Class-based Continuous Query Scheduling for Data Streams

Lory Al Moakar[1], Thao N. Pham[1], Panayiotis Neophytou[1],
Panos K. Chrysanthis[1], Alexandros Labrinidis[1], Mohamed Sharaf[2]

[1] University of Pittsburgh

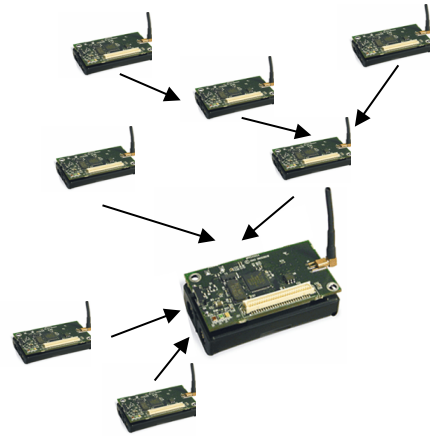
[2] University of Toronto

Did you say STREAMS?

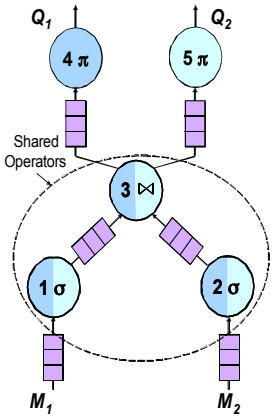
- Yes! ☺

Data Acquisition

Web Data Management



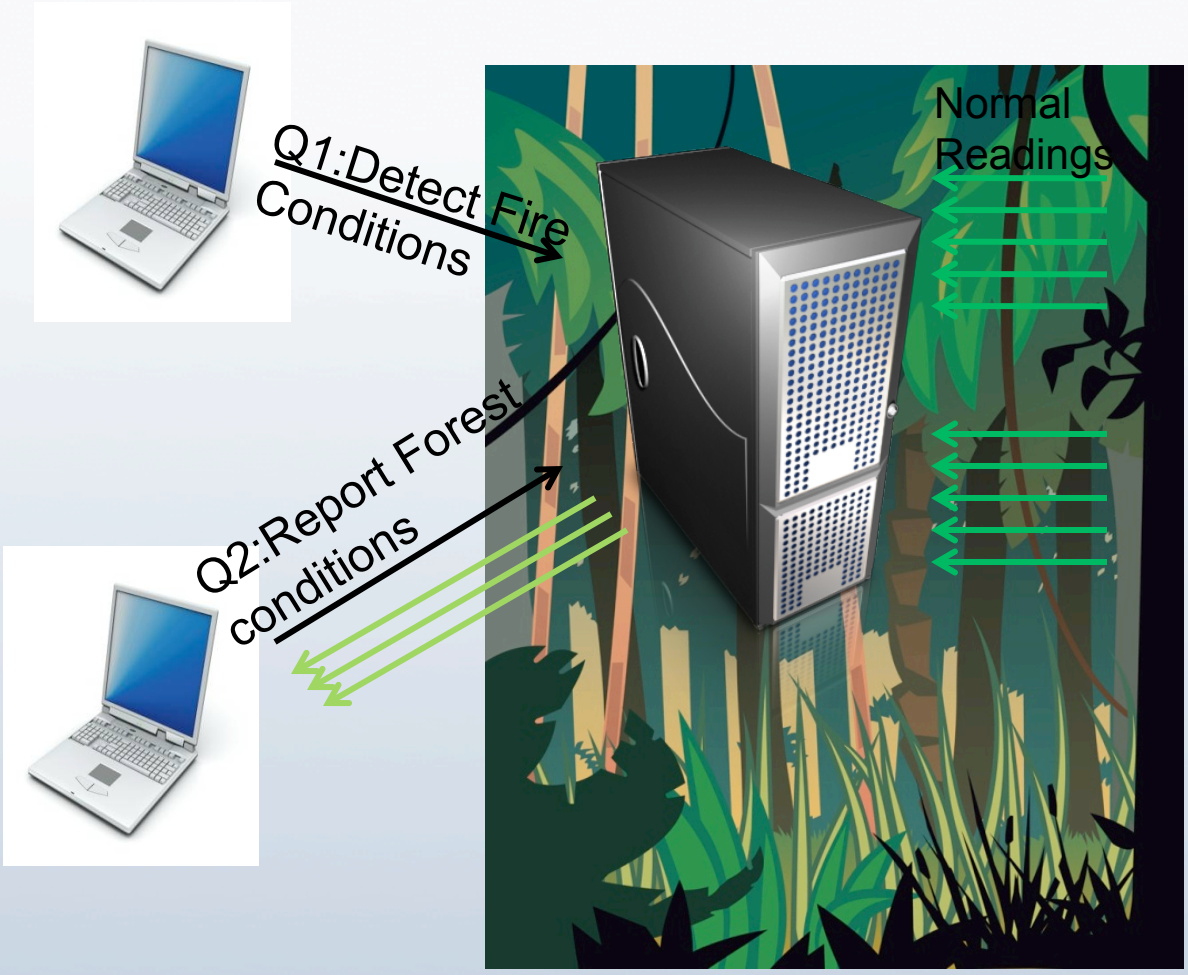
Data Flow



Data Stream Processing

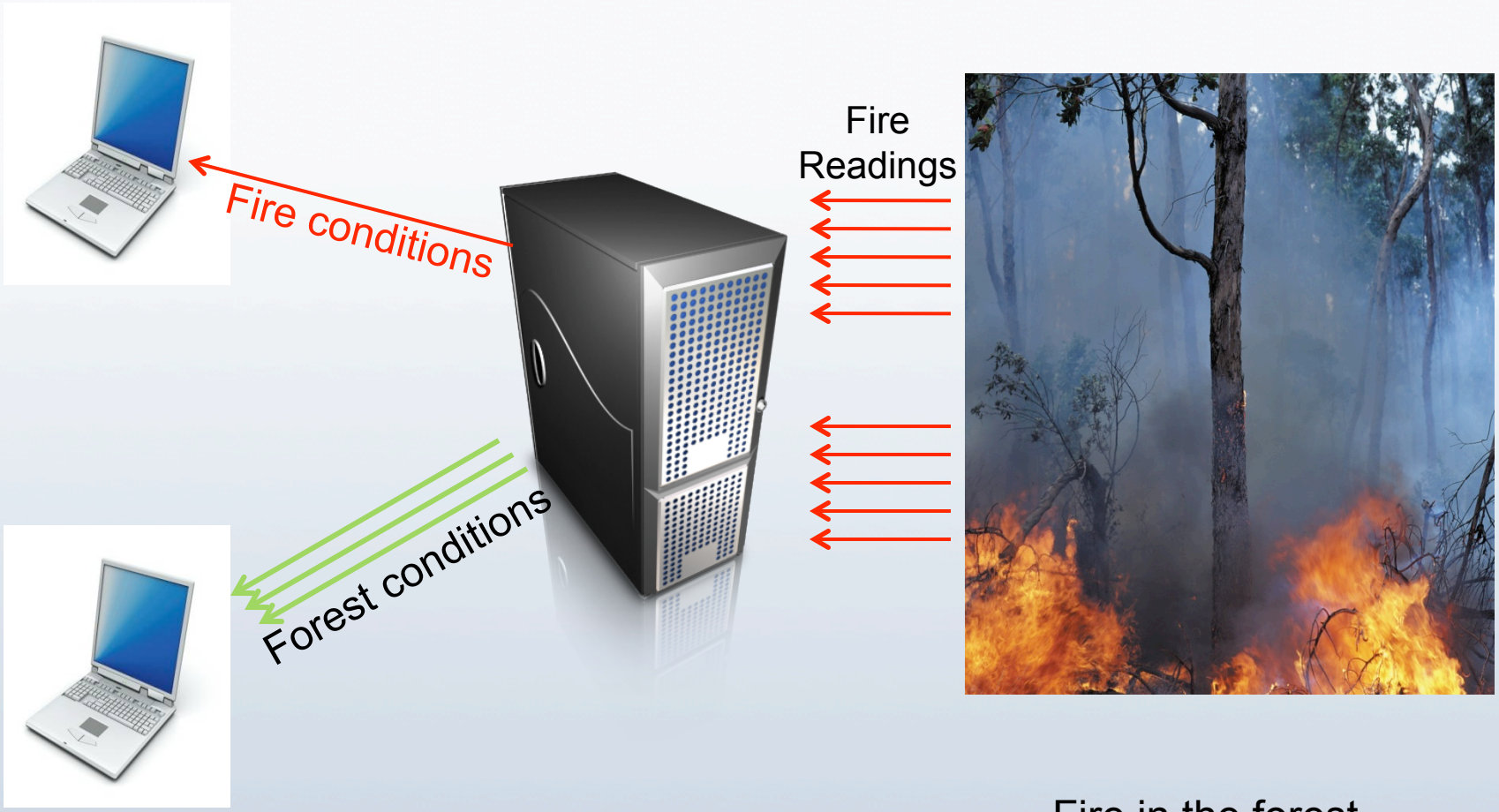
Data Dissemination

Motivation



Forest under normal conditions

Motivation

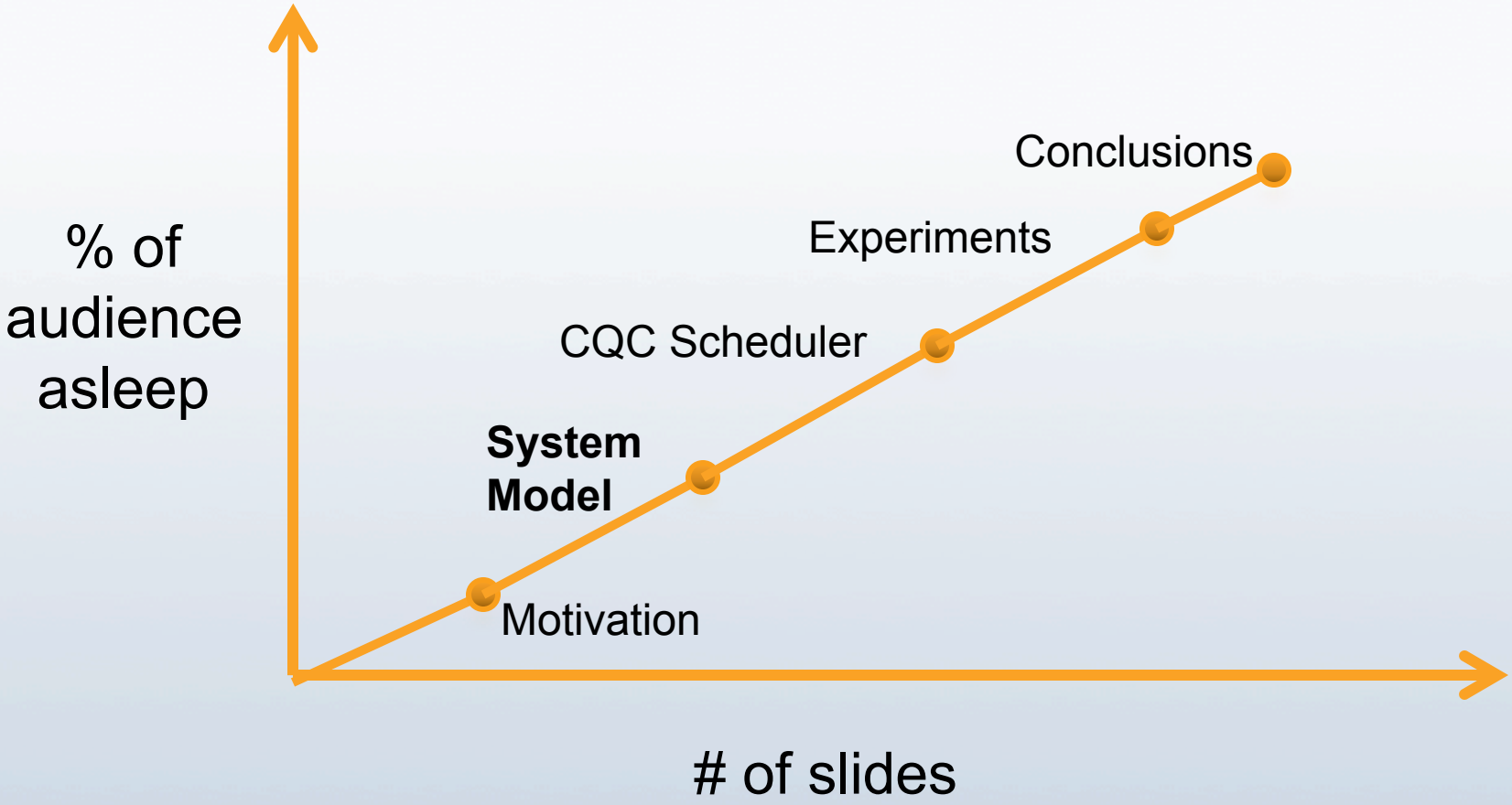


Fire in the forest

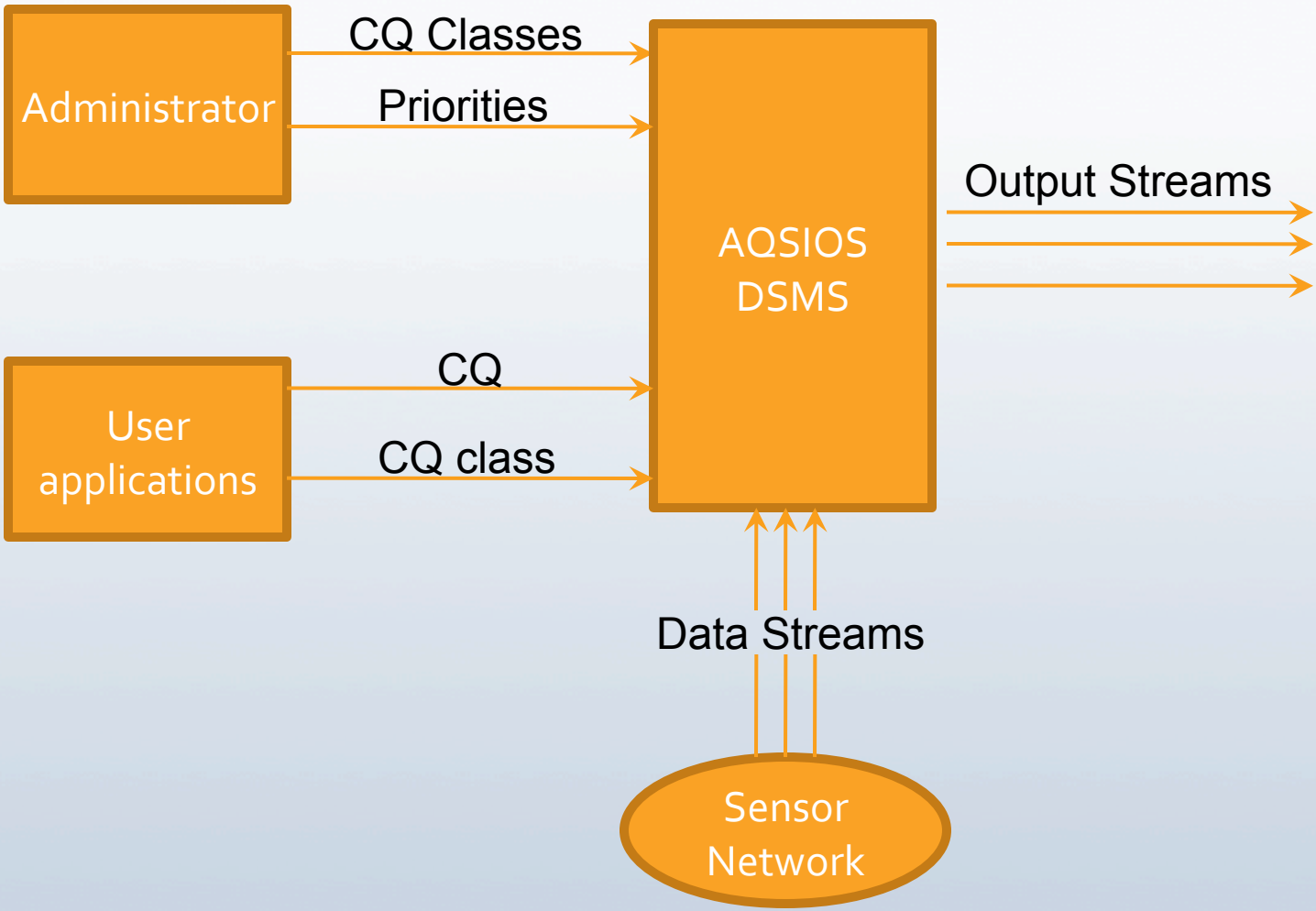
Problem Statement

- How do you schedule continuous queries with different priorities such that
  - the highest priority ones always have preference (lowest average response time)
  - while the lowest priority ones never starve when the system is loaded?
- **Our contribution**:
  - Best-effort fair scheduler.

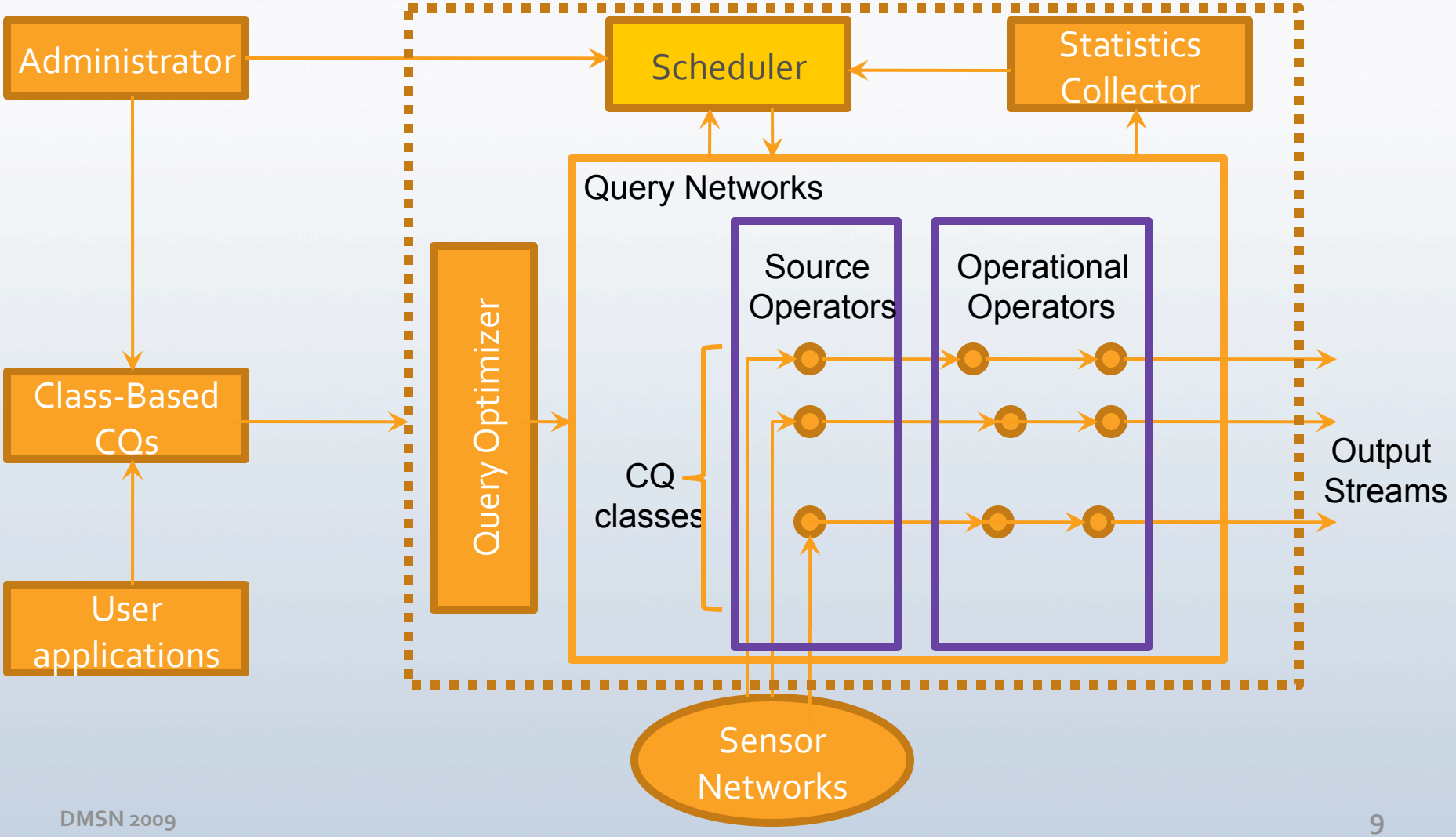
Roadmap


% of audience asleep
Conclusions
Experiments
CQC Scheduler
System Model
Motivation
# of slides

System Model



System Model



Assumptions

- No restrictions on the query operators.
- CQs belong to query classes.
- No shared operators across query classes (including source operators).
- Query classes are not associated with a specified QoS.

RR Scheduler

- **Round Robin Scheduler**: Inherited from STREAM.
  - Operators are scheduled in a round robin fashion.
  - Each operator executes all the tuples in its input queue.

  → does not optimize for more critical queries

HR Scheduler

- Optimizes the **average response time**.
- Priority of an operator *i* is:

$$P_i = \overline{S}_i^x \Big/ \overline{C}_i^x$$

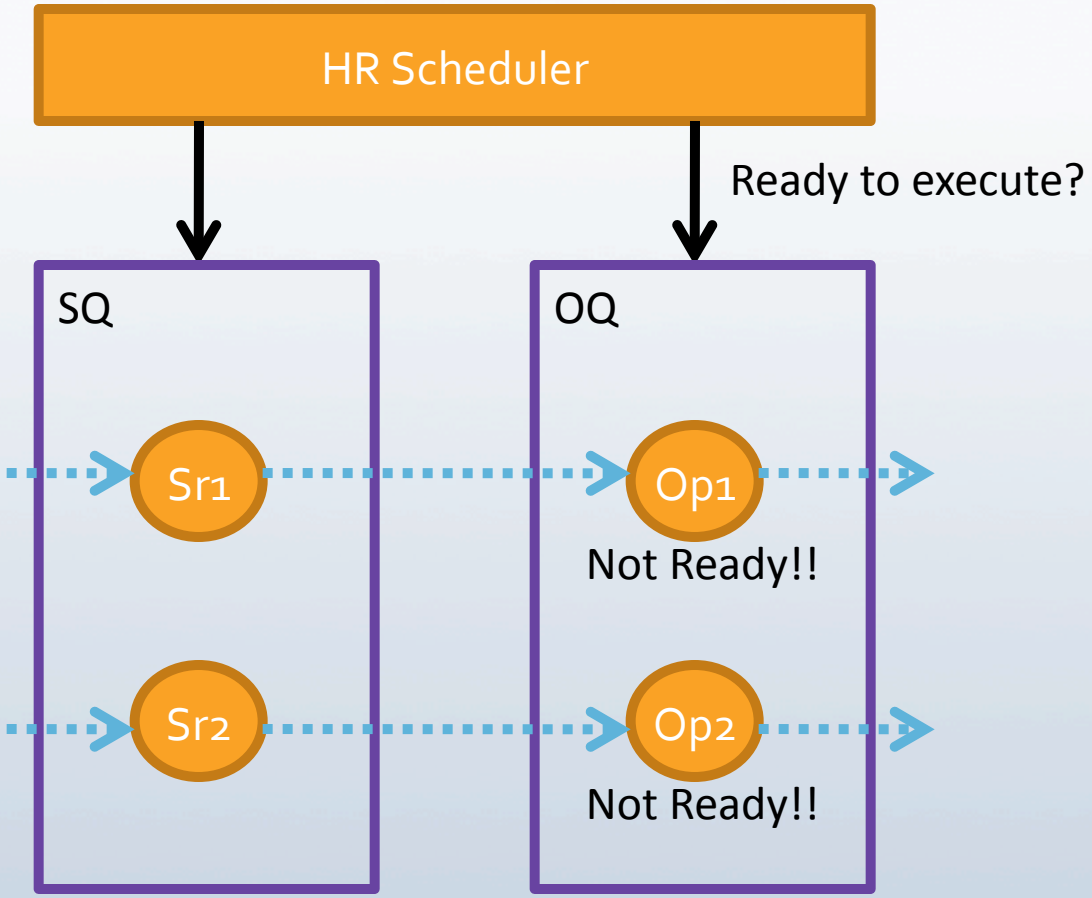
- Example:
  - 2 operators i and j
  - Oi has an output rate of 0.5 tuples/msec and Oj has an output rate of 0.75 tuples/msec
  - Oi will execute only if Oj has no tuples to process.

→ does not optimize for critical queries but optimizes the average response time.

HR scheduler as implemented under AQSIOS

- Under AQSIOS, the source operators need to be scheduled as part of the scheduler.
- HR maintains two queues:
  - **SQ**: a queue for the **source operators**
    - scheduled in round robin order.
    - SQ is scheduled only when all the operators in OQ are **not** ready to execute
    - When SQ is scheduled, all the source operators execute.
  - **OQ**: a queue for the rest of the **operators**
    - Operational operators and output operators
    - Scheduled according to HR.

Example of HR Scheduling in AQSIOS



Obvious extensions to support CQ classes (1)

- **WRR (Weighted Round Robin):**
  - extend Round Robin scheduler by assigning each operator a time-slice proportional to its CQ class priority.

  → does not optimize for the response time of the queries.

Obvious extensions to support CQ classes (2)

- **WHR (Weighted HR):**
  - extend HR scheduler by setting the priority of an operator *i* :

$$P_i = P_i^{HR} \times P_{Qclass(i)}$$

→ Starvation possibility

→ No guarantee that higher priority class operators will be scheduled first.

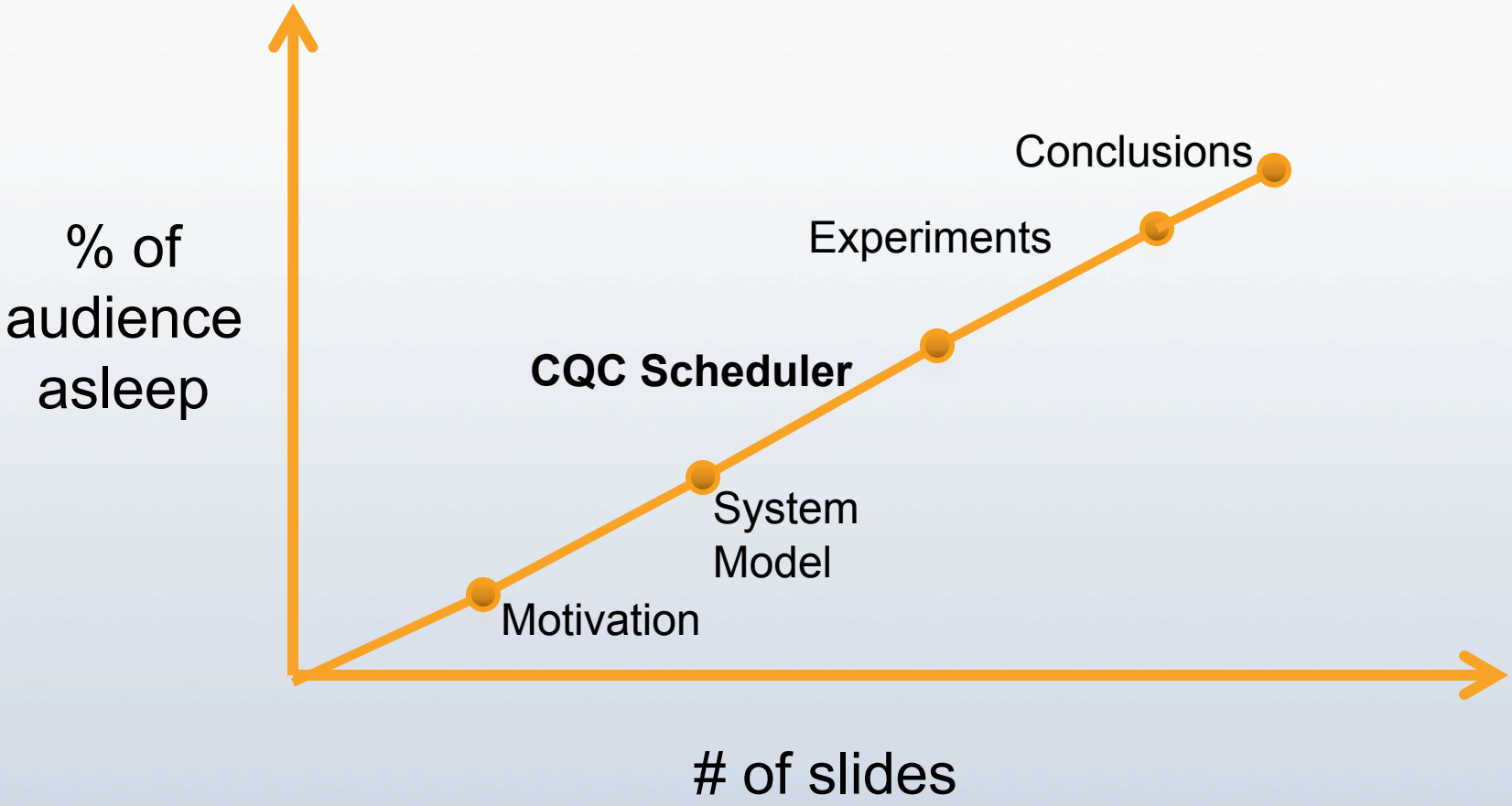
Why WHR fails to optimize for more critical classes?

- Consider the following two operators: O_i and O_j

Operator	HR Priority	Class Priority	WHR Priority
O_i	6	2	12
O_j	3	3	9

- Even though O_j belongs to a more critical class, it is scheduled behind O_i.

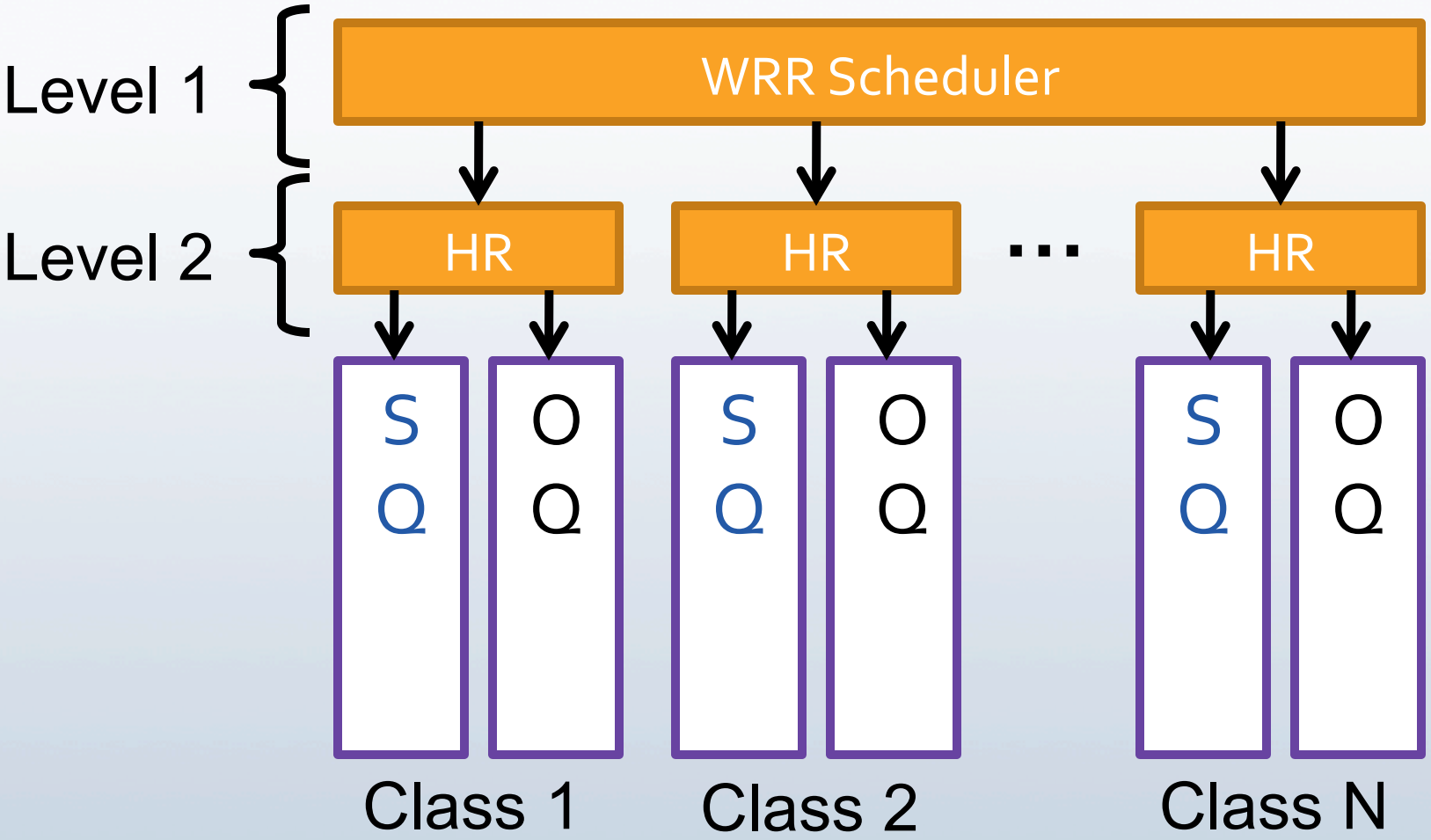
Roadmap


% of audience asleep
Conclusions
Experiments
CQC Scheduler
System Model
Motivation
# of slides

Continuous Query Class Scheduler (CQC Scheduler)

- Optimizes the response time of critical queries.
- Consists of **two levels**:
  1) Weighted Round Robin scheduler (WRR) that dispatches between the HR schedulers
  2) HR schedulers: each HR scheduler is responsible to schedule operators belonging to the same class.

CQC Scheduler



CQC Scheduler

- Parameters inputted by the administrator:
  - Input to **WRR Scheduler**:
    - K: time period
    - N: number of query classes
    - P_1, P_2, … P_N: priority of query classes
  - Input to **HR schedulers**
    - Δ: number of times an HR scheduler instance is allowed to pull its sources before returning control to the WRR scheduler.

Level 1: WRR scheduler

- Allocates to each scheduler a maximum amount of time or quota T_i to execute for.
- Each scheduler *i* is allocated a time quota

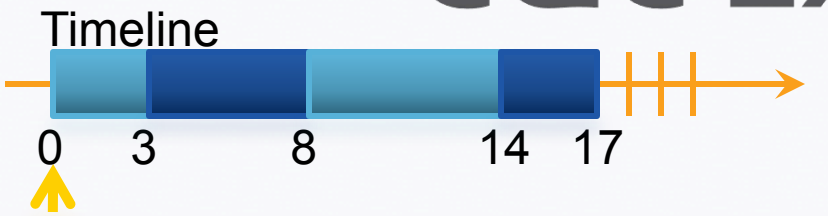
$$T_i = K \times \frac{P_i}{\sum P_j}$$

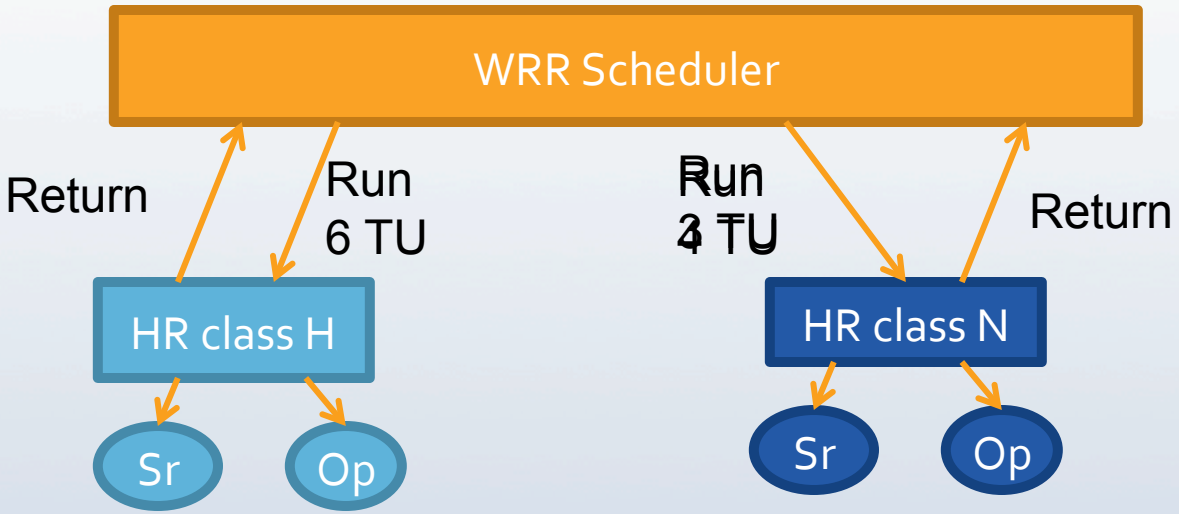
- WRR uses a negative credit system where if a scheduler exceeds its quota then this excess will be deducted from its future quotas.

CQC Example

- Example:
  - 2 query classes: H and N with priorities 0.6 and 0.4.
  - One query per class formed of a source operator and an operational operator.
  - Time period = 10 timeunits → TH is 6 and TN is 4
  - $\Delta = 2$

CQC Example

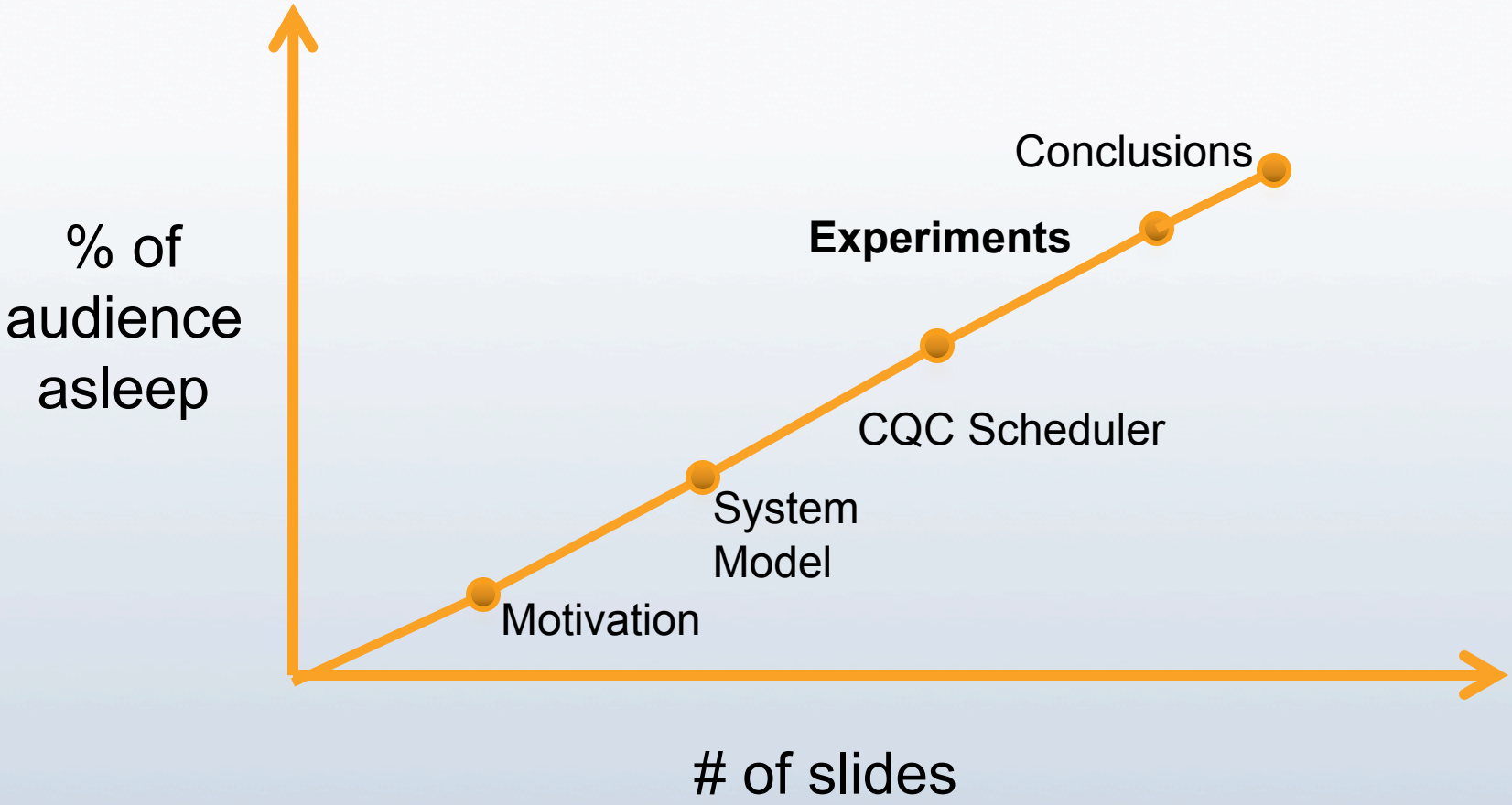




No tuples to execute!!
Time quota expired!!
No tuples to execute!!

Time quota expired!!
Ran for 5 TU!!
Next quota = 3 TU

Roadmap


% of audience asleep
Conclusions
Experiments
CQC Scheduler
System Model
Motivation
# of slides

Experiments

- Implemented HR scheduler and CQC scheduler under AQSIOS (real system, based on STREAM)
- Measured the average response time of each CQ class.

Experimental Setup

System Parameters	k (time period)	1000 time units
	Time unit	1 micro second
Query Load Specifications	Number of queries	21
	Number of query classes	3
	Types of queries	Select, Aggr, 2-way Joins
	Window Size	10 time units
	Selectivity of Selections	[0.25 – 1]

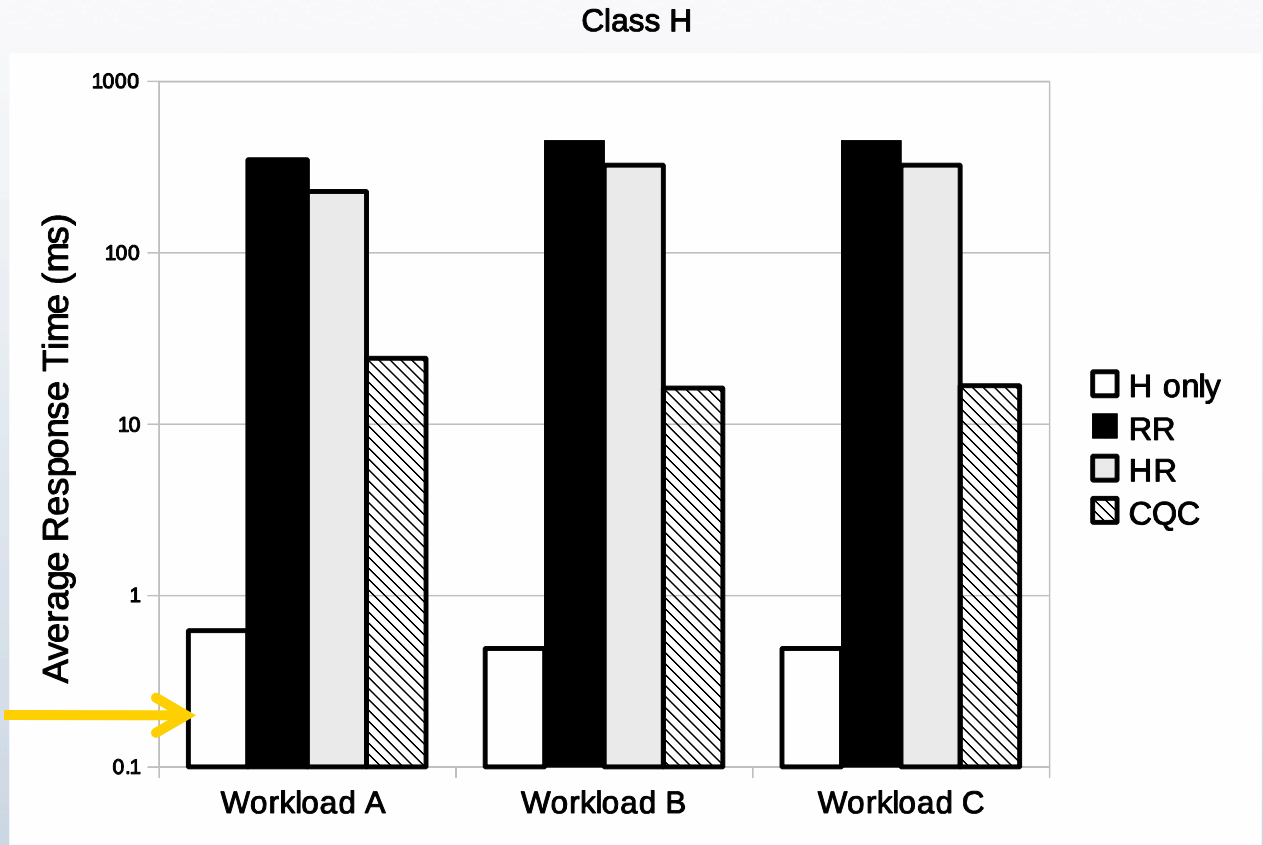
Experimental Setup - Data Stream Specifications

Data Stream Specifications	Humidity	Uniform [0 – 100]
	Tuple arrival rate/stream	1500 tuples/second
	Temperature	Uniform [0 – 40]
	Location	20 locations
	Number of tuples/stream	10,000
	Number of input streams	27

Experimental Setup

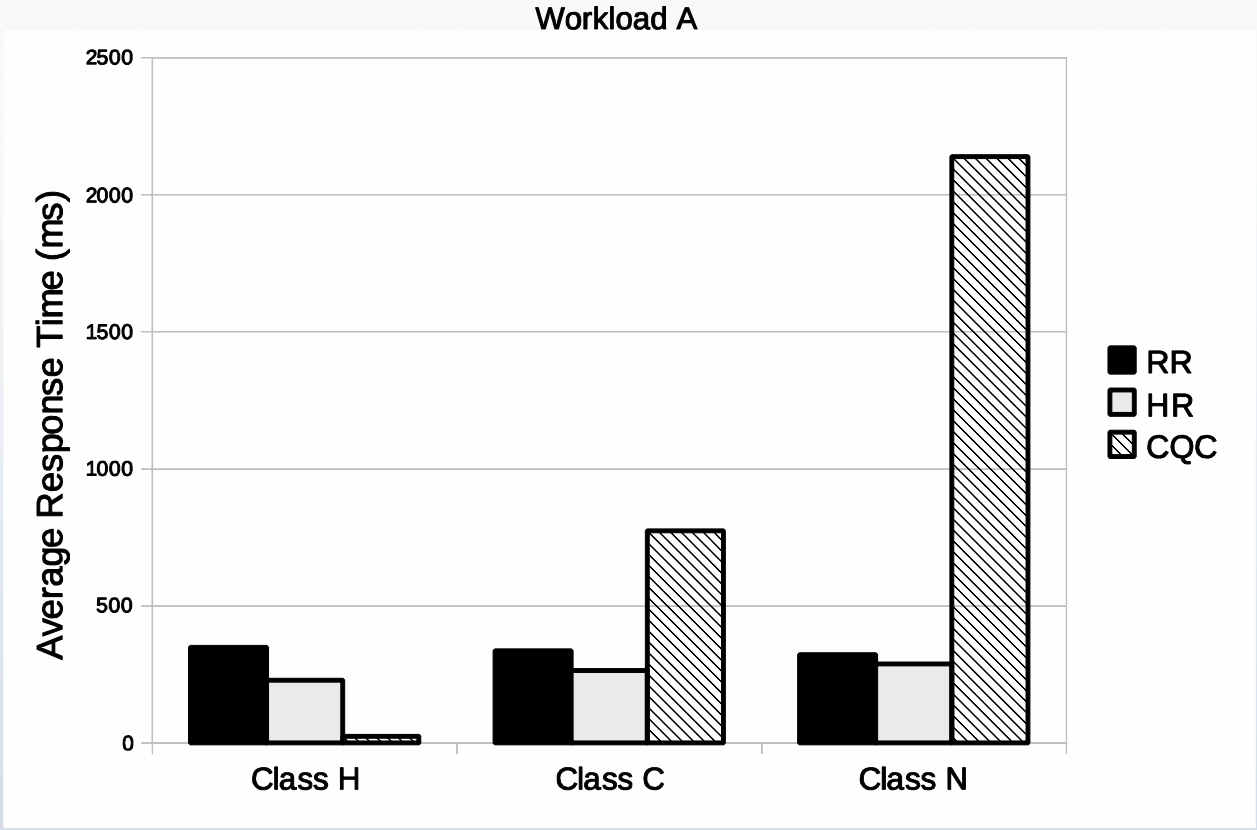
		Class H	Class C	Class N
Workload A	Number of queries	7	7	7
	Types of queries	All	All	All
	Priorities	6	3	1
Workload B	Number of queries	2	8	11
	Types of queries	Join	All	All
	Priorities	3	2	1
Workload C	Number of queries	2	8	11
	Types of queries	Join	All	All
	Priorities	6	3	1

Class H queries



System is under-loaded

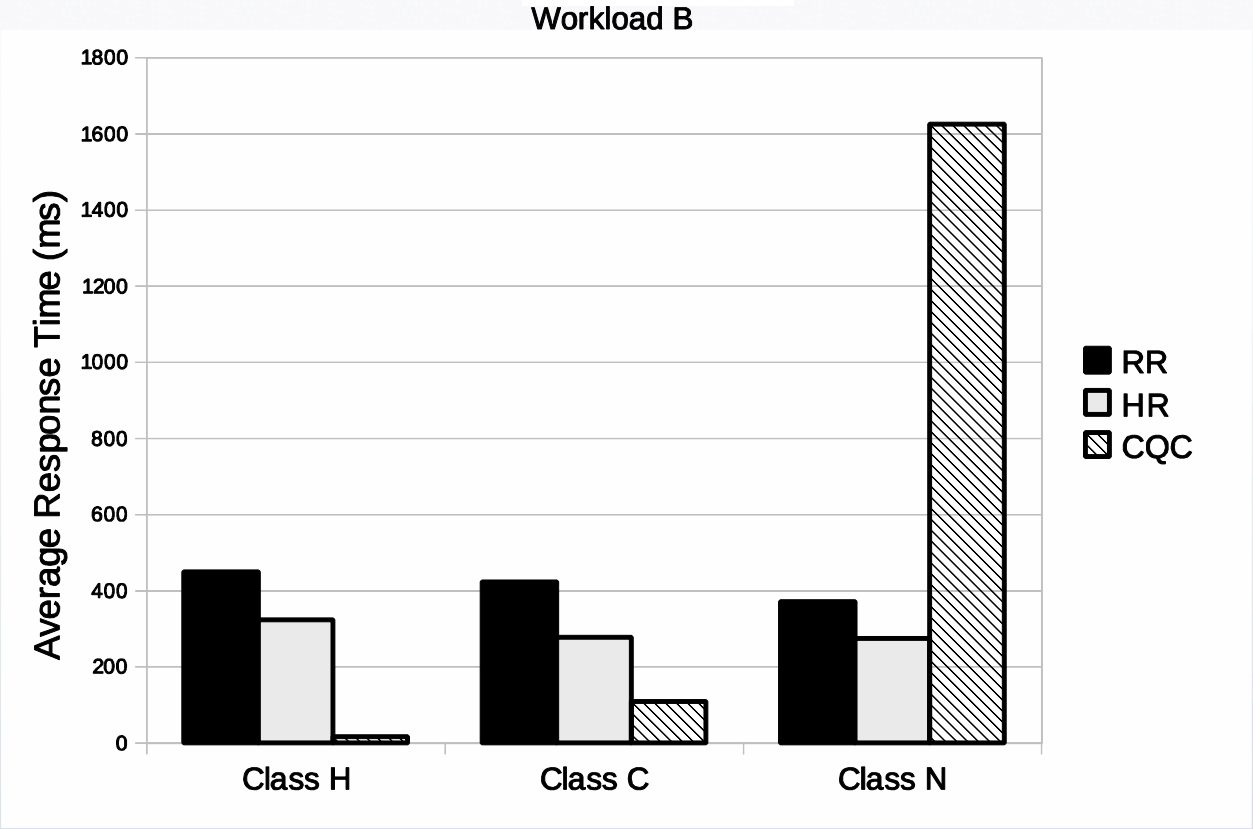
Workload A



Number of queries	7	7	7
Types of queries	All	All	All
Priorities	6	3	1

Improves the response time of class H queries by 9.4 times.

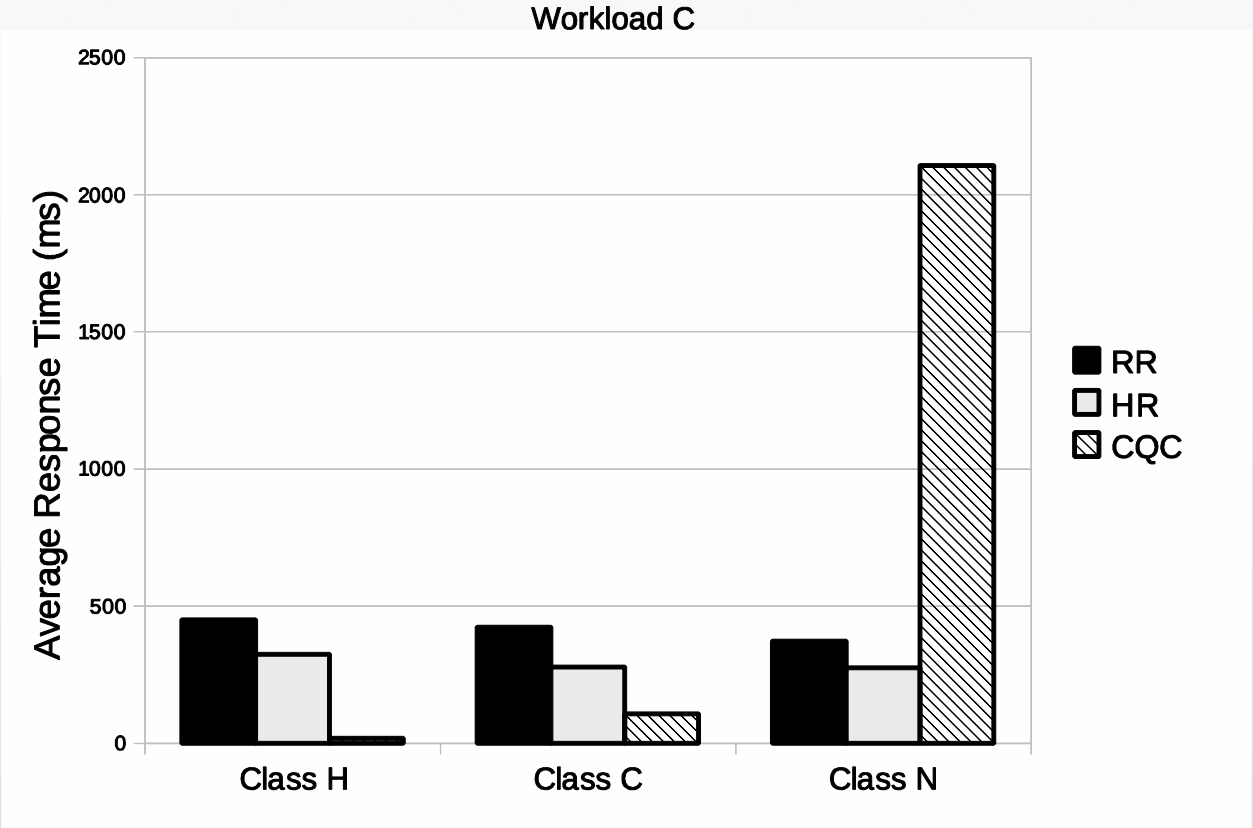
Workload B



Number of queries	2	8	11
Types of queries	Join	All	All
Priorities	3	2	1

Improves the response time of class H queries by 19.8 times and class C queries by 2.5 times

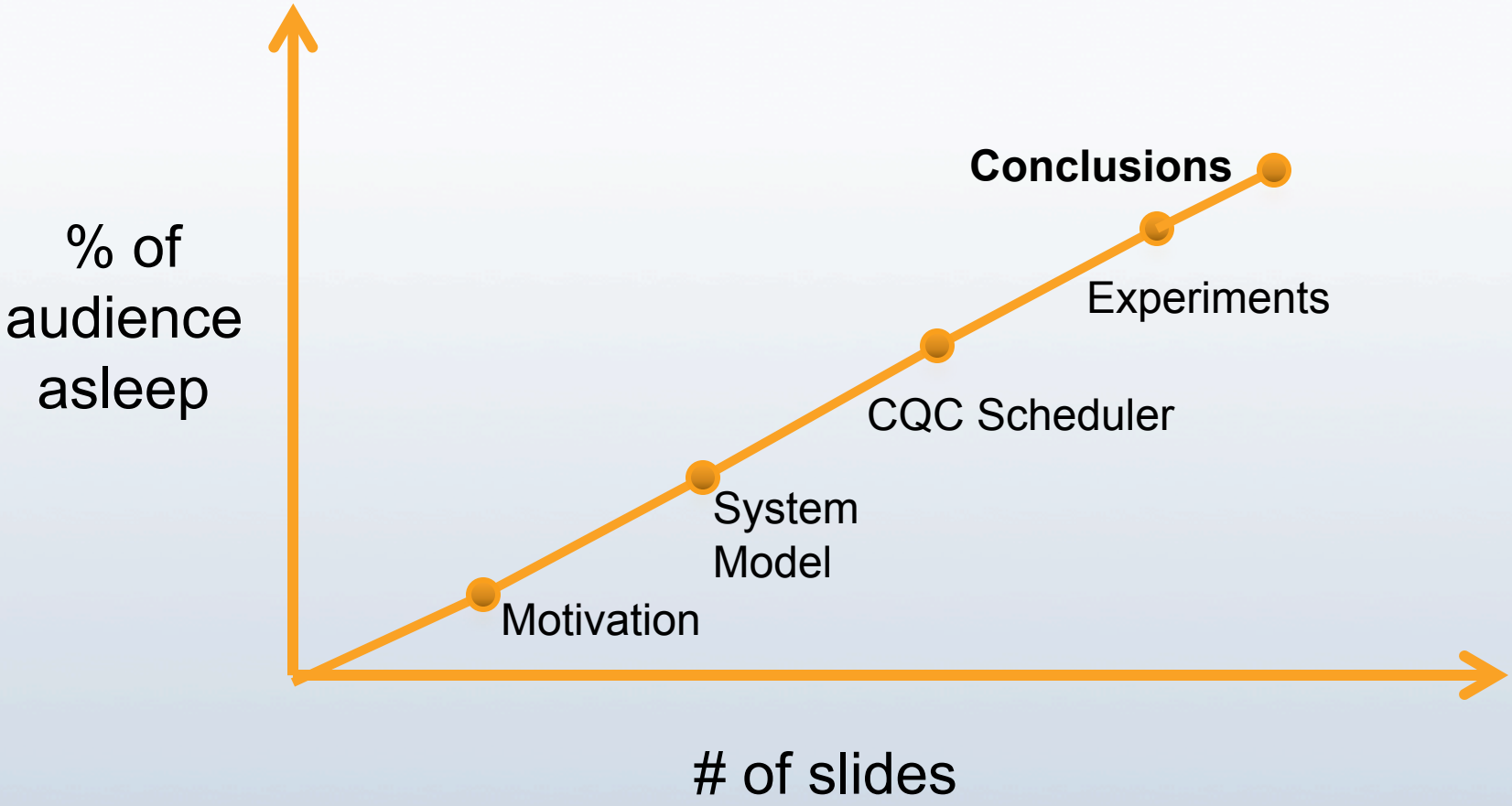
Workload C



Number of queries	2	6	11
Types of queries	Join	All	All
Priorities	3	2	1

Improves the response time of class H queries by 19.3 times and class C queries by 2.5 times

Roadmap


% of audience asleep
Conclusions
Experiments
CQC Scheduler
System Model
Motivation
# of slides

Conclusions

- We presented a scheduling policy that optimizes for the average response time of critical classes.
- We implemented and evaluated our scheduler on our DSMS prototype AQSIOS.
- We showed that our scheduler improves the average response time of critical queries.

Questions?