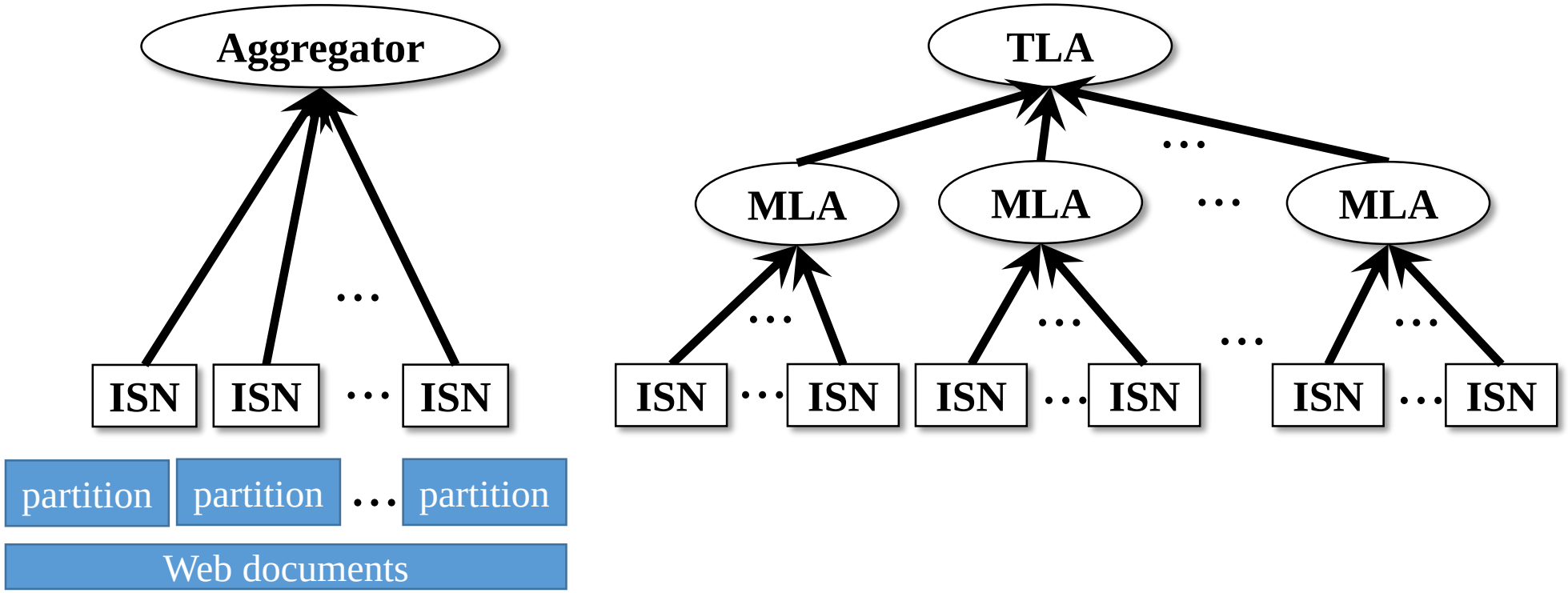
Optimal Aggregation Policy for Web Search

Jeong-Min Yun[1], Yuxiong He[2], Sameh Elnikety[2], Shaolei Ren[3]

[1]POSTECH, [2]Microsoft Research, [3]Florida International University

Web Search Architecture

- Billions of web documents are partitioned among many servers
- Distributed system with aggregators and index serving nodes (ISNs)



Aggregation Policy

- Decide how long aggregators wait for ISNs

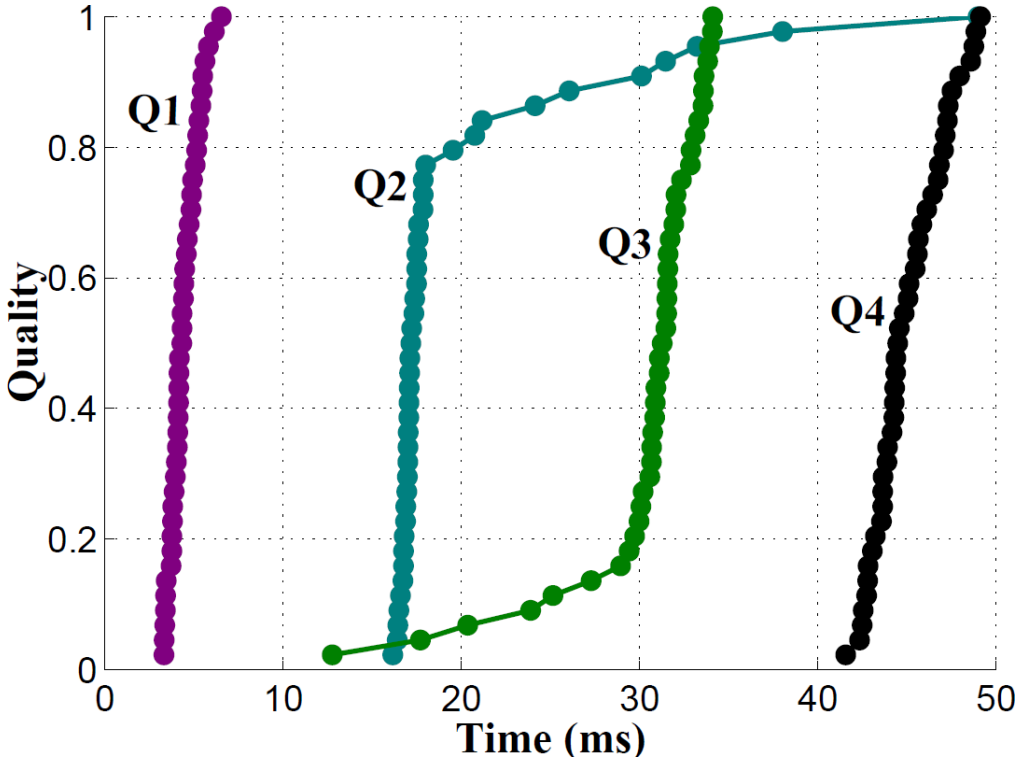
- Latency: tail latency for consistently fast responses
- Quality: fraction of ISNs whose results are returned
- Latency quality tradeoff
  - No waiting policy gives zero latency but zero quality
  - Wait all policy gives perfect quality but maximum latency

- Our objective: reduce tail latency while meeting quality requirements

Challenges

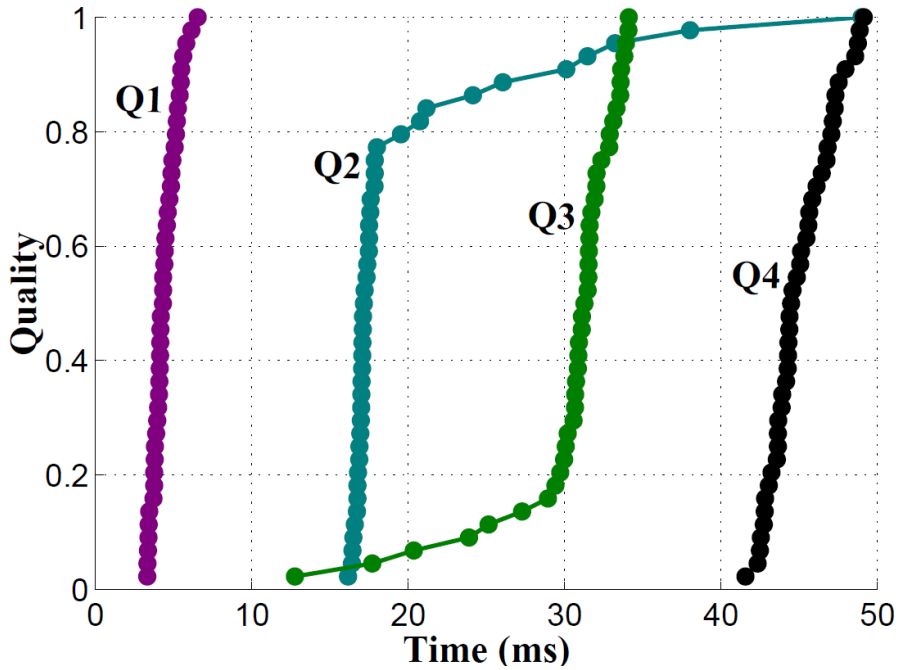
- Online decision
  - Aggregators do not know when ISNs will return their results

- Different queries exhibit highly variable service demand
- ISN response times vary significantly even for a single query



Prior Work

- Wait for all
- Wait by time *t*
- Wait until quality *q*
- Jointly consider time and quality



Which query should be terminated?

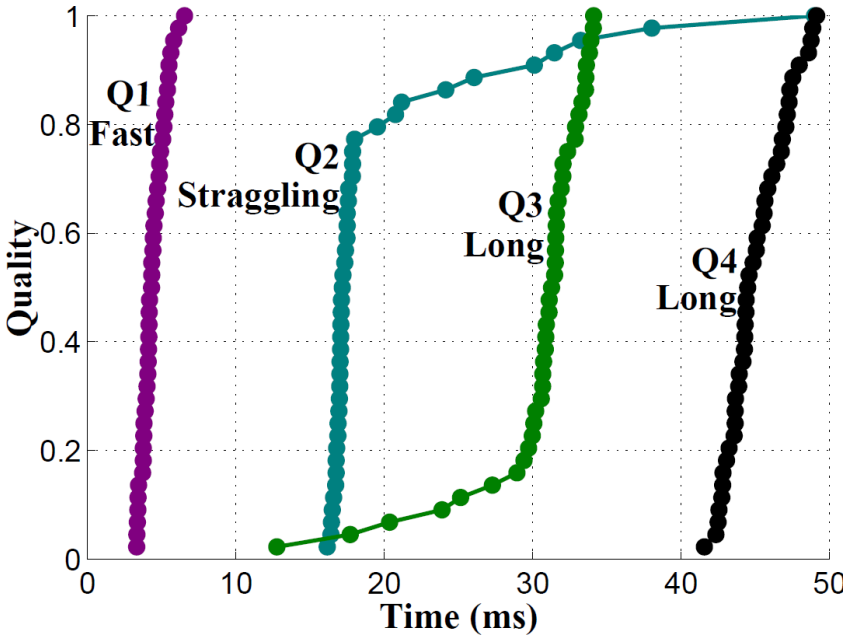
- Limitations
  - Heuristic algorithms, missing potential latency improvement
  - None of them cannot address multilevel aggregation

Summary of Contributions

- Workload characterization and key intuitions
- FSL: a new aggregation policy with optimality proof
  - Performs as well as optimal policy!
- Extension to multilevel aggregation
- Experimental evaluation
  - Microsoft Bing search and Advertisement production traces
  - Reduces tail latency by 36% over the best prior work

Intuitions

- Workload characterization: three types of queries
  - Fast query: responses from all ISNs arrive quickly
  - Straggling query: most responses arrive quickly with a few stragglers
  - Long query: most responses take a long time
- Key intuition
  - Complete fast & long queries for quality
  - Terminate straggling queries to reduce latency



Intuitions by Example

- Goal: Minimize 95-th percentile latency with average quality ≥ 0.99
- Fast query: their completion time does not affect 95-th tail latency

- Straggling query:
  - Miss at most 1 – 0.99 = 1% of ISN responses
  - Allocate 1% quality loss to straggling queries to maximize latency reduction

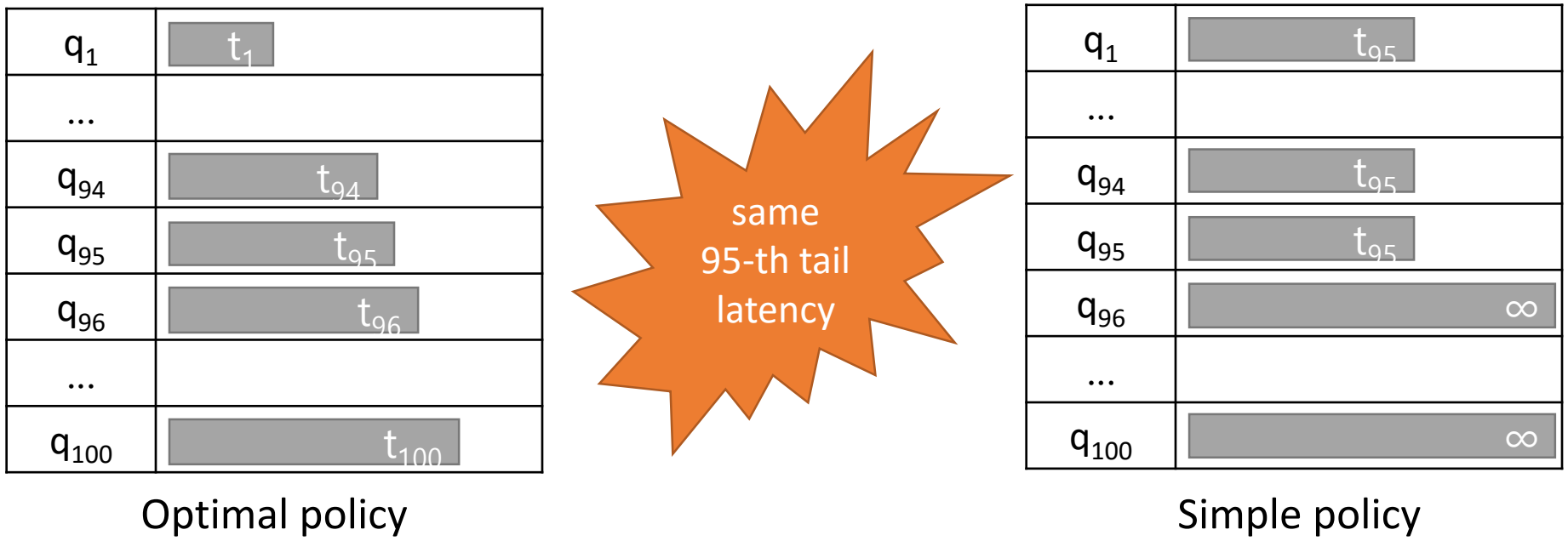
- Long query: to minimize 95-th tail latency, < 5% long queries may respond slowly with full quality without affecting latency

FSL Aggregation Algorithm
- for Fast, Straggling, Long queries

- Single time threshold and quality threshold
  - Differentiate fast, straggling and long queries with proper actions
- Data-driven approach
  - Offline processing: find best time and quality threshold using data traces
  - Online processing: Terminate query at time threshold if its quality is less than quality threshold
- Optimality proof: FSL performs as well as the offline optimal policy

FSL: Key Idea

- There exists a simple policy with one time threshold and one quality threshold whose tail latency is equivalent to that of any optimal policy
  - Example: for 100 queries, termination time of i-th query (q_i) from an optimal policy is t_i, $t_1 \leq t_2 \leq \ldots \leq t_{100}$, $\exists$ latency and quality equivalent simple policy



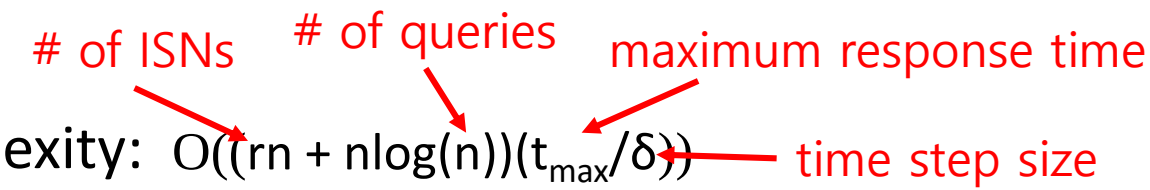
FSL: Online Processing

- Time threshold t* and quality threshold u*

- At time t*,
  - If all responses are returned
    - Do nothing (fast query)
  - If quality u ≥ u*
    - Terminate the query (straggling query)
  - If quality u < u*
    - Run query until completion (long query)

FSL: Offline Processing

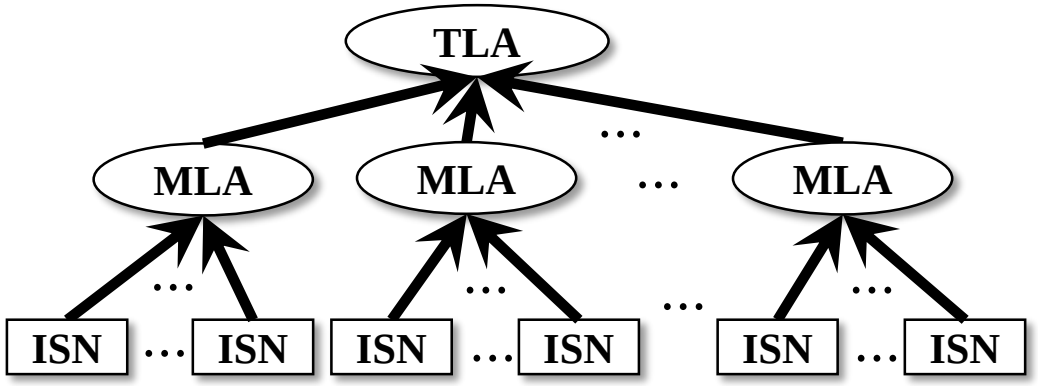
- How to compute time threshold t* and quality threshold u*?
  - For each candidate time threshold,
    - ① Assign quality 1 to long queries
    - ② check whether it satisfies all quality requirements
  - Time threshold is the minimum of them who satisfies all quality requirements
  - Quality threshold is the lowest quality straggling query at that time
  - Time complexity: $O((rn + n\log(n))(t_{max}/\delta))$



- Any given workload only requires offline processing ONCE; online decision for a query is a simple comparison incurring constant cost

Extension to Multilevel Aggregation

- New challenges
  - Aggregators' decisions on different levels are coupled
  - Communications between different levels of aggregators are essential to check query progress, but the amount of communication must be small



TLA doesn't know quality of the current query unless all MLAs send their progress

For an MLA to know the quality, TLA should send back computed value to MLA

FSL for Two-Level Aggregation

- Known messaging times
  - Almost same as the single aggregator case (optimality proof is still possible!)
- Bounded messaging times
  - Approximation error bound is derived
- Unknown messaging times
  - Proposed heuristic (no optimality guarantee) forces all MLAs to send their partial results at the same time point

Experimental Setup

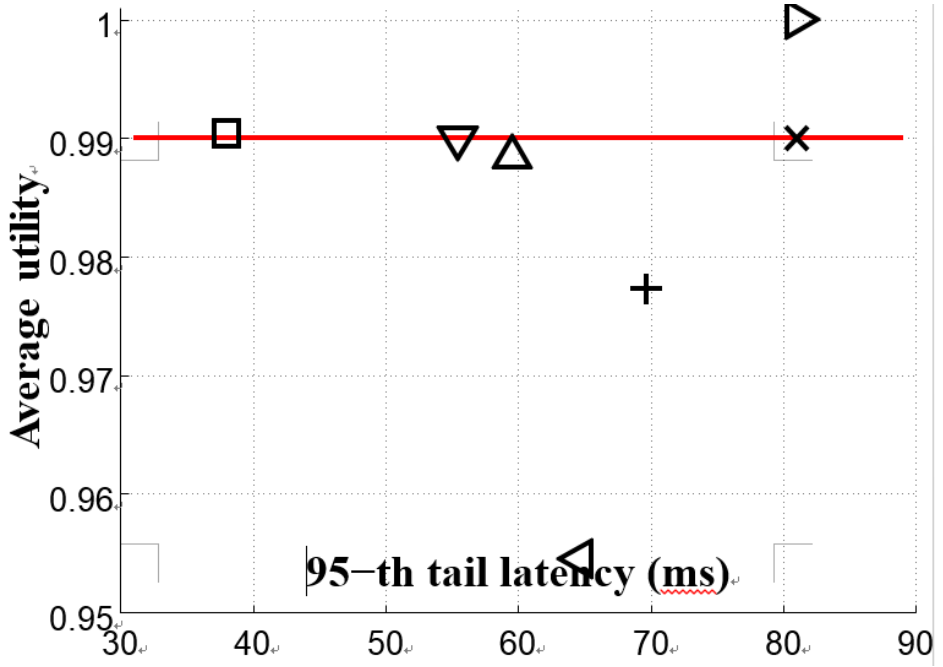
- Workload
  - Single Aggregator – Microsoft Bing production traces
  - Two level aggregation – Microsoft Bing Ads production traces
  - Rich set of synthetic workloads
- Algorithms in comparison
  - Wait all: wait responses of all ISNs
  - Time only: return results at time *t*
  - Quality only: return results at quality *q*
  - Kwiken [1]: jointly consider time and quality thresholds

[1] V. Jalaparti, P. Bodik, S. Kandula, I. Menache, M. Rybalkin, and C. Yan. Speeding up distributed request-response workflows. In SIGCOMM ’13, 2013.

Experiments: Single Aggregator

- Microsoft Bing search engine production traces
  - Latency of 44 ISNs over 66,922 queries (10,000 for training, 56,922 for test)
- Goal: minimize 95-th tail latency while average quality ≥ 0.99

- FSL reduces tail latency
  by 53% over wait all
  by 36% over the best alternative



	Policy
▷	Wait-All
◁	Utility-Only(42/44)
+	Utility-Only(43/44)
X	Time-Only
▽	Time-Utility
△	Kwiken
□	FSL

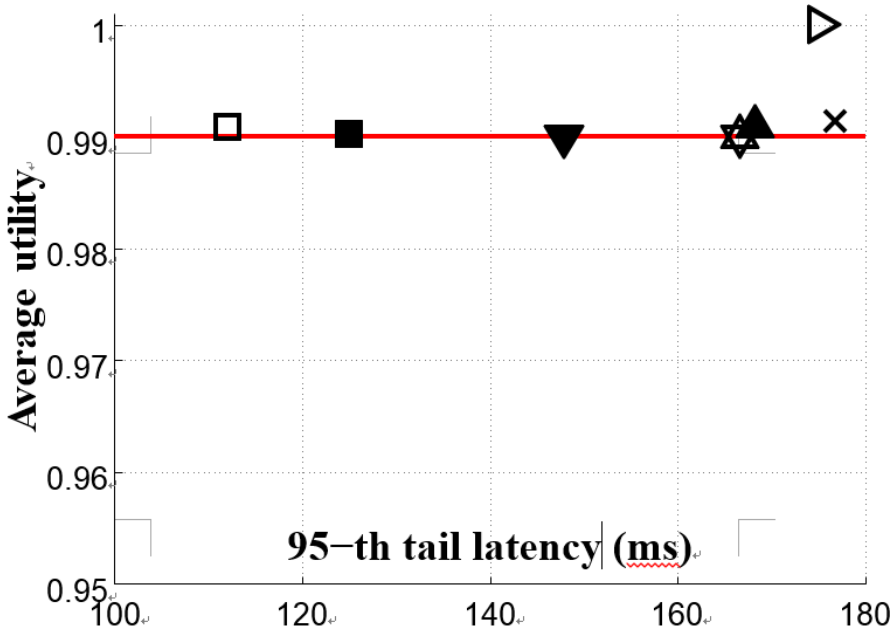
Experiments: Multilevel Aggregation

- Microsoft Advertisement engine production traces
  - 1 TLA, 16 MLAs, 64 ISNs (4 per MLA). 10,000 for training, 6,311 for test
- Goal: minimize 95-th tail latency while average quality ≥ 0.99

- FSL-U is within 12% of the optimal (FSL-K)

  Reduces tail latency by 15% over best alternative

Symbol		▷	◁	◀	X	▽	▼	△	▲	□	■
Policy	MLA	Wait-All	Wait-All	Utility-Only	Time-Only	Time-Utility	Wait-All	Kwiken	Wait-All	FSL-K	FSL-U
	TLA	Wait-All	Utility-Only	Utility-Only	Time-Only	Wait-All	Time-Utility	Wait-All	Kwiken		



Conclusion

- FSL: optimal online aggregation policy
- Extension to multilevel aggregation
  - Optimal for known messaging time between aggregators
  - Empirically-effective policy for unknown messaging time
- Experimental evaluation
  - Microsoft Bing search and Advertisement production traces
  - Reduces tail latency by 36% over the best prior work