E2E: Embracing User Heterogeneity to Improve Quality of Experience on the Web

Xu Zhang[1], Siddhartha Sen[2], Daniar Kurniawan[1], Haryadi Gunawi[1], Junchen Jiang[1]

[1]University of Chicago, [2]Microsoft Research

Page load time matters!

How One Second Could Cost Amazon $1.6 Billion In Sales

Users are happier with faster page load time!

How Website Performance Affects Conversion Rates | Cloudflare

https://www.cloudflare.com/learning/.../more/website-performance-conversion-rates/

impact on your entire marketing and sales pipelines. Up to 79% of customers who are dissatisfied with a website's performance say they're less likely to buy from the same site again. **But can big business issues actually be solved simply**

Conventional wisdom

Cut all server-side processing delays

- Minimize mean delay
- Minimize P99 delay
- Minimize rate of missing a deadline

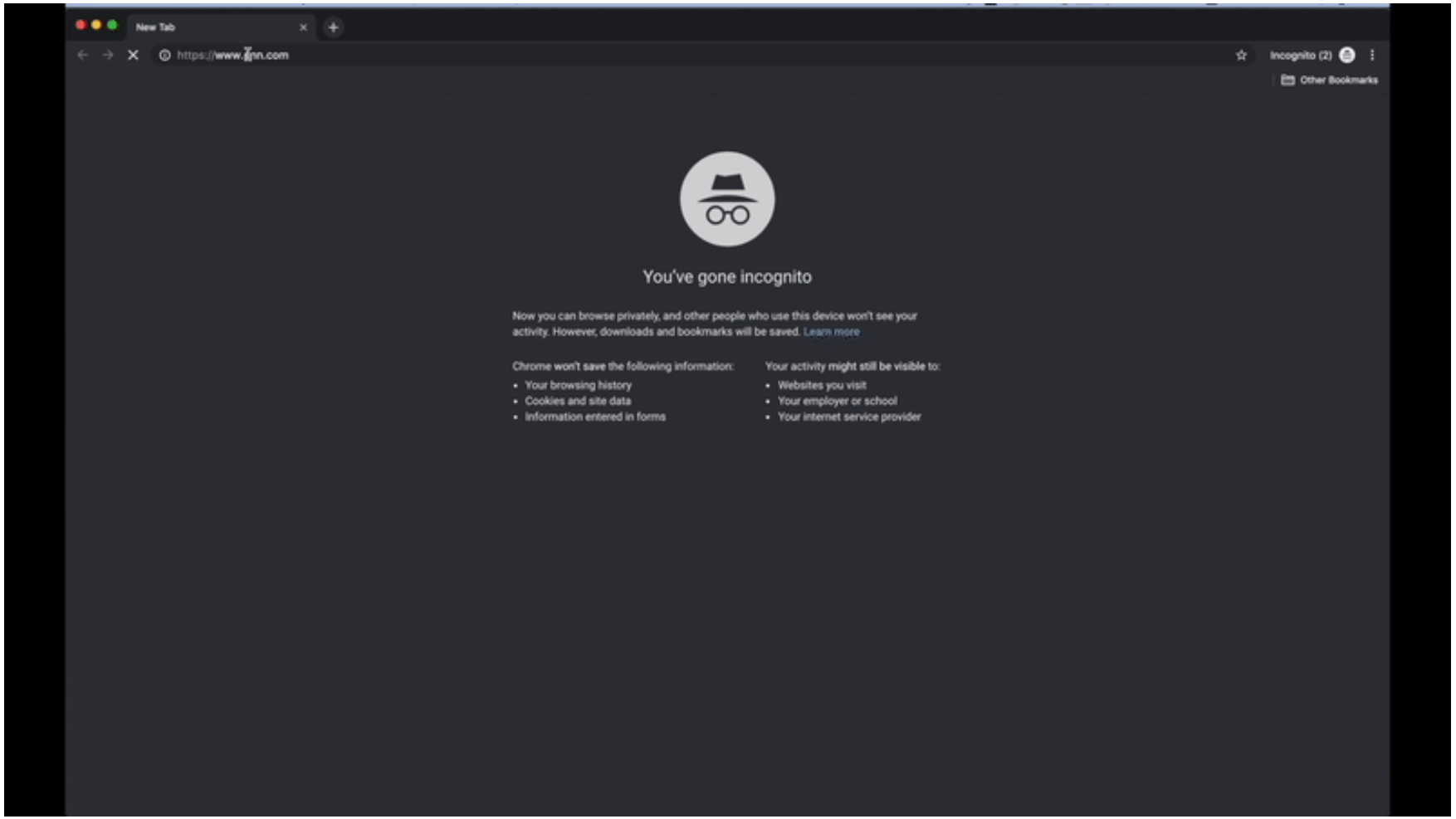
Q: Should we treat all requests in the *SAME* way?

Experiment: Observe the difference in the quality of page load events

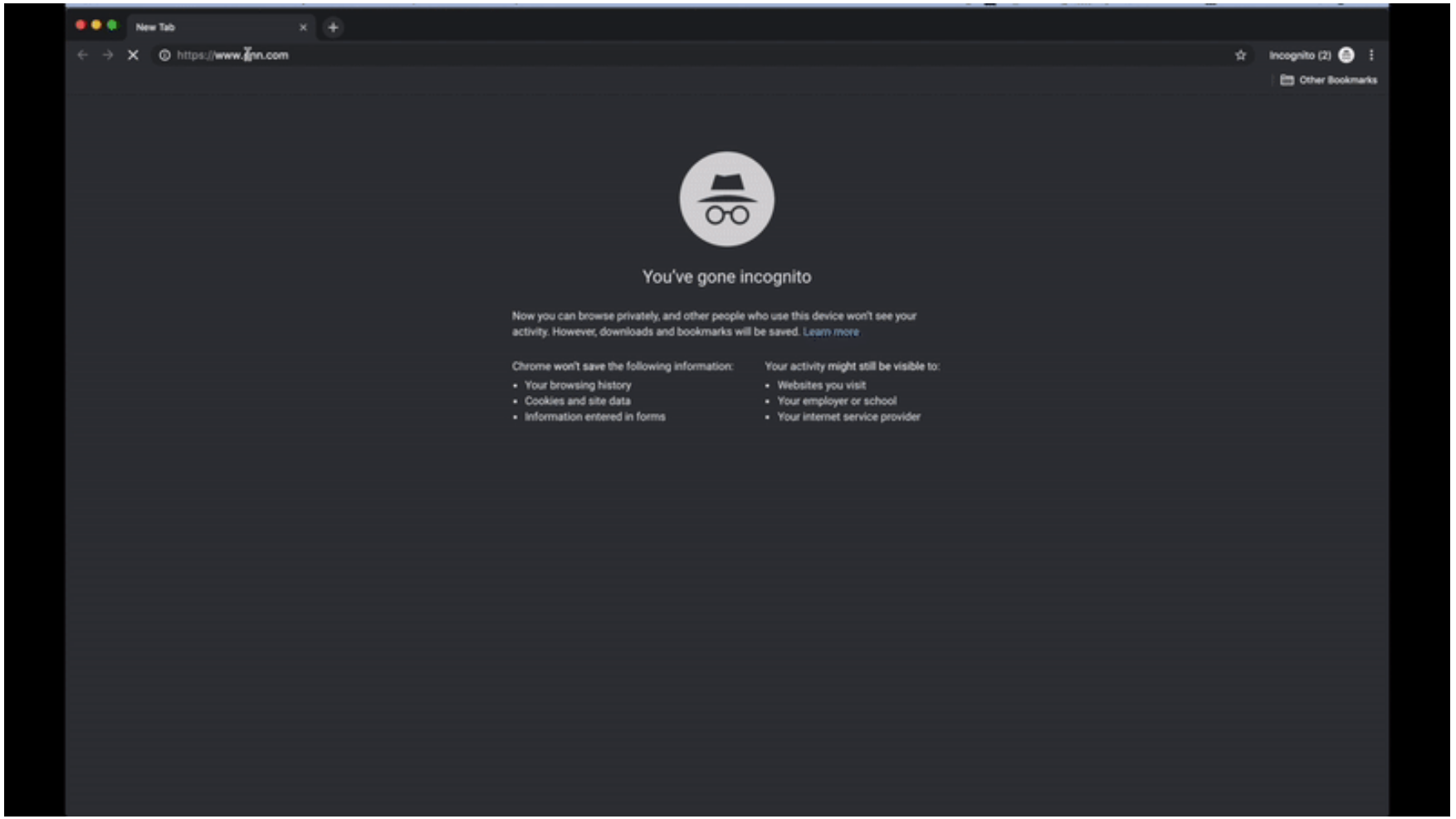
Can you see significant improvement?

Set #A

Set #A: Before improvement



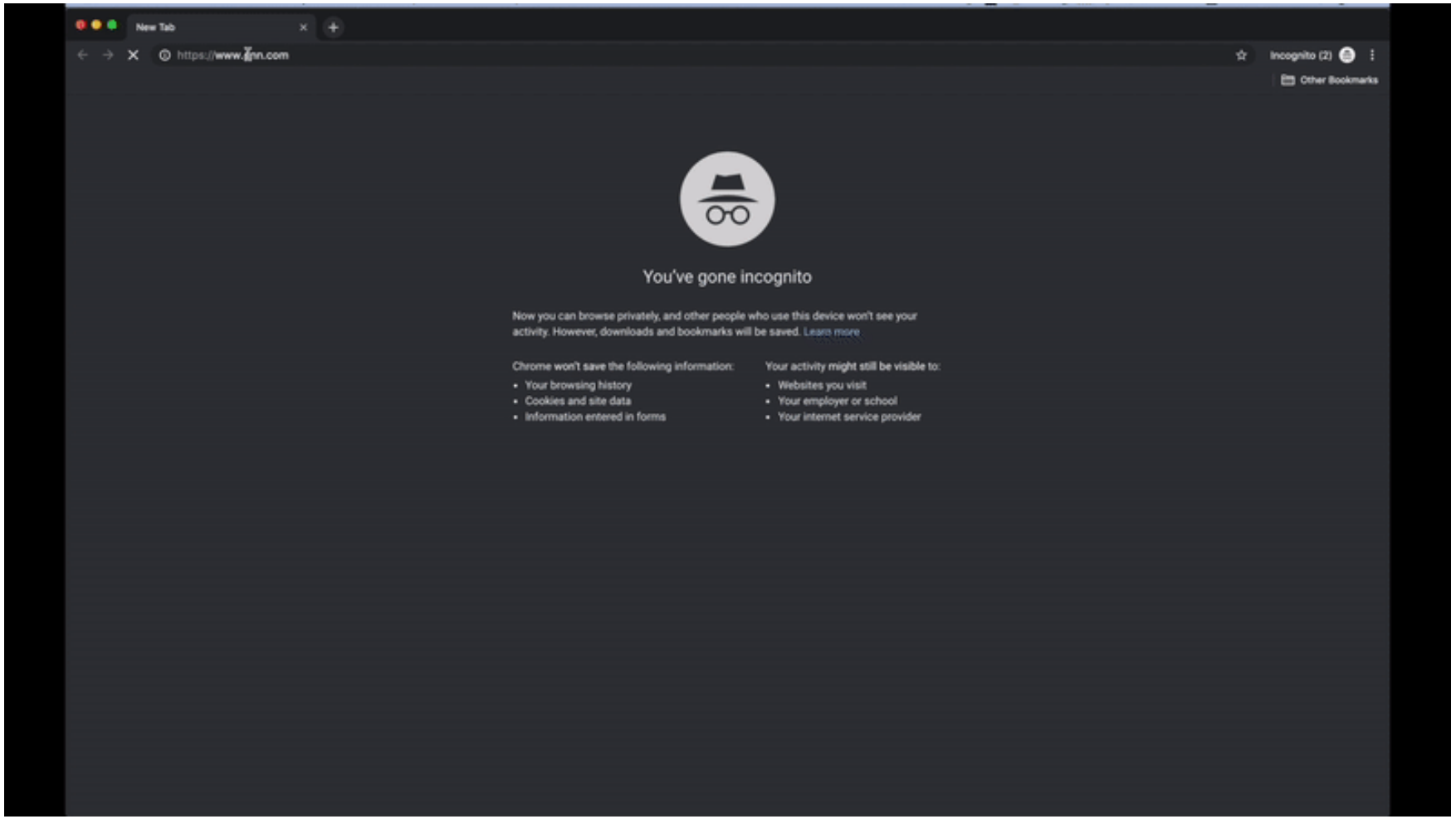
Set #A: After improvement



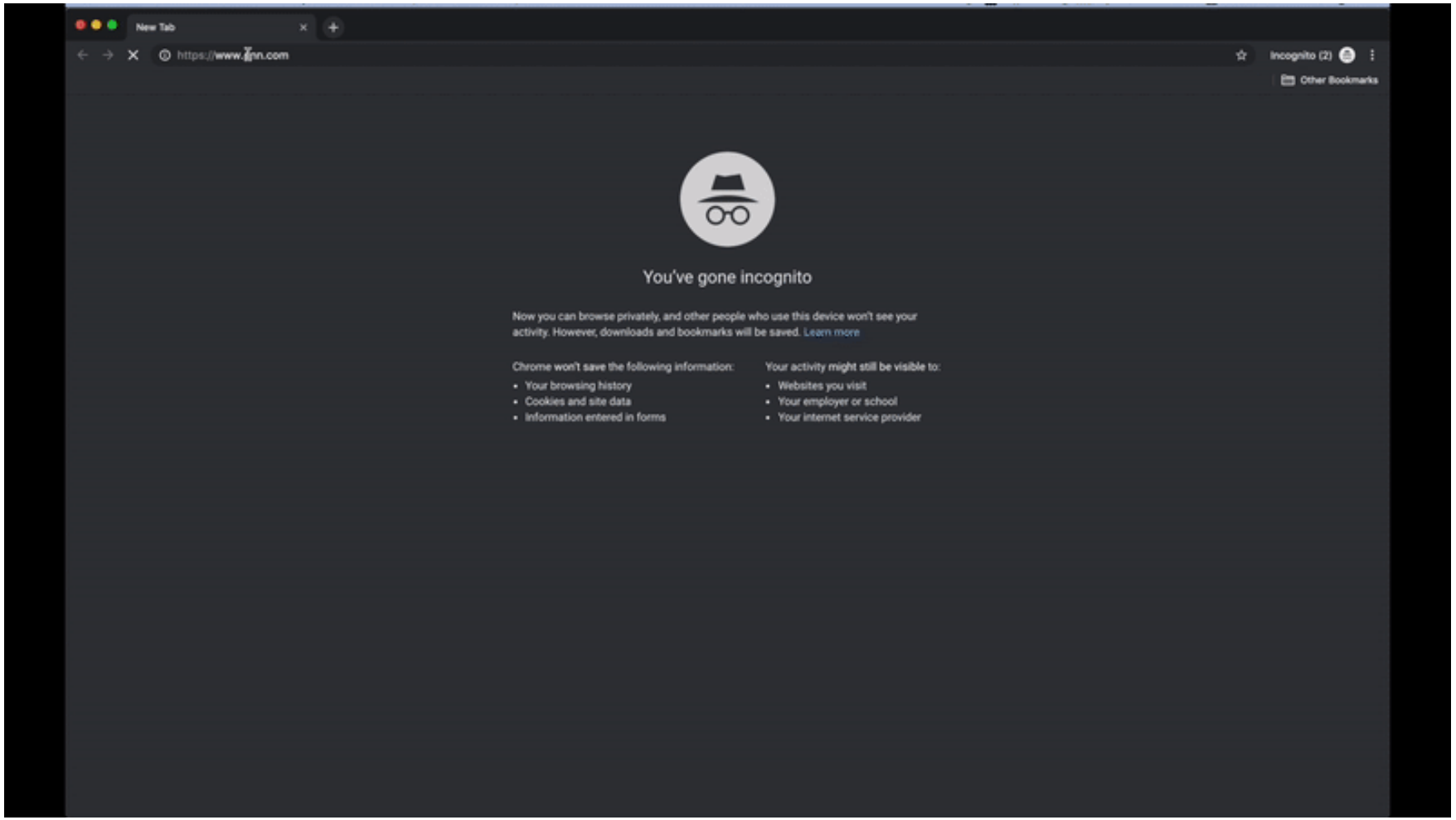
Can you see significant improvement?

Set #B

Set #2: Before improvement



Set #2: After improvement



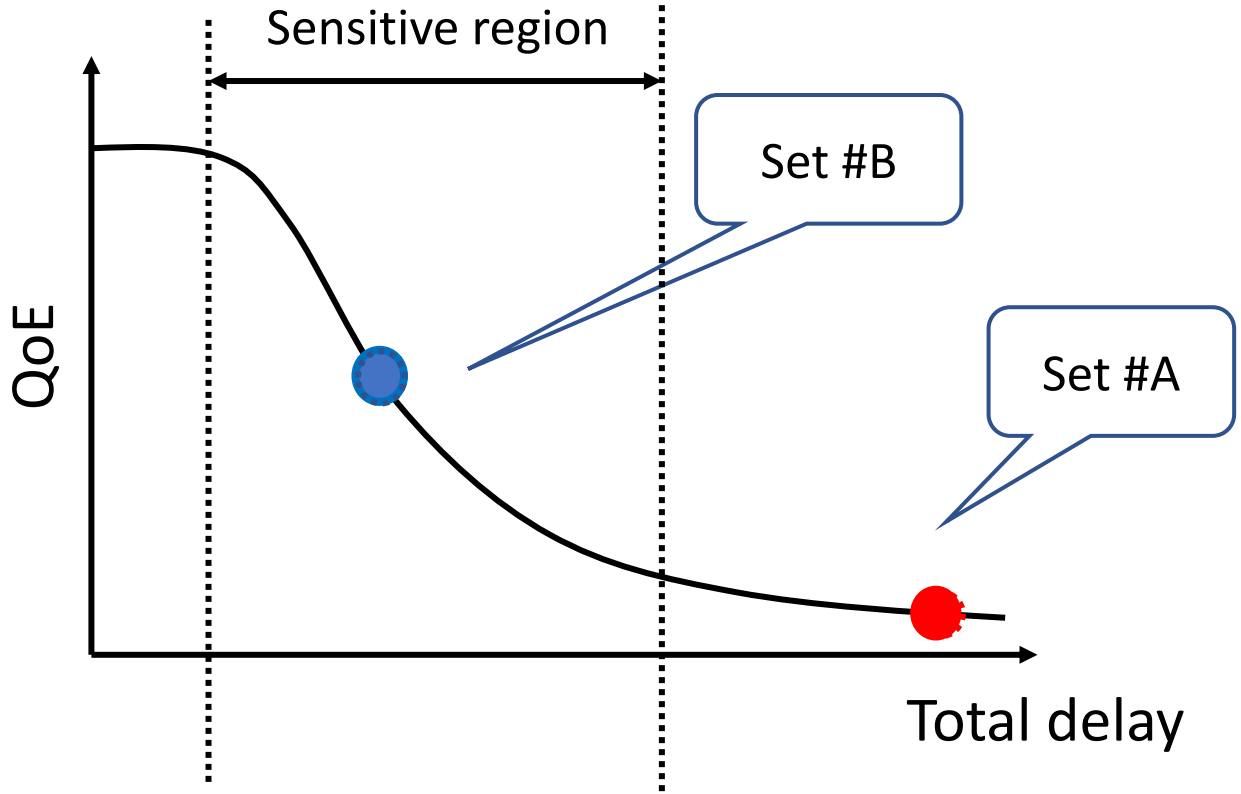
Can you see significant improvement?

Does it mean Set #B has a bigger delay reduction?

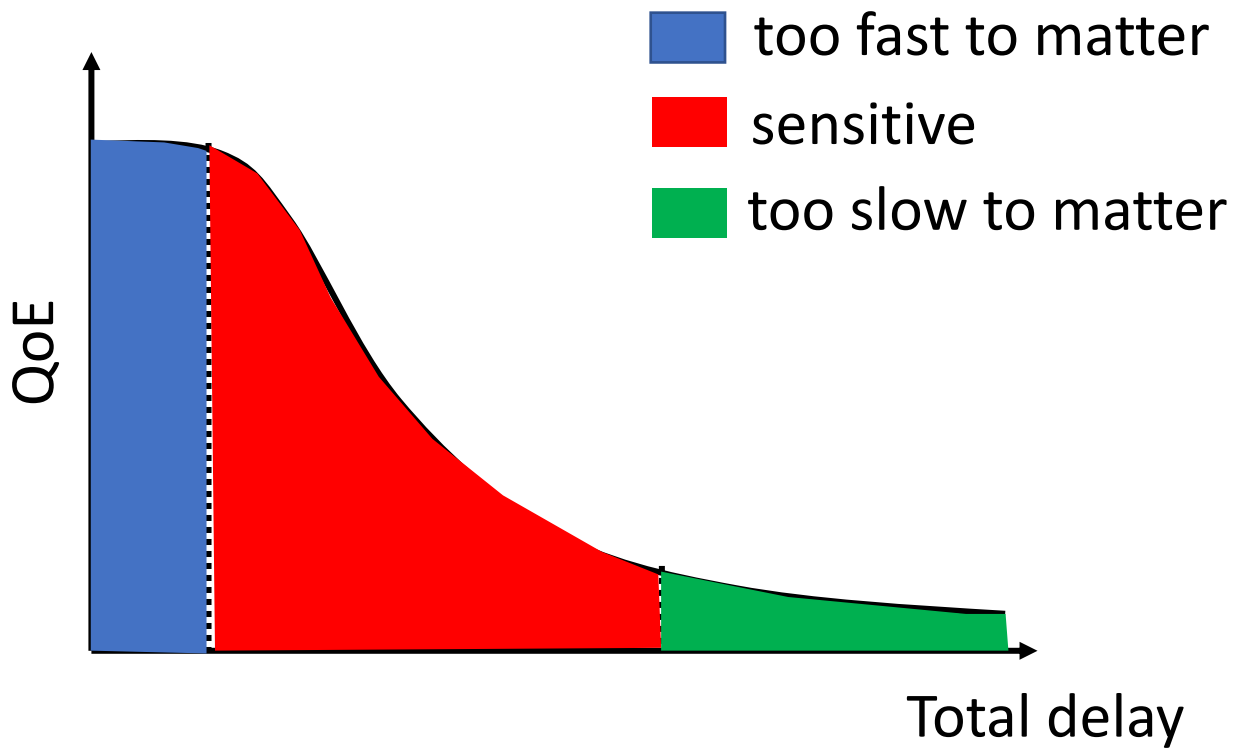
No! The two sets improved by the *SAME* amount of delay!

Same delay reduction, but different improvements

In sensitive region, people are more sensitive to additional delay.

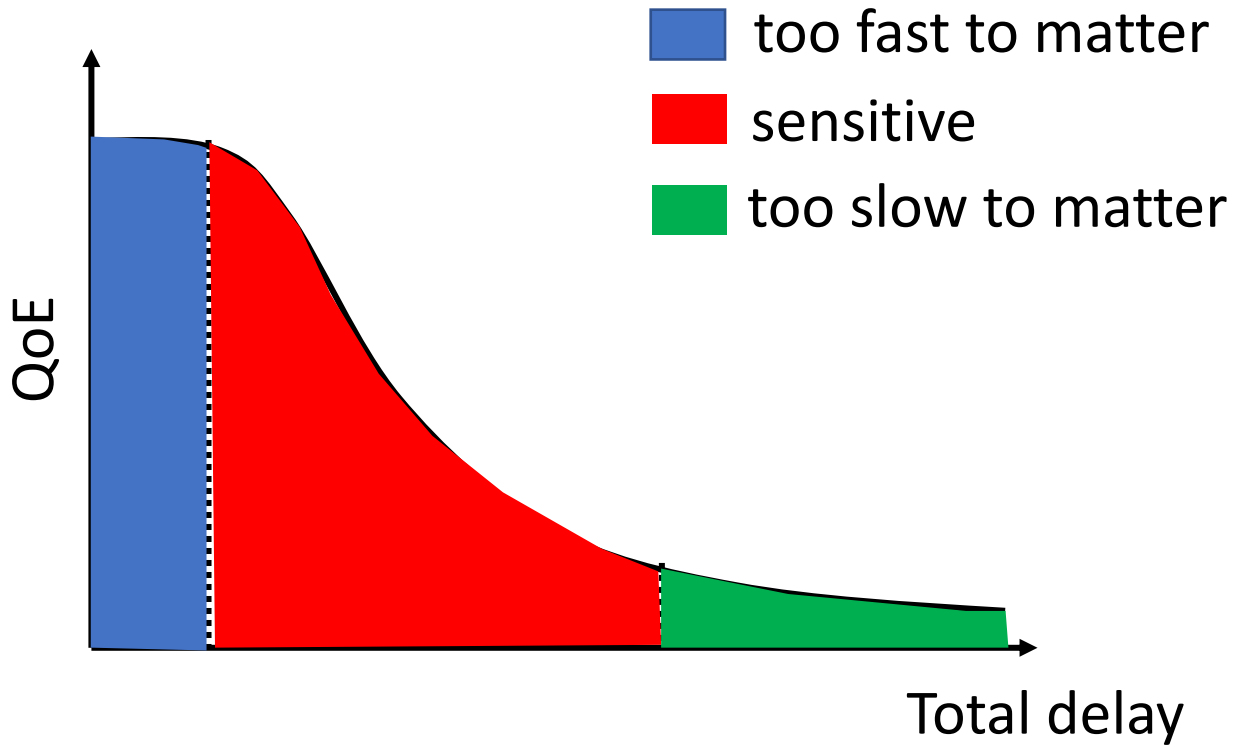


Requests have different sensitivities to additional delay

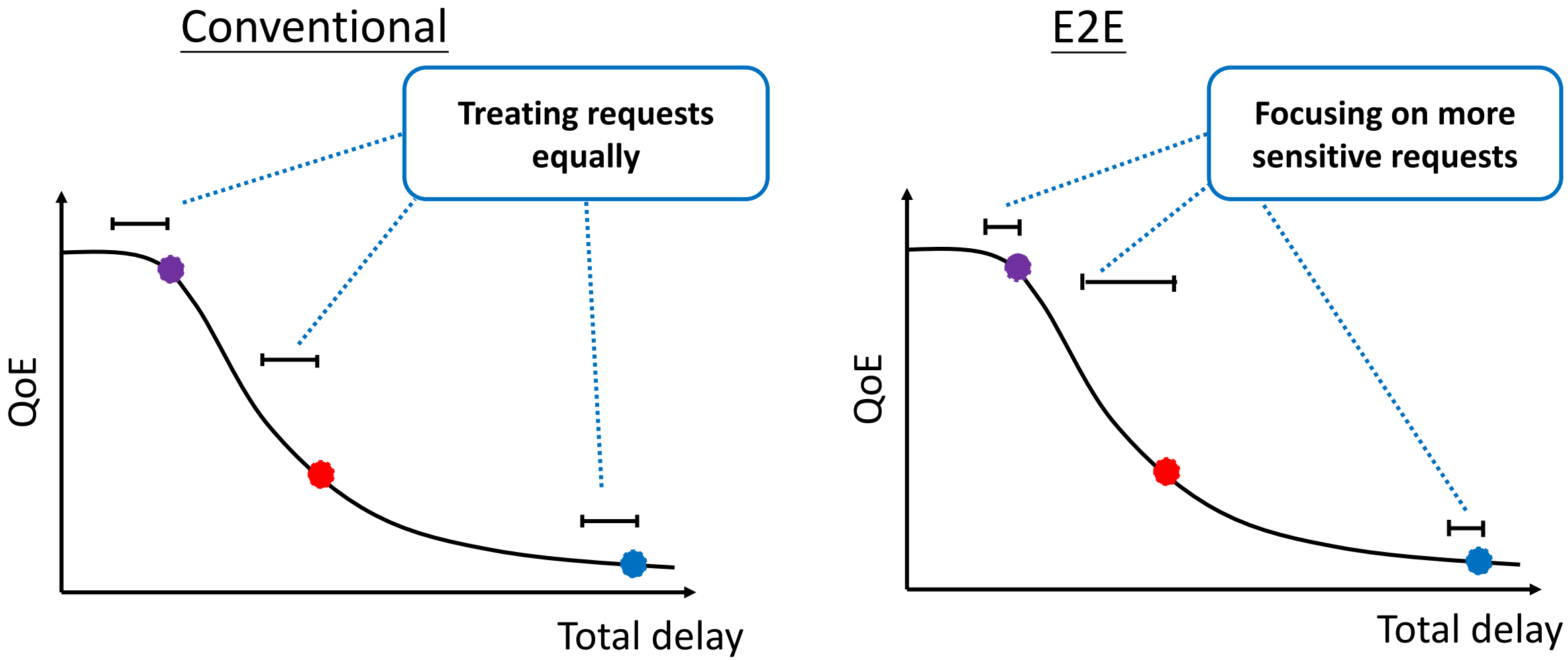


Requests have different sensitivities to additional delay

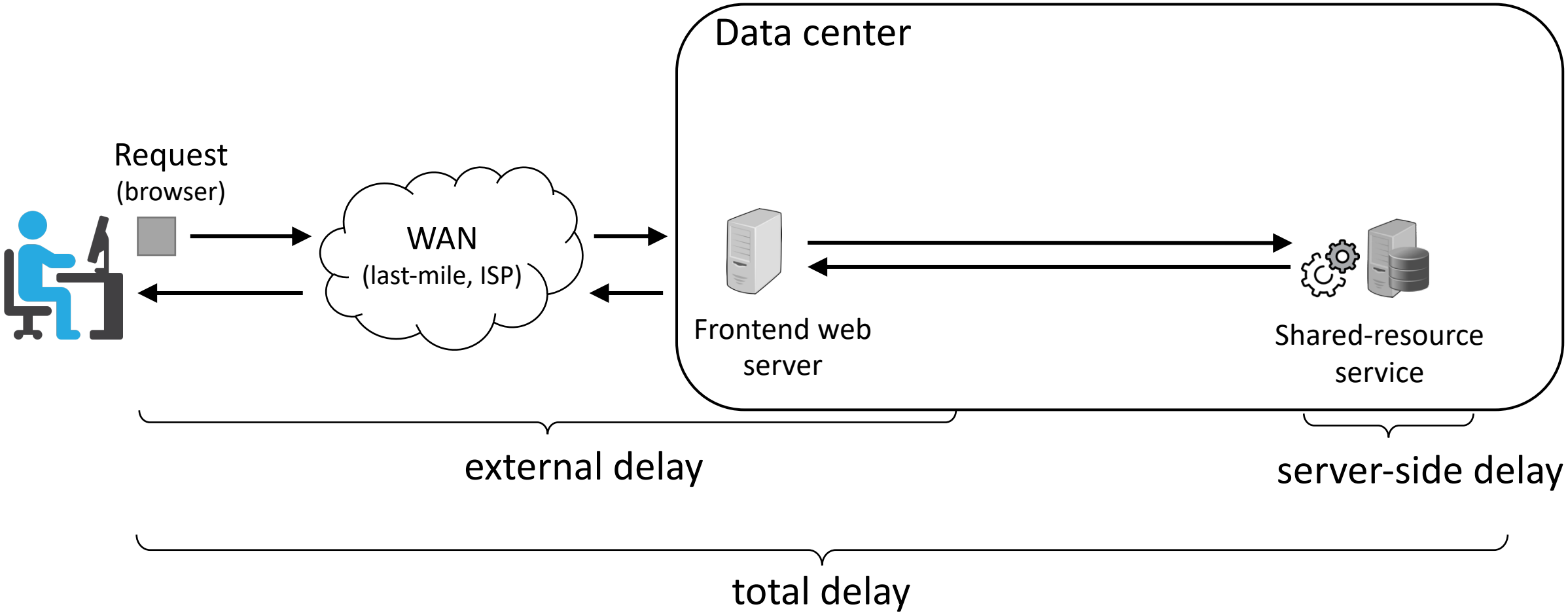
Analysis from Microsoft online store traces and user study on MTurk



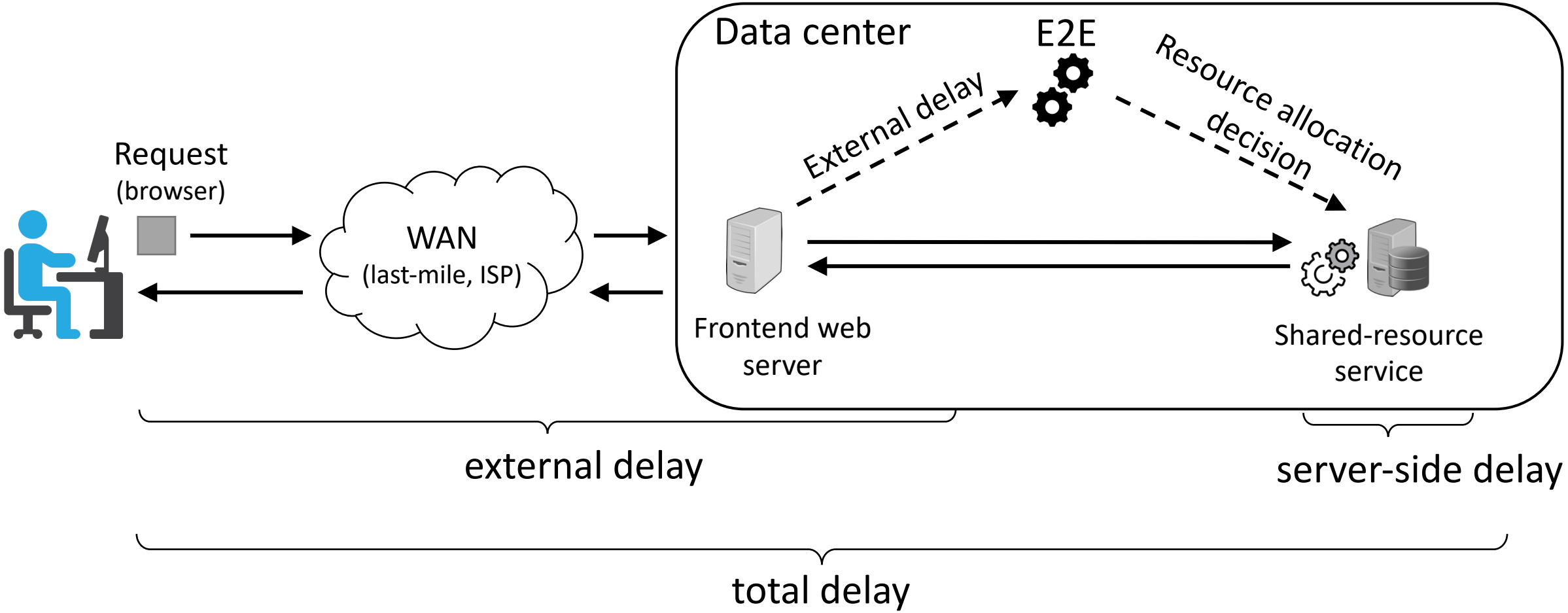
Idea: Focusing on more sensitive requests



Data center without E2E

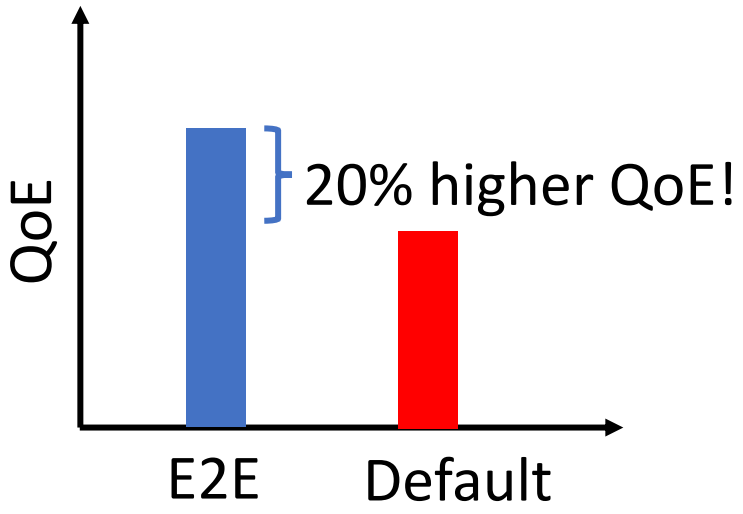


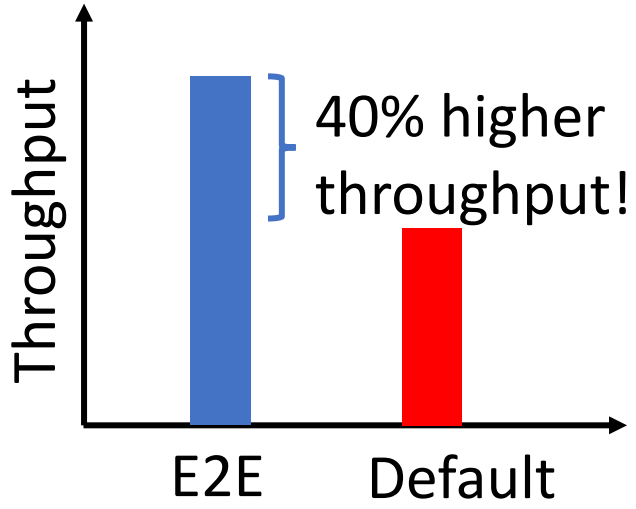
Data center with E2E



Potential gain

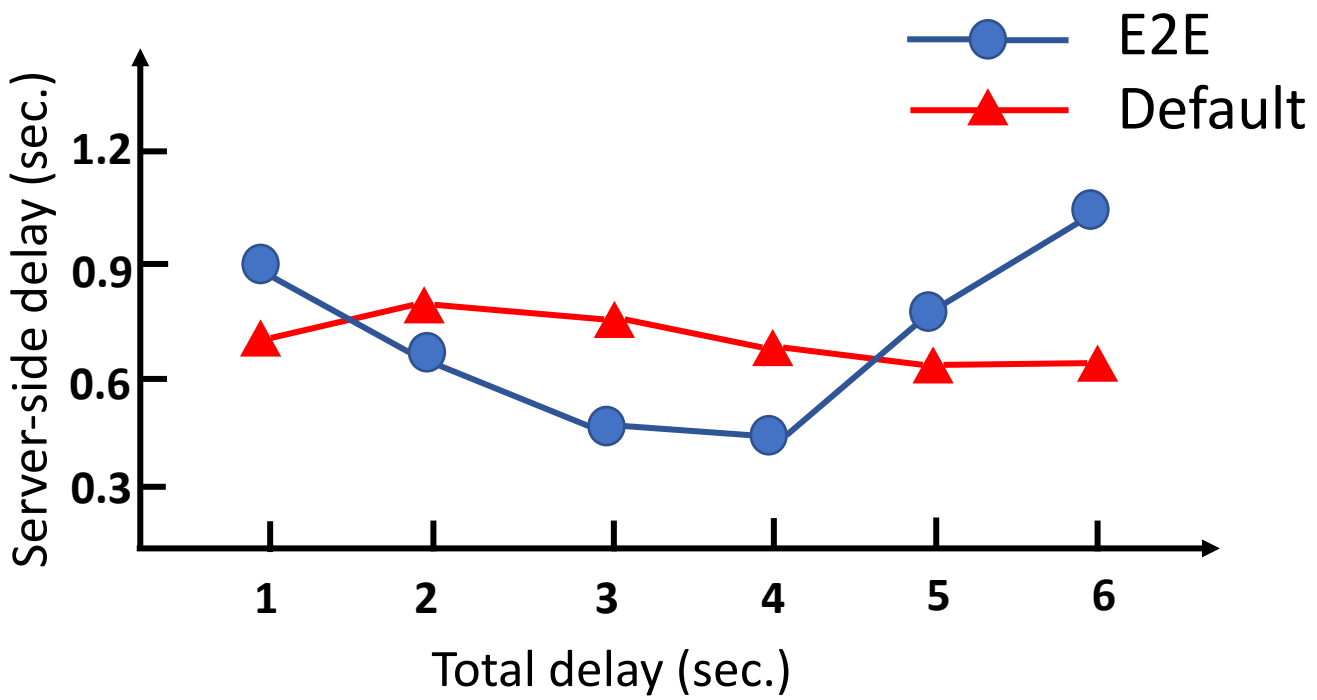
- We reshuffle the server-side delays between concurrent requests
- More sensitive requests get smaller delays





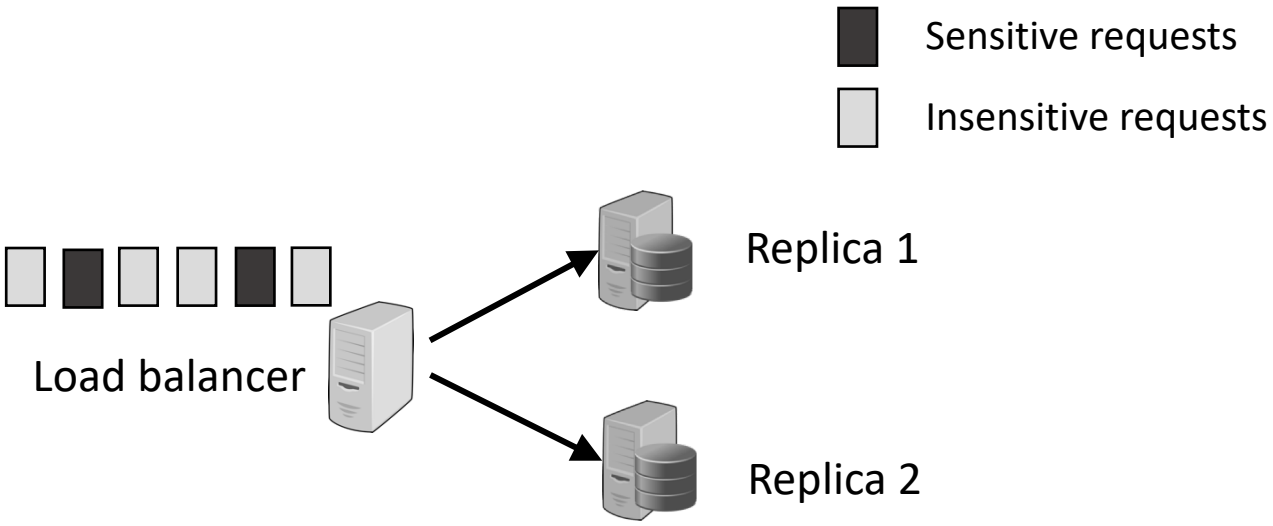
Our opportunity

- Current content providers do not distinguish the requests.



Case study: replica selection

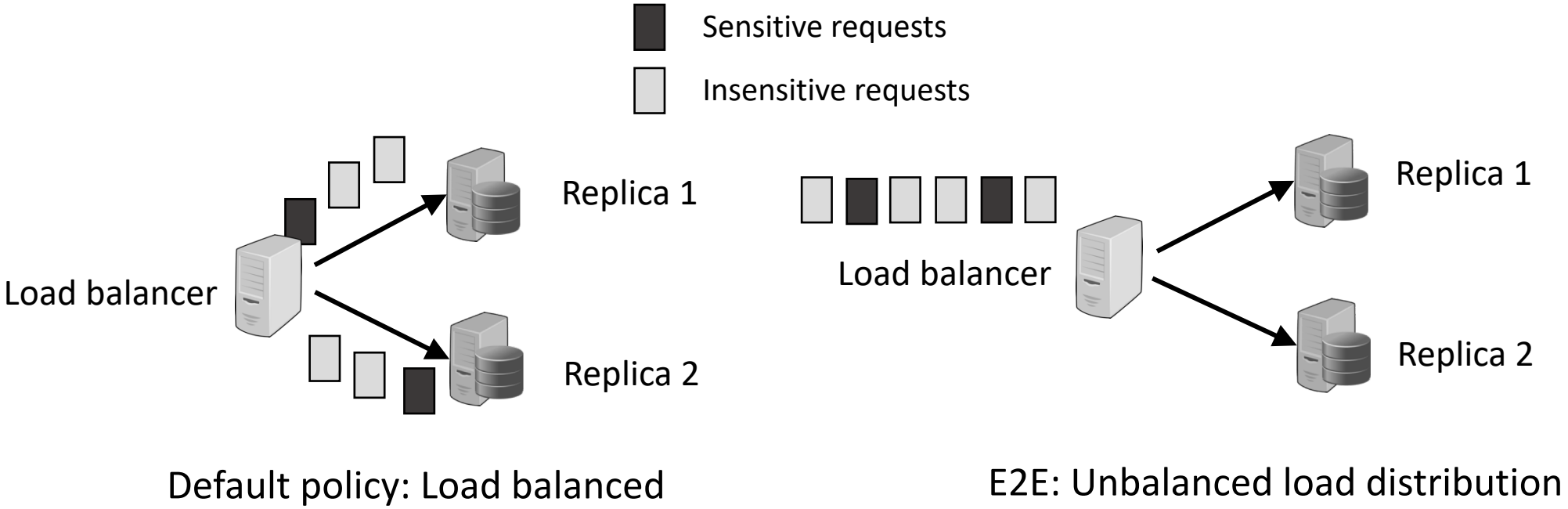
- Assign sensitive requests to the fast replica



Default policy: Load balanced

Case study: replica selection

- Assign sensitive requests to the fast replica

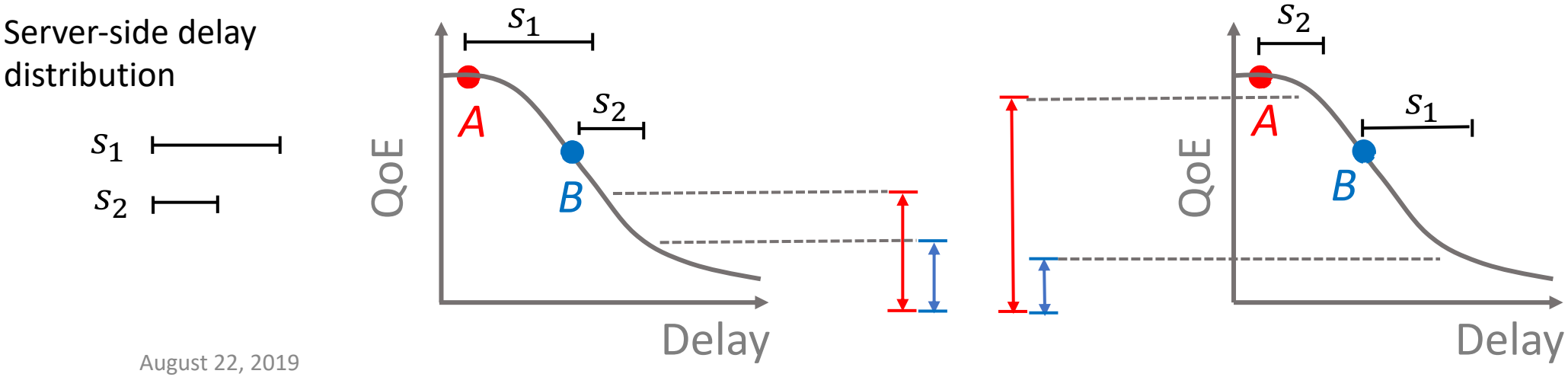


Default policy: Load balanced

E2E: Unbalanced load distribution

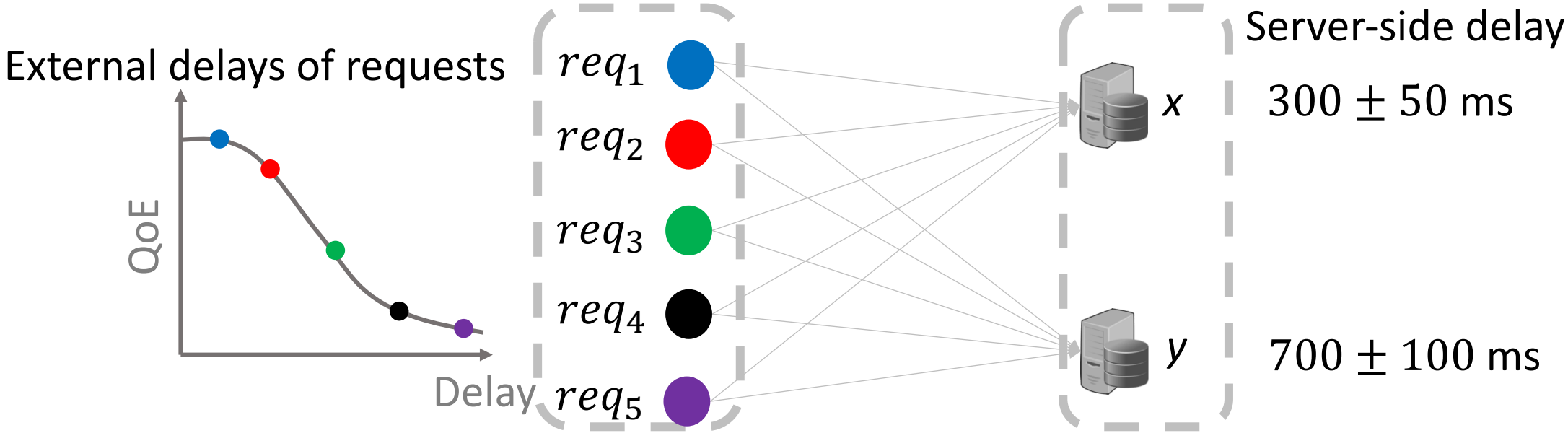
How do we decide a request's sensitivity?

- **Goal**: Sensitive requests will be sent to fast replicas
- **Challenge**: a request's sensitivity is not an inherent property
  - Strawman: A request's sensitivity is the slope of this request's external delay
- **Observation**: The optimal replica selection depends on the server-side delay distribution.



How to select replicas for heterogenous requests?

- Send requests to replicas
- Maximize $\sum_i QoE(external\_delay_i + server\_delay_i)$
- Classical maximum bipartite graph matching problem



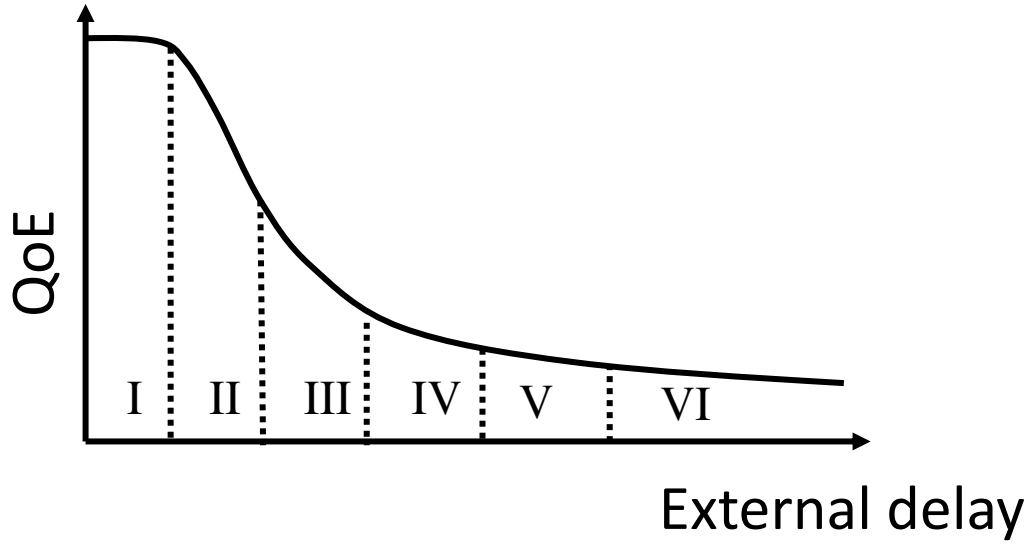
Need to reduce the decision-making overhead!

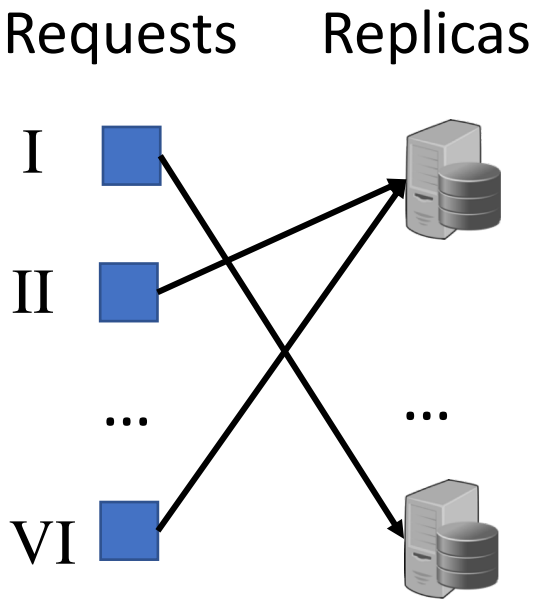
Reduce the time consumption of running request-replica matching algorithm

Reduce the frequency of decision-making

Idea #1: Grouping requests by their external delays

- spatial coarsening of E2E decision-making





Idea #2: Reducing decision update frequency

- Temporal coarsening of E2E decision making
- Cache decision
- No need to compute the table per request

External delay	Cached Decision
<500ms	Replica_x
500-1200ms	Replica_y
>1200ms	Replica_x

Evaluation

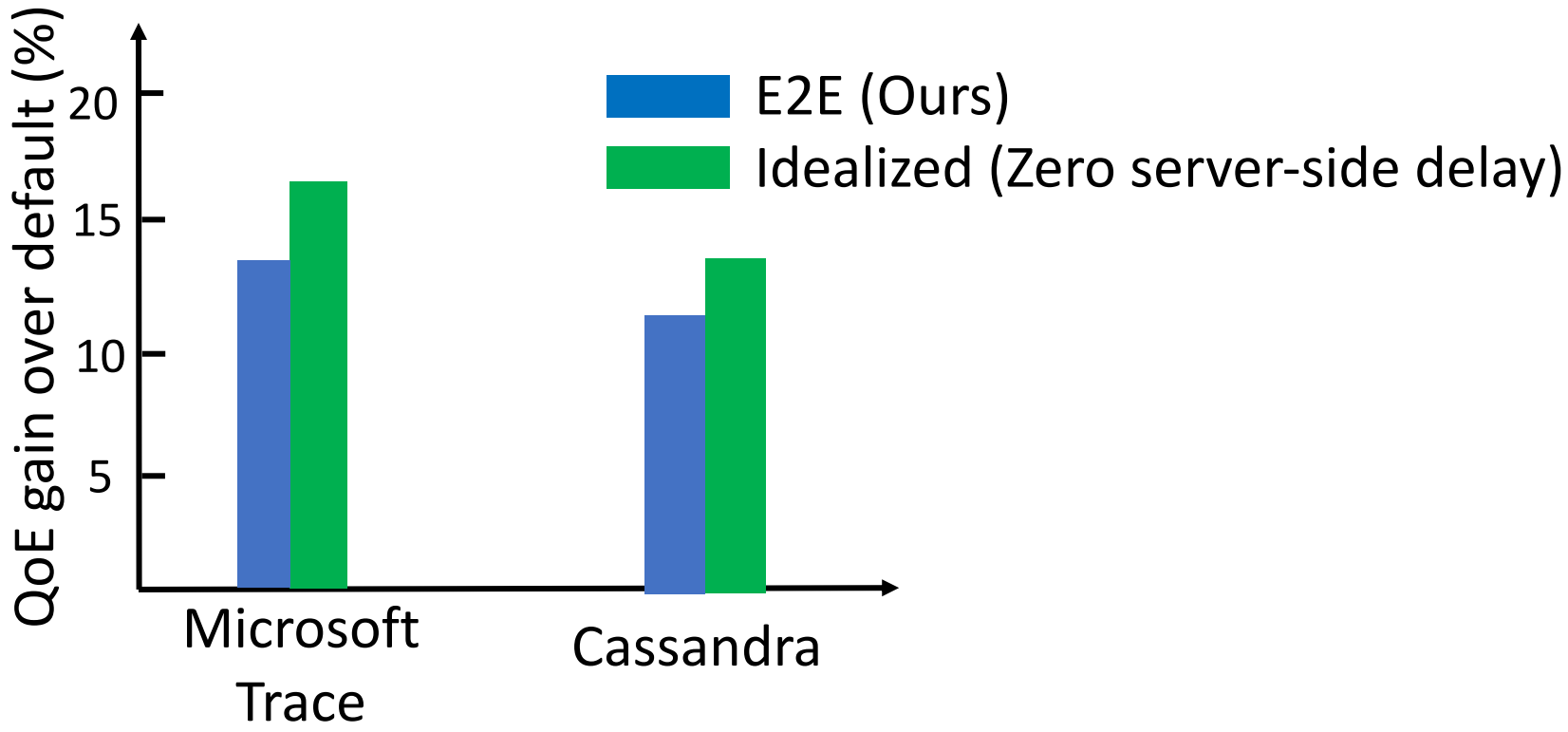
Set up
- Dataset: Real-world external delays from Microsoft traces
- Benchmark
  - Default: Load balanced replica selection
  - Idealized: Server-side delay is zero

Performance evaluation
- Overall performance
- E2E vs prior work
- E2E's overhead

Overall performance of E2E

Microsoft trace: reshuffle the server-side delays vs default server-side delays

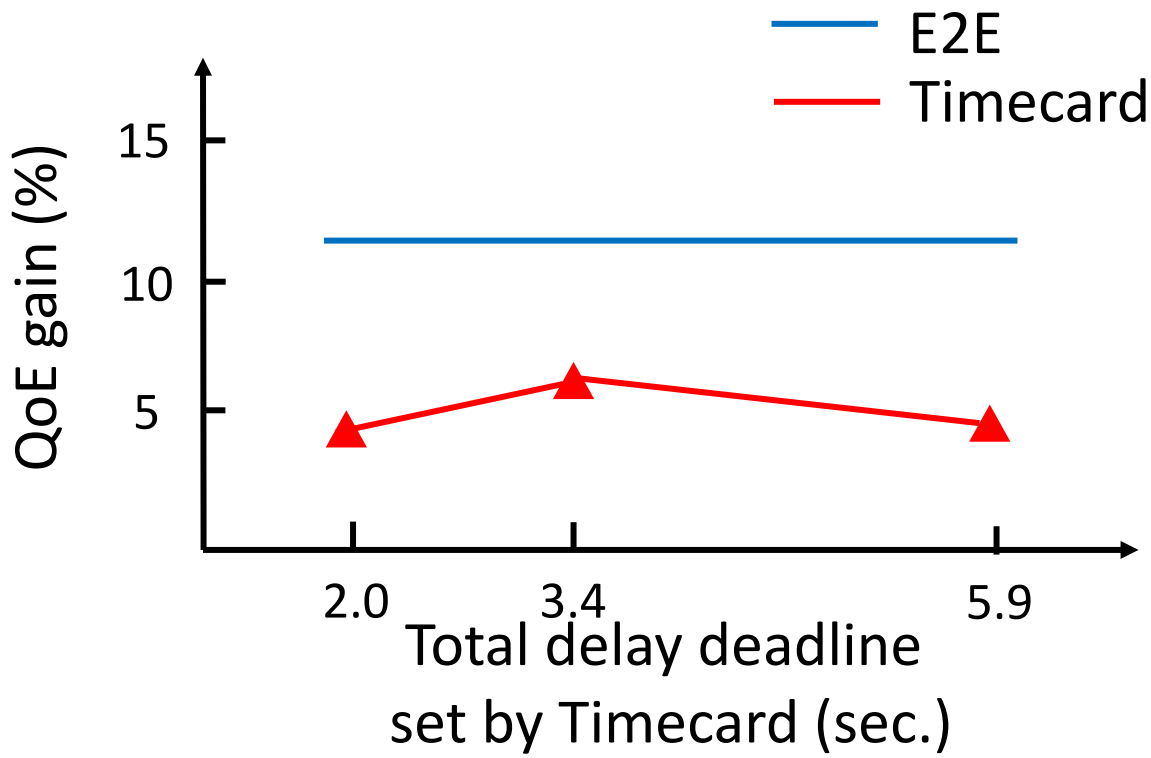
Distributed database: replica selection in Cassandra



E2E vs Prior work

E2E vs deadline-driven algorithm (Timecard [SOSP'13])

Timecard: shortest-remaining time first

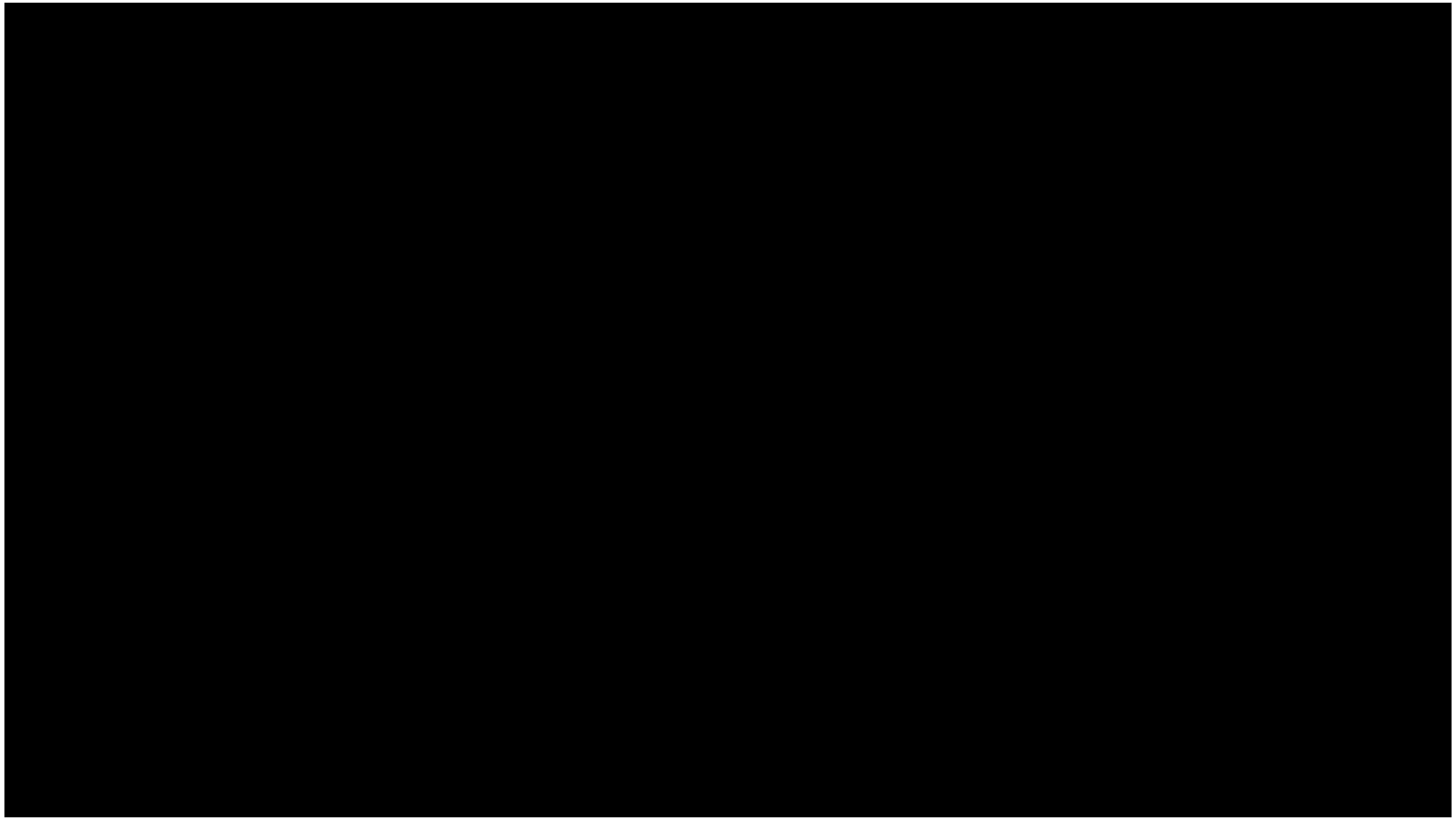


E2E's overhead

- Machines in testbed: 3.0GHz Intel Xeon processor, 2GB RAM, 2GB RAM, 146G HDD and 1Gbps Ethernet link.

	Time consumption per request (ms)	Additional memory	Additional CPU	QoE gain
E2E (basic)	~100,000	~100%	>100%	11.8%
E2E w/ grouping requests & cache decision	**~0.1**	**~7%**	**~2%**	**10.4%**

Demo: How E2E works



Conclusion

Concurrent users have **different** sensitivities to server-side delays

Key idea: **Embracing heterogenous user sensitivities leads to higher QoE**

E2E: A concrete design to improve web QoE by allocating resource in accordance to user sensitivity

E2E improves QoE by up-to 15.4%, with negligible computing overhead

More details about E2E can be found in: https://people.cs.uchicago.edu/~zhangxu/e2e.html