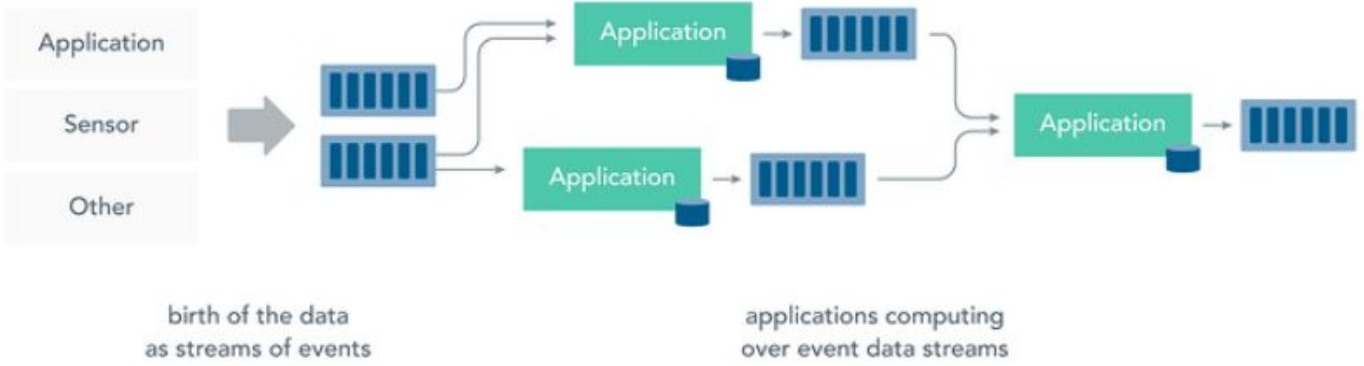
Real-Time Middleware for Cyber-Physical Event Processing

Chao Wang, Christopher Gill, Chenyang Lu
Presented by: Eric Wendt

Stream Processing

Streaming Data is data that is generated continuously by thousands of data sources, which typically send in the data records simultaneously, and in small sizes (order of Kilobytes).

What is Middleware?



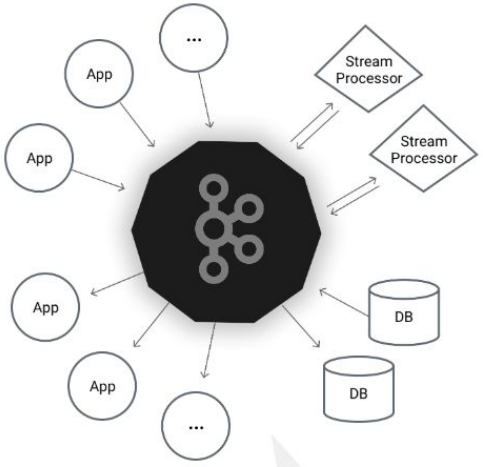
Extended features beyond what OS provides.

Why use a middleware?

- Quick reaction between data aggregation and application response
- Handling large quantities of data
- Processing of data happen in real-time
- Less storage for processing

The past: Apache Kafka

- Load balancing of time-stamped log messages
- Fault tolerance
- Interface for message processing
- NO PRIORITY LEVELS



The past: Apache Flink

- Stream processing framework
- High throughput
- Low Latency
- Does NOT support absolute and relative time consistency

What should a data-streaming engine offer?

- Configurable processing operations
- Process prioritization and sharing
- Efficient concurrent processing
- Enforcement of temporal validity and shedding

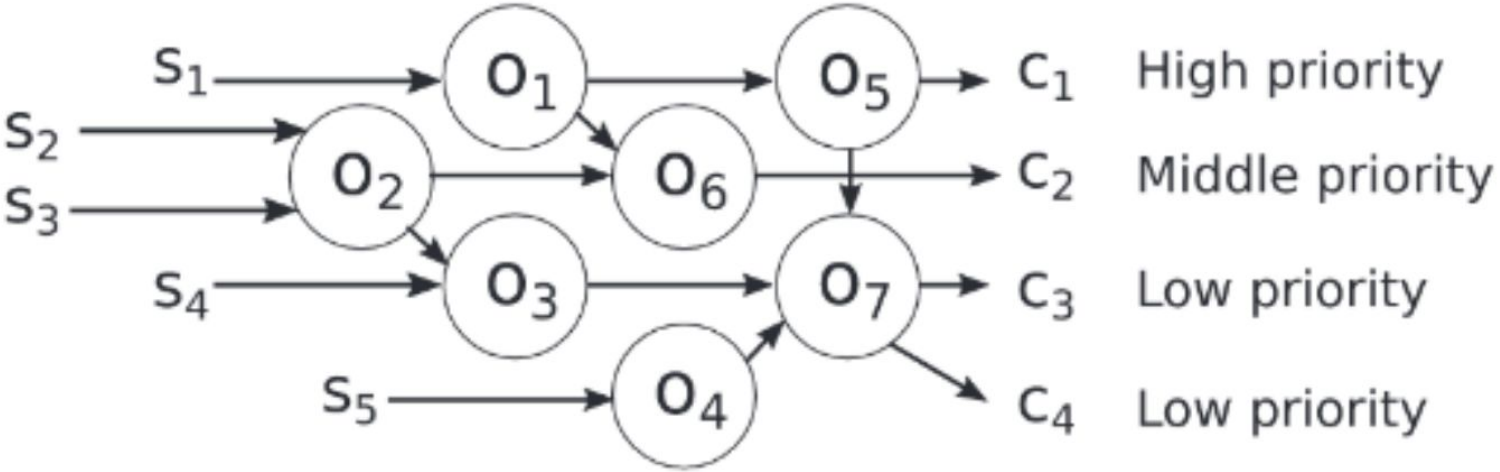
Enter CPEP!

Cyber-Physical Event Processing

Challenges CPEP addresses

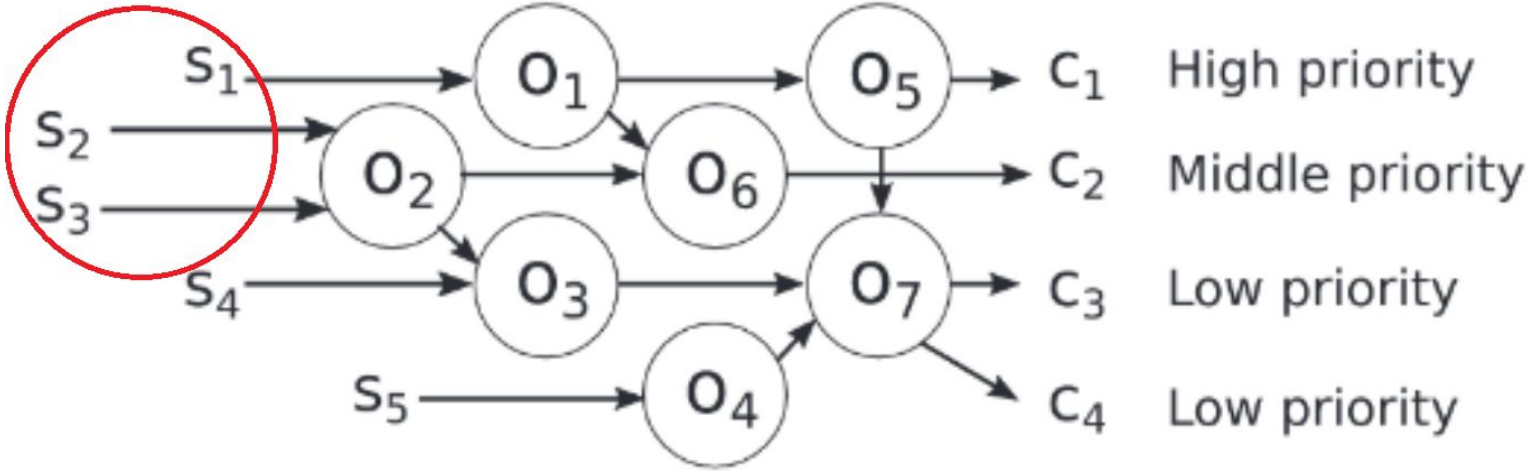
- Cut latency down
- Enforce absolute and relative consistency
- Increase CPU utilization/efficiency
- Let users prioritize!!!

Event Processing



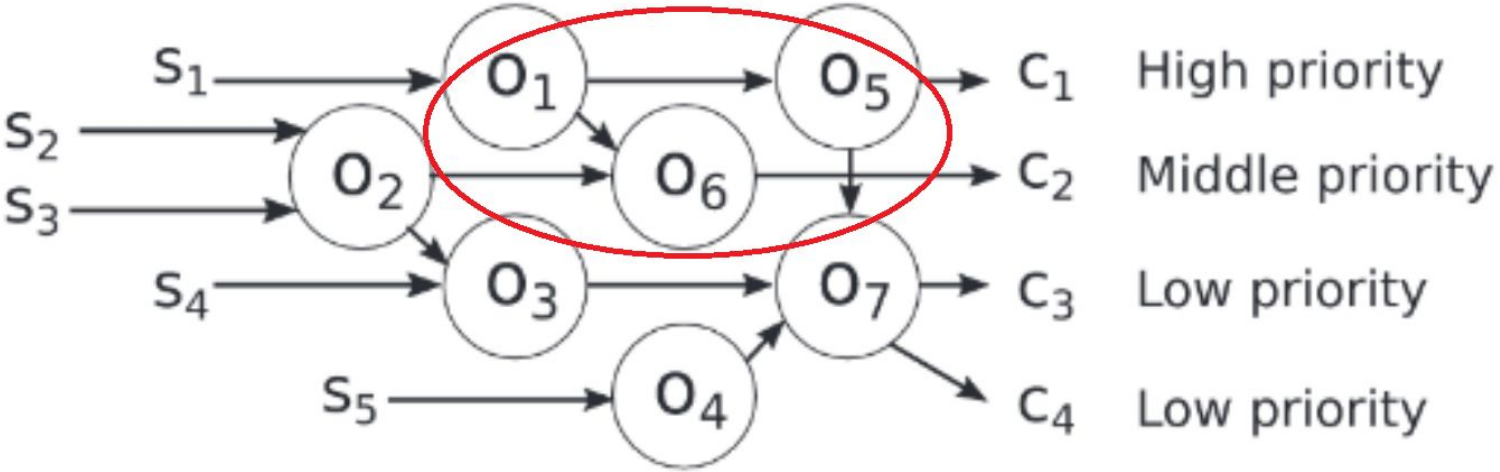
Event Processing

Supplier events: processes that gather information from sensors/physical components, pushed to “operators”



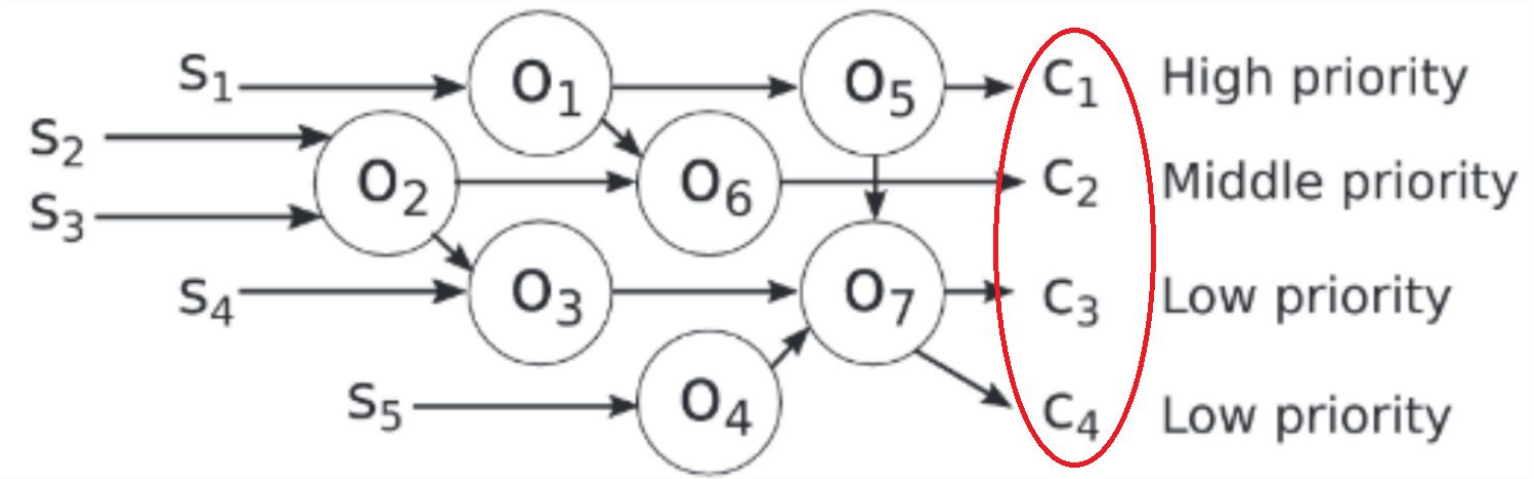
Event Processing

Internal Events: events pushed from one operator to another

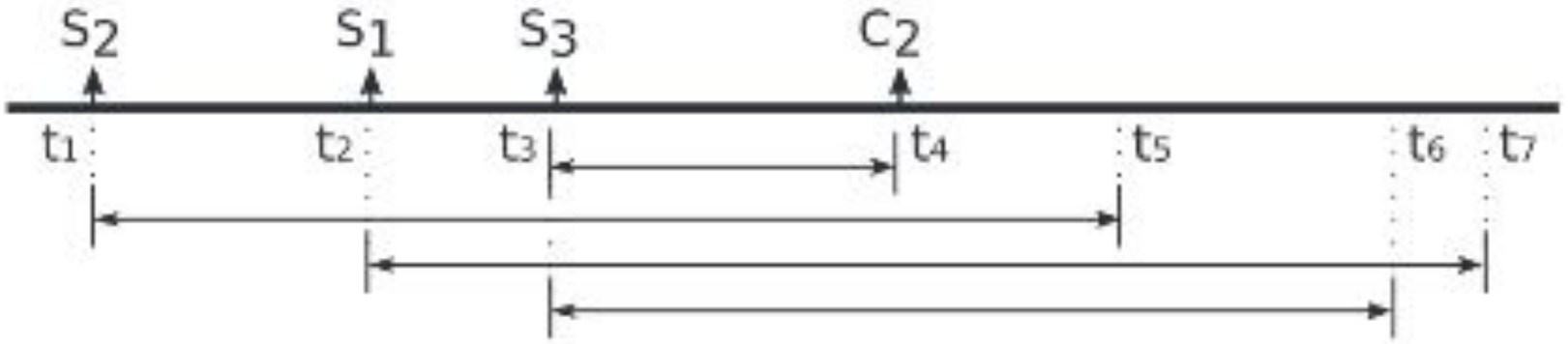


Event Processing

Consumer Events: events pushed from one operator to another



Absolute Time Consistency of Events



Absolute Time Consistency of Events

- Defined as the maximum overlap of an events supplier events

Event: e_i

Begin time: $t_b(e_i)$

End time: $t_e(e_i)$

Absolute Time Consistency of Events

- Defined as the maximum overlap of an events supplier events

Event: e_i

Begin time: $t_b(e_i)$

End time: $t_e(e_i)$

$$\mathrm{abs}(e_i) = [t_b(e_i), t_e(e_i))$$

Absolute Time Consistency of Events

- Defined as the maximum overlap of an events supplier events

Event: e_i

Begin time: $t_b(e_i)$

End time: $t_e(e_i)$

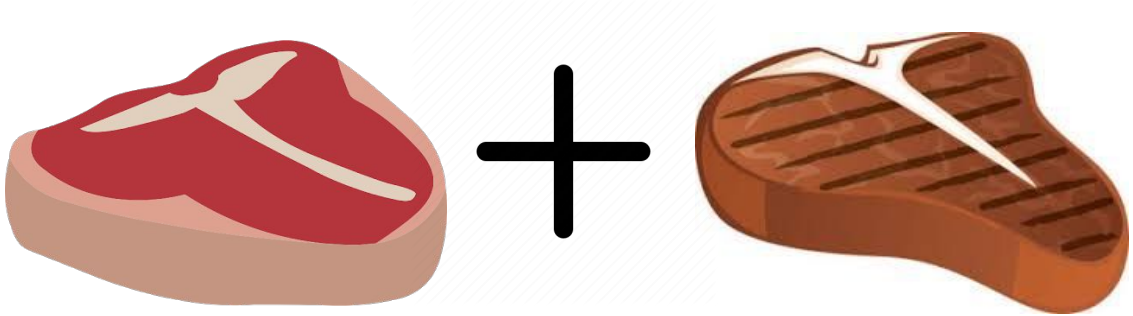
$$\text{abs}(e_i) = [t_b(e_i), t_e(e_i))$$

$$t_b(e_i) = \max\{t_b(u) \mid u \in I_{e_i}\}$$

$$t_e(e_i) = \min\{t_e(u) \mid u \in I_{e_i}\}$$

Relative Time Consistency

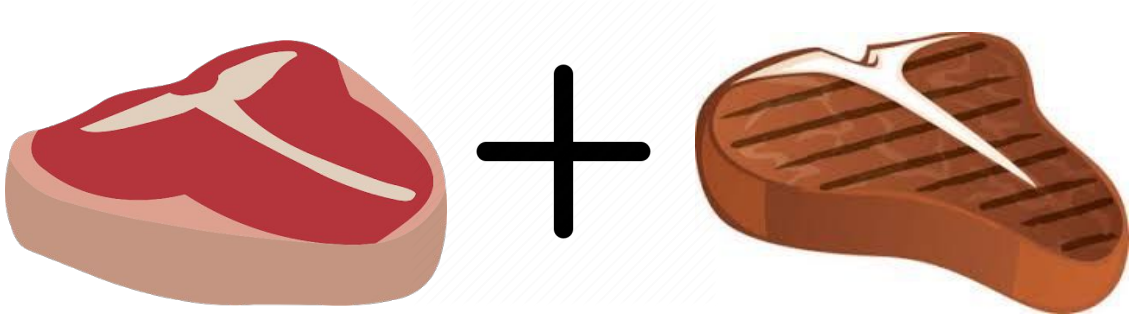
Dependent events are operated within a maximum time-frame Time to window: 20 min



Relative Time Consistency

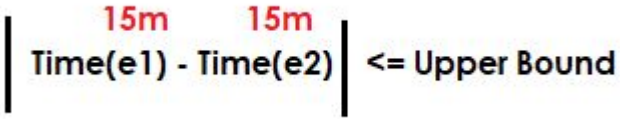
Dependent events are operated within a maximum time-frame Time to window: 20 min

Time to cook EACH steak: 15 min

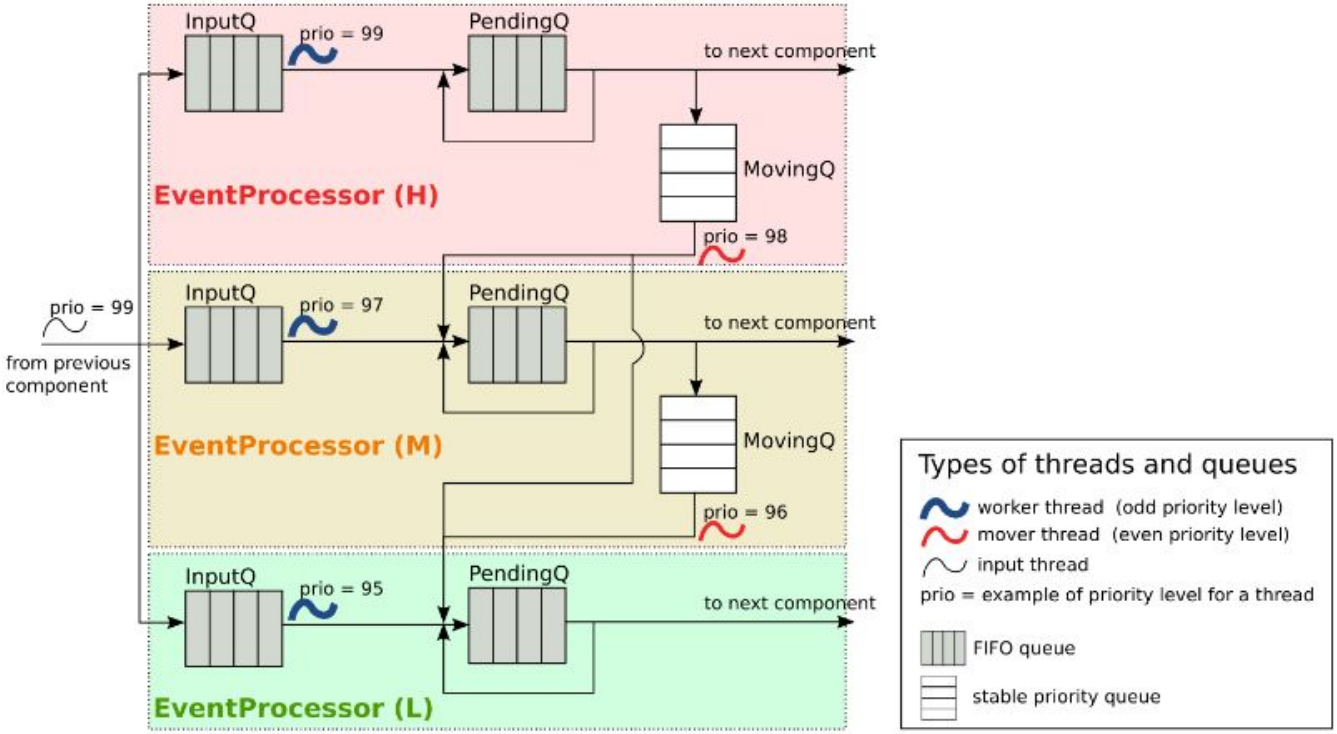


Relative Time Consistency

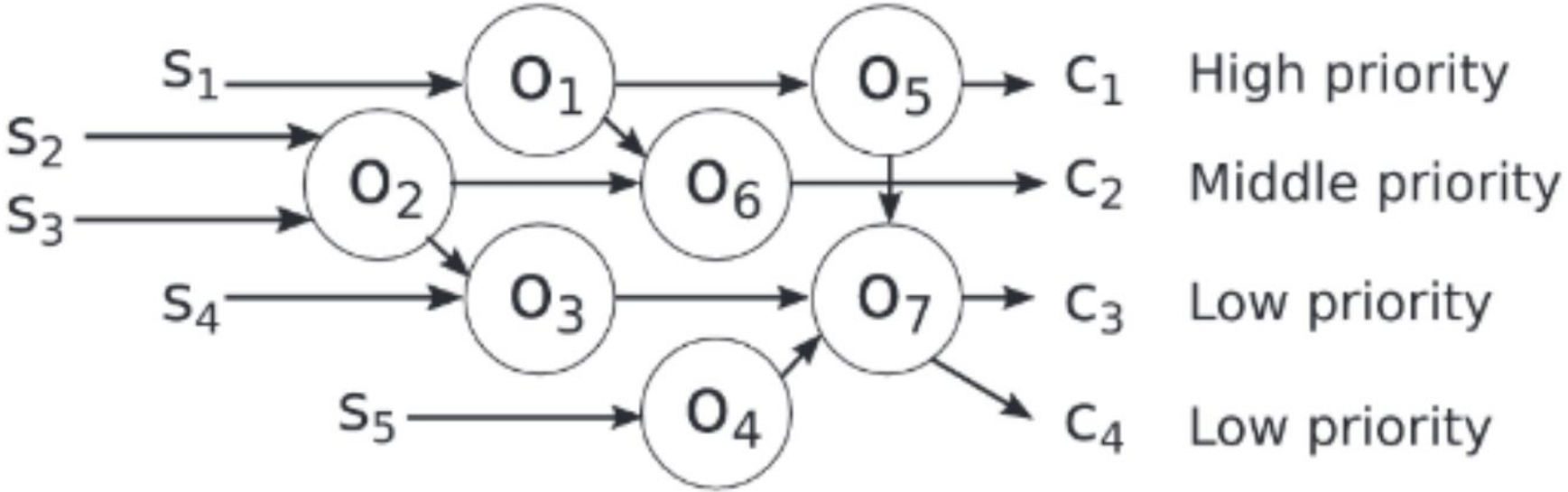
Maximum time between two events assigned to an operator



Event flow - data representation



Event Flow - visual representation

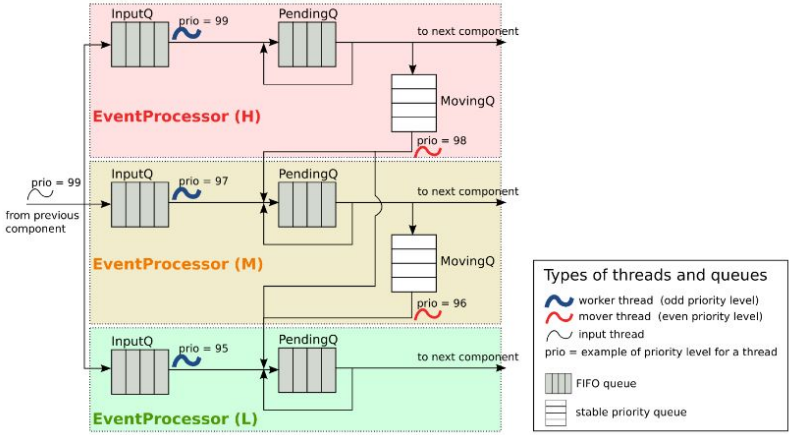


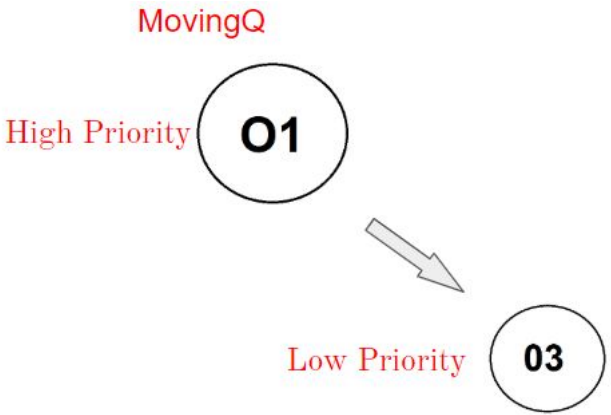
Event Sharing

- One Event Processor(EP) per priority
- Each EP has one thread for current events and one for moving to another priority

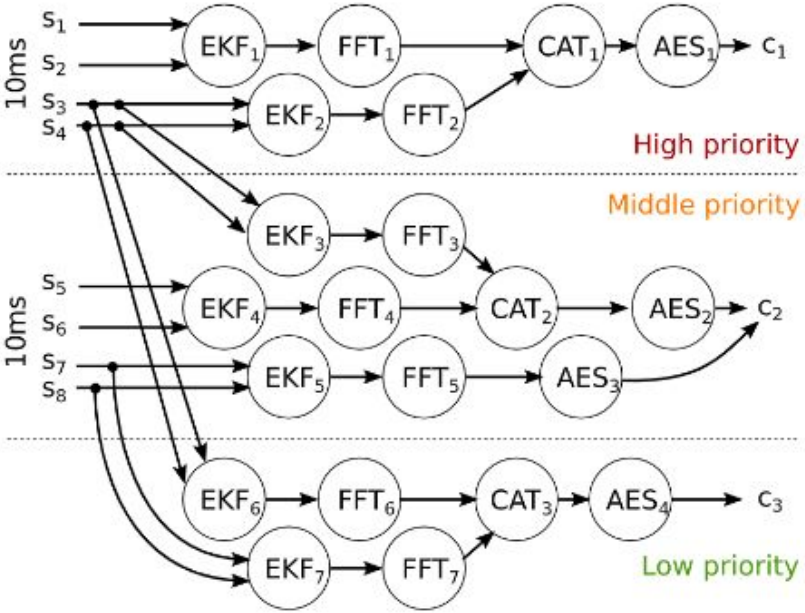
- Events can preempt Operators at same priority
- Processing threads will preempt lower priority EPs if needed

- InputQ holds supplier events
- MovingQ holds events for cross priority sharing



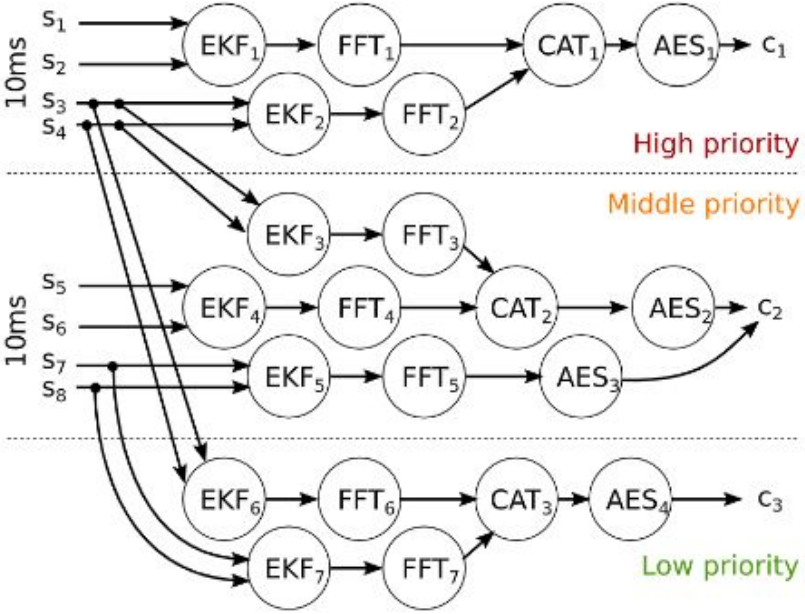


Event Sharing vs. NOT

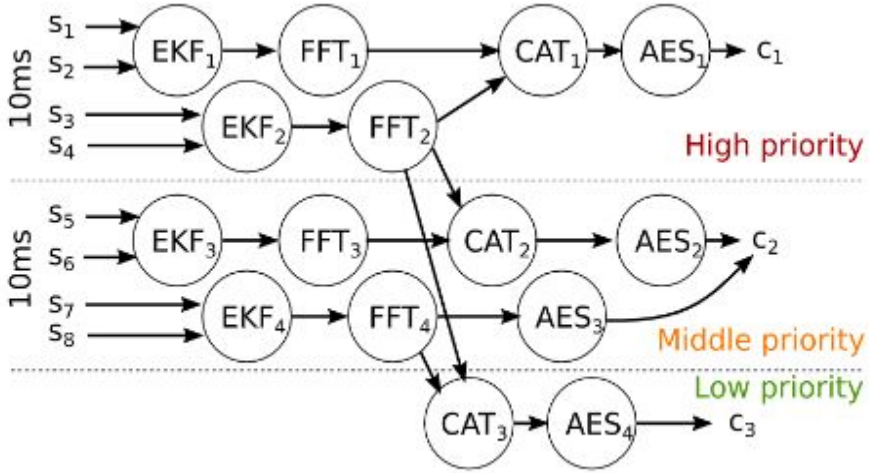


(b) Without sharing operators.

Event Sharing vs. NOT



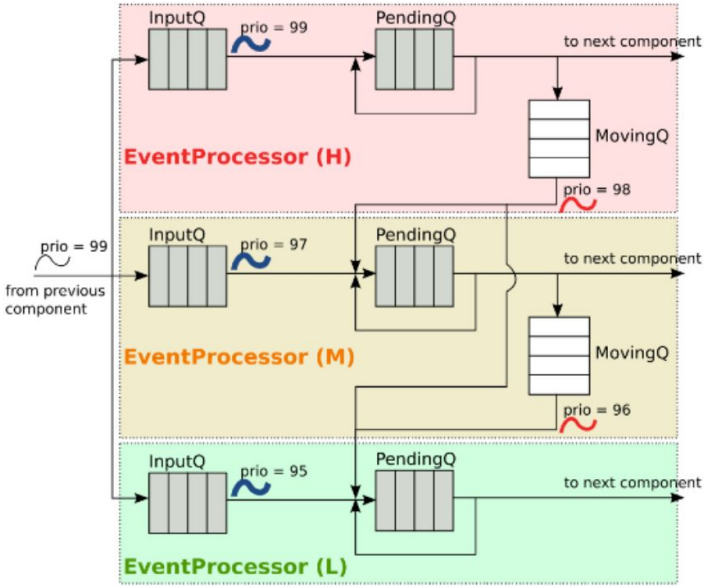
(b) Without sharing operators.



(a) With sharing operators.

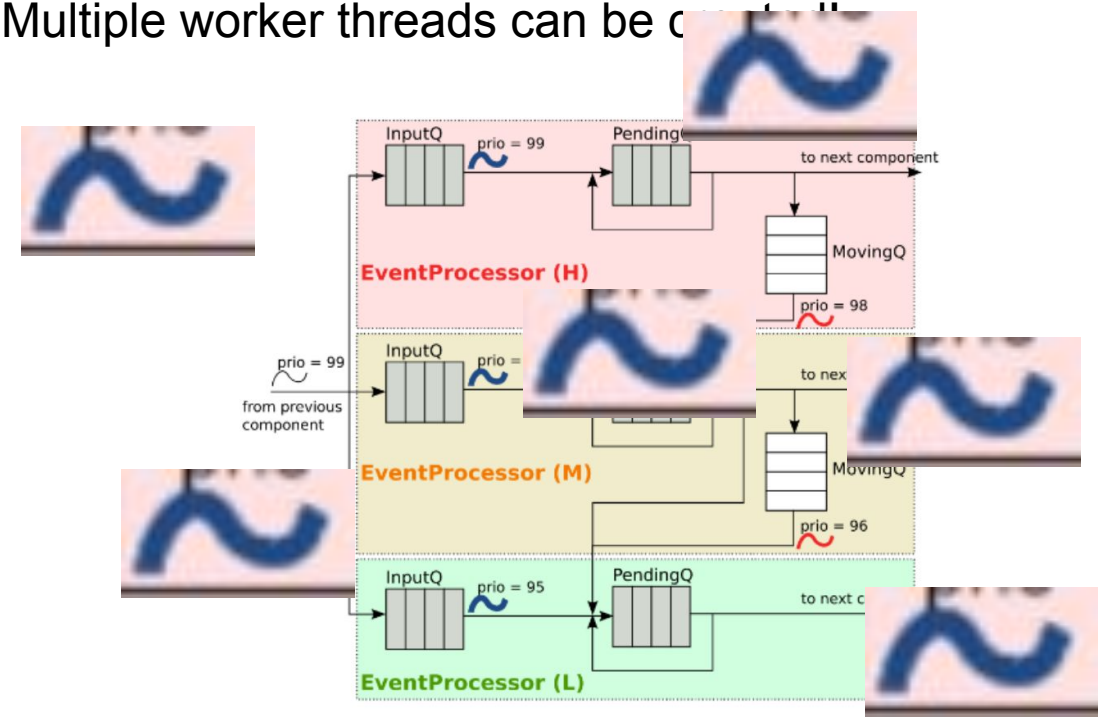
Concurrent Processing and Replacement

Multiple worker threads can be created!



Concurrent Processing and Replacement

Multiple worker threads can be c



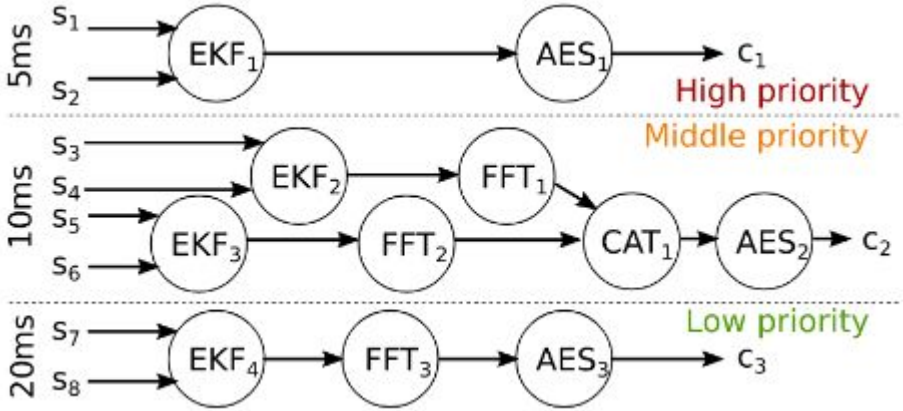
Time Consistency Enforcement and Shedding

- Time Consistency Validation:
  - If an absolute or relative time constraint is broken by any of the operations, all current events are “shedded” in favor of the upcoming events.
  - Specified by user!

How is this implemented?

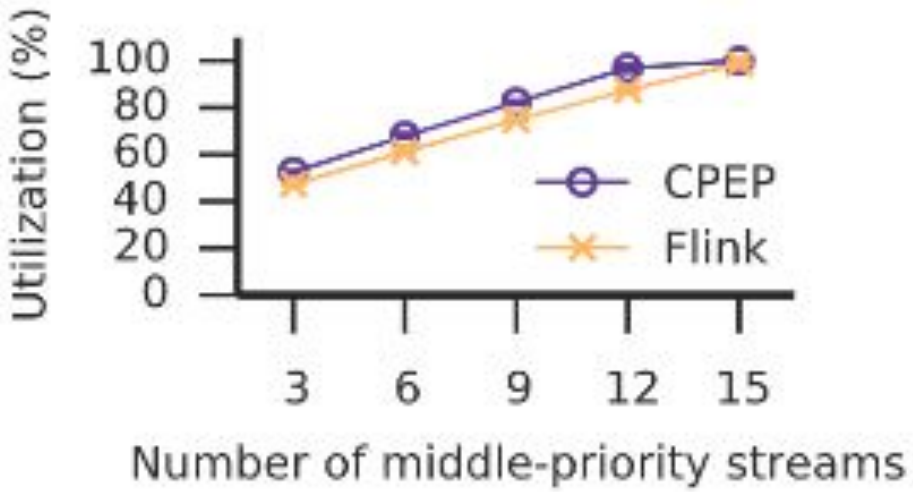
- MovingQ is c++ priority queue
- PendingQ is c++ FIFO queue
- InputQ is an array ring buffer
  - All of these data structures are read/write lock protected

- Graph of event processing is an array of 464bytes structs



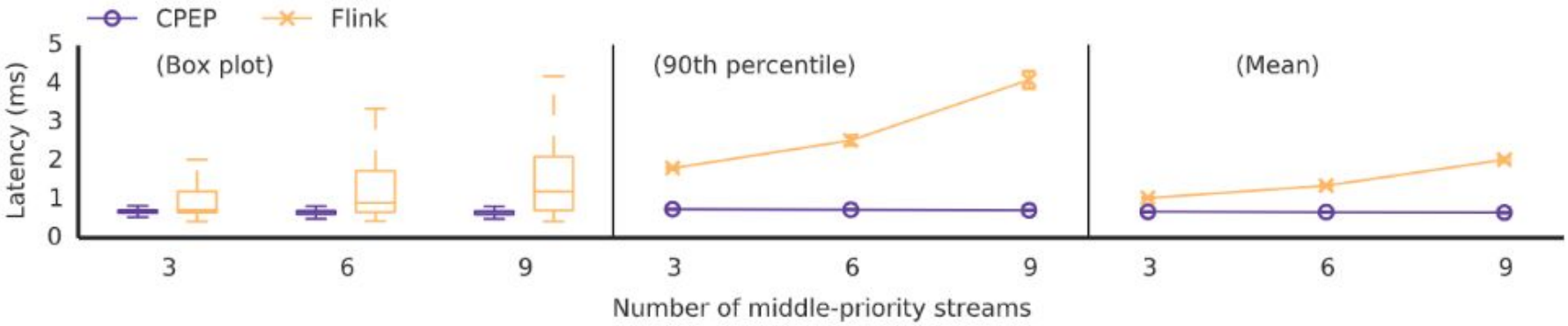
Comparison Tests

In comparison with Apache Flink -CPU usage



(a) CPU utilization.

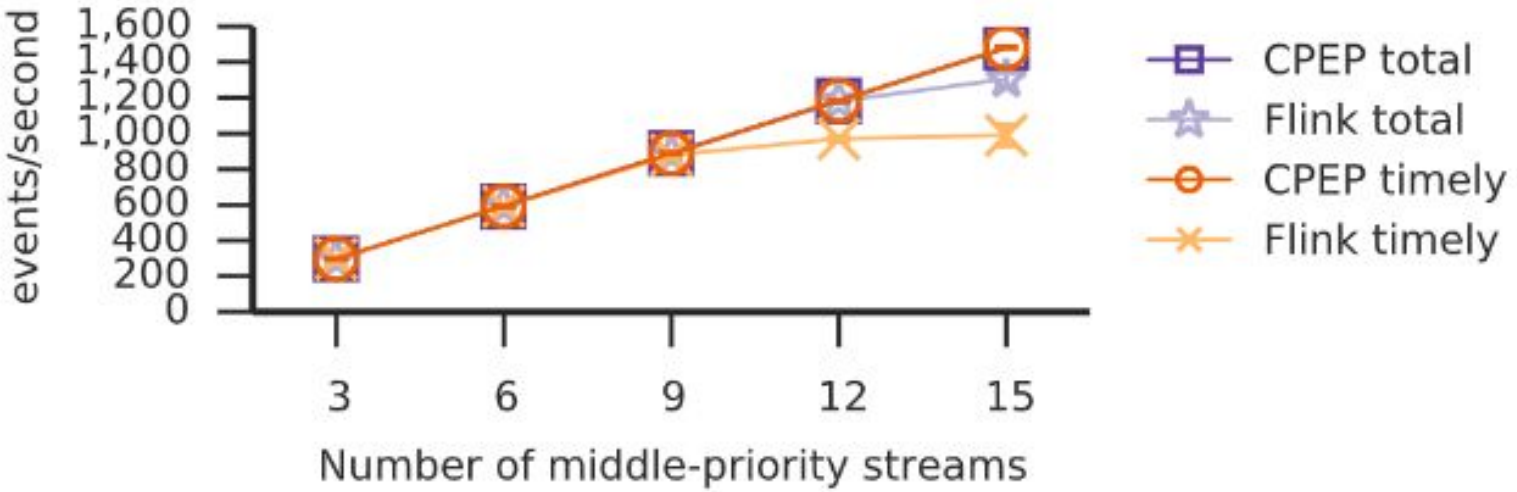
In comparison with Apache Flink -Latency



(b) Moderate workload (CPU utilization = 40%–80%).

Beats Flink in Moderate and Heavy workloads!

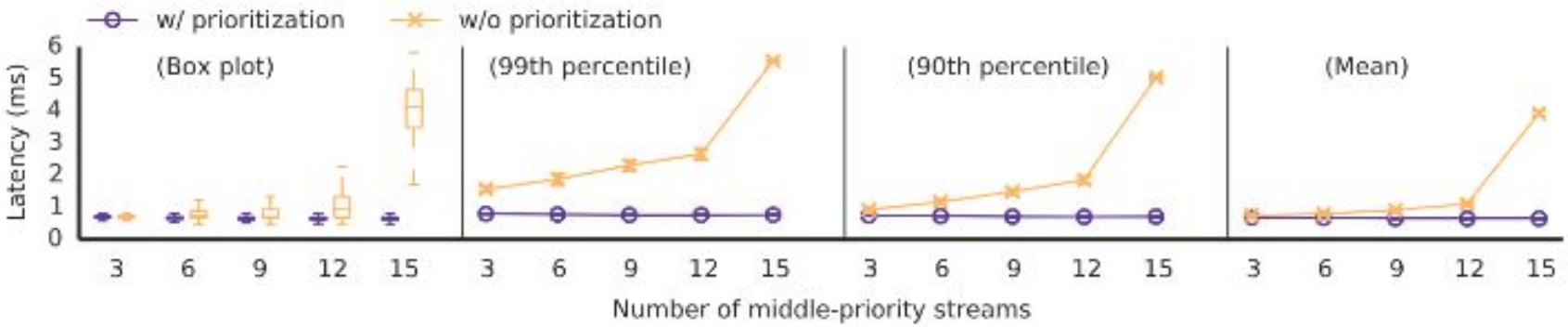
In comparison with Apache Flink -Throughput



(b) Middle priority.

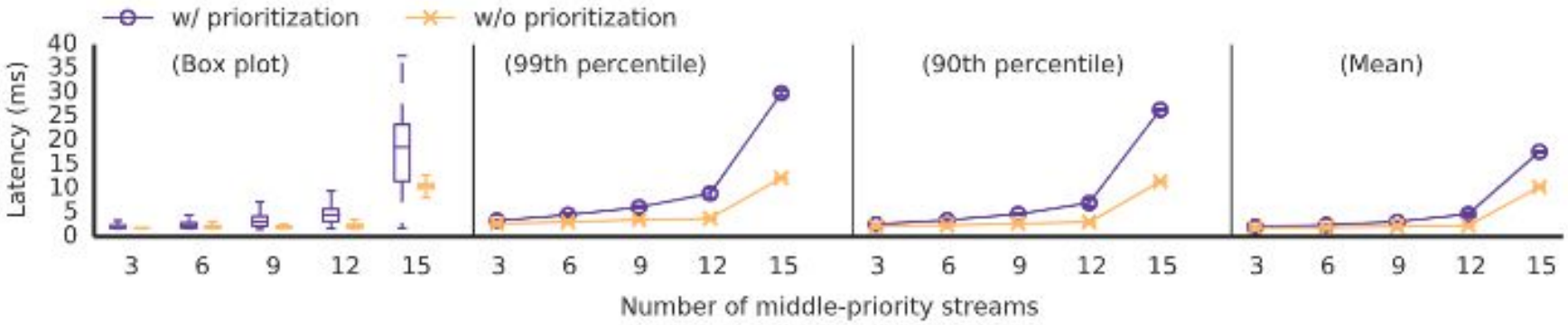
CPEP
Self-Tests

Prioritization Latency -High Priority



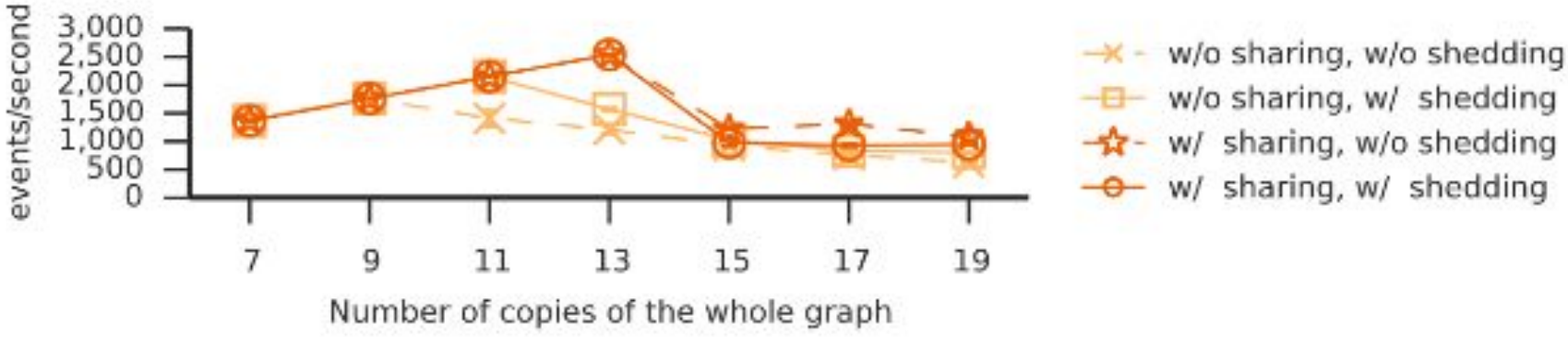
(b) High priority.

Prioritization Latency -Low Priority



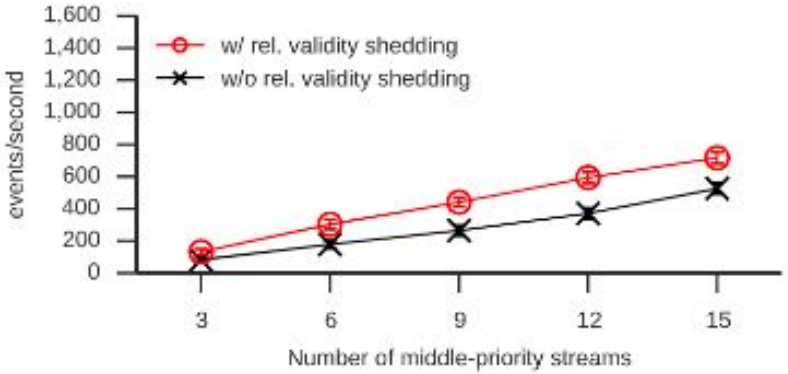
(d) Low priority.

Absolute Shedding Throughput

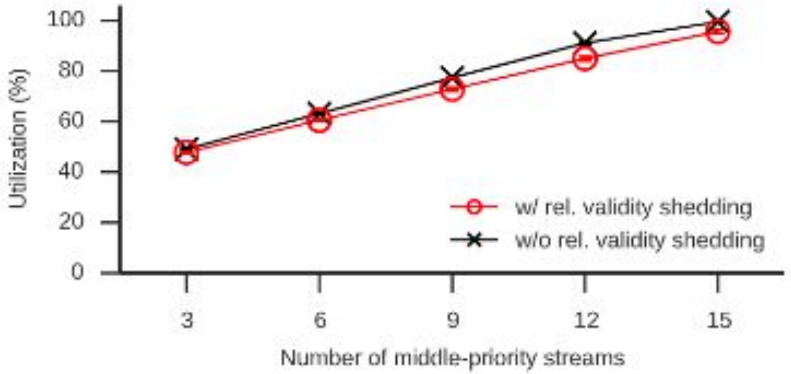


(c) Middle priority.

Effects of Relative validity shedding -Throughput

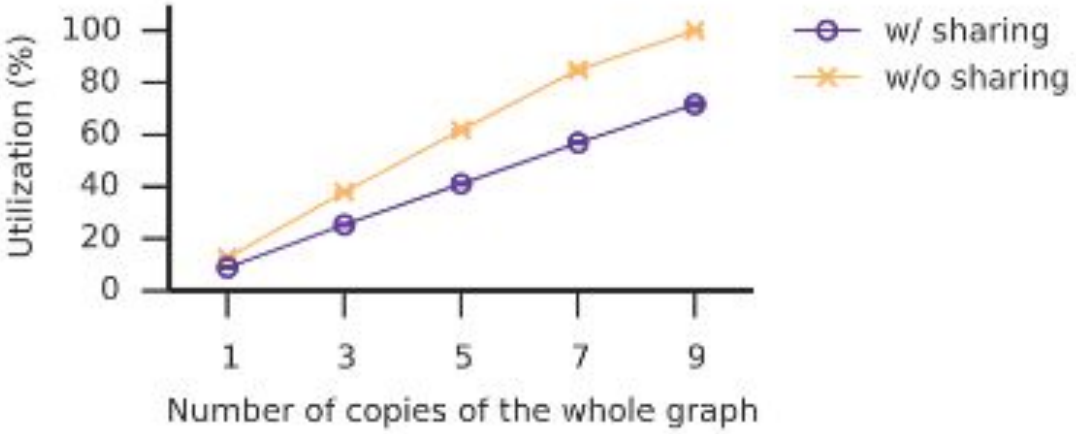


(a) Middle-priority throughput.



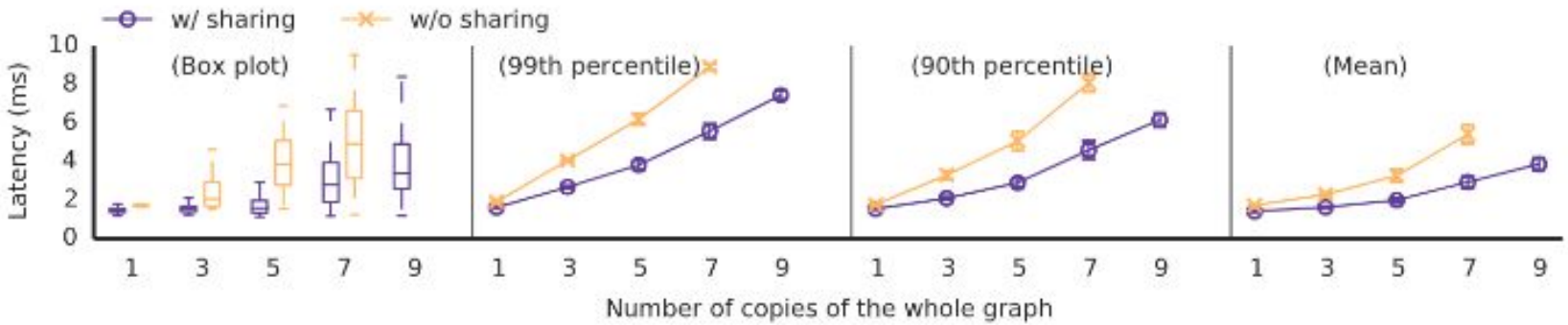
(b) CPU utilization.

Sharing events vs. NOT sharing -Utilization



(a) CPU utilization.

Sharing events vs. NOT sharing CONTINUED



(d) Low priority.

Greatest strengths:

- High Customizability
- Time violation detection
- Shared events across priority levels
- Higher throughput at mid-priority
- Lower latency

Weakness of the paper

- No overarching example to communicate pipeline
- Vague and confusing distinctions made between events and operators
- Event sharing between priorities unrealistically complicated to visualize
- Experiments were horribly laid out, very in-depth, but labeled poorly

Conclusion

- CPEP offers unique advantages over competitors.
- Needs more testing to be deemed a viable competitor
- Paper is certainly confusing, but core idea is brilliant and the technology is undoubtedly promising.

Questions

- Can this system work outside of TAO?
- Is comparing CPEP to Apache Flink a fair comparison?
- Security Flaws?
- How do Operators generate events?
- Disadvantages to using Abs/Rel constraints?
- Real World examples?
- How is concurrency ensured without trampling over shared memory?

Questions

- How much context switching is occurring by not assigning Operators to CPU cores?
- Why was Apache Flink the comparison of choice?
- What happens when two high priority Supplier Events come in rapid succession?
- What is a tuple?