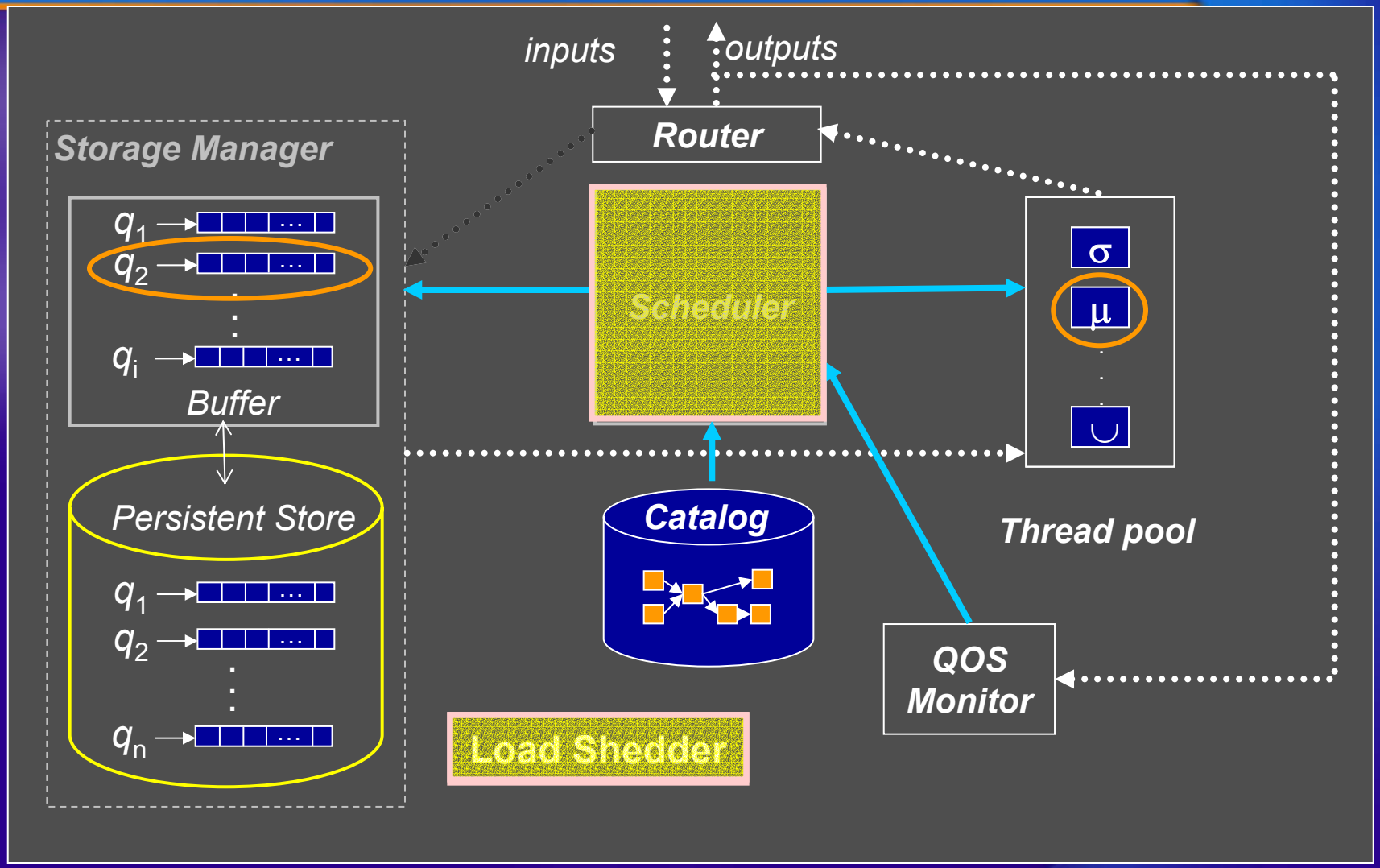
(Over)Load Management in Stream Processing Engines

ENGINE PANEL
SWIM MEETING

Uğur Çetintemel
Brown University

Aurora Run-Time Engine



Control is good!

- How to handle "load"?
  Need fine grained resource control -> new way of performance optimization
  - Execution models
  - Scheduling issues
- How to handle "overload"?
  Need graceful degradation in performance
  - Load shedding

What is the right execution model?

- Thread-driven execution
  - Use a thread per query, per operator, etc.
  - Easy to program
  - Not scalable (scheduling overhead, lock contention, TLB misses, destroys cache locality)
  - At the mercy of OS
- Tuple-driven execution
  - Small number of threads (typically one per CPU)
  - Loop continuously, processing events from queues
    much like an FSM
  - More control, better throughput (robust to load)
  - Challenge: Scheduling
    when, what, and how many to process?

Fine-Grained Scheduling

- Multi-level scheduling plans
  - Inter-query scheduling
    "which query to schedule?"
    (sharing makes things more interesting)
  - Intra-query scheduling
    "in which order to schedule the individual ops?"
- Batching control
  - How many tuples to process within each invocation of a box (**train scheduling**)
  - How many boxes to execute within a single scheduling decision (**superbox scheduling**)
  - Knob to trade off throughput and latency
- State Monitoring (number of tuples, latencies, etc.)
  - Incrementally and approximately

Overload Management- Load Shedding

- Load shedding
  - "Drop excess load (tuples) from the system"
    - where to place drop boxes?
    - how much to drop?
- Driving factors;
  - Priority/QoS; semantic values
- Efficiency
  - Dynamic approaches might be high-overhead
  - materialize incremental shedding plans; instantiate the "right" one on-the-fly