BMW-Tree = **Balanced Multi-way Sorting Tree**

*Operation*: pop

- Pop the element with the **smallest** value in the root node
- Counter - 1
Data Structure

BMW-Tree = Balanced Multi-way Sorting Tree

Operation: pop
Data Structure

BMW-Tree = Balanced Multi-way Sorting Tree

Operation: pop
An $L$ level $M$ way BMW-Tree supports $N = \frac{M(M^L - 1)}{(M - 1)}$ flows.

=> Hardware frequency is independent of the tree depth in theory.
Hardware Designs

Design #1: Register-based BMW-Tree (R-BMW)

- Modularized
  - connect the parent and child nodes
- Pipelined
  - $O(1)$ ranking time
    - 1 push per cycle
    - 1 pop every 2 cycles
    - push + pop 2 cycles
- Resources
  - $O(N)$ register consumption

R-BMW Building Block

High Throughput
Medium Scale
Hardware Designs

Design #2: **RPU (Ranking Processing Unit)** - driven **BMW**-Tree (RPU-BMW)

![Diagram of RPU-BMW Tree]

- **RPU (Ranking Processing Unit)**
- **SRAM**

**Load** and **Write**

Only one sub-tree was enabled during an operation.

```
<table>
<thead>
<tr>
<th></th>
<th>55</th>
<th>25</th>
<th>10</th>
<th>17</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>3</td>
</tr>
</tbody>
</table>
```

```
<table>
<thead>
<tr>
<th>78</th>
<th>77</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>
```

```
<table>
<thead>
<tr>
<th>57</th>
<th>35</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>
```

```
<table>
<thead>
<tr>
<th>36</th>
<th>33</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>
```

```
<table>
<thead>
<tr>
<th>37</th>
<th>29</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>
```
Hardware Designs

Design #2: RPU (Ranking Processing Unit) - driven BMW-Tree (RPU-BMW)

Load ①, write ①
Load ⑤, write ⑤
Hardware Designs

Design #2: **RPU (Ranking Processing Unit)** - driven **BMW-Tree (RPU-BMW)**

![Diagram showing RPU and SRAM with numbers and load/write operations]

- **Large Scale**
- **Low Throughput**

<table>
<thead>
<tr>
<th>1</th>
<th>3</th>
<th>3</th>
<th>3</th>
<th>3</th>
<th>57</th>
<th>35</th>
<th>36</th>
<th>37</th>
<th>29</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>
Design #2: **RPU (Ranking Processing Unit) - driven BMW-Tree (RPU-BMW)**

**Hardware Designs**

**RPU-BMW Architecture**

\[
\frac{M(M^{L-1}-1)}{M-1} + 1 \sim \frac{M(M^{L-1}-1)}{M-1}
\]
Hardware Designs

Design #2: RPU (Ranking Processing Unit) - driven BMW-Tree (RPU-BMW)

RPU-BMW Architecture
Design #2: RPU (Ranking Processing Unit) - driven BMW-Tree (RPU-BMW)
Hardware Designs

Design #2: **RPU**-driven **BMW**-Tree (**RPU-BMW**)  

- Modularized
- 4 operation primitives supported  
  - push, pop, load, write
- Pipelined  
  - \( O(1) \) ranking time  
    - 1 push per cycle
    - 1 pop every 2 cycles
    - push + pop 3 cycles
- Resources  
  - \( O(\log M N) \) register consumption

High Throughput
Large Scale

RPU-BMW Architecture
Evaluation

Settings:
- **FPGA**: Implement R-BMW and RPU-BMW on Xilinx Alveo U200 Data Center Accelerator Card with XCU200 FPGA
  - 1182k LUTs
  - 591k LUTRAMs
  - 2364k flip flops
- **ASIC**: Synthesize RPU-BMW in GlobalFoundries 28 nm process
- **Simulation**: NS3, packet level evaluation

Experiments:
- Evaluate the performance of R-BMW and RPU-BMW with different parameters
- Compare PIFO with R-BMW and RPU-BMW
- Measure the flow completion time (See details in paper)
Performance of R-BMW on FPGA

- Scale to 5460 flows
- 192Mpps, 4.8 x of PIFO
- Recommend 2-order R-BMW
Performance of RPU-BMW on FPGA

- scale to 87k flows
- Frequency is significantly higher than that of PIFO.
- Recommend 4-order RPU-BMW
Compare RPU-BMW with R-BMW on FPGA

- RPU-BMW costs much fewer LUT and FF than R-BMW.
- RPU-BMW has a higher frequency than R-BMW when supporting $\geq 4680$ flows.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>R-BMW</th>
<th>RPU-BMW</th>
</tr>
</thead>
<tbody>
<tr>
<td>M</td>
<td>L</td>
<td>Capacity</td>
</tr>
<tr>
<td>2</td>
<td>11</td>
<td>4094</td>
</tr>
<tr>
<td>4</td>
<td>6</td>
<td>5460</td>
</tr>
<tr>
<td>8</td>
<td>4</td>
<td>4680</td>
</tr>
</tbody>
</table>
Compare RPU-BMW with PIFO on 28nm ASIC

- RPU-BMW supporting 37k flows has an even smaller area than a 1k PIFO.

<table>
<thead>
<tr>
<th></th>
<th>M</th>
<th>L</th>
<th>Capacity</th>
<th>Meets Timing at 600 MHz</th>
<th>Chip Area / mm² (%)</th>
<th>Off-chip Mem (MB)</th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>8</td>
<td>87380</td>
<td>Yes</td>
<td>1.043 (0.522%)</td>
<td>0.57</td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>5</td>
<td>37448</td>
<td>Yes</td>
<td>0.127 (0.064%)</td>
<td>0.25</td>
<td></td>
</tr>
<tr>
<td>PIFO</td>
<td>1024</td>
<td>Yes</td>
<td>0.404 (0.202%)</td>
<td>-</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Note: Chip area is set as 200 mm².
Summary

BMW-Tree = Balanced Multi-way Sorting Tree

A novel data structure
- Modularized, insertion-balanced, pipeline-friendly with autonomous nodes

Two hardware designs
- Scheduling in $O(1)$ time
- $O(N)$ and $O(\log_M N)$ register consumption of R-BMW and RPU-BMW respectively
- RPU-BMW is the first accurate PIFO implementation supporting over 80k flows at as fast as 200Mpps
Thank You!

Q & A