DRACONIS: NETWORK-ACCELERATED SCHEDULING FOR MICROSECOND-SCALE WORKLOADS

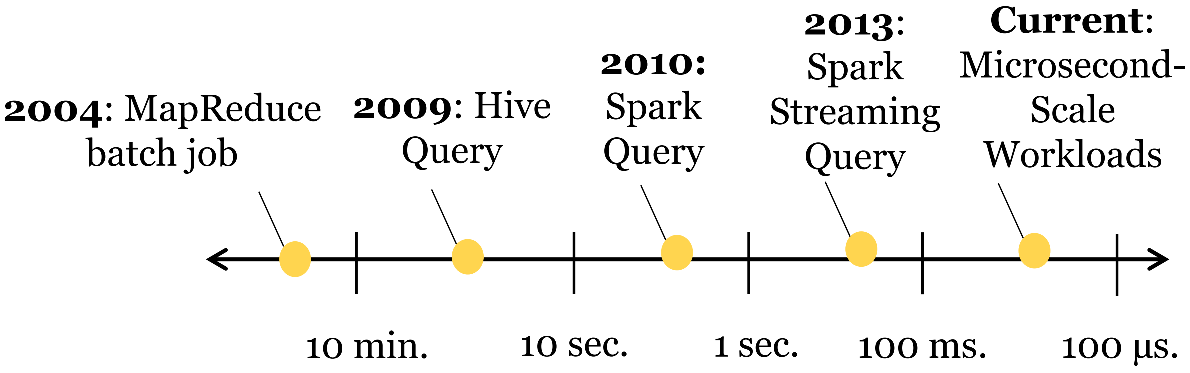
Sreeharsha Udayashankar, Ashraf Abdel-Hadi, Ali Mashtizadeh and Samer Al-Kiswany

UNIVERSITY OF WATERLOO

Microsecond-Scale Workloads

- Recent datacenter advances enable microsecond-scale workloads
  - Storage class memory, accelerators, and RDMA

- Applications are getting shorter despite accessing large datasets [1]
  - Financial analytics and algorithmic smart trading [2]
  - Realtime IoT analytics [3]
  - Rapid object detection [4]
  - Low latency web services [5]



[1] Kay Ousterhout, Aurojit Panda, Joshua Rosen, et al. The case for tiny tasks in compute clusters. *HotOS. 2013*

[2] Xinhui Tian, Rui Han, Lei Wang, et al. Latency critical big data computing in finance. *The Journal of Finance and Data Science*. 2015

[3] S. Verma, Y. Kawamoto, Z. M. Fadlullah, et al. A Survey on Network Methodologies for Real-Time Analytics of Massive IoT Data and Open Research Issues. *IEEE Communications Surveys & Tutorials*. 2017

[4] Tan Zhang, Aakanksha Chowdhery, Paramvir Bahl, et al. The design and implementation of a wireless video surveillance system. *MobiCom*. 2015

[5] Jeffrey Dean and Luiz André Barroso. The Tail at Scale. *Commun. ACM* .2013

UNIVERSITY OF WATERLOO

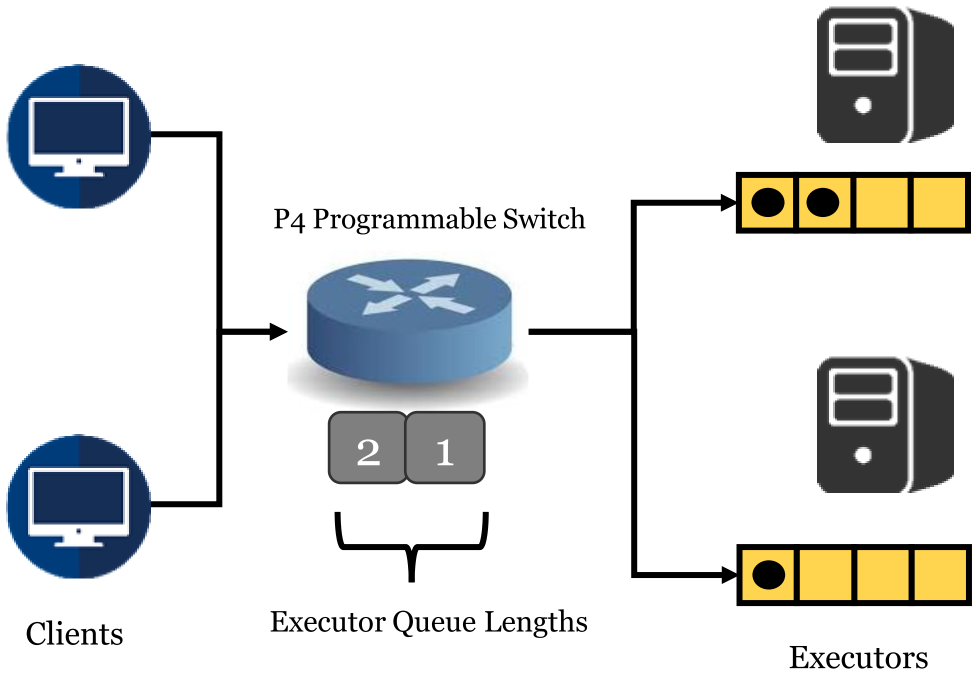
Scheduling Microsecond-scale Workloads

- Low *Scheduling Tail Latency*
  - Accurate scheduling decisions
- High *Scheduling Throughput*
  - Millions of scheduling decisions per second [6]

[6] Philipp Moritz, Robert Nishihara, Stephanie Wang, et al. Ray: A Distributed Framework for Emerging AI Applications. *OSDI 2018*

Modern Network-Accelerated Scheduling

- Distributed queue design
- **Disadvantages:**
  - Suboptimal - *Node-level blocking*
  - Inefficient implementations
    - R2P2 [7] – Recirculation
    - RackSched [8]– Sampling



[7] Marios Kogias, George Prekas, Adrien Ghosn, Jonas Fietz, and Edouard Bugnion. R2p2: Making rpcs first-class datacenter citizens. 2019 USENIX Annual Technical Conference (ATC 19), 2019

[8] Hang Zhu, Kostis Kaffes, Zixu Chen, Zhenming Liu, Christos Kozyrakis, Ion Stoica, and Xin Jin. Racksched: A microsecond-scale scheduler for rack-scale computers. the Proceedings of the 14th USENIX Symposium on Operating Systems Design and Implementation, 2020

Programmable Switches

- **Challenges**
  - No loops / recursion
  - Limited pipeline stages and memory
  - *Access a memory register only once per packet!*
- "Switches cannot house dynamic data structures such as queues"

Draconis Overview

- A novel in-network scheduling paradigm
  - Centralized in-switch queue -> Eliminates node-level blocking
  - Supports complex scheduling policies

- **Evaluation Highlights**:
  - **61%** lower tail latency over network-accelerated designs
  - ***52x*** throughput over server-based designs

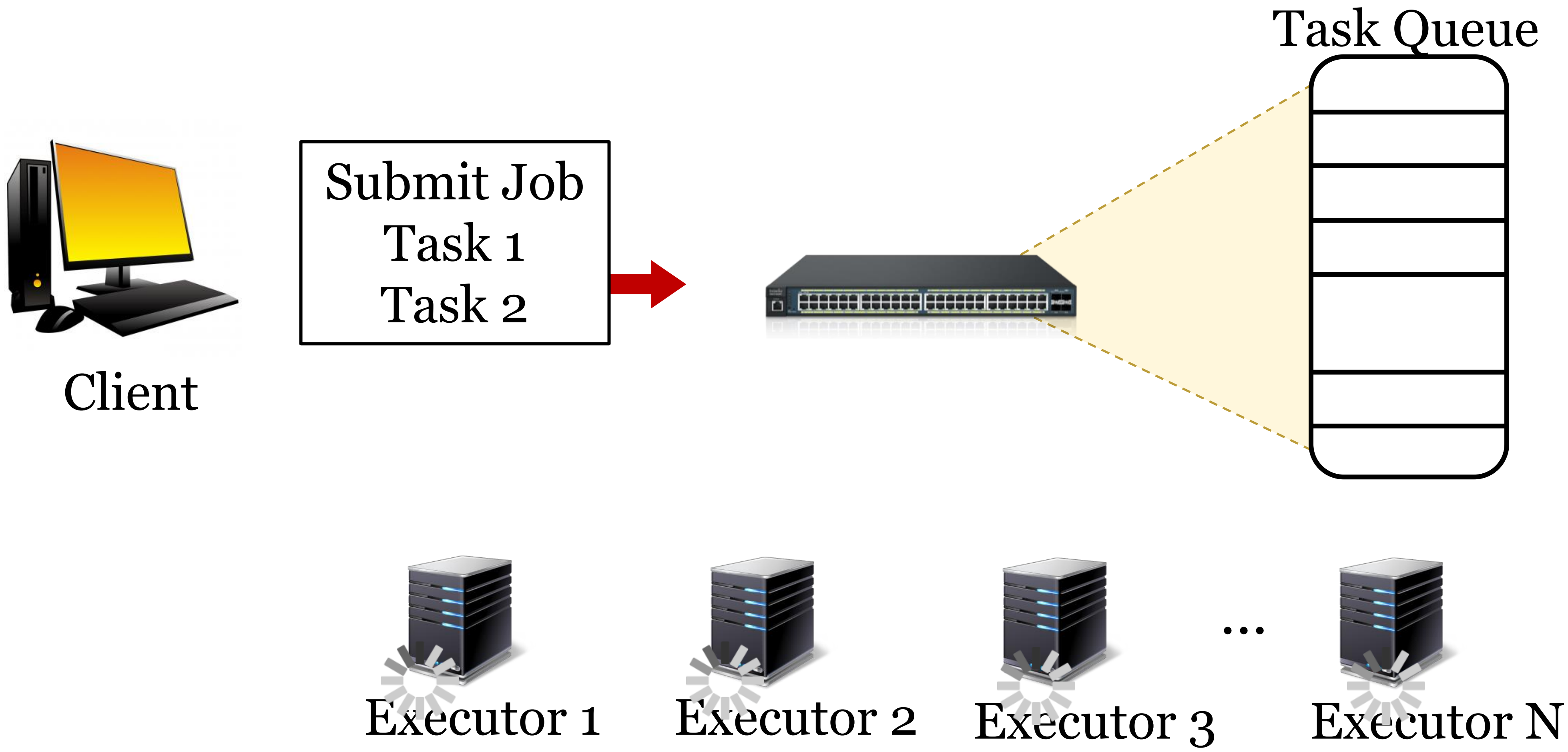
UNIVERSITY OF WATERLOO

Outline

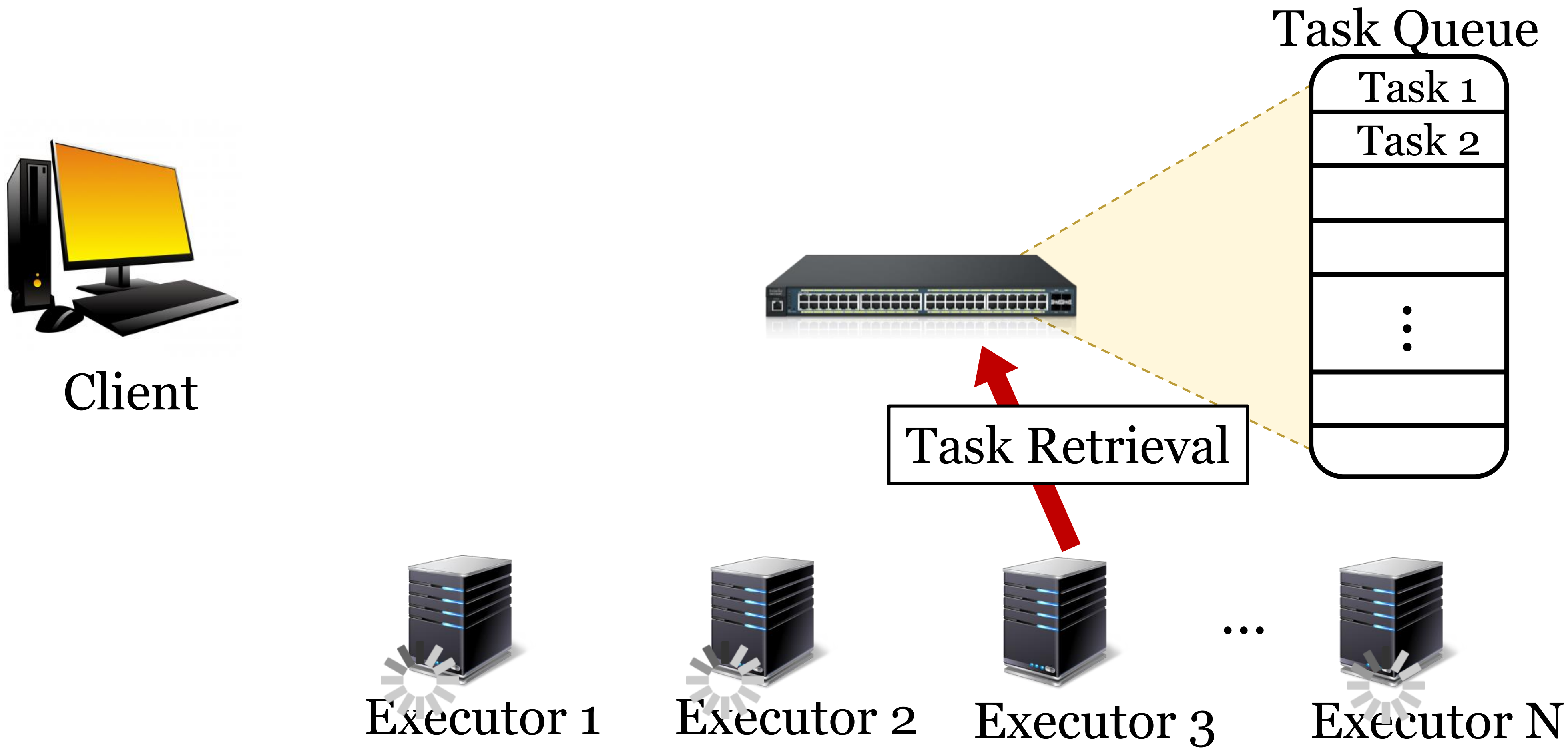
- Introduction
- Background
- FIFO Scheduling
- Complex Policies
- Evaluation
- Conclusion

UNIVERSITY OF WATERLOO

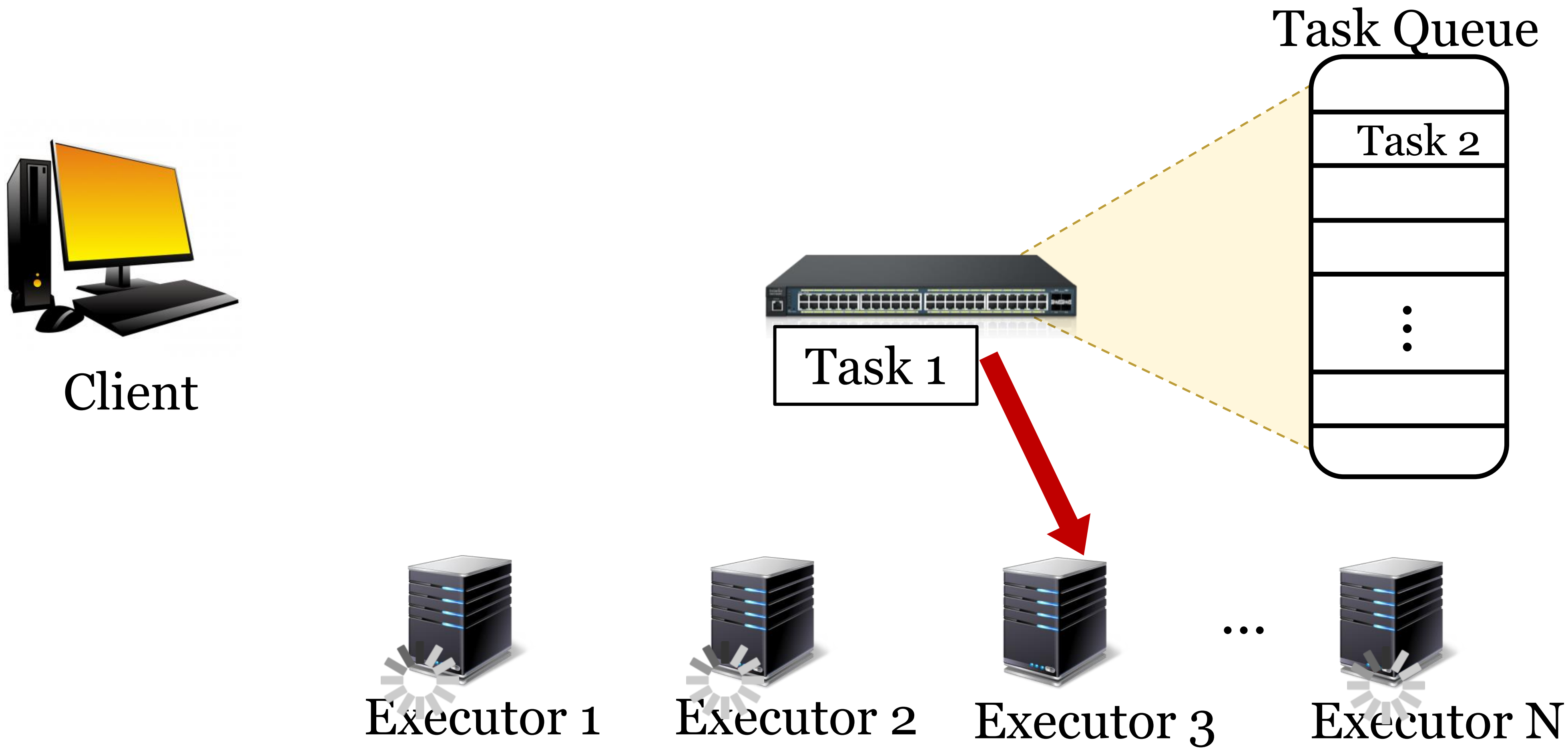
Draconis – FIFO Scheduling



Draconis – FIFO Scheduling



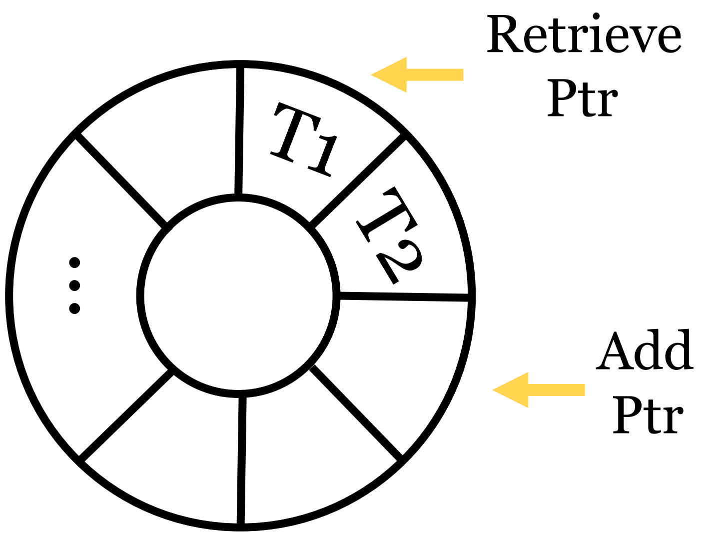
Draconis – FIFO Scheduling



UNIVERSITY OF WATERLOO

Design Components – Task Queue

- Tasks are stored in a circular queue
- Simple yet tricky to implement on modern programmable switches

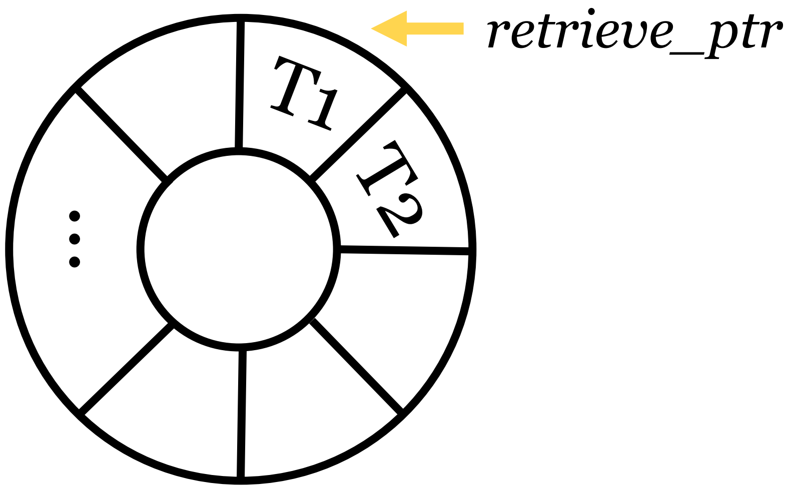


Draconis's Task Queue Design

UNIVERSITY OF WATERLOO

Design Components – Task Queue

```
on_retrieve {
    rtrv_ptr = read(rtrv_ptr)
    if( queue contains a task )
    {
        // Schedule the task
        rtrv_ptr ++
    } else {
          // Queue is empty
    }
}
```



Challenge: This accesses the pointer twice!

Design Components – Task Queue

```
on_retrieve {
    rtrv_ptr = read(rtrv_ptr)
    if( queue contains a task )
    {
        // Schedule the task
        rtrv_ptr ++
    } else {
          // Queue is empty
    }
}
```

```
on_add {
    add_ptr = read(add_ptr)
    if( queue has space )
    {
        // Enqueue the task
        add_ptr++
    } else {
          // Queue is full
    }
}
```

Challenge: This accesses the pointer twice!

Design Components – Task Queue

```
on_retrieve{
    rtrv_ptr = read_and_increment(rtrv_ptr)
    if(queue contains a task){
        // Schedule the task
    }
    else
        // rtrv_ptr needs fixing
}
```

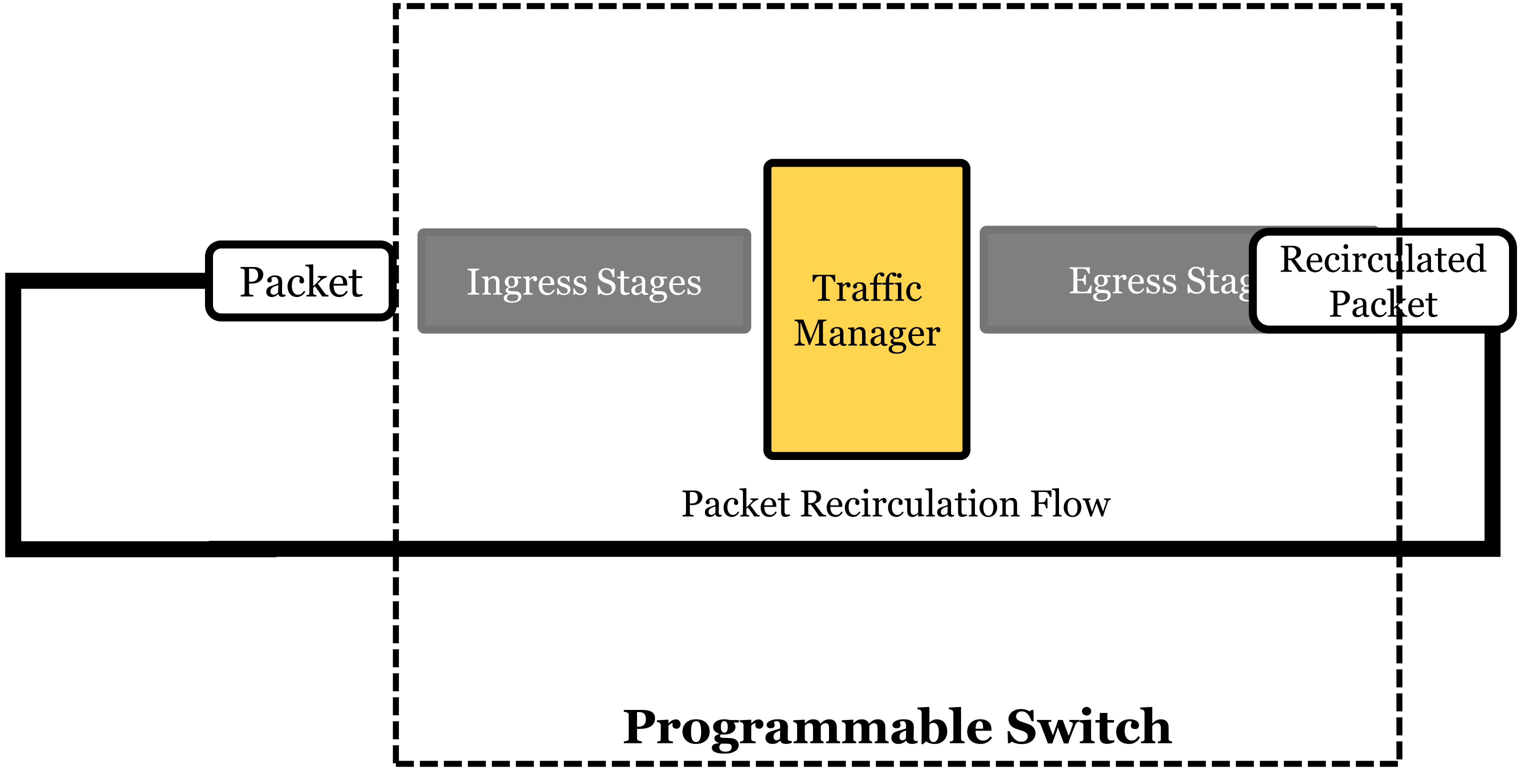
```
on_add(task) {
    add_ptr = read_and_increment(add_ptr)
    if (queue is full){
        //add_ptr needs fixing
        fix(add_ptr)
    }

    if (rtrv_ptr needs fixing)
        fix(rtrv_ptr)

    // Enqueue the task
}
```

read_and_increment() - Optimistic atomic read and increment of pointers

fix() – Fixing pointer values as required

Design Components - Pointer Fixing

- Use packet recirculation
- Adjust pointer values as needed
  - Use Boolean flags to prevent race conditions




```
fix(pointer){
    send_fix_pkt ( pointer, new_val )
}
```

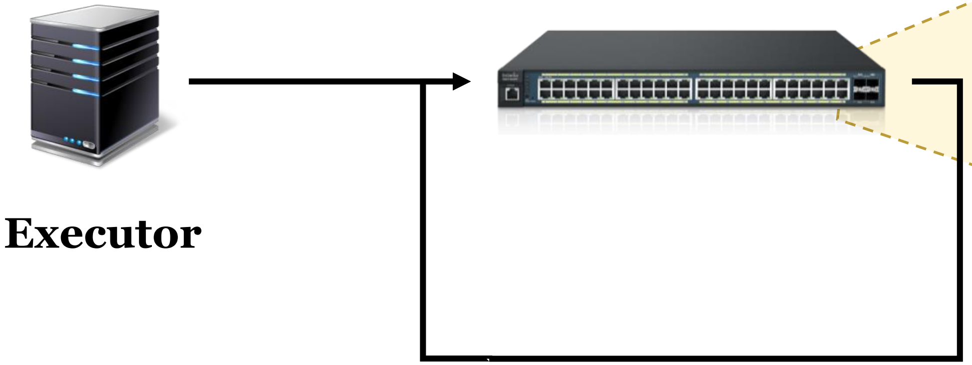
Complex Scheduling Policies

- Classes of service
  - Priority Scheduling
- Constraint based
  - **Resource-Aware Scheduling**
  - Locality Scheduling

Task Swapping : Resource-Aware Scheduling

Task Retrieval

Executor_Rsrc:
R2



Executor

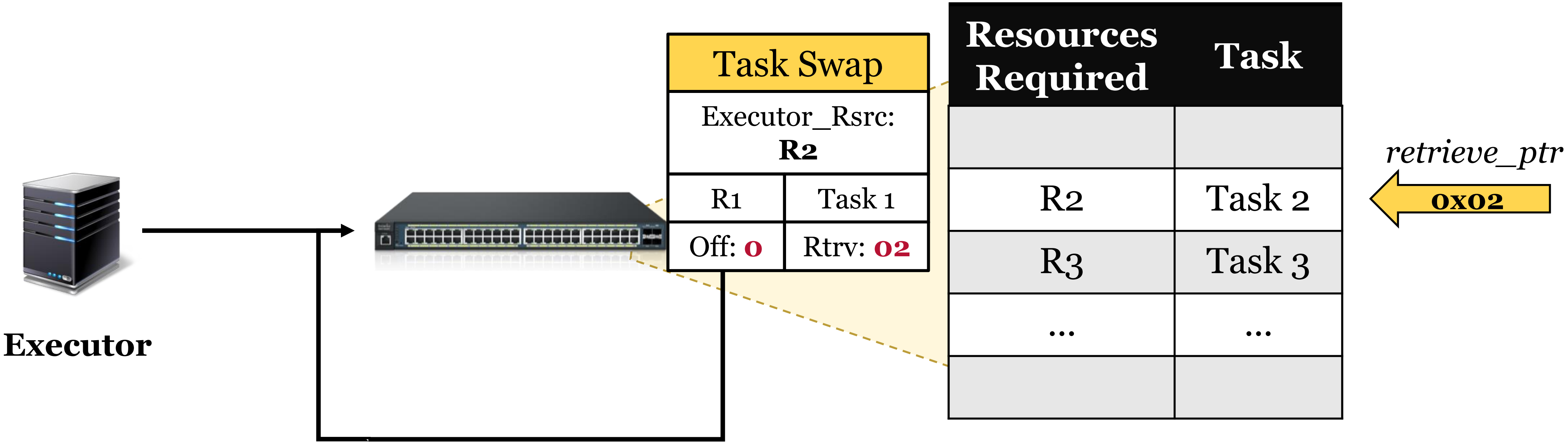
Resources Required	Task
R1	Task 1
R2	Task 2
R3	Task 3
…	…

retrieve_ptr

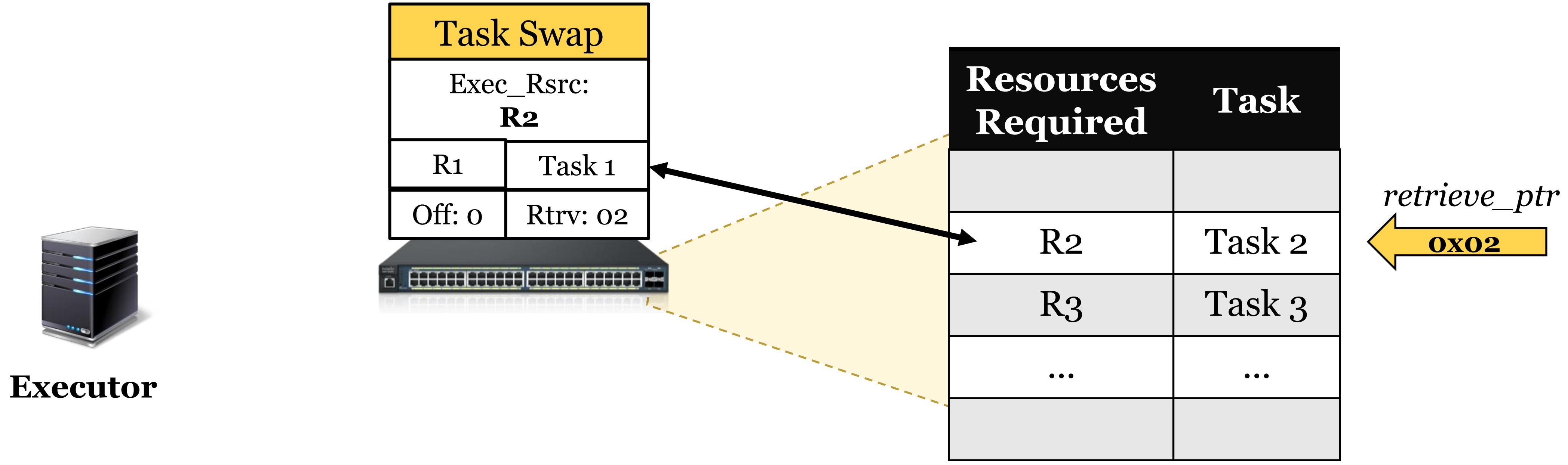
0x01

Task requires resources unavailable on executor

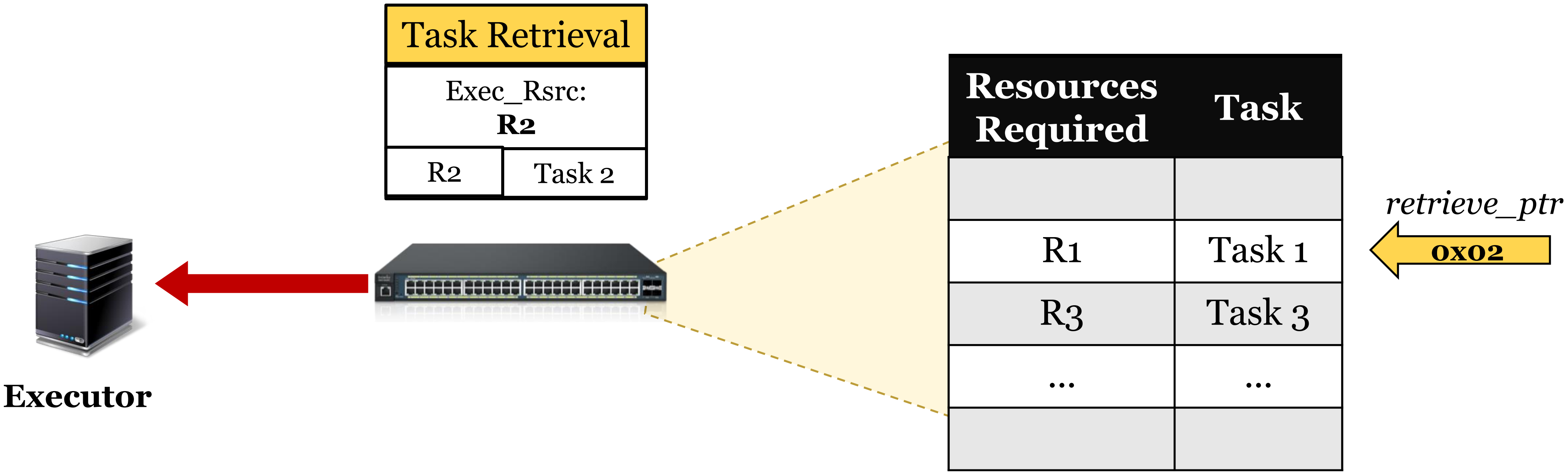
Task Swapping : Resource-Aware Scheduling



Task Swapping : Resource-Aware Scheduling



Task Swapping : Resource-Aware Scheduling



Evaluation Overview

Testbed

- 13 node cluster
  - Intel Xeon 10 core CPU with HT
  - 48GB RAM
  - 100GBps Mellanox NIC
- EdgeCore Wedge 100BF-32x
  - Intel Tofino ASIC

Workloads

- Synthetic Suite
  - Uniform Service Times (100, 250 & 500 µs)
  - Bimodal and Trimodal
  - Exponential – Mean 250 µs
- Google Cluster Traces

UNIVERSITY OF WATERLOO

[9] John Wilkes. Google cluster data 2011 traces. GitHub. https://github.com/google/cluster-data.

Evaluation Overview

Alternatives

- R2P2 [7]
- RackSched [8]
- Sparrow [9]
- Draconis-Socket-Server
- Draconis-DPDK-Server

Metrics

- 99^{th} Percentile Scheduling Latency
- Scheduling Throughput
- Effectiveness of complex scheduling policies

[7] Marios Kogias, George Prekas, Adrien Ghosn, Jonas Fietz, and Edouard Bugnion. R2p2: Making rpcs first-class datacenter citizens. 2019 USENIX Annual Technical Conference (ATC 19), 2019

[8] Hang Zhu, Kostis Kaffes, Zixu Chen, Zhenming Liu, Christos Kozyrakis, Ion Stoica, and Xin Jin. Racksched: A microsecond-scale scheduler for rack-scale computers. the Proceedings of the 14th USENIX Symposium on Operating Systems Design and Implementation, 2020

[9] Kay Ousterhout, Patrick Wendell, Matei Zaharia et al. Sparrow: Distributed low latency scheduling. *SOSP 2013*

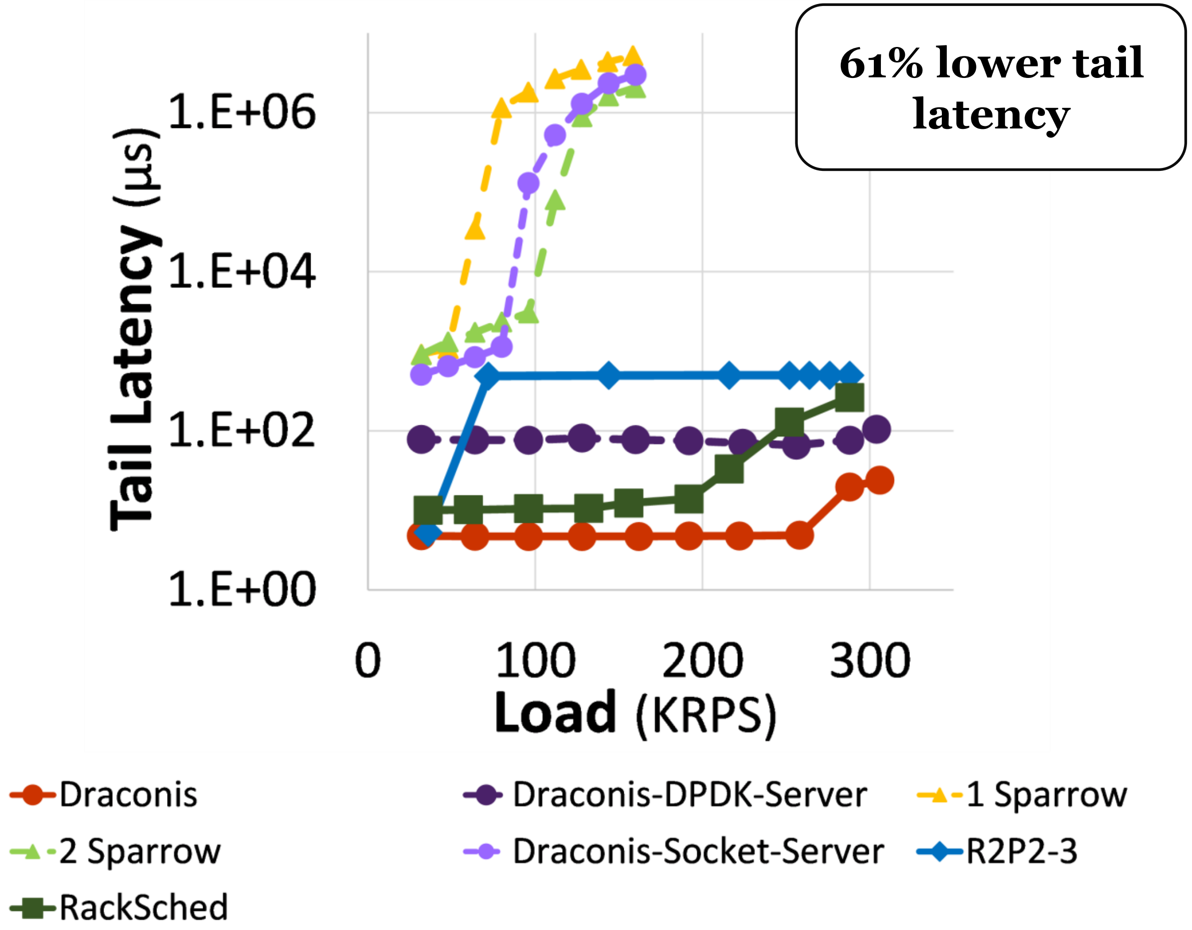
UNIVERSITY OF WATERLOO

Scheduling Latency

Experimental Setup:

- Synthetic workload consisting of 500 µs tasks
- Tail Latency – 99th %ile

Draconis outperforms all other alternatives by at least ***61%***



Concluding Remarks

- Draconis overcomes the shortcomings of modern scheduling paradigms
  - Novel in-network centralized scheduling approach
- Supports complex policies with generic design principles
  - Task swapping and Queue replication

- **Evaluation Highlights:**
  - **61%** lower latencies over network-accelerated scheduling
  - ***52x*** higher scheduling throughput over server-based scheduling

- Code: https://github.com/UWASL/Draconis

UNIVERSITY OF WATERLOO