Event-Based Scheduling for Energy-Efficient Quality of Service (eQoS) in Mobile Web Applications

Yuhao Zhu, Matthew Halpern, Vijay Janapa Reddi
Department of Electrical and Computer Engineering
The University of Texas at Austin

HPCA — Feb. 9th, 2015

TRINITY

Responsiveness

Responsiveness

- 100 ms latency is the limit for having users feel the system is reacting responsively [Miller 1968; Card et al. 1991]

- 100 ms latency is the limit for having users feel the system is reacting responsively [Miller 1968; Card et al. 1991]

Responsiveness

- 64% of mobile users will not revisit a slow Web app [Source: Akamai]

- 100 ms latency is the limit for having users feel the system is reacting responsively [Miller 1968; Card et al. 1991]

Responsiveness

- 64% of mobile users will not revisit a slow Web app [Source: Akamai]

- Amazon found every 100 ms of latency costs them 1% in sales per year [Source: Amazon]

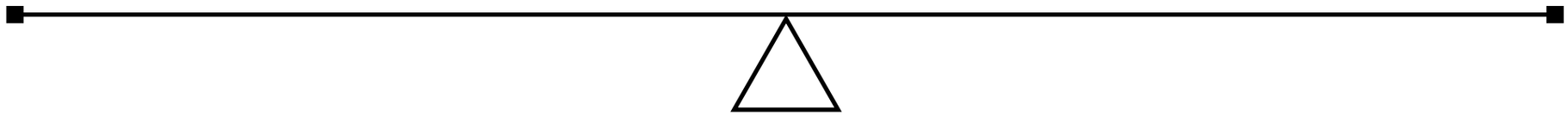
Responsiveness

Energy-Efficiency

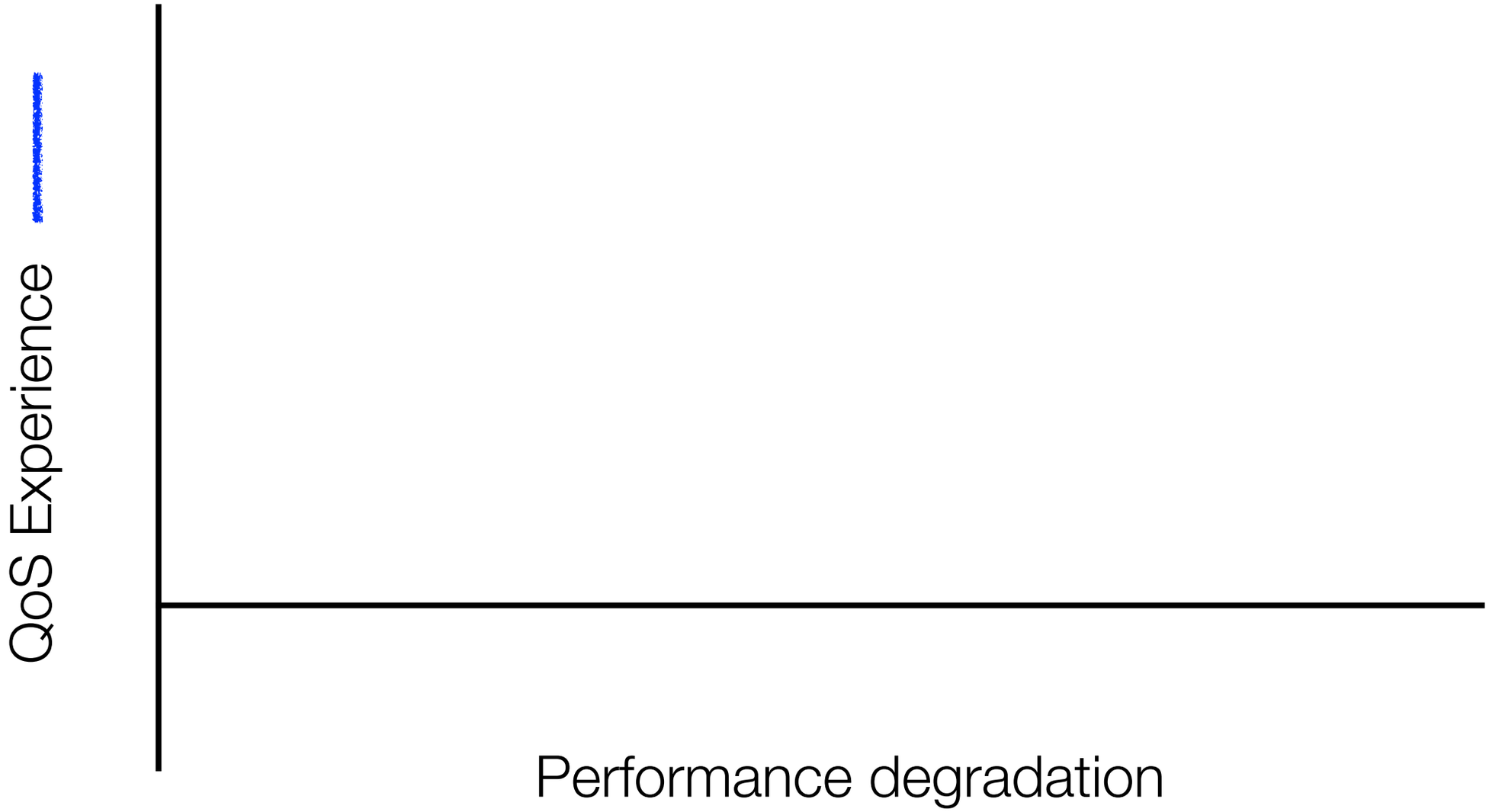
Responsiveness

Conflicting requirements

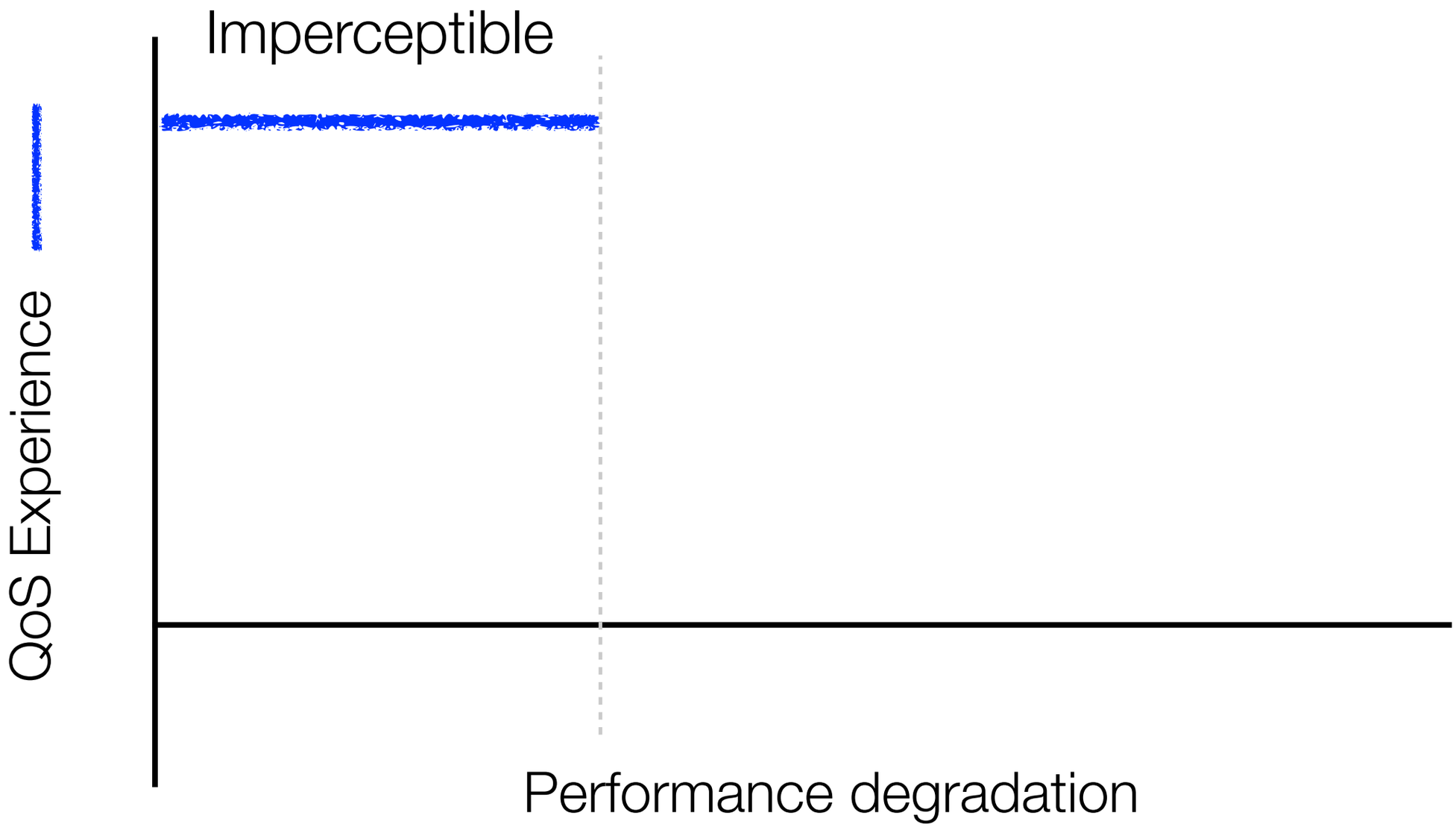
Energy-Efficiency



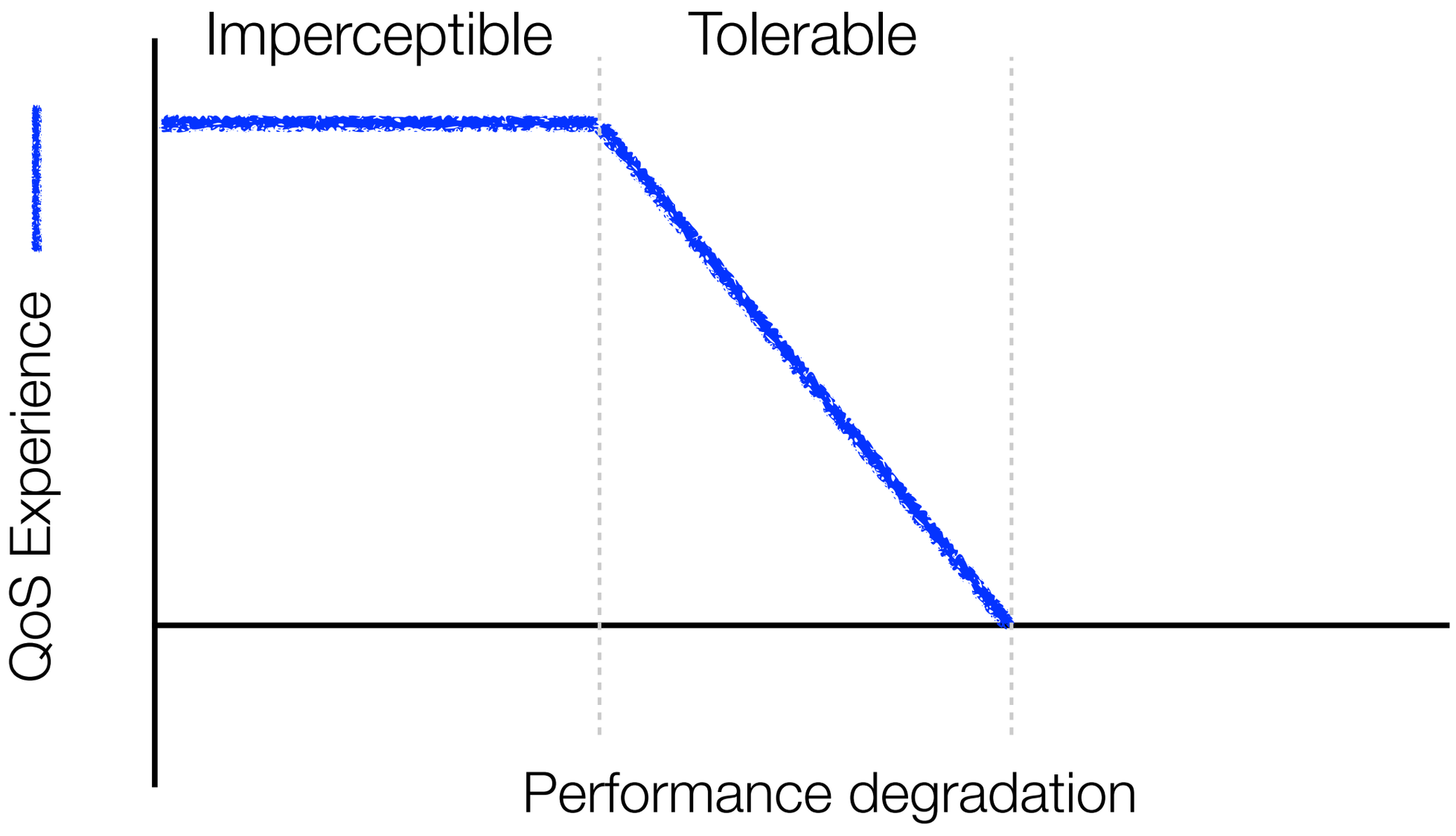
Making a Calculated Trade-off



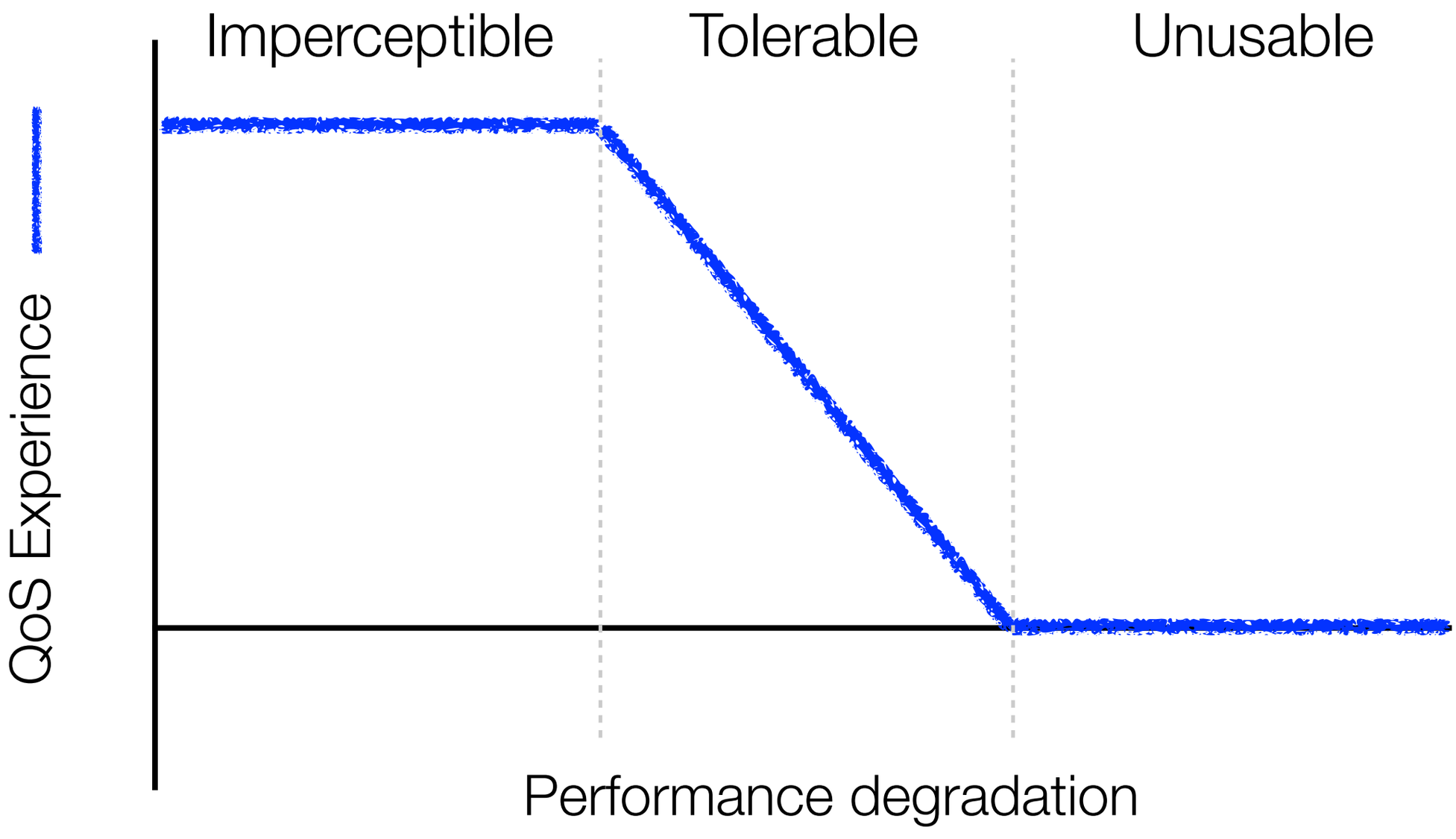
Making a Calculated Trade-off



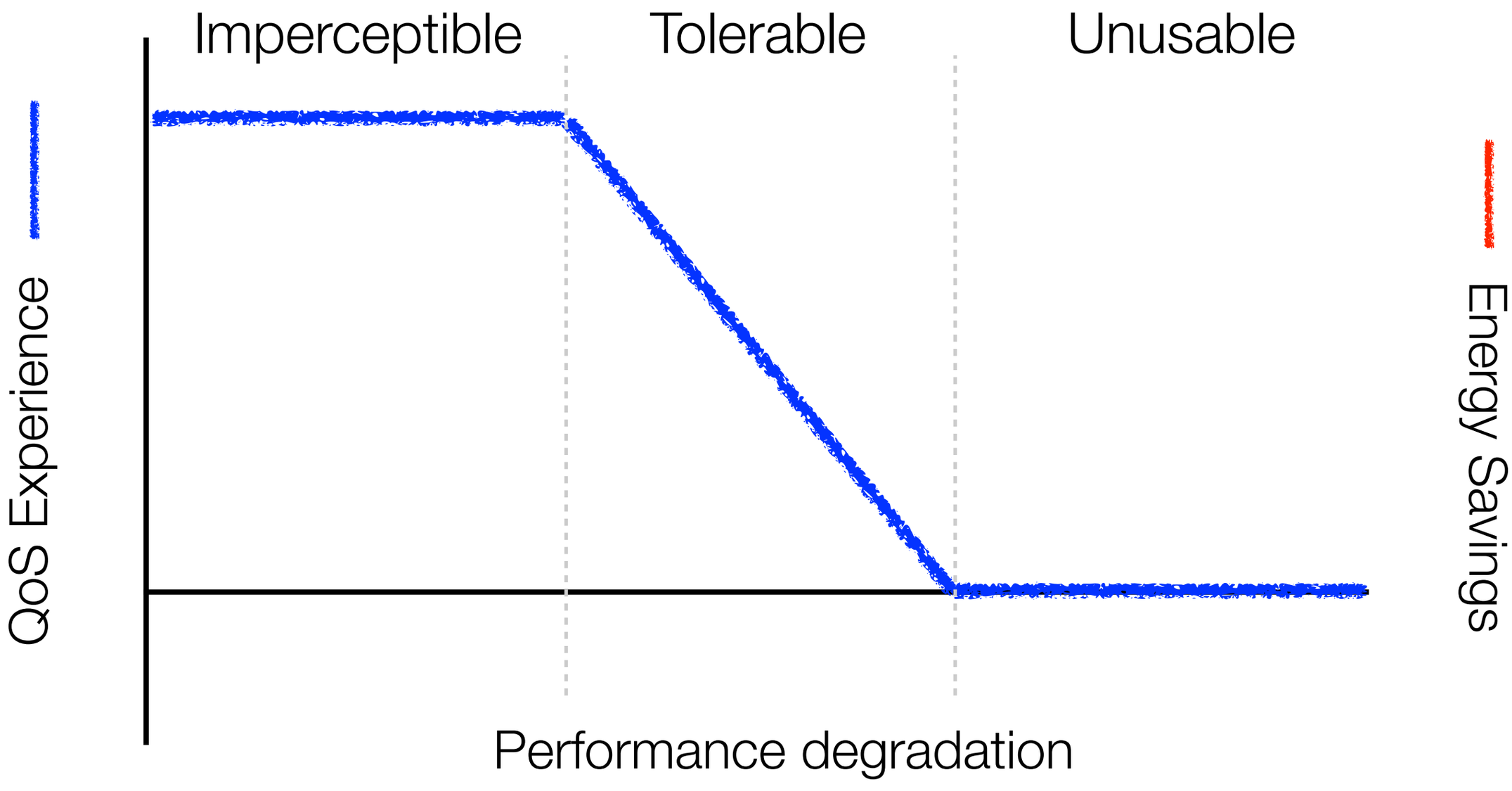
Making a Calculated Trade-off



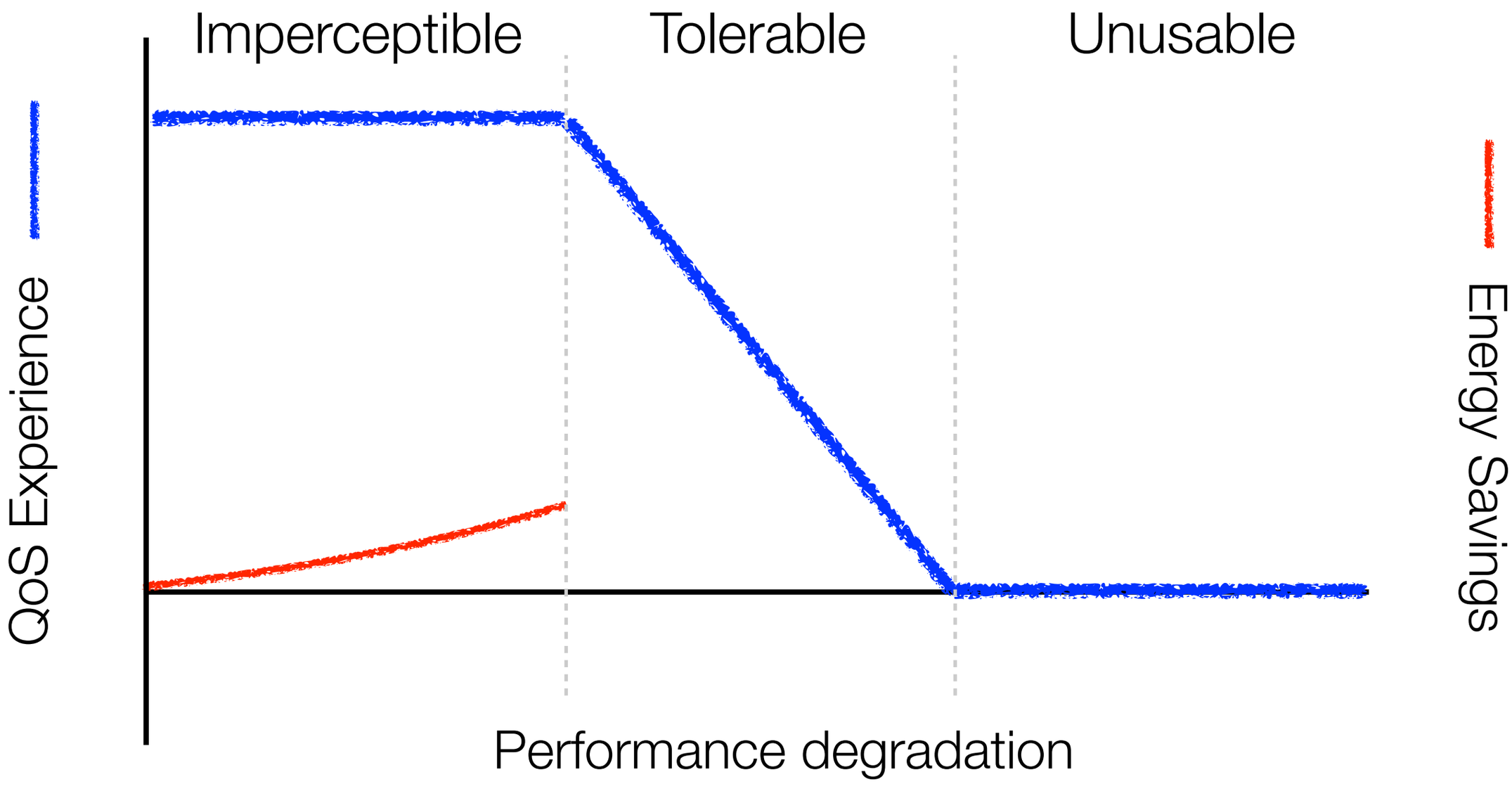
Making a Calculated Trade-off



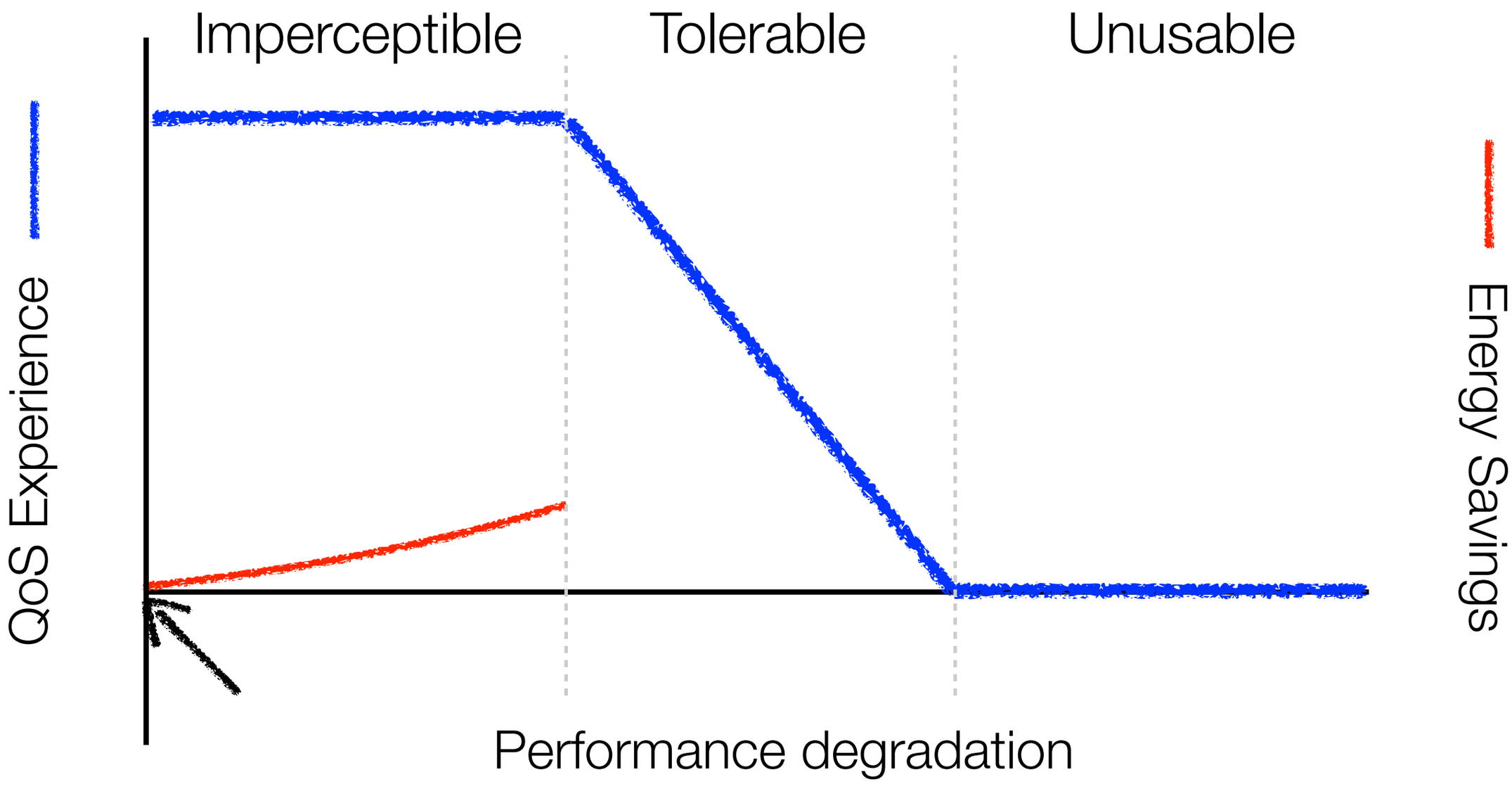
Making a Calculated Trade-off



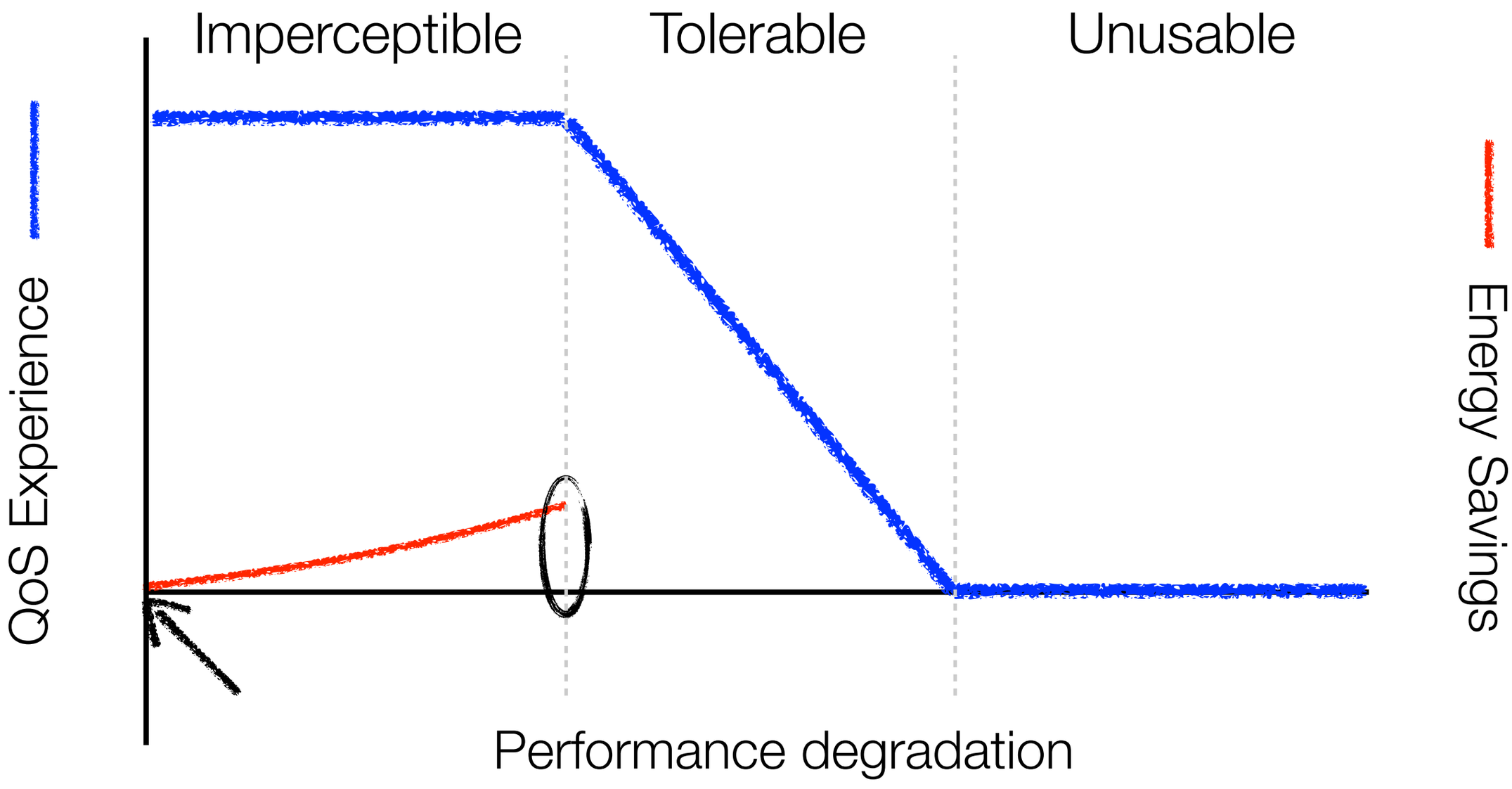
Making a Calculated Trade-off



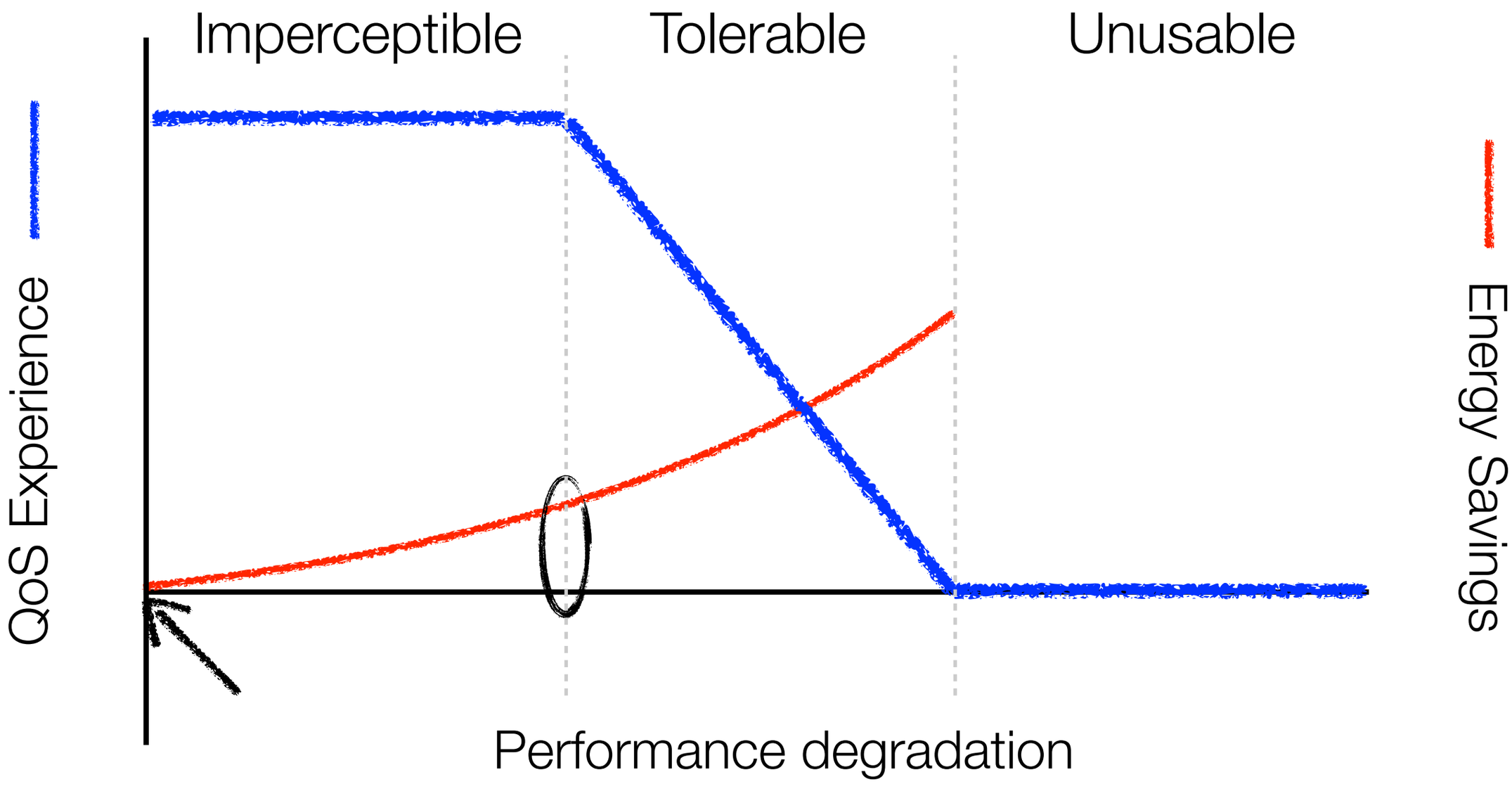
Making a Calculated Trade-off



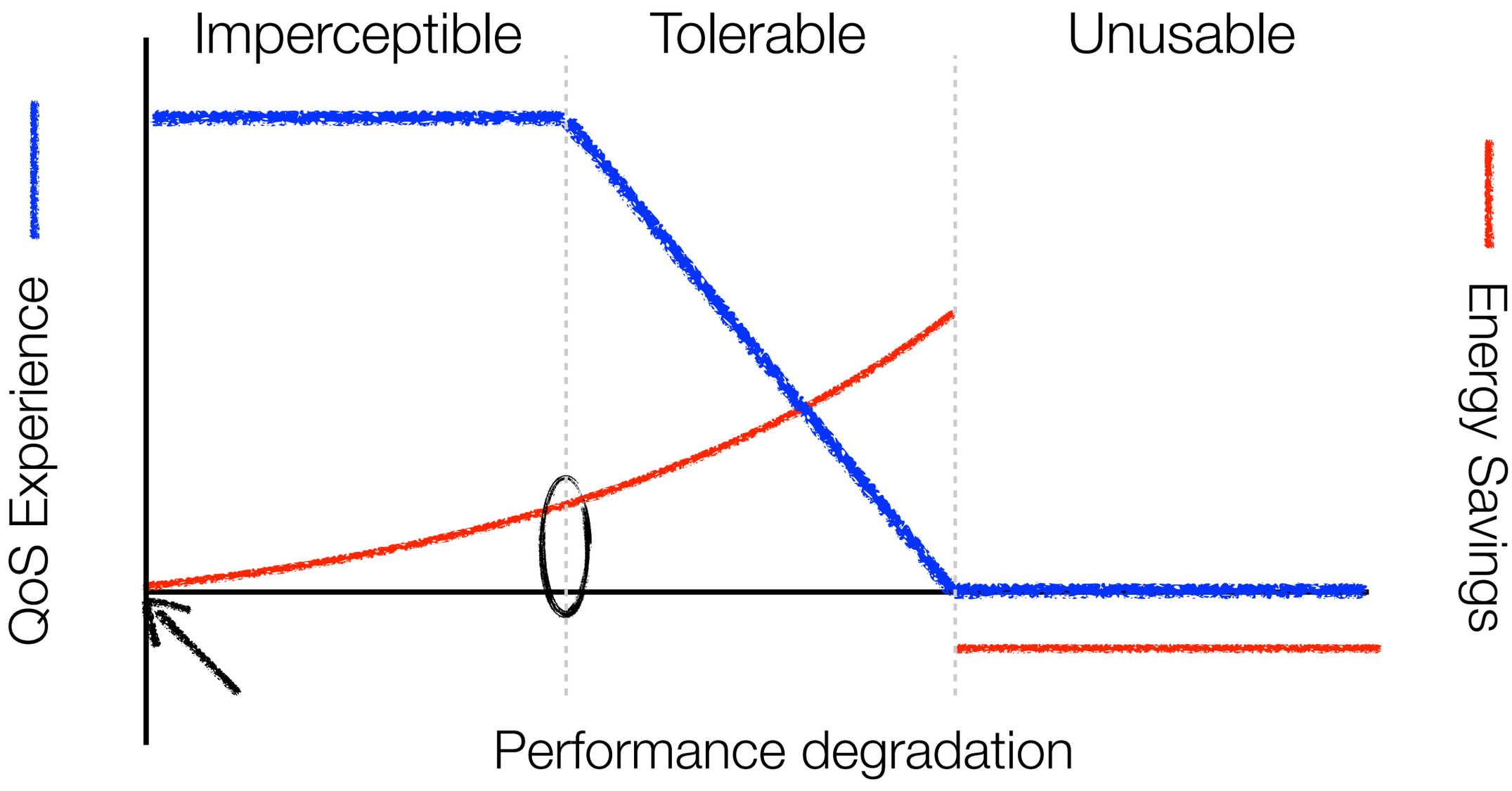
Making a Calculated Trade-off



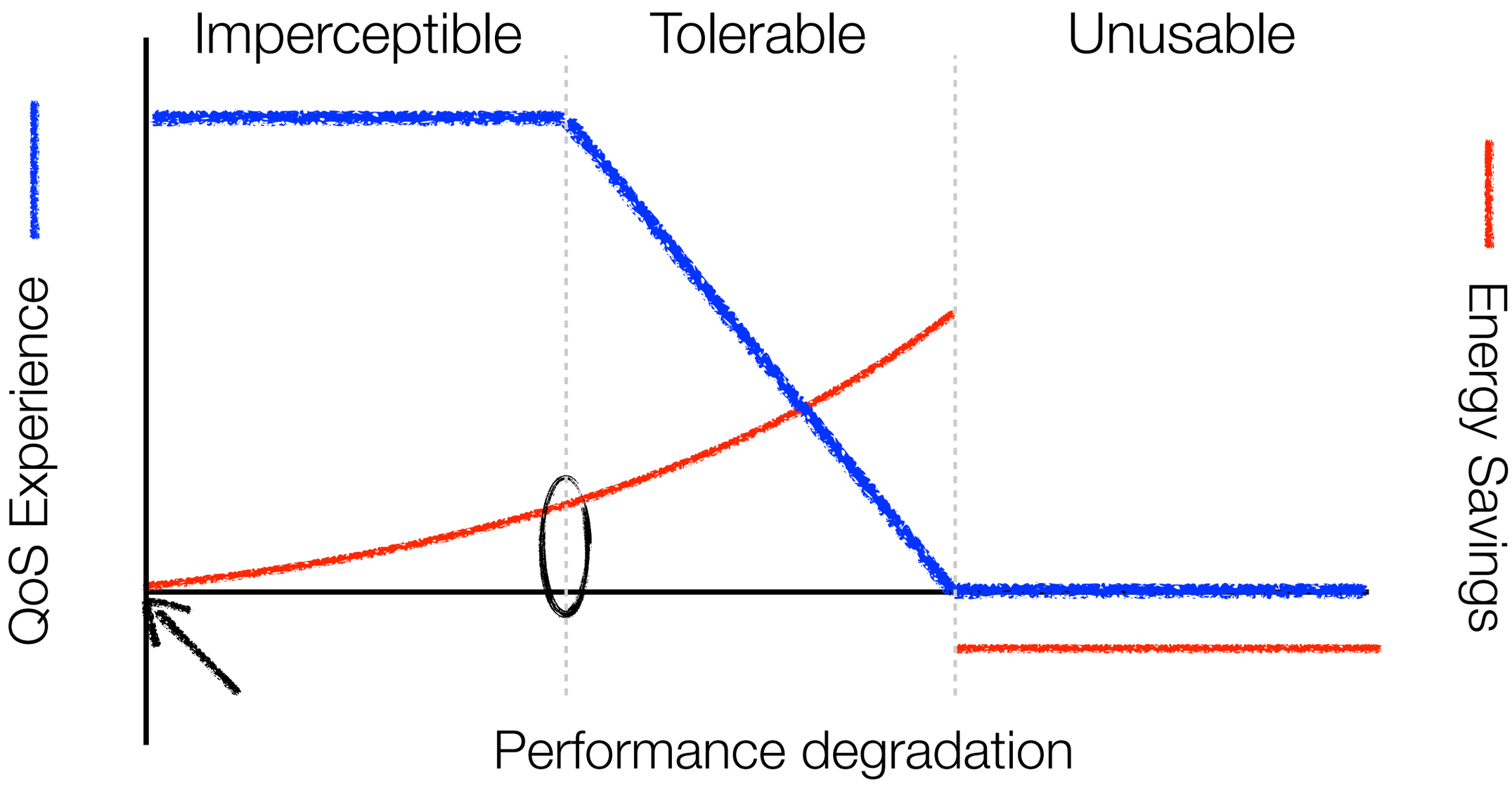
Making a Calculated Trade-off



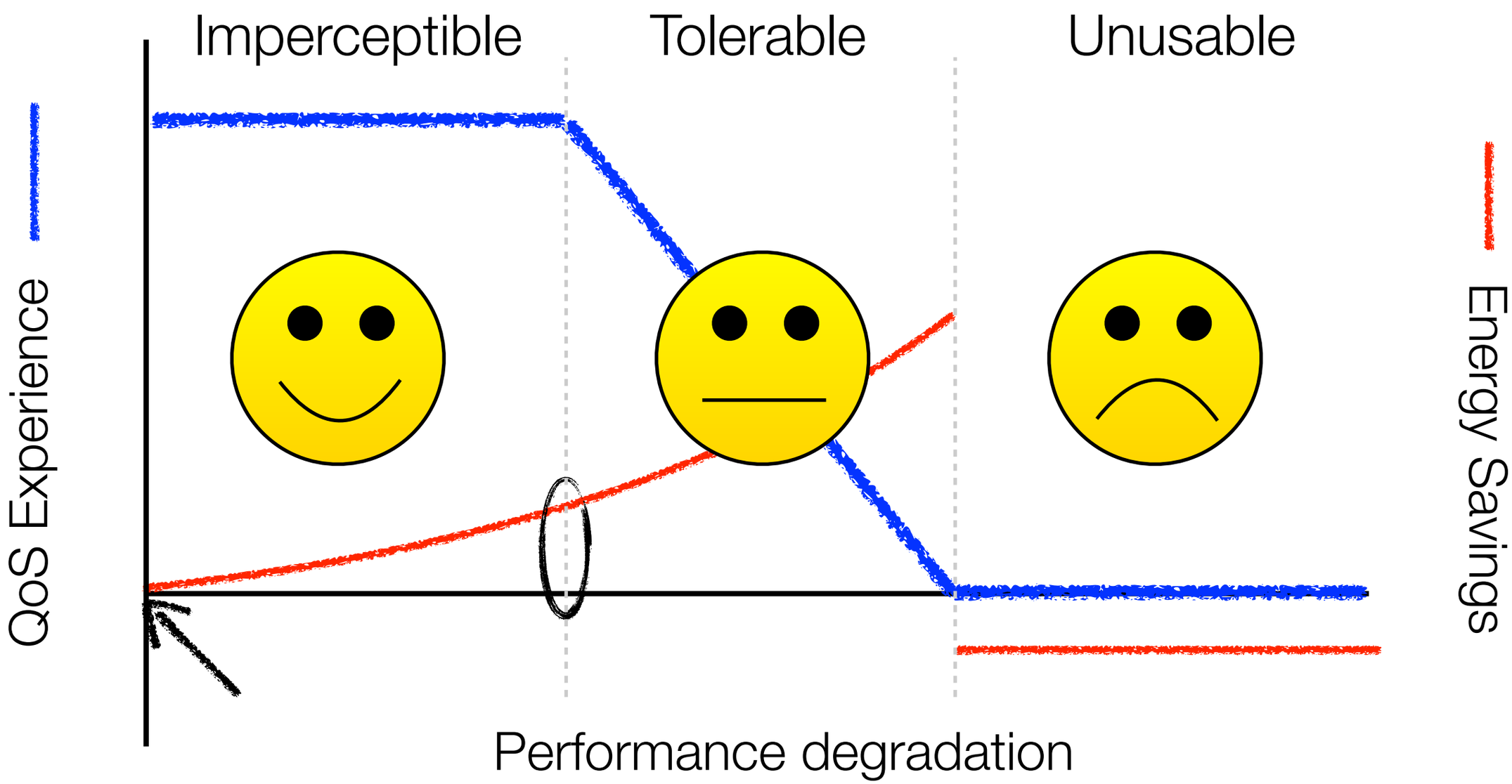
Making a Calculated Trade-off



eQoS: Making a Calculated Trade-off

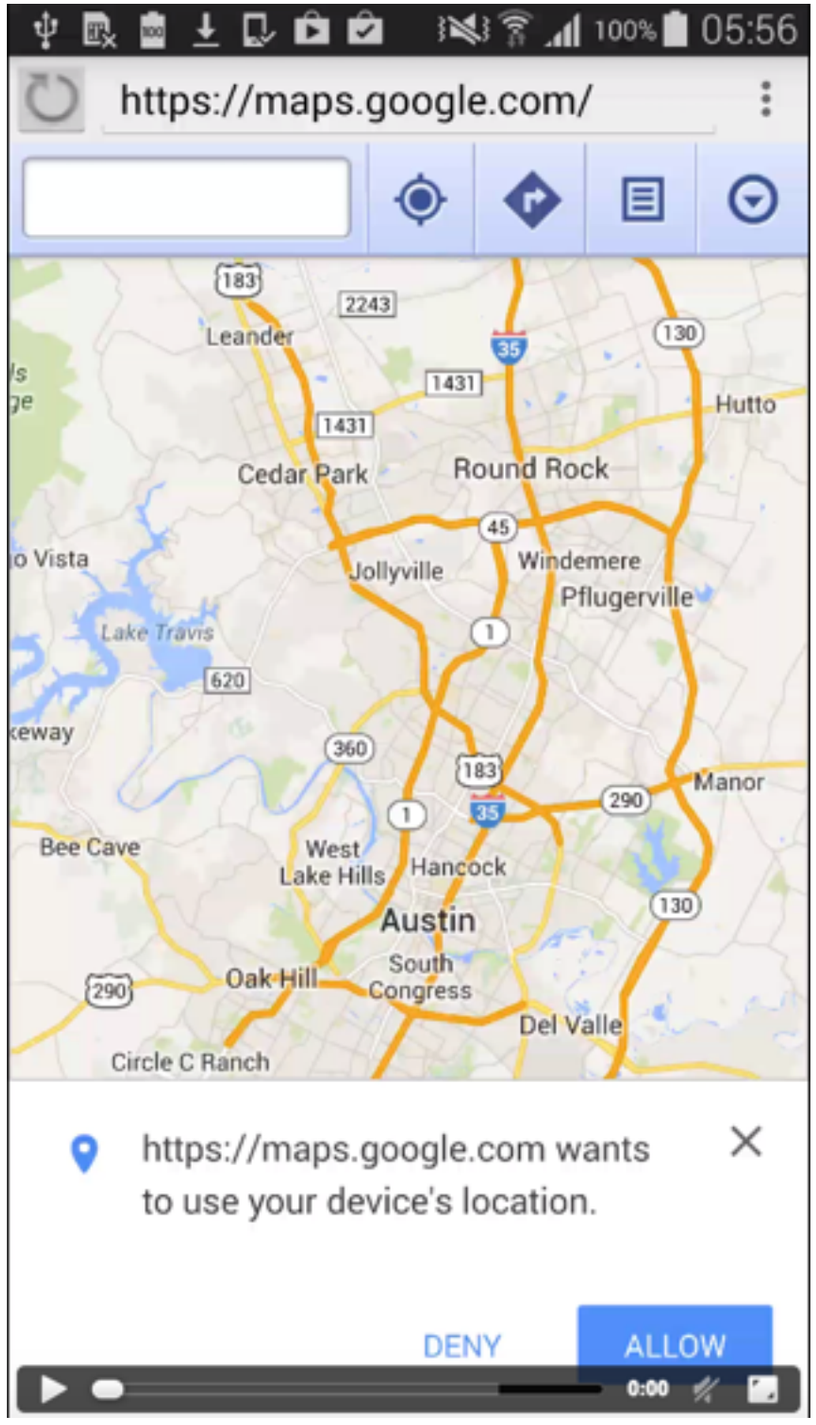


eQoS: Making a Calculated Trade-off



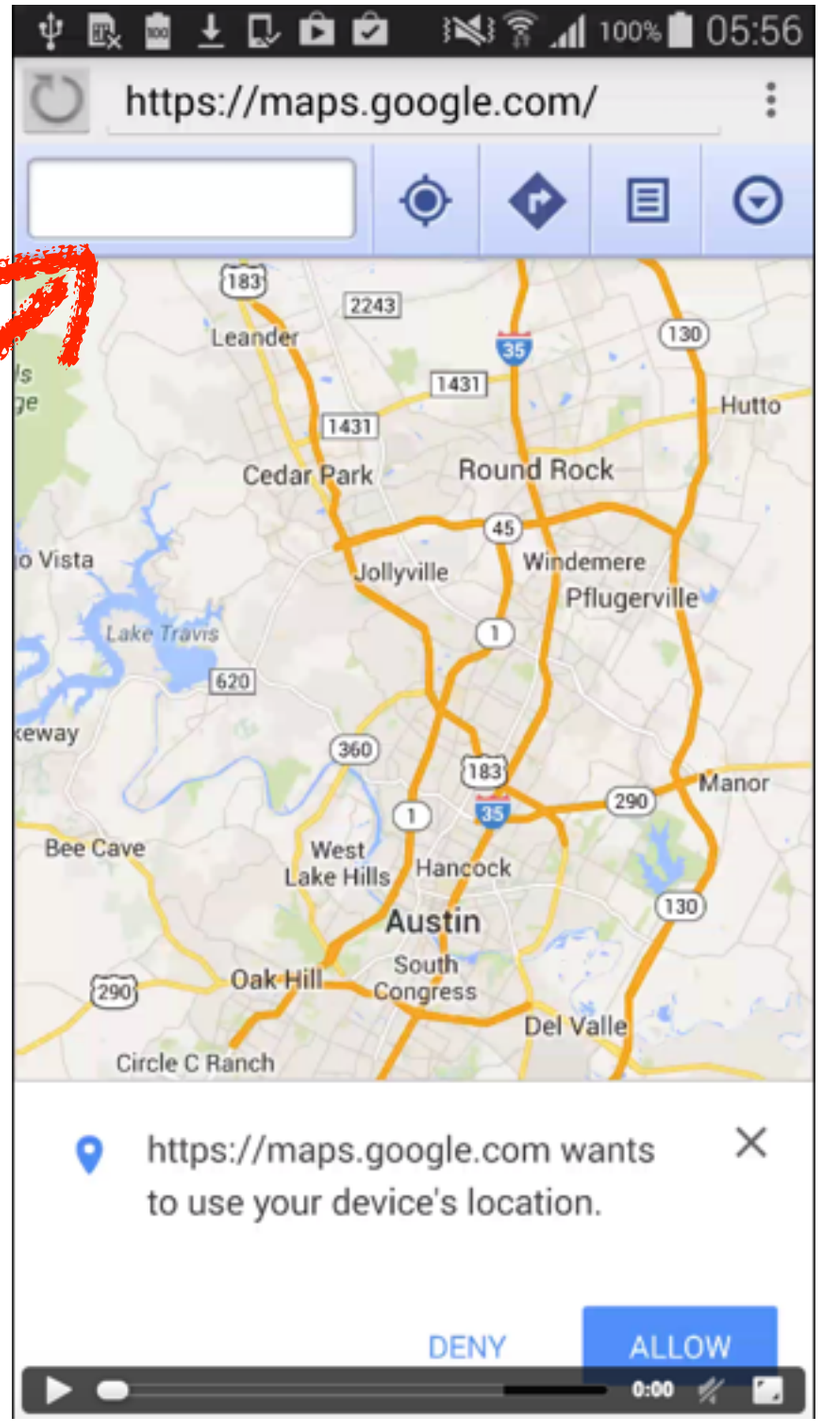
Interacting with an App

Interacting with an App



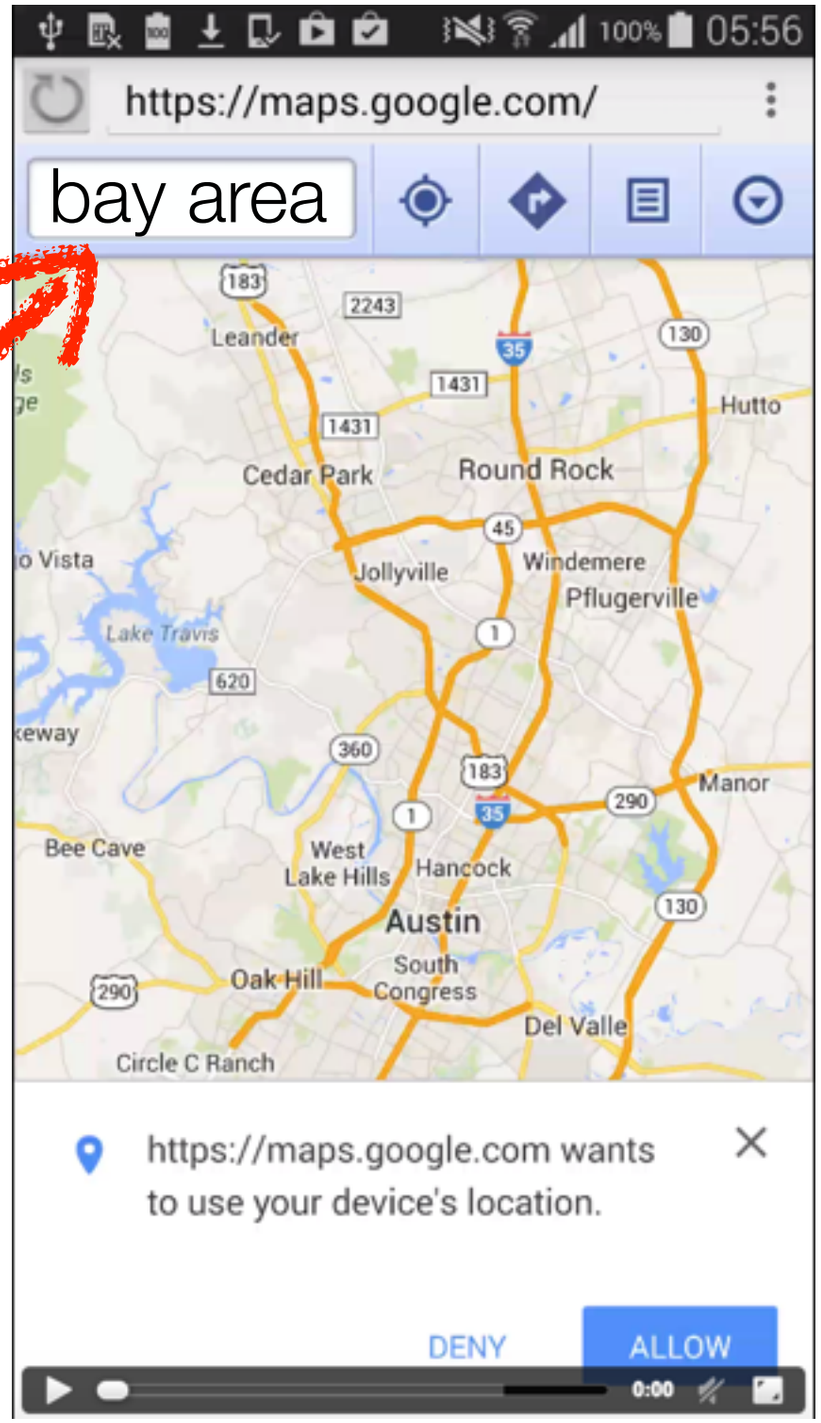
http://yuhaozhu.com/googlemaps.mp4

Interacting with an App



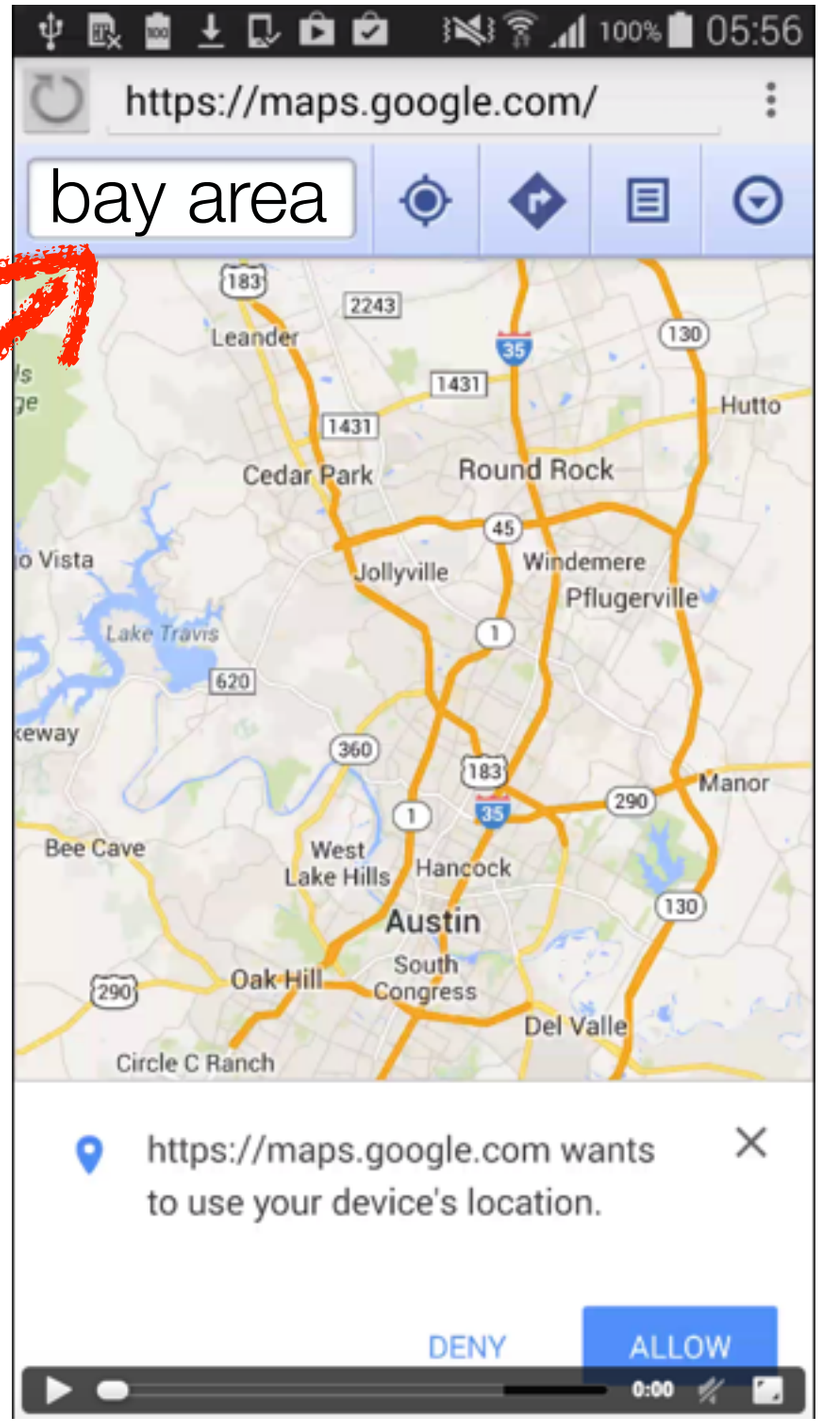
http://yuhaozhu.com/googlemaps.mp4

Interacting with an App



http://yuhaozhu.com/googlemaps.mp4

Interacting with an App

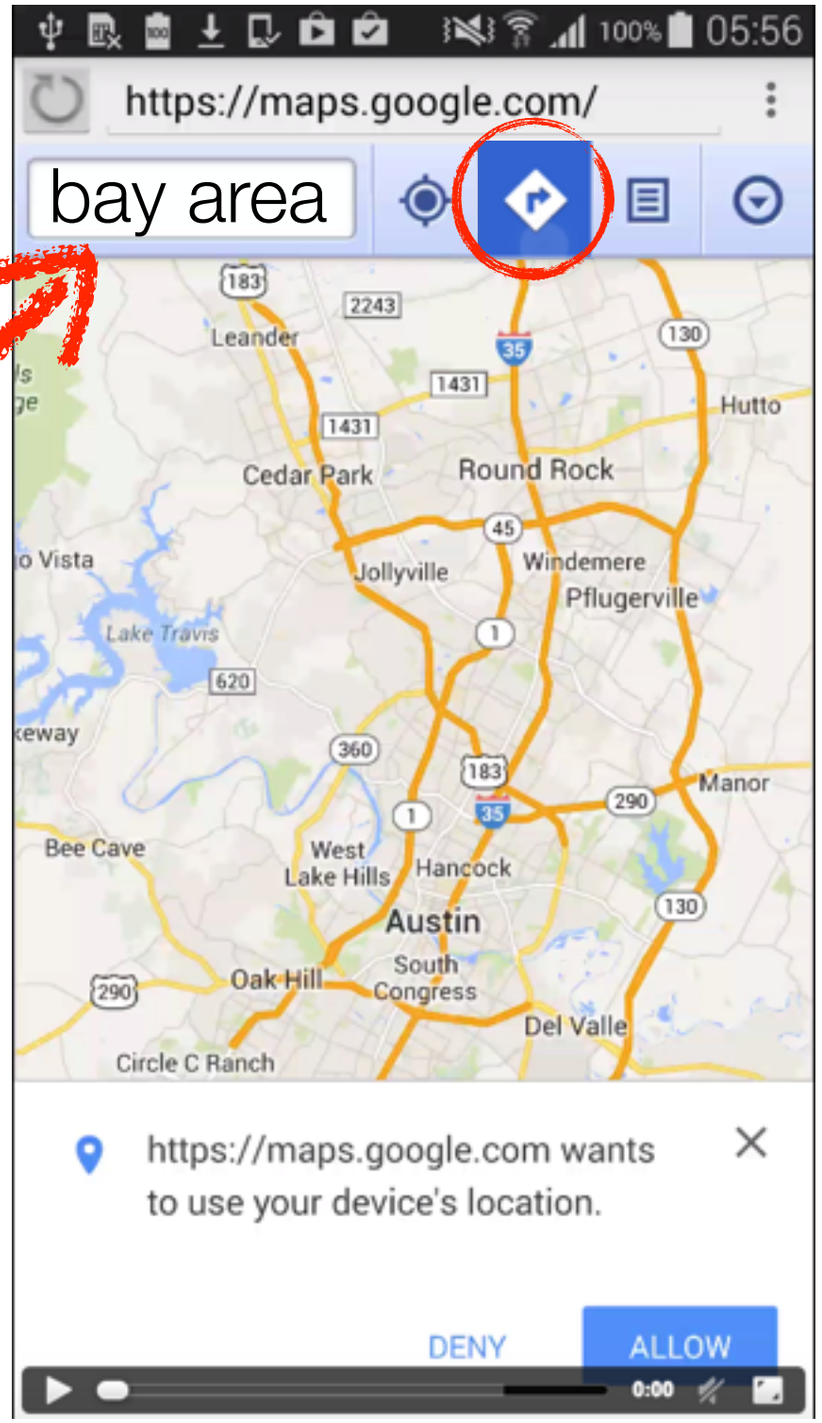


Interactions

typing

http://yuhaozhu.com/googlemaps.mp4

Interacting with an App

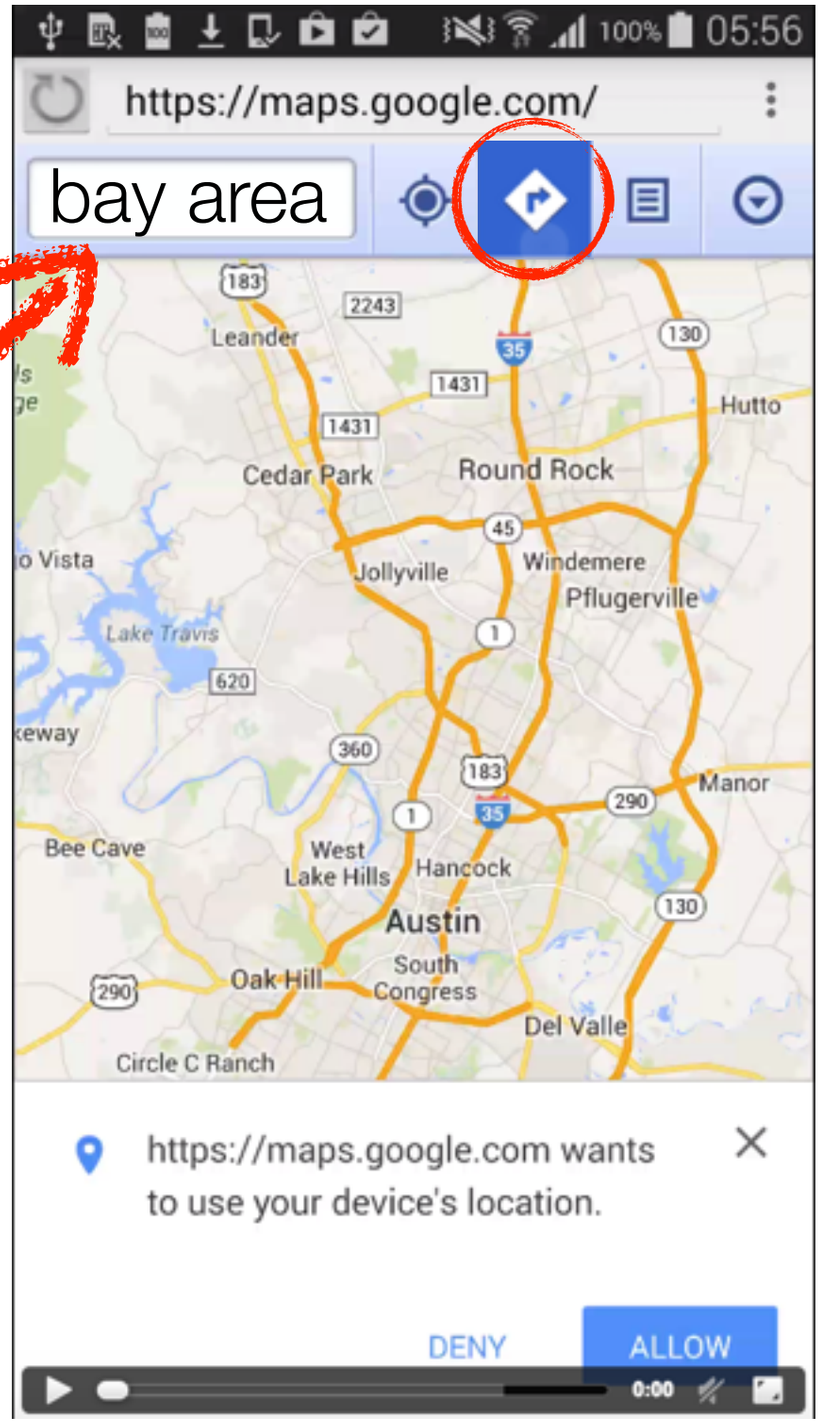


Interactions

typing

http://yuhaozhu.com/googlemaps.mp4

Interacting with an App



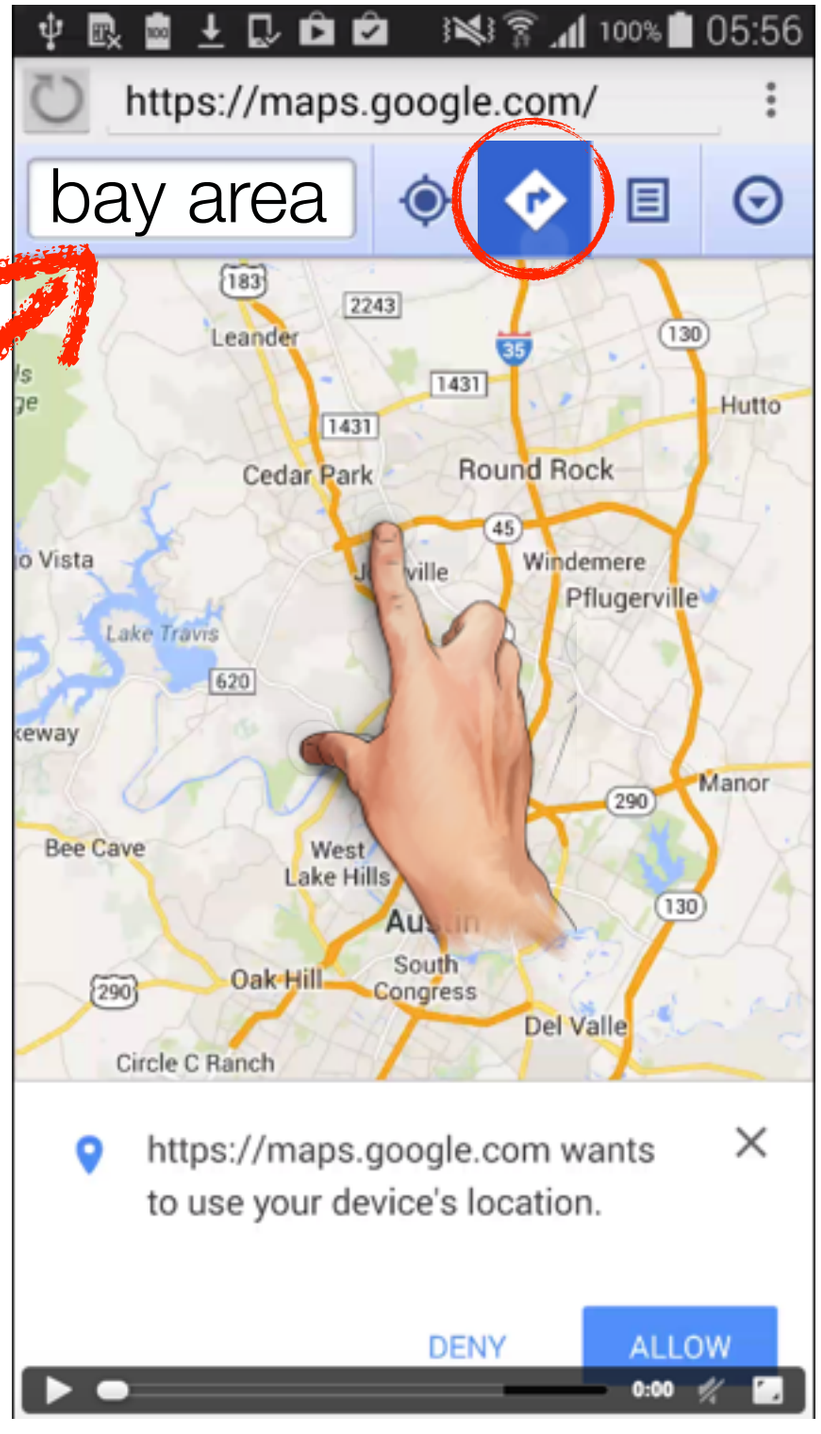
Interactions

typing

pressing

http://yuhaozhu.com/googlemaps.mp4

Interacting with an App



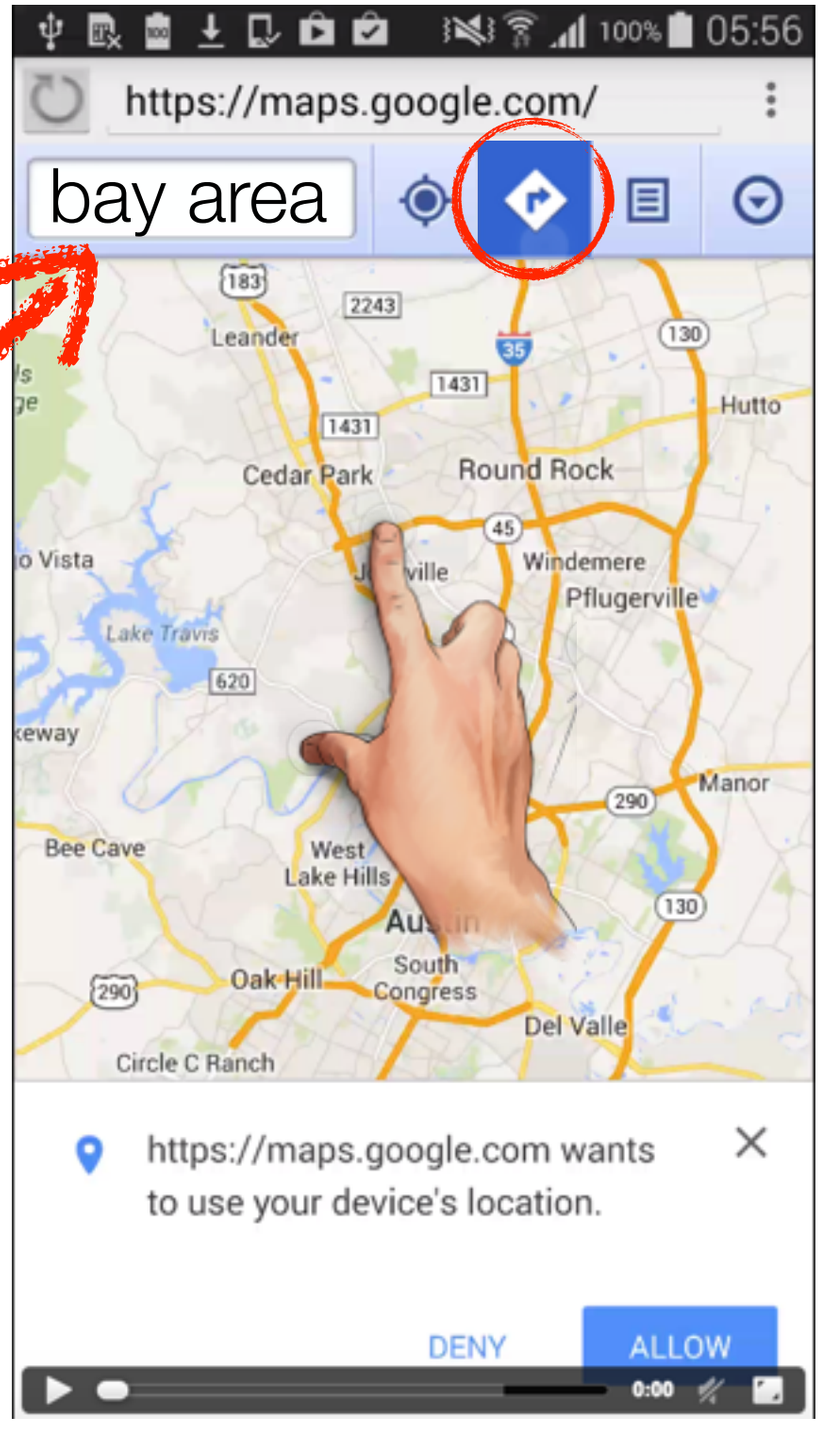
Interactions

typing

pressing

http://yuhaozhu.com/googlemaps.mp4

Interacting with an App



Interactions

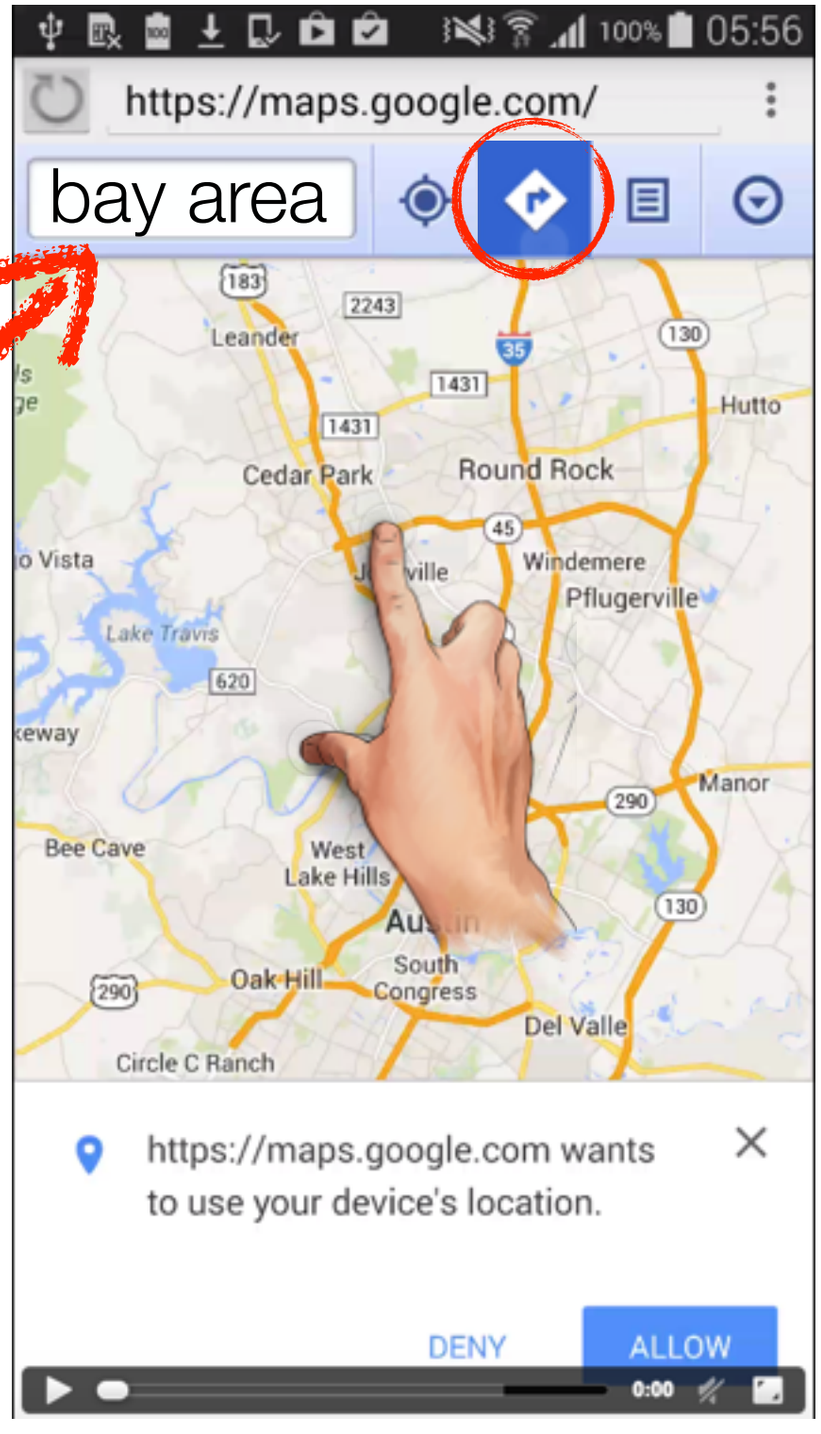
typing

pressing

moving

http://yuhaozhu.com/googlemaps.mp4

Interacting with an App



Interactions → Events

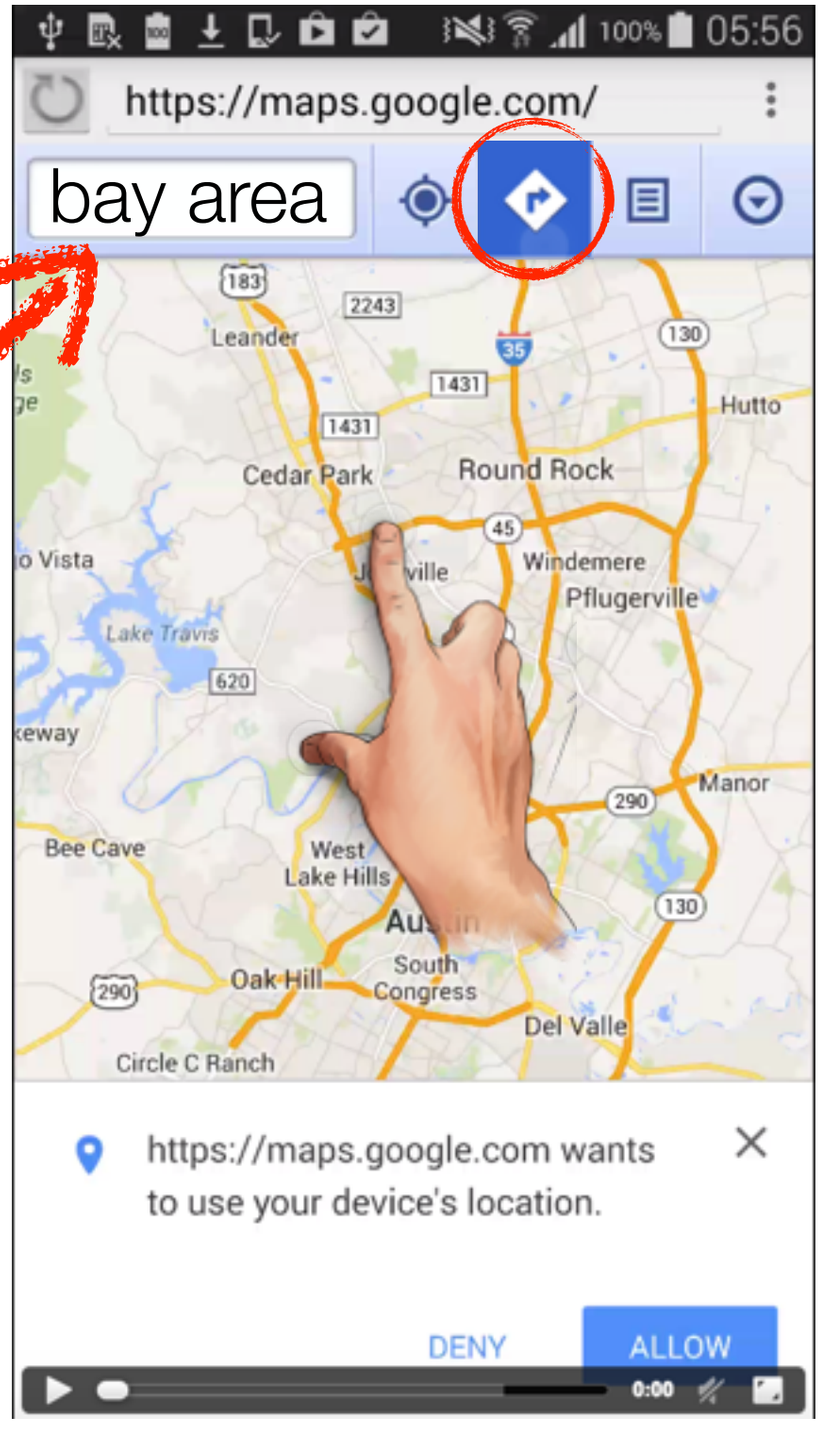
typing

pressing

moving

http://yuhaozhu.com/googlemaps.mp4

Interacting with an App



Interactions → Events

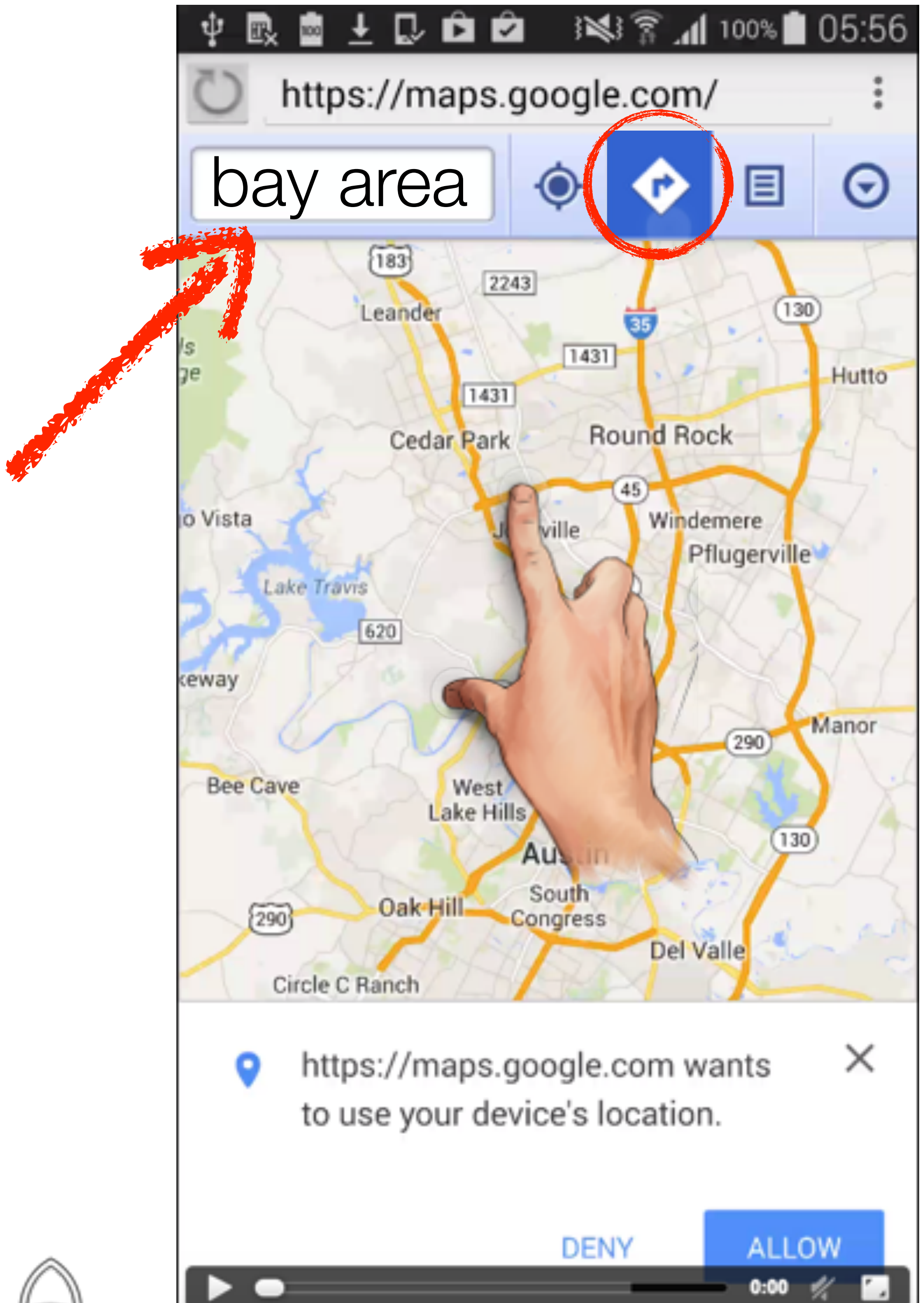
typing → `keypress`

pressing → `click`

moving → `touchmove`

http://yuhaozhu.com/googlemaps.mp4

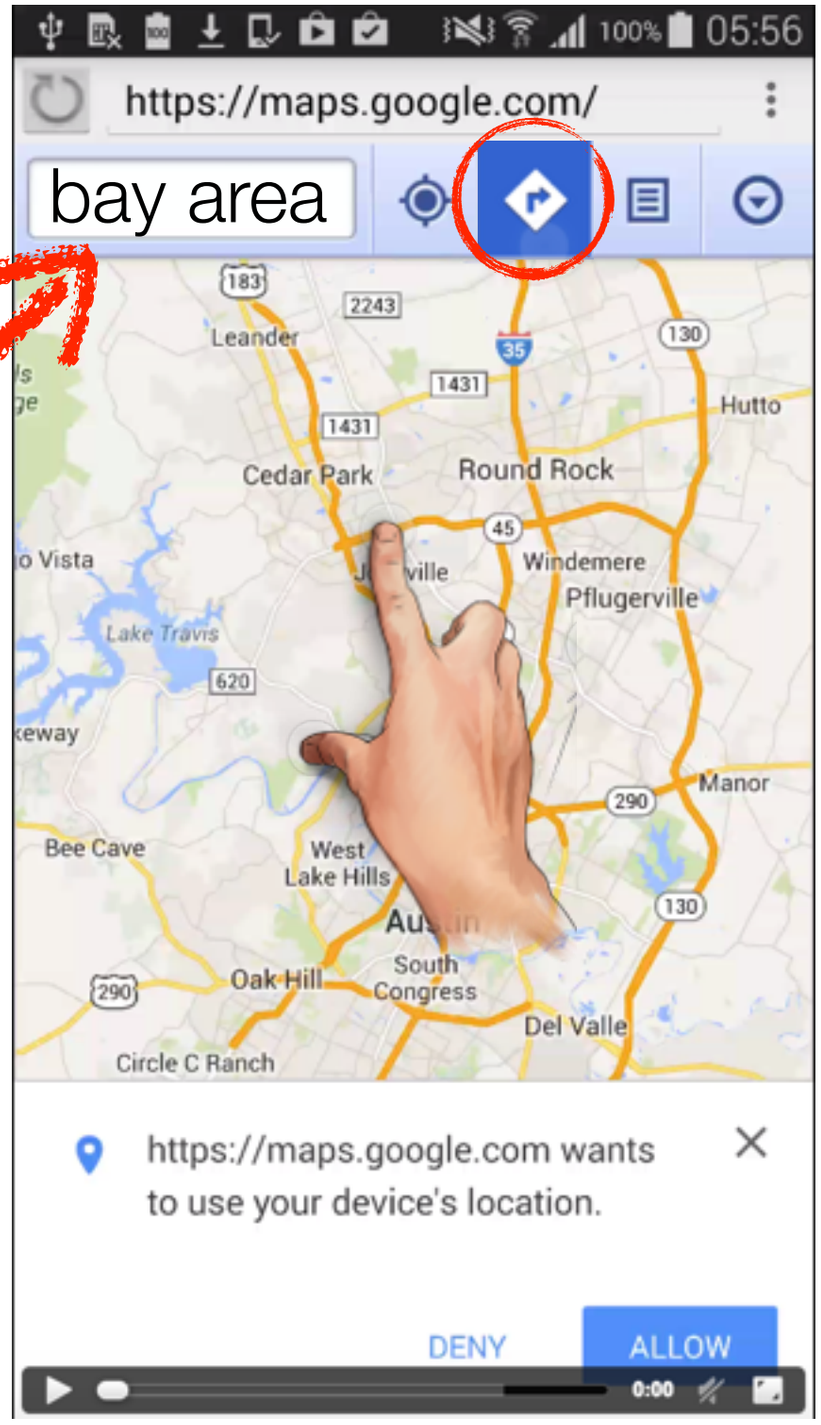
Interacting with an App



http://yuhaozhu.com/googlemaps.mp4

```
DOMFocusIn @ INPUT (16, 1)
selectstart @ DIV (1, 0)
mouseup @ DIV (1, 0)
DOMActivate @ DIV (1, 0)
DOMActivate @ DIV (1, 0)
click @ DIV (1, 0)
selectionchange @ #document (0, 0)
load @ SCRIPT (3, 59)
error @ IMG (0, 0)
error @ IMG (0, 0)
load @ SCRIPT (3, 59)
load @ SCRIPT (3, 59)
keydown @ INPUT (16, 1)
webkitBeforeTextInserted @ DIV (1, 0)
webkitEditableContentChanged @ DIV (1, 0)
input @ DIV (1, 0)
textInput @ INPUT (16, 1)
keypress @ INPUT (16, 1)
keyup @ INPUT (16, 1)
selectionchange @ #document (0, 0)
load @ SCRIPT (3, 58)
keydown @ INPUT (16, 1)
webkitBeforeTextInserted @ DIV (1, 0)
webkitEditableContentChanged @ DIV (1, 0)
input @ DIV (1, 0)
textInput @ INPUT (16, 1)
keypress @ INPUT (16, 1)
keyup @ INPUT (16, 1)
selectionchange @ #document (0, 0)
load @ SCRIPT (3, 58)
keydown @ INPUT (16, 1)
webkitBeforeTextInserted @ DIV (1, 0)
webkitEditableContentChanged @ DIV (1, 0)
input @ DIV (1, 0)
textInput @ INPUT (16, 1)
keypress @ INPUT (16, 1)
keyup @ INPUT (16, 1)
selectionchange @ #document (0, 0)
load @ SCRIPT (3, 58)
keydown @ INPUT (16, 1)
webkitBeforeTextInserted @ DIV (1, 0)
webkitEditableContentChanged @ DIV (1, 0)
input @ DIV (1, 0)
textInput @ INPUT (16, 1)
keypress @ INPUT (16, 1)
keyup @ INPUT (16, 1)
selectionchange @ #document (0, 0)
load @ SCRIPT (3, 58)
keydown @ INPUT (16, 1)
webkitBeforeTextInserted @ DIV (1, 0)
```

Interacting with an App



http://yuhaozhu.com/googlemaps.mp4

```
DOMFocusIn @ INPUT (16, 1)
selectstart @ DIV (1, 0)
mouseup @ DIV (1, 0)
DOMActivate @ DIV (1, 0)
DOMActivate @ DIV (1, 0)
click @ DIV (1, 0)
selectionchange @ #document (0, 0)
load @ SCRIPT (3, 59)
error @ IMG (0, 0)
error @ IMG (0, 0)
load @ SCRIPT (3, 59)
load @ SCRIPT (3, 59)
keydown @ INPUT (16, 1)
webkitBeforeTextInserted @ DIV (1, 0)
webkitEditableContentChanged @ DIV (1, 0)
input @ DIV (1, 0)
textInput @ INPUT (16, 1)
keypress @ INPUT (16, 1)
keyup @ INPUT (16, 1)
selectionchange @ #document (0, 0)
load @ SCRIPT (3, 58)
keydown @ INPUT (16, 1)
webkitBeforeTextInserted @ DIV (1, 0)
webkitEditableContentChanged @ DIV (1, 0)
input @ DIV (1, 0)
```

1487 Events

```
keydown @ INPUT (16, 1)
webkitBeforeTextInserted @ DIV (1, 0)
webkitEditableContentChanged @ DIV (1, 0)
input @ DIV (1, 0)
textInput @ INPUT (16, 1)
keypress @ INPUT (16, 1)
keyup @ INPUT (16, 1)
selectionchange @ #document (0, 0)
load @ SCRIPT (3, 58)
keydown @ INPUT (16, 1)
webkitBeforeTextInserted @ DIV (1, 0)
webkitEditableContentChanged @ DIV (1, 0)
input @ DIV (1, 0)
textInput @ INPUT (16, 1)
keypress @ INPUT (16, 1)
keyup @ INPUT (16, 1)
selectionchange @ #document (0, 0)
load @ SCRIPT (3, 58)
keydown @ INPUT (16, 1)
webkitBeforeTextInserted @ DIV (1, 0)
```

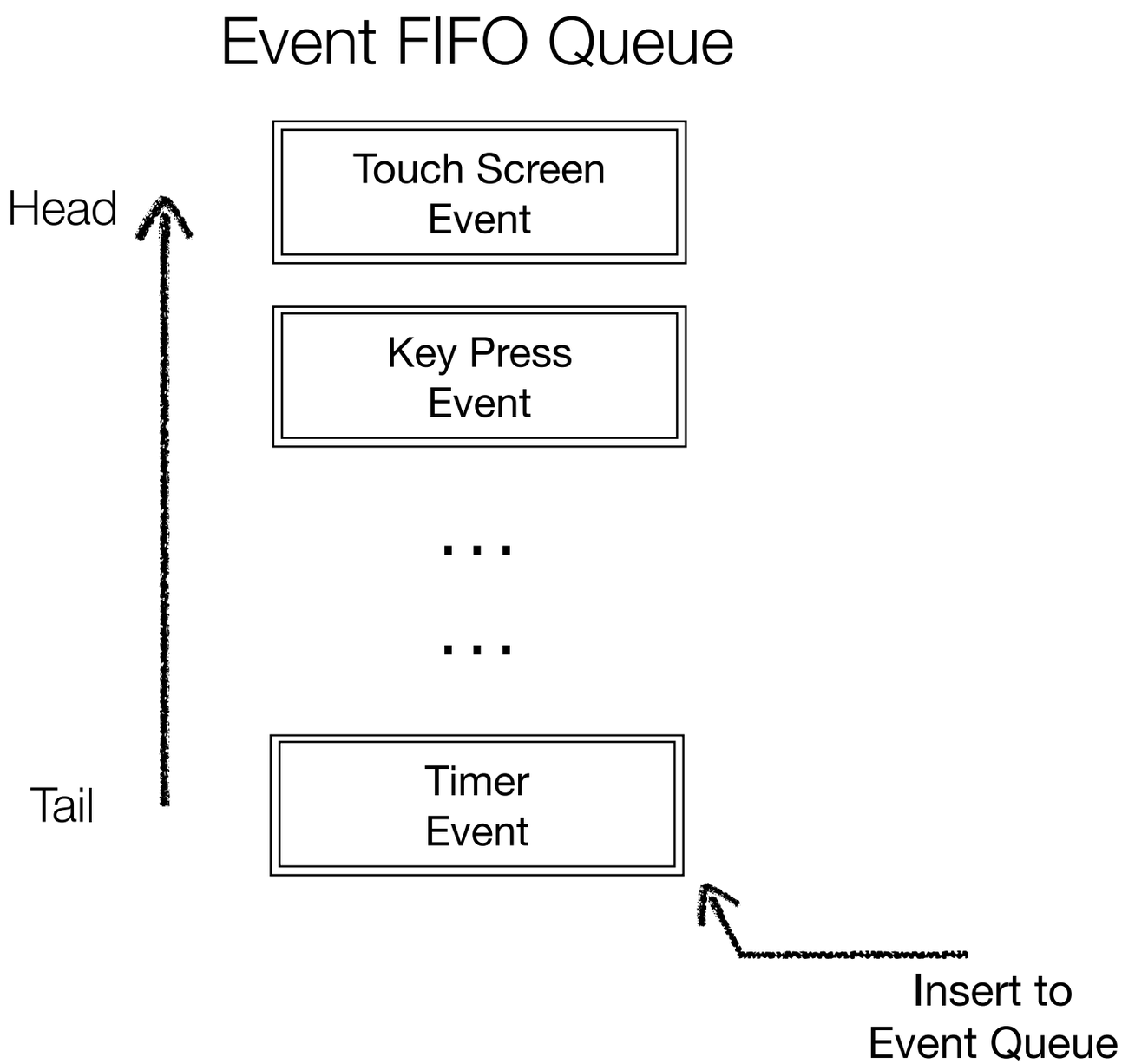
Mobile Web Applications are Event-Driven

Mobile Web Applications are Event-Driven

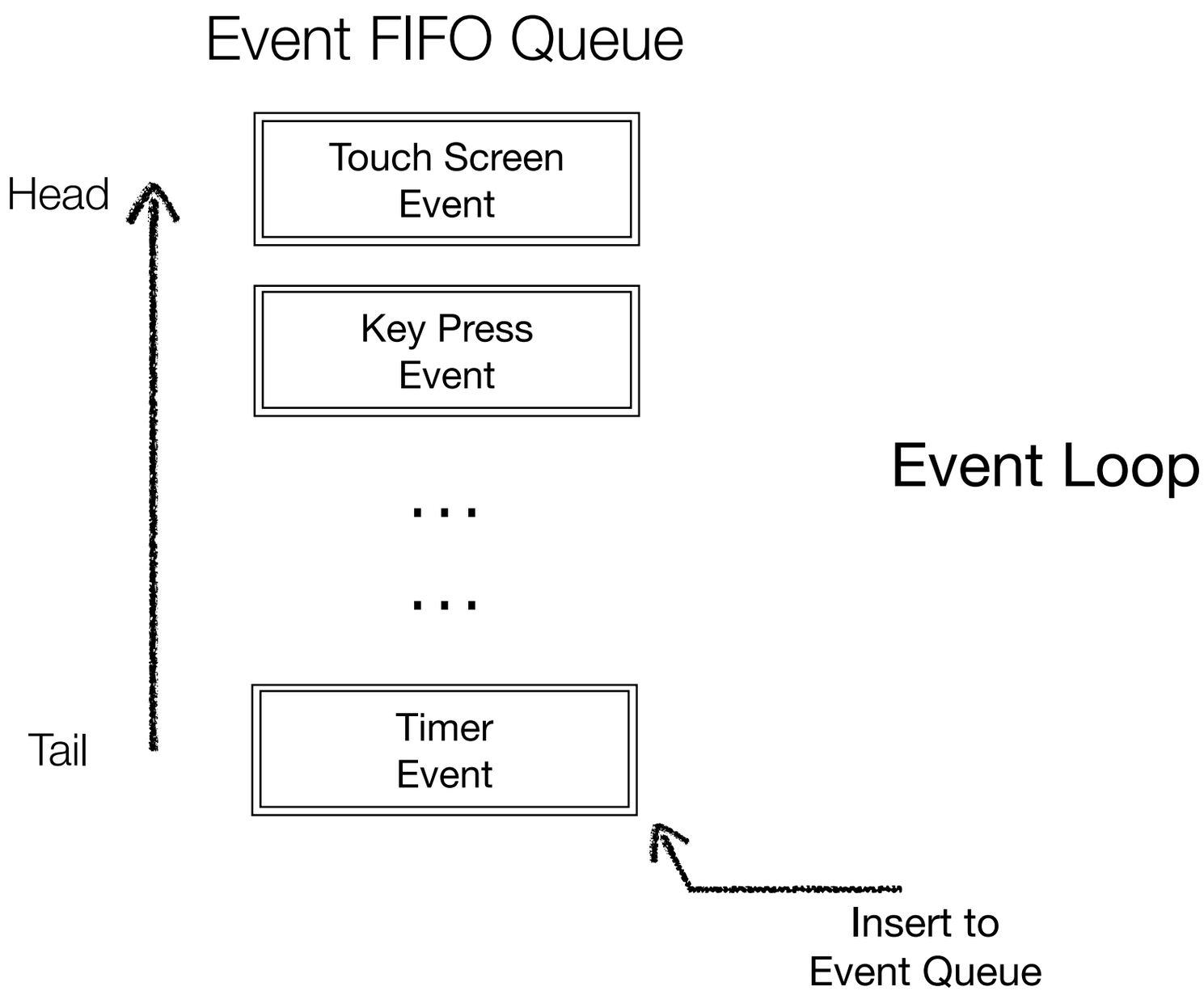
Event FIFO Queue

Mobile Web Applications are Event-Driven

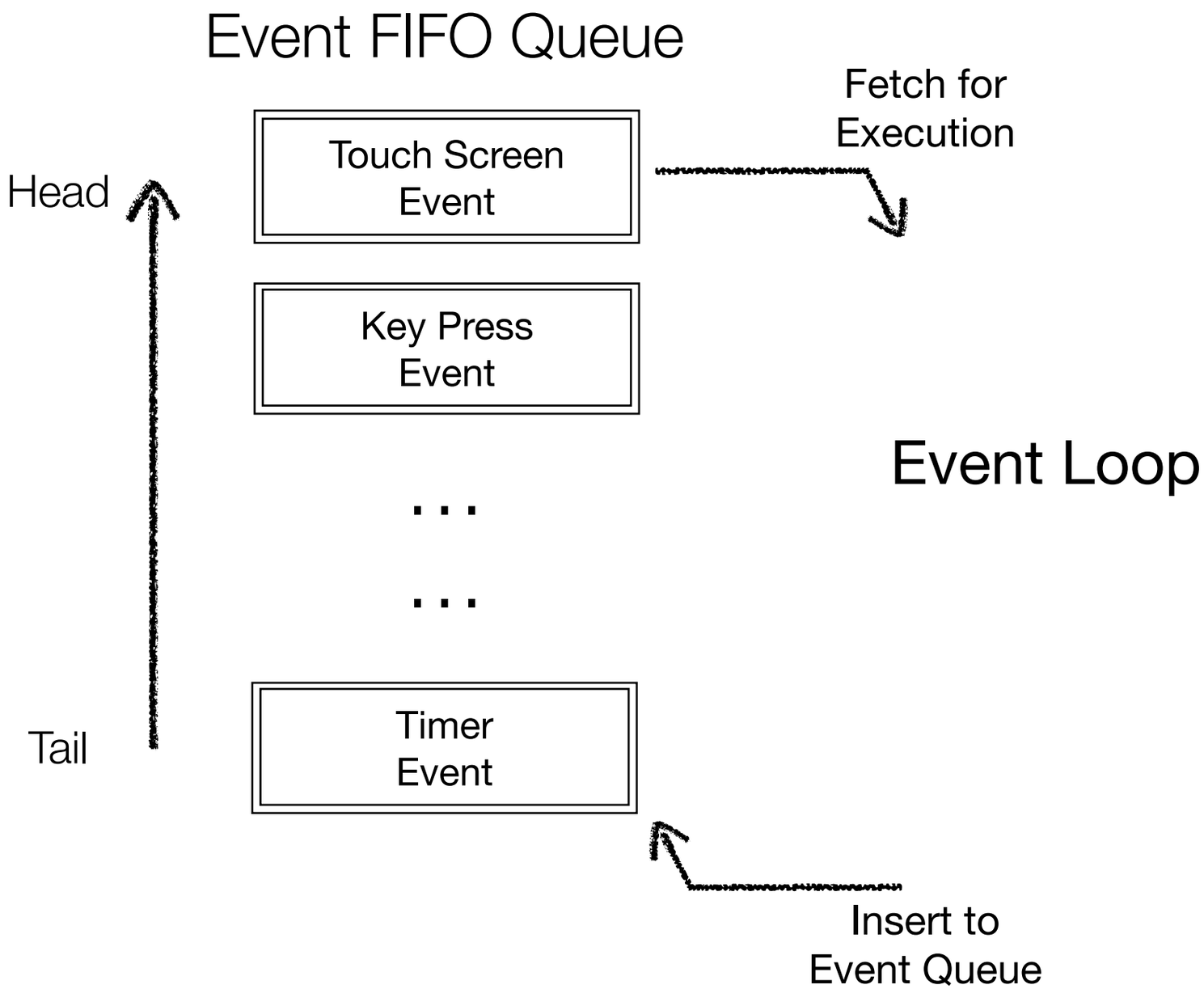
Mobile Web Applications are Event-Driven



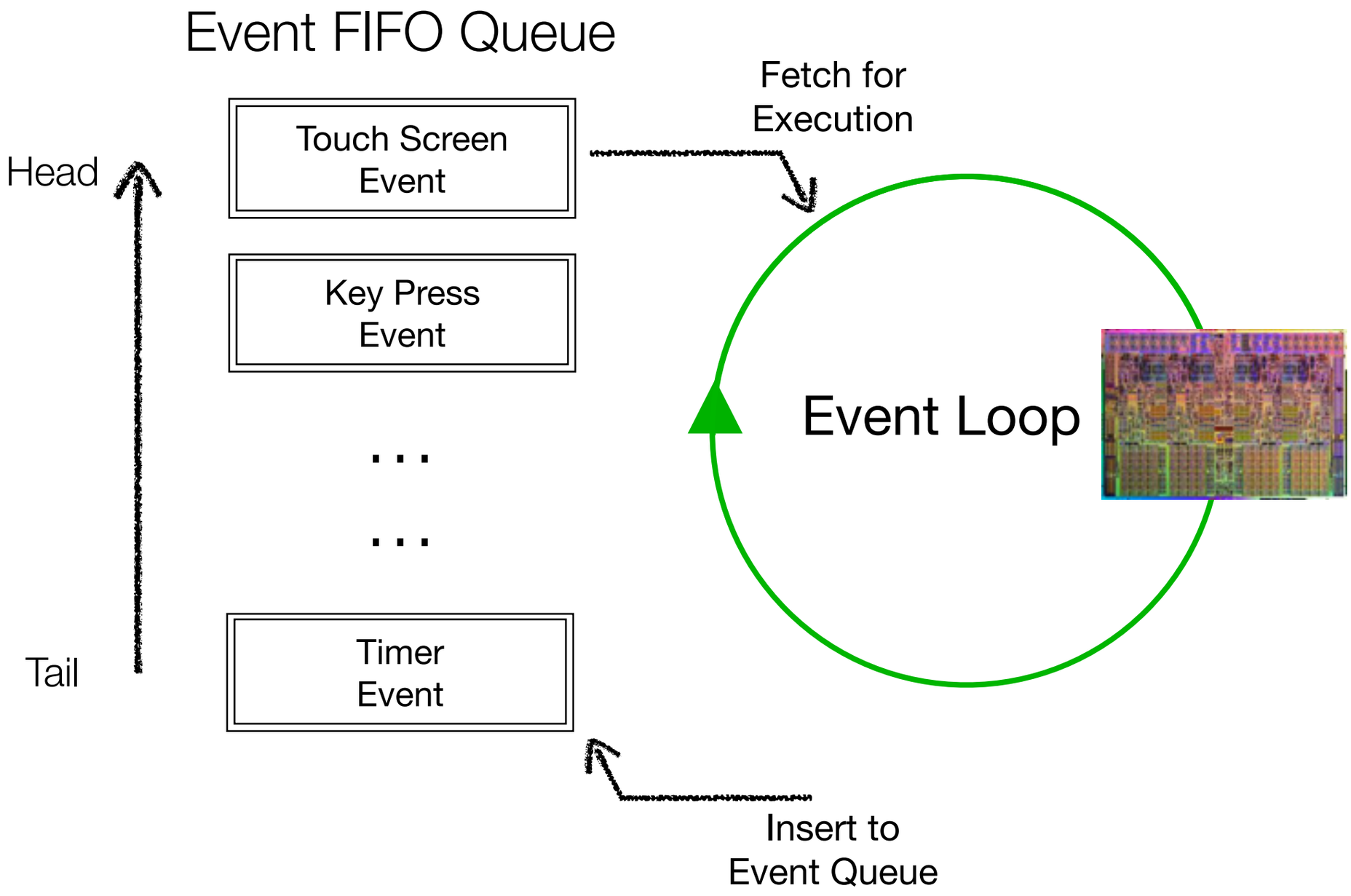
Mobile Web Applications are Event-Driven



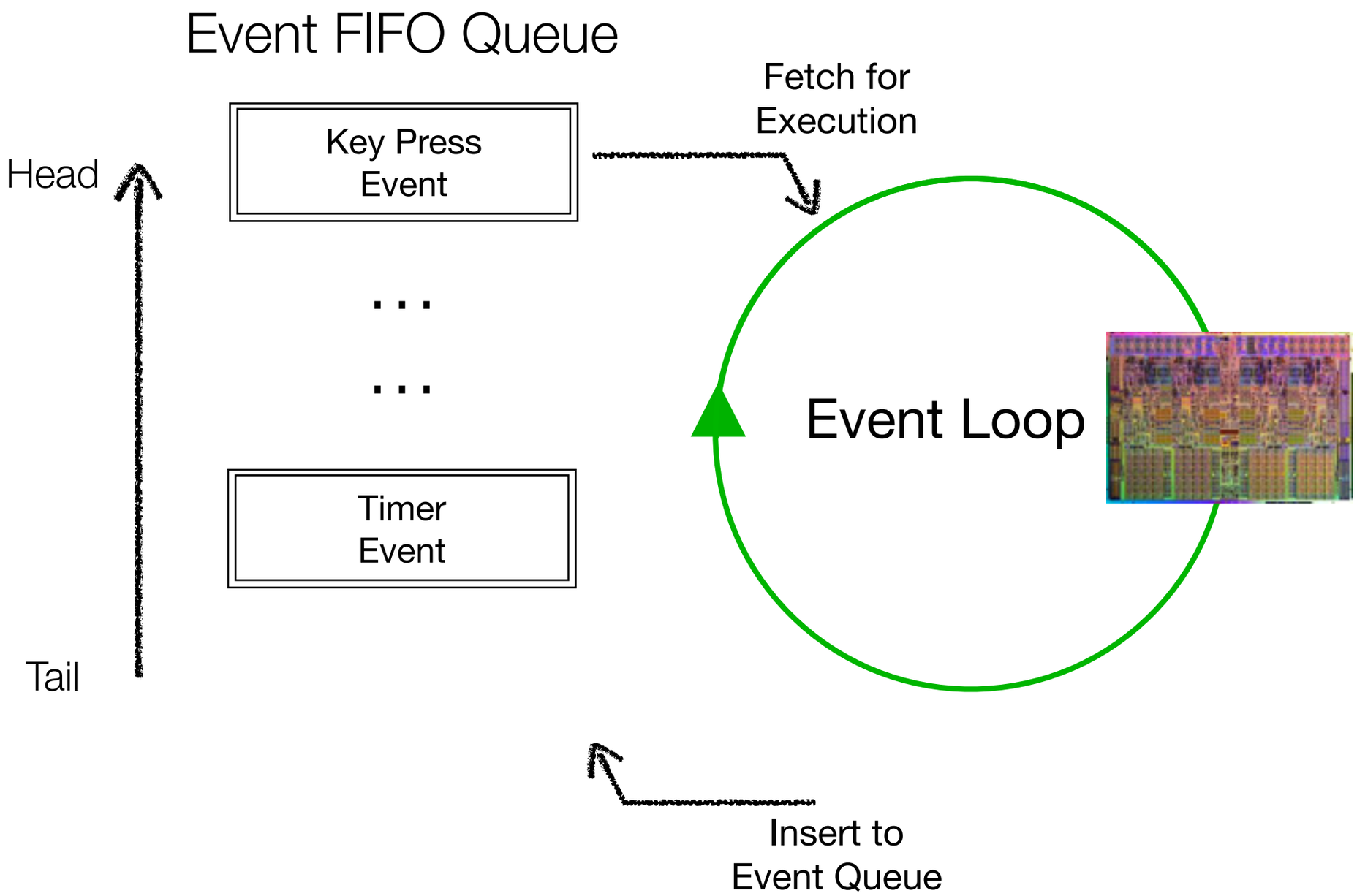
Mobile Web Applications are Event-Driven



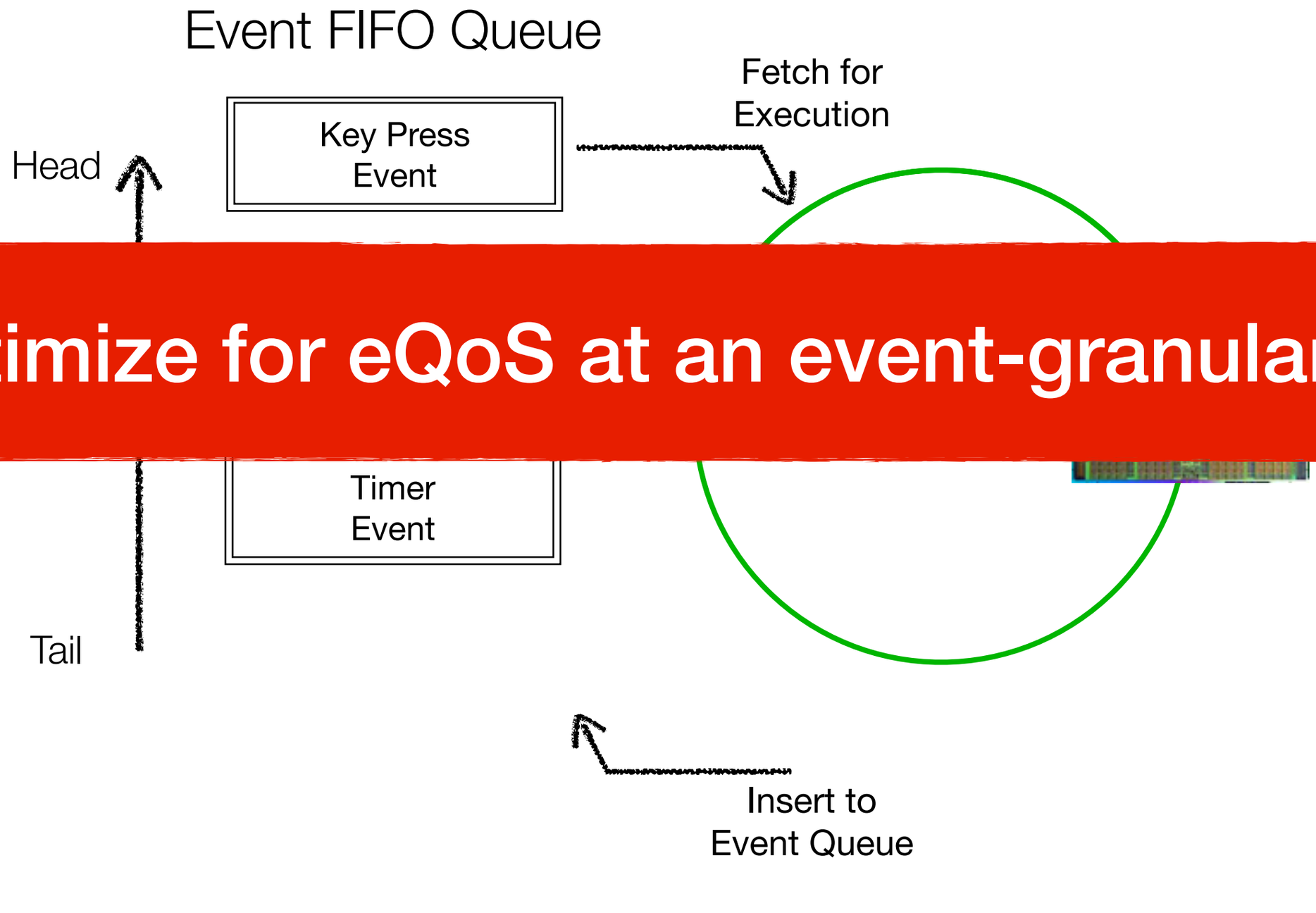
Mobile Web Applications are Event-Driven



Mobile Web Applications are Event-Driven



Mobile Web Applications are Event-Driven



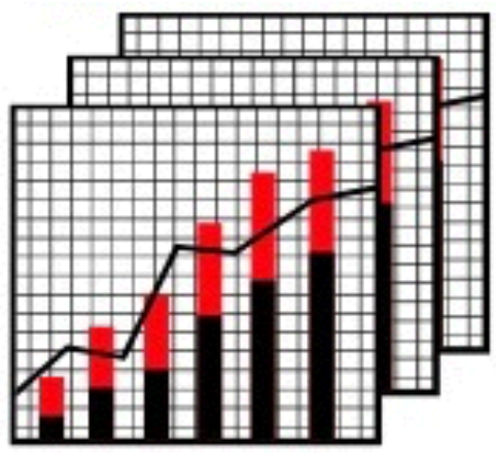
Optimize for eQoS at an event-granularity

Executive Summary

- **Key idea**: Execute **events** with just-enough energy to meet user QoS expectation
  - Develop a runtime system that leverages the large scheduling space of heterogeneous CPUs

Executive Summary

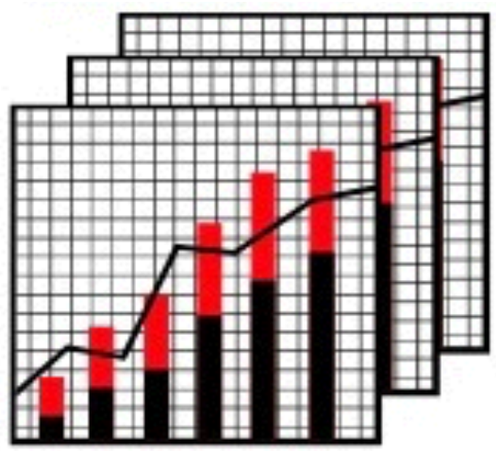
- **Key idea**: Execute **events** with just-enough energy to meet user QoS expectation
  - Develop a runtime system that leverages the large scheduling space of heterogeneous CPUs



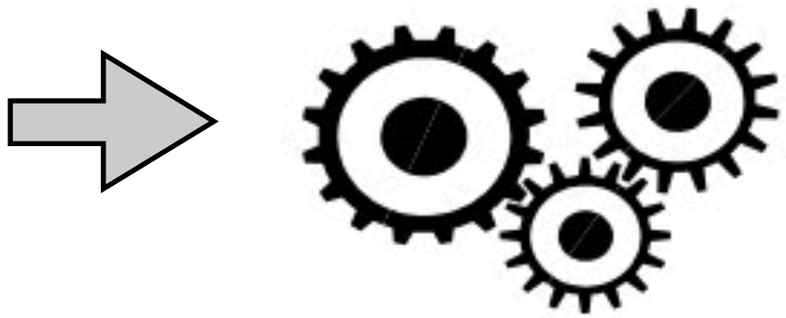
Event-level Characterization

Executive Summary

- **Key idea**: Execute **events** with just-enough energy to meet user QoS expectation
  - Develop a runtime system that leverages the large scheduling space of heterogeneous CPUs



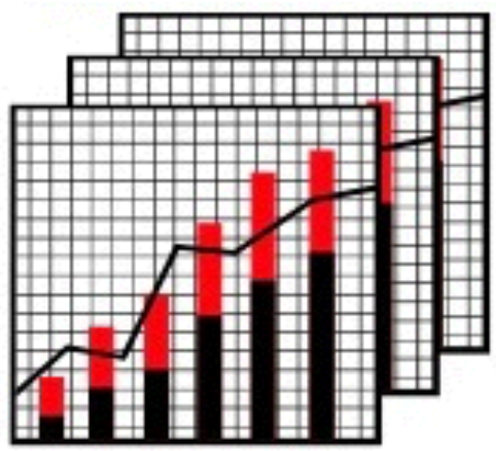
Event-level Characterization



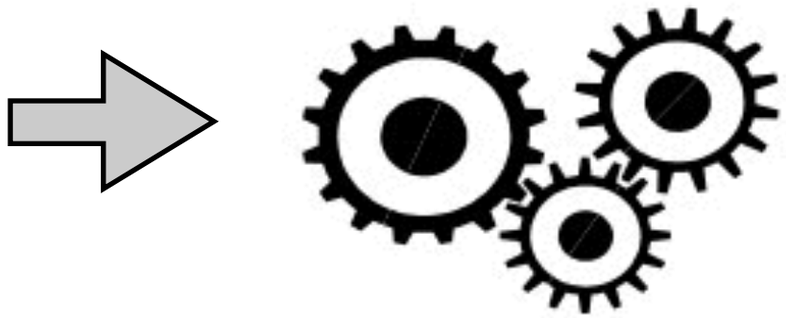
Runtime Mechanics to Exploit Latency Slack

Executive Summary

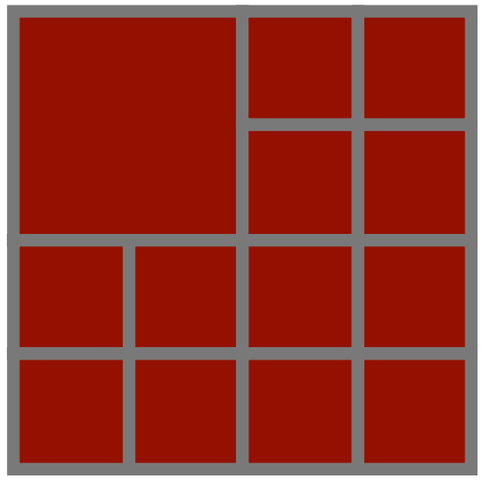
- **Key idea**: Execute **events** with just-enough energy to meet user QoS expectation
  - Develop a runtime system that leverages the large scheduling space of heterogeneous CPUs



Event-level Characterization



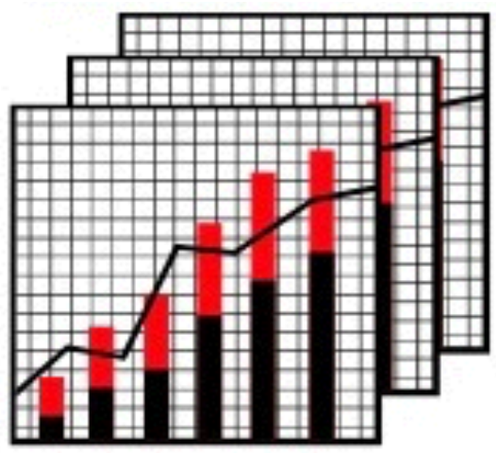
Runtime Mechanics to Exploit Latency Slack



Heterogeneous Resource Utilization

Executive Summary

- **Key idea**: Execute **events** with just-enough energy to meet user QoS expectation
  - Develop a runtime system that leverages the large scheduling space of heterogeneous CPUs

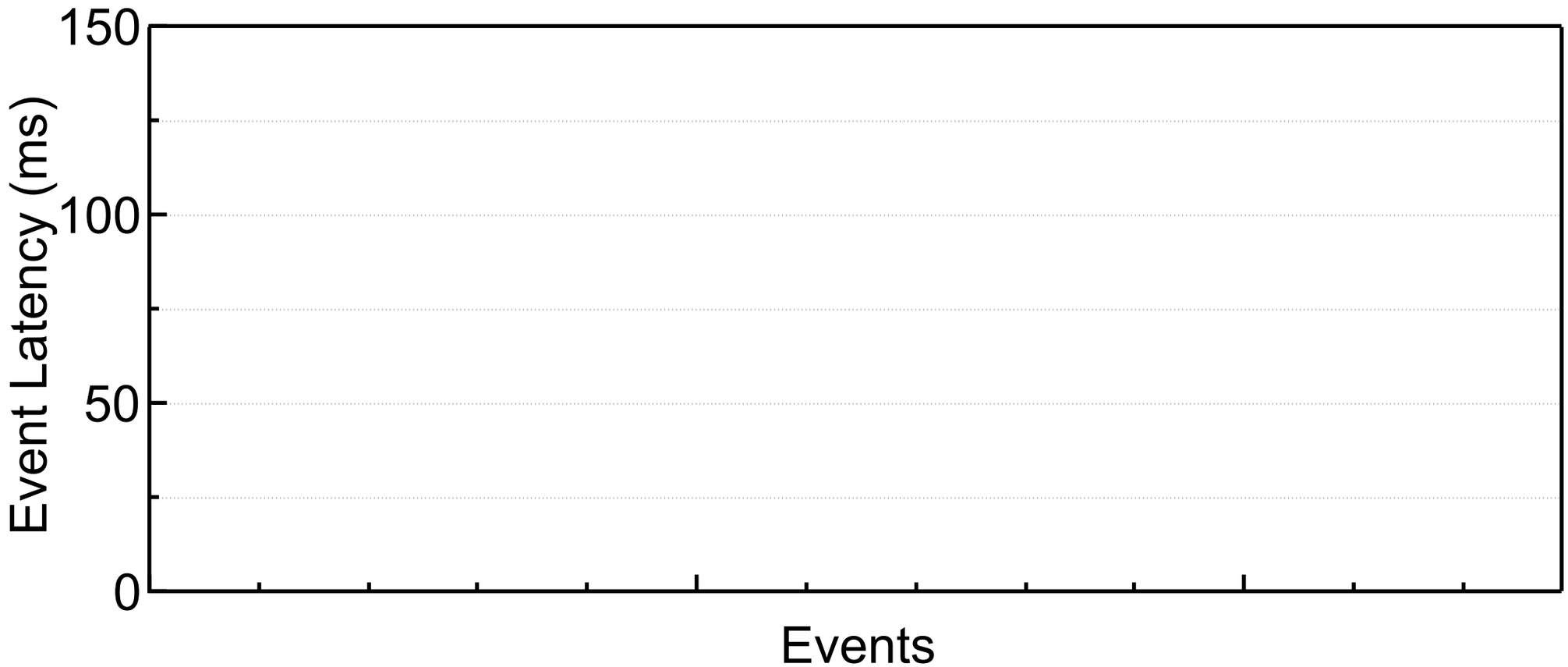


Event-level Characterization

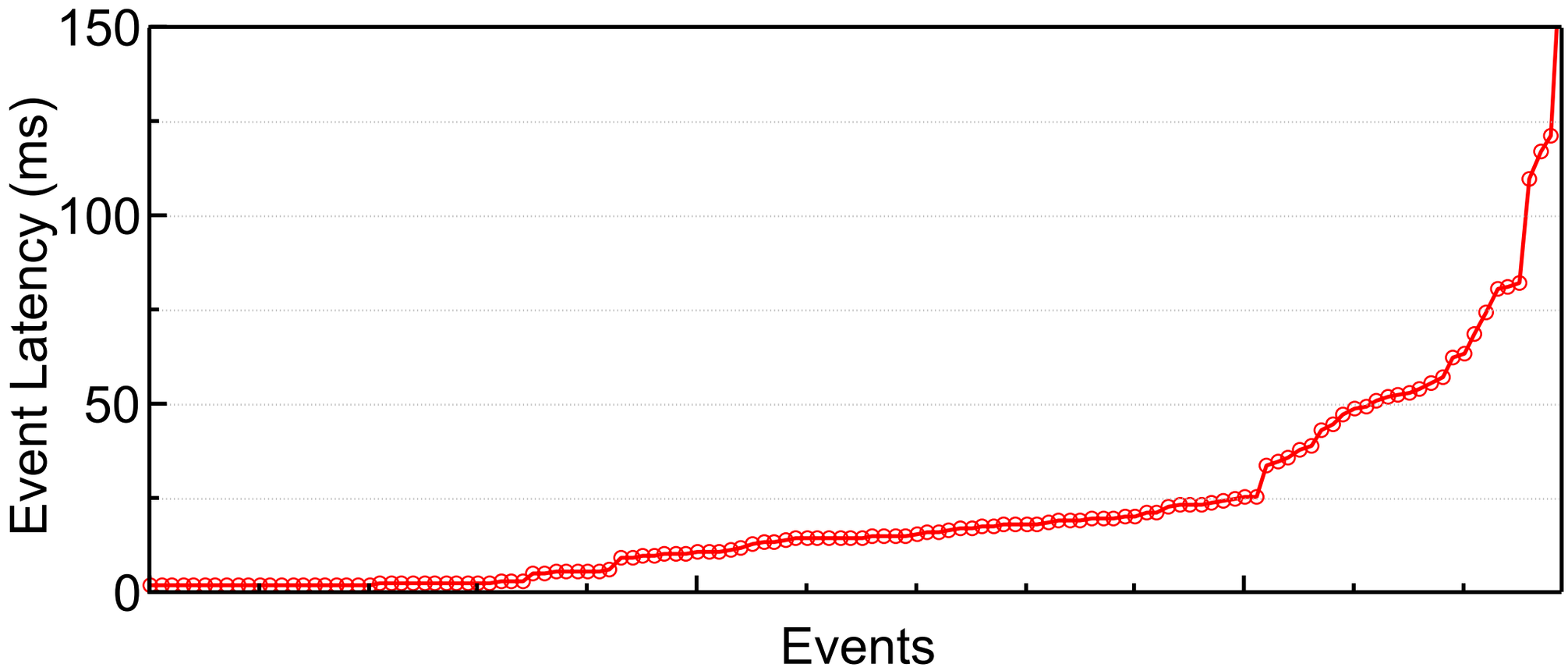
Runtime Mechanics to Exploit Latency Slack

Heterogeneous Resource Utilization

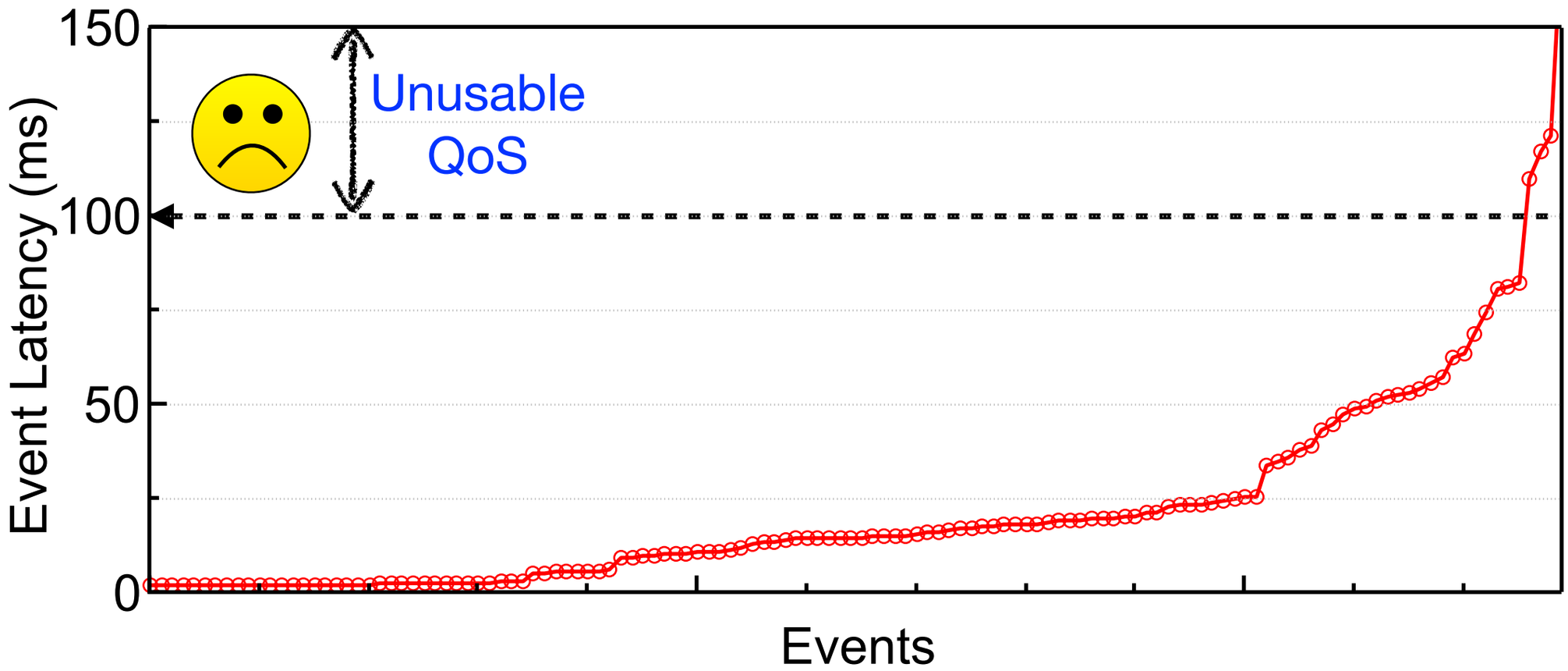
Exploit the Slack in Events



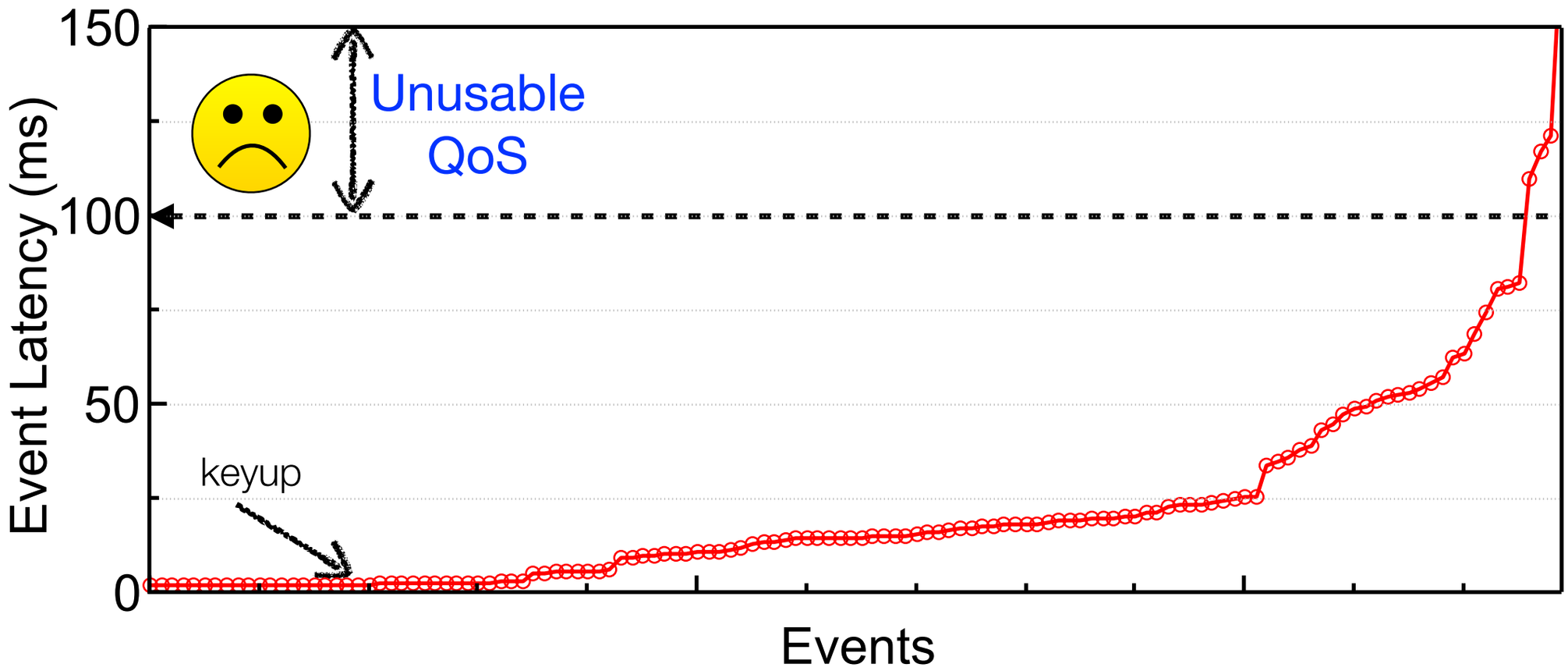
Exploit the Slack in Events



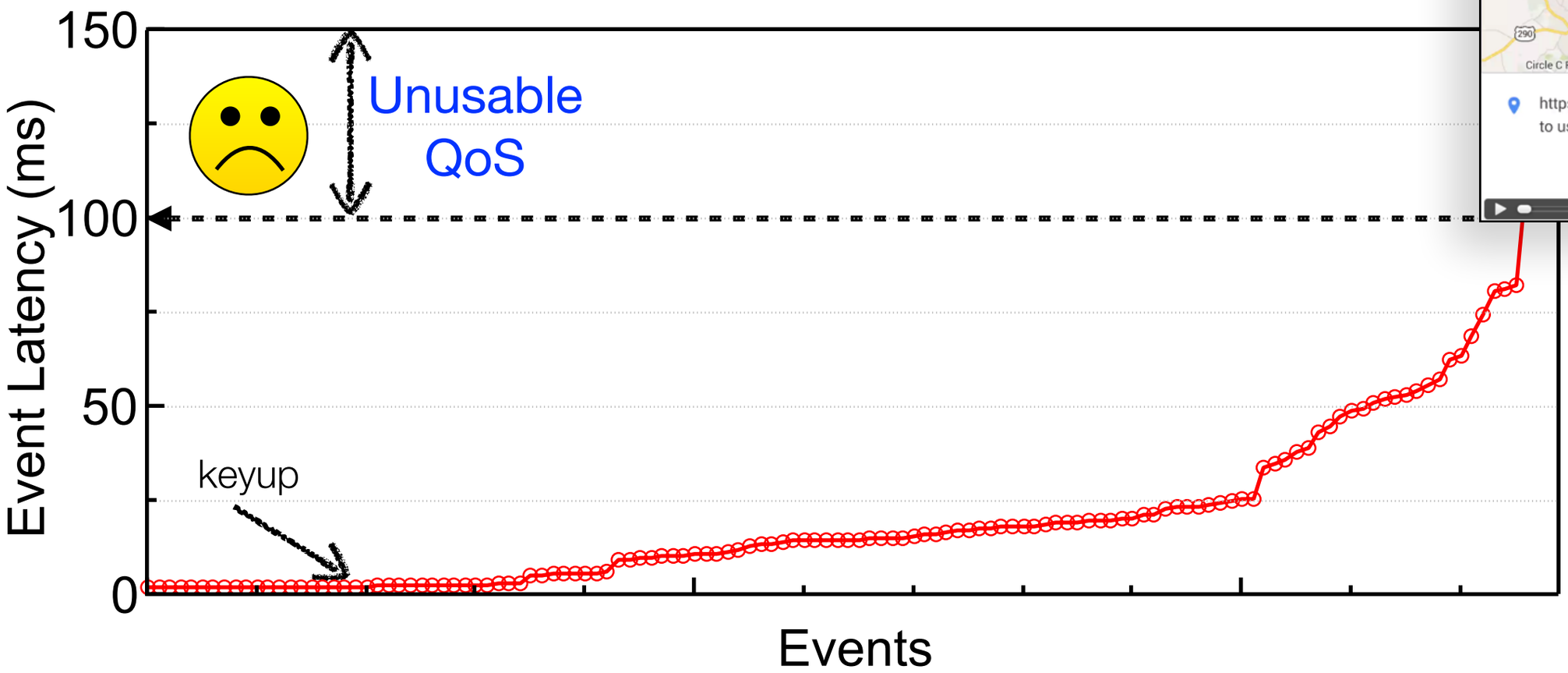
Exploit the Slack in Events

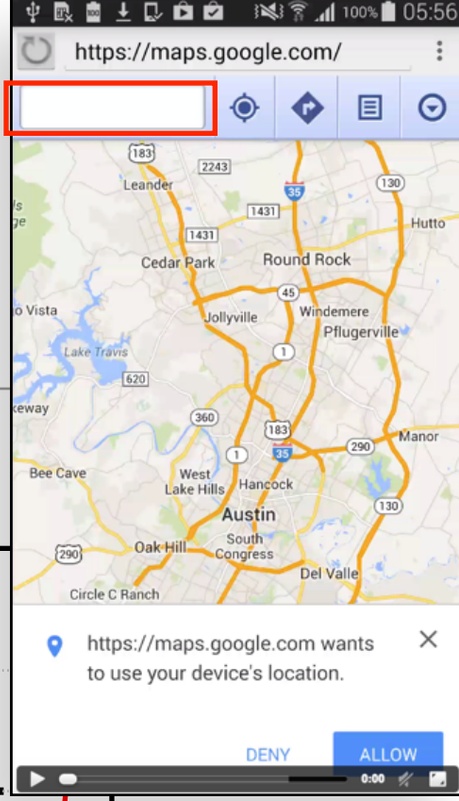


Exploit the Slack in Events

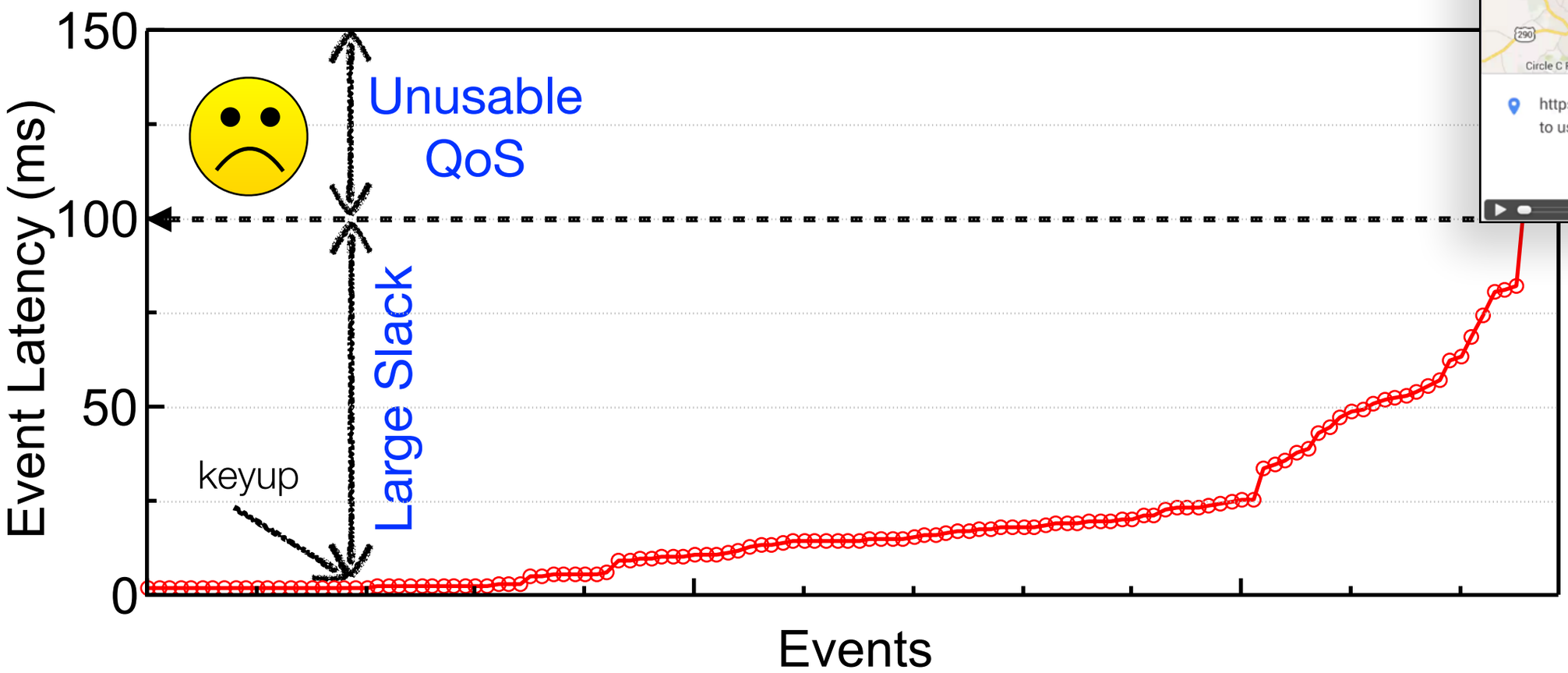


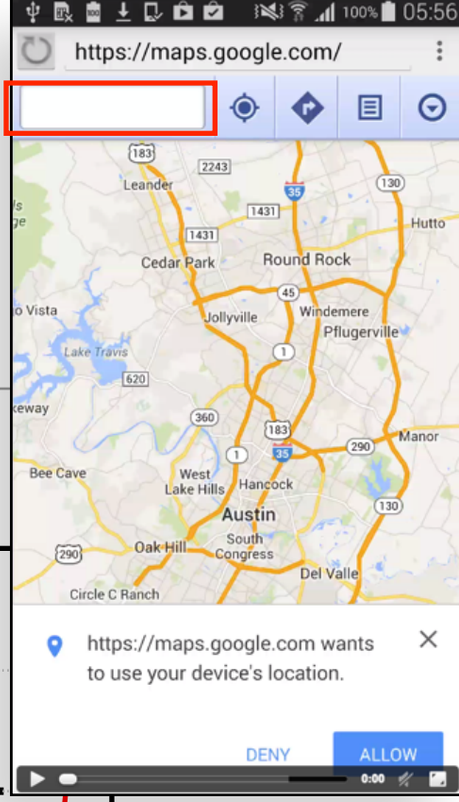
Exploit the Slack in Events



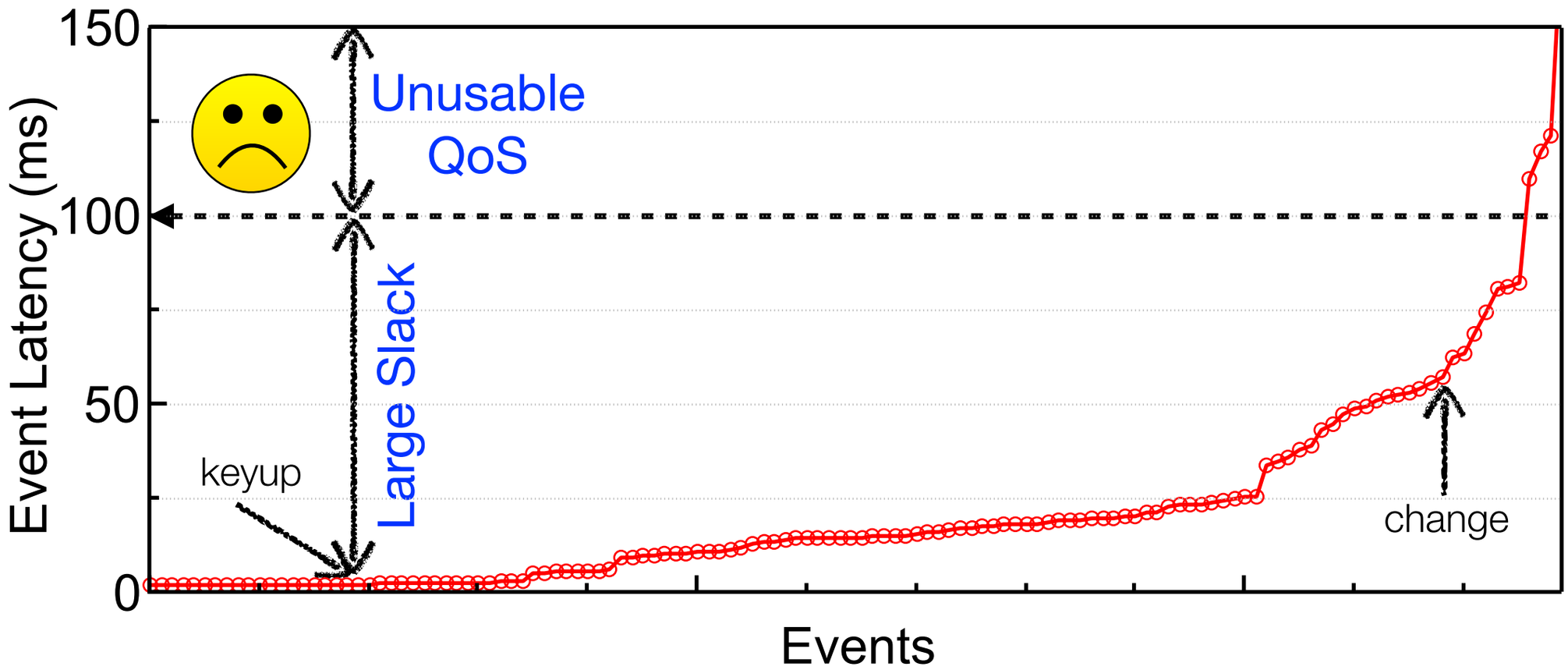


Exploit the Slack in Events

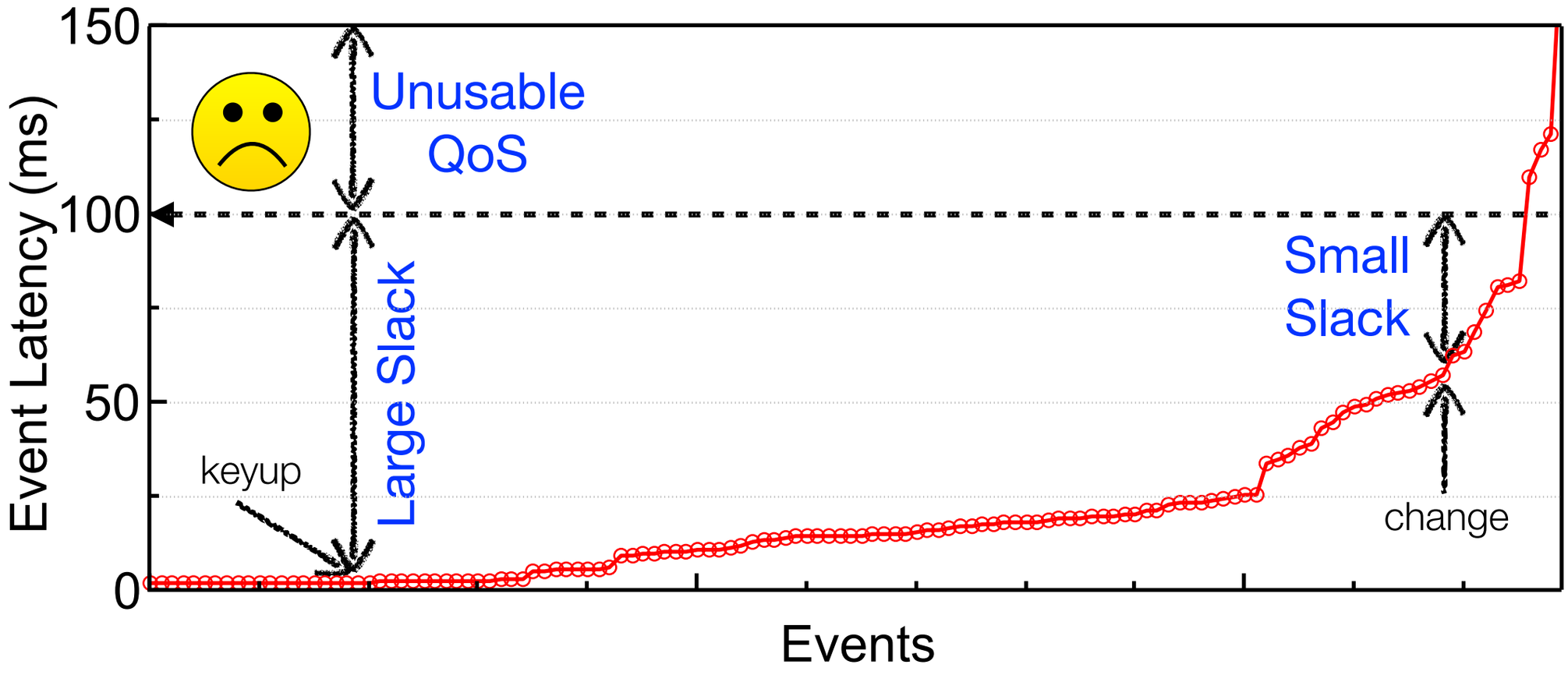




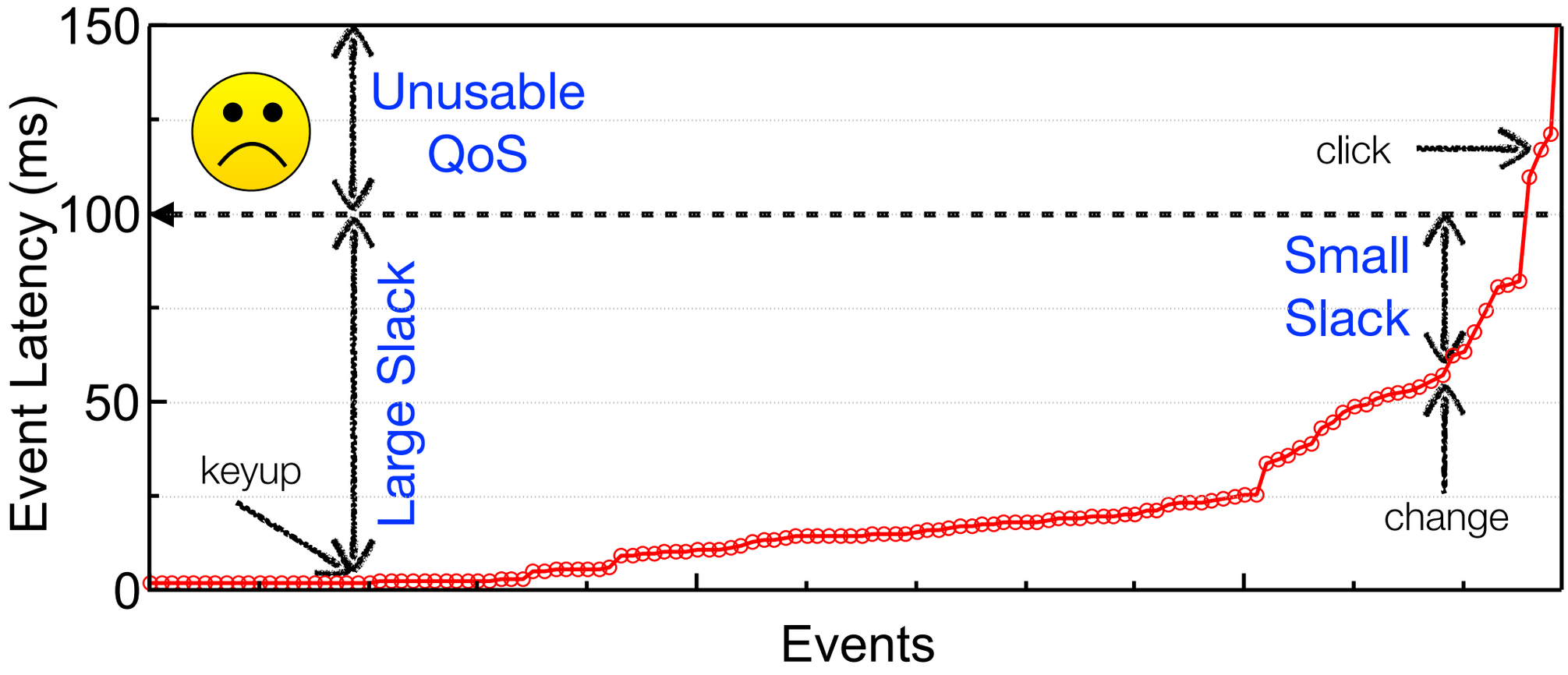
Exploit the Slack in Events



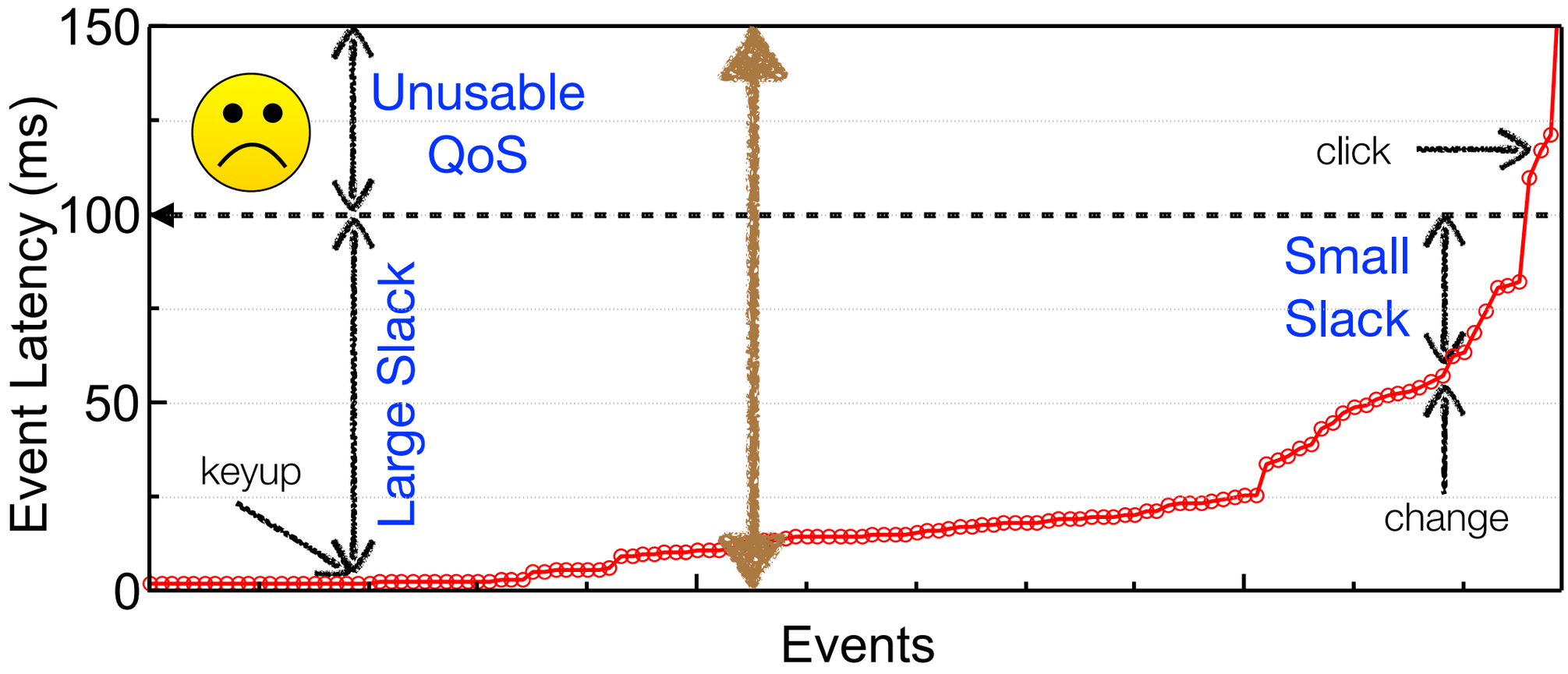
Exploit the Slack in Events



Exploit the Slack in Events



Exploit the Slack in Events



- Wide distribution of event latencies. Events exhibit different slacks.
  - How to exploit different slacks for different events?

Event-Based Scheduler (EBS) Overview

- **Goal**: Find the most energy-efficient architectural configuration that meets the QoS target for each event

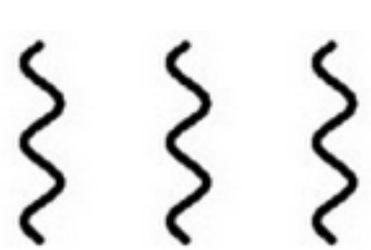
Event-Based Scheduler (EBS) Overview

- **Goal**: Find the most energy-efficient architectural configuration that meets the QoS target for each event

Thread Scheduling

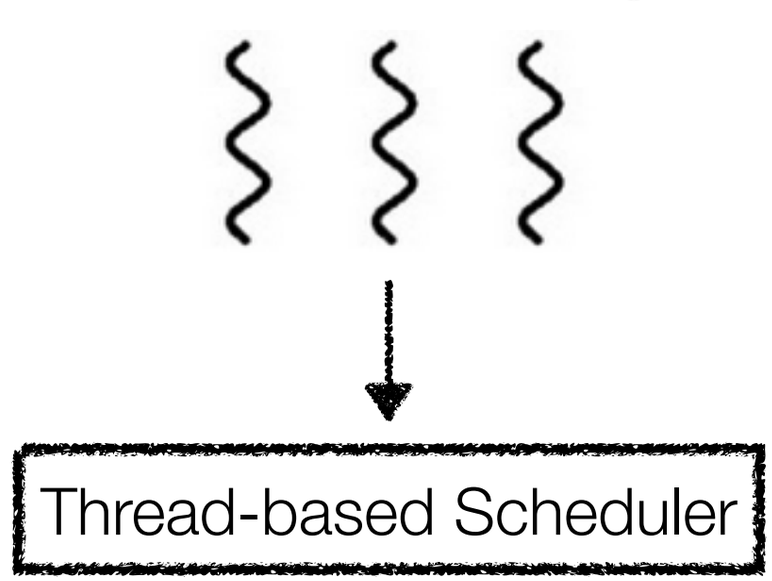
Event-Based Scheduler (EBS) Overview

- **Goal**: Find the most energy-efficient architectural configuration that meets the QoS target for each event



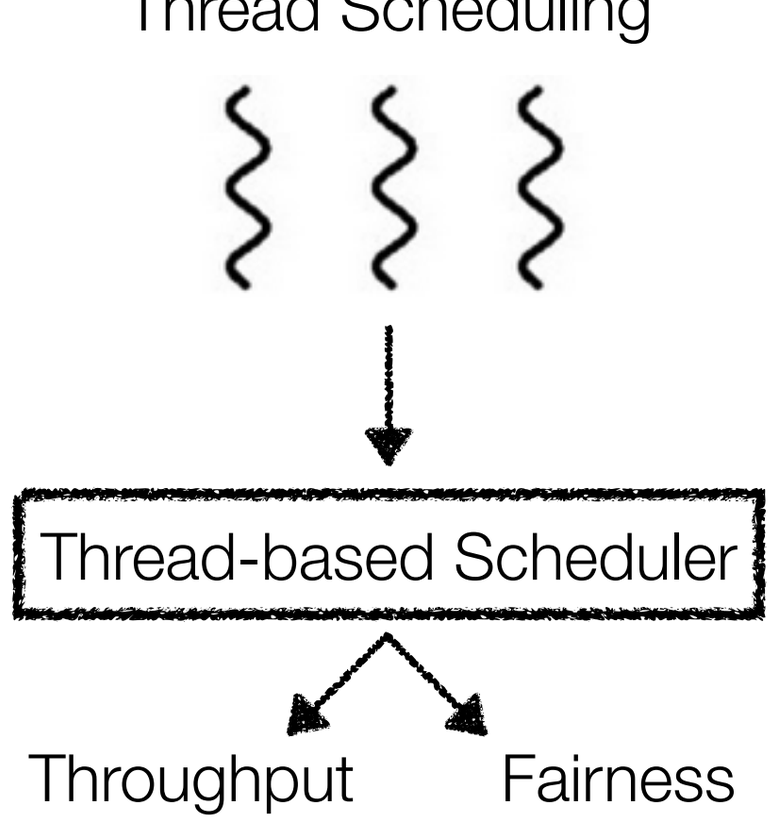
Event-Based Scheduler (EBS) Overview

- **Goal**: Find the most energy-efficient architectural configuration that meets the QoS target for each event



Event-Based Scheduler (EBS) Overview

- **Goal**: Find the most energy-efficient architectural configuration that meets the QoS target for each event



Event-Based Scheduler (EBS) Overview

- **Goal**: Find the most energy-efficient architectural configuration that meets the QoS target for each event

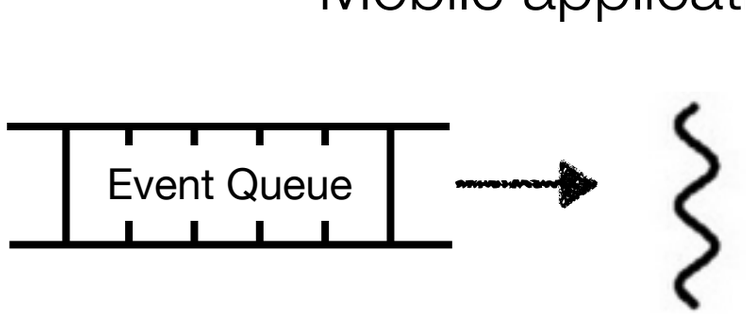
Event-Based Scheduler (EBS) Overview

- **Goal**: Find the most energy-efficient architectural configuration that meets the QoS target for each event

Mobile applications

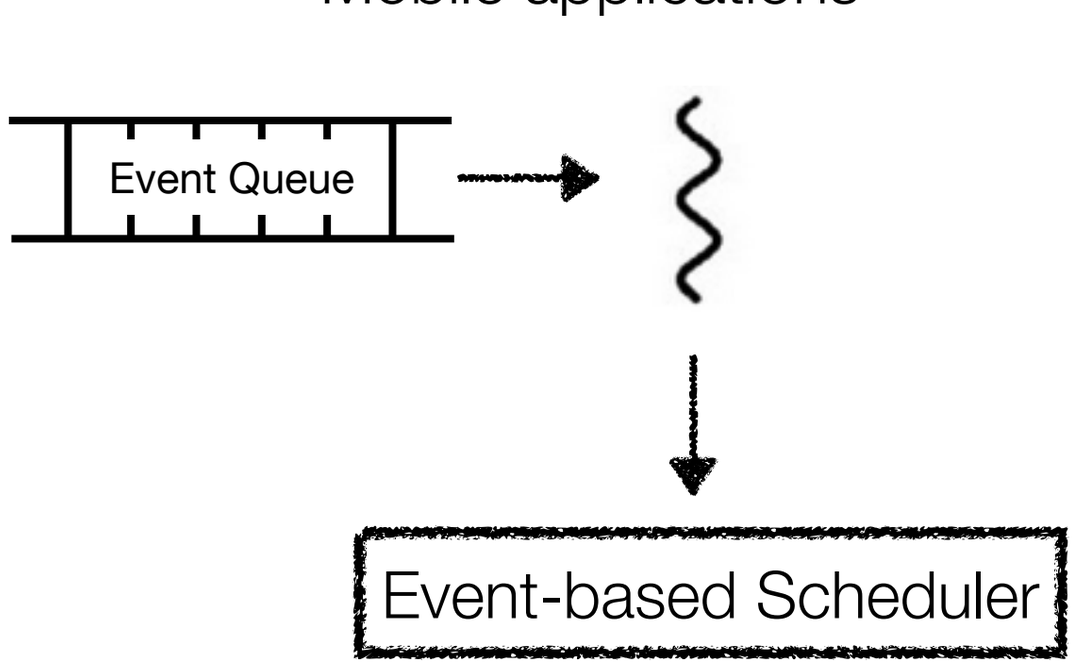
Event-Based Scheduler (EBS) Overview

- **Goal**: Find the most energy-efficient architectural configuration that meets the QoS target for each event



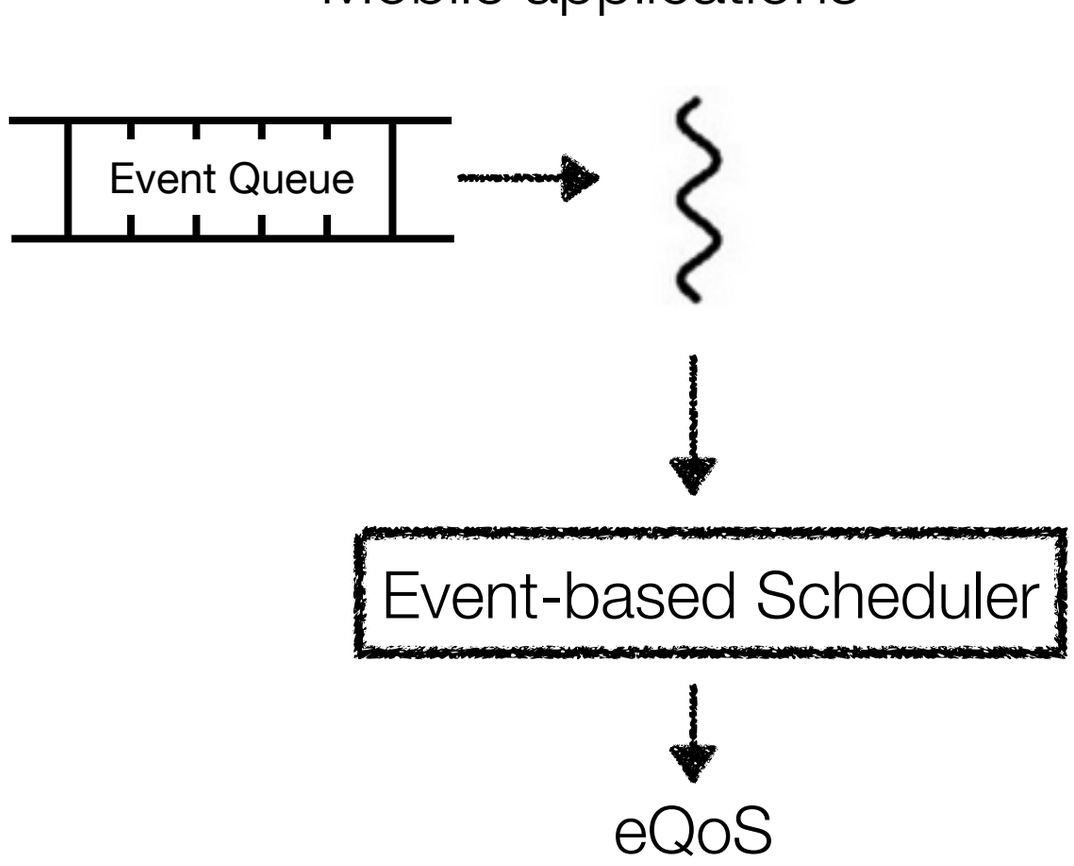
Event-Based Scheduler (EBS) Overview

- **Goal**: Find the most energy-efficient architectural configuration that meets the QoS target for each event



Event-Based Scheduler (EBS) Overview

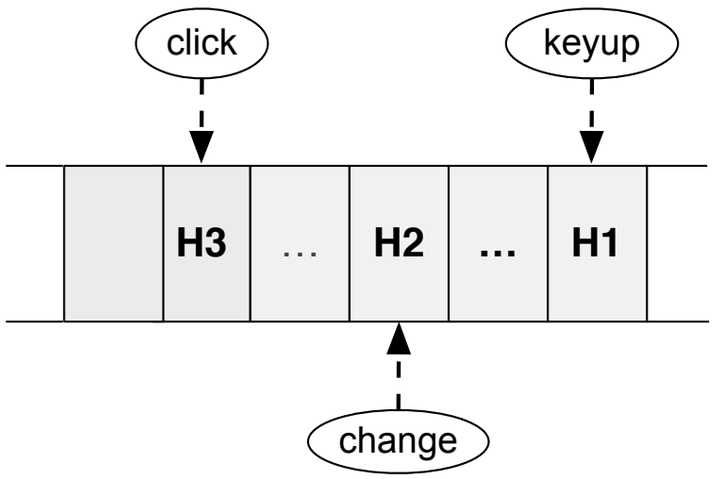
- **Goal**: Find the most energy-efficient architectural configuration that meets the QoS target for each event



Event-Based Scheduler (EBS) Overview

- **Goal**: Find the most energy-efficient architectural configuration that meets the QoS target for each event

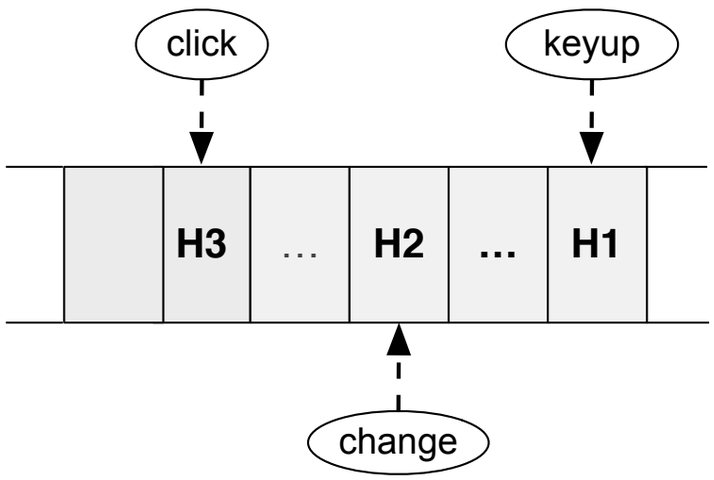
Event Queue



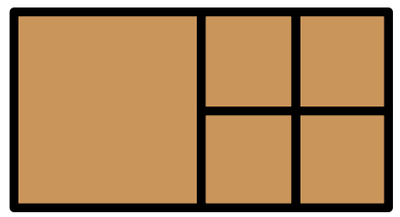
Event-Based Scheduler (EBS) Overview

- **Goal**: Find the most energy-efficient architectural configuration that meets the QoS target for each event

Event Queue



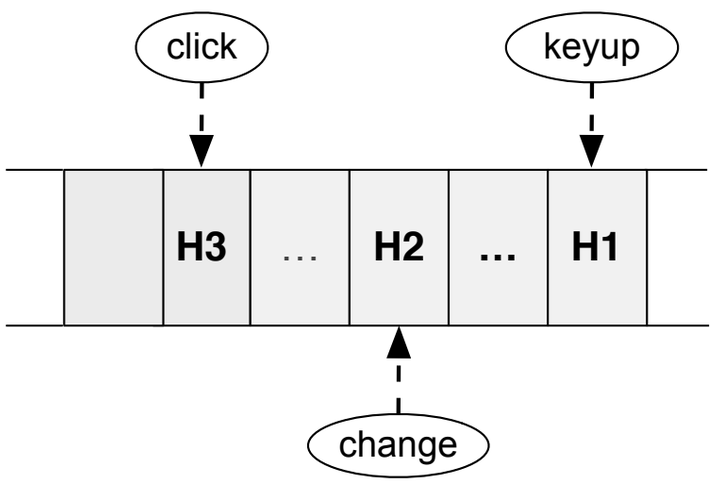
Big/Little Hardware



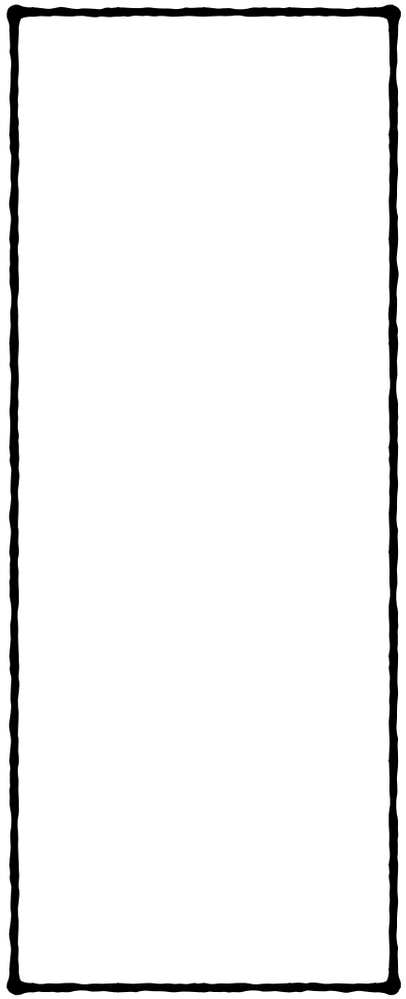
Event-Based Scheduler (EBS) Overview

- **Goal**: Find the most energy-efficient architectural configuration that meets the QoS target for each event

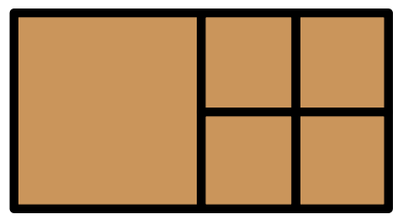
Event Queue



Event-Based Scheduler

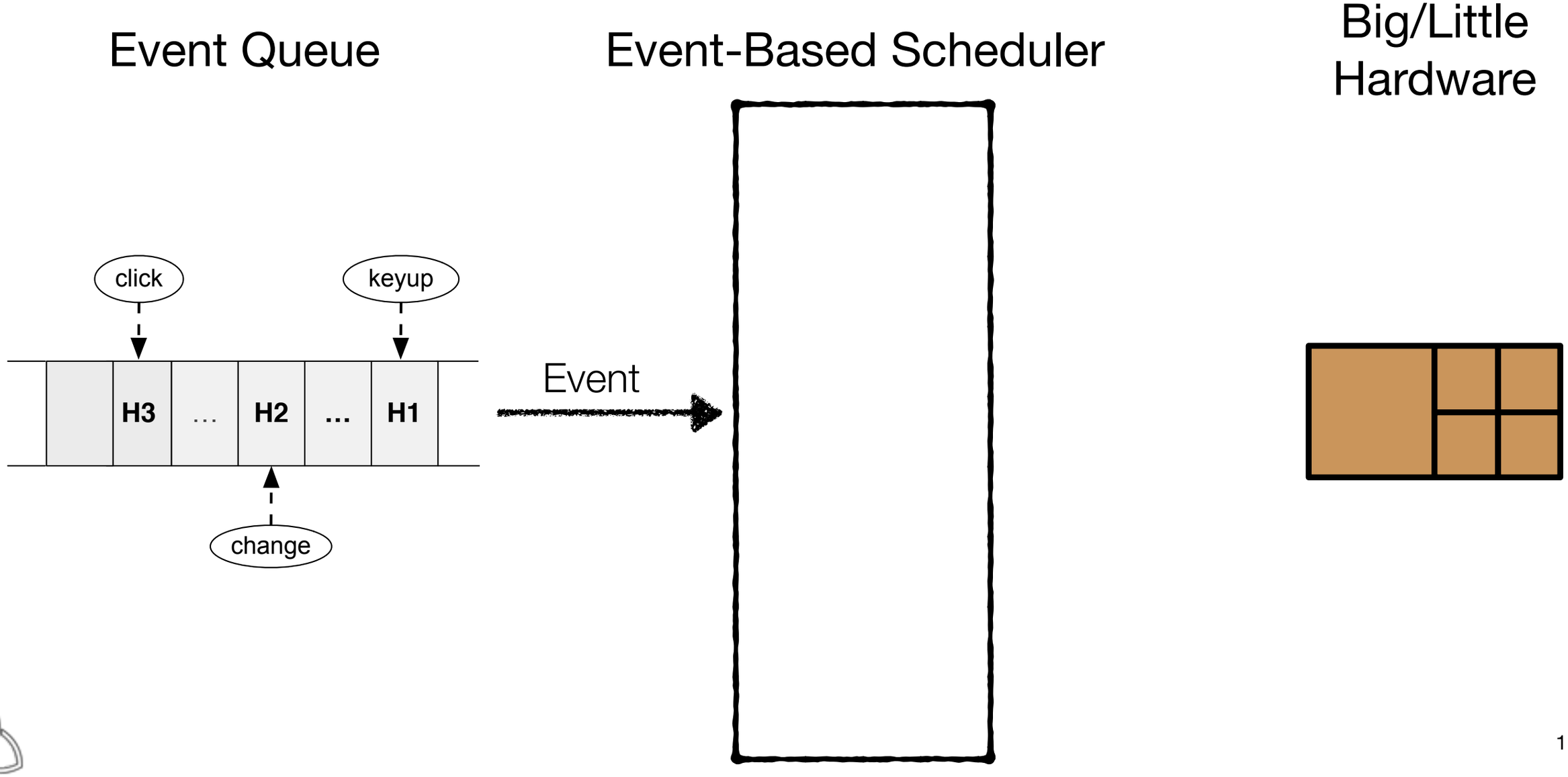


Big/Little Hardware



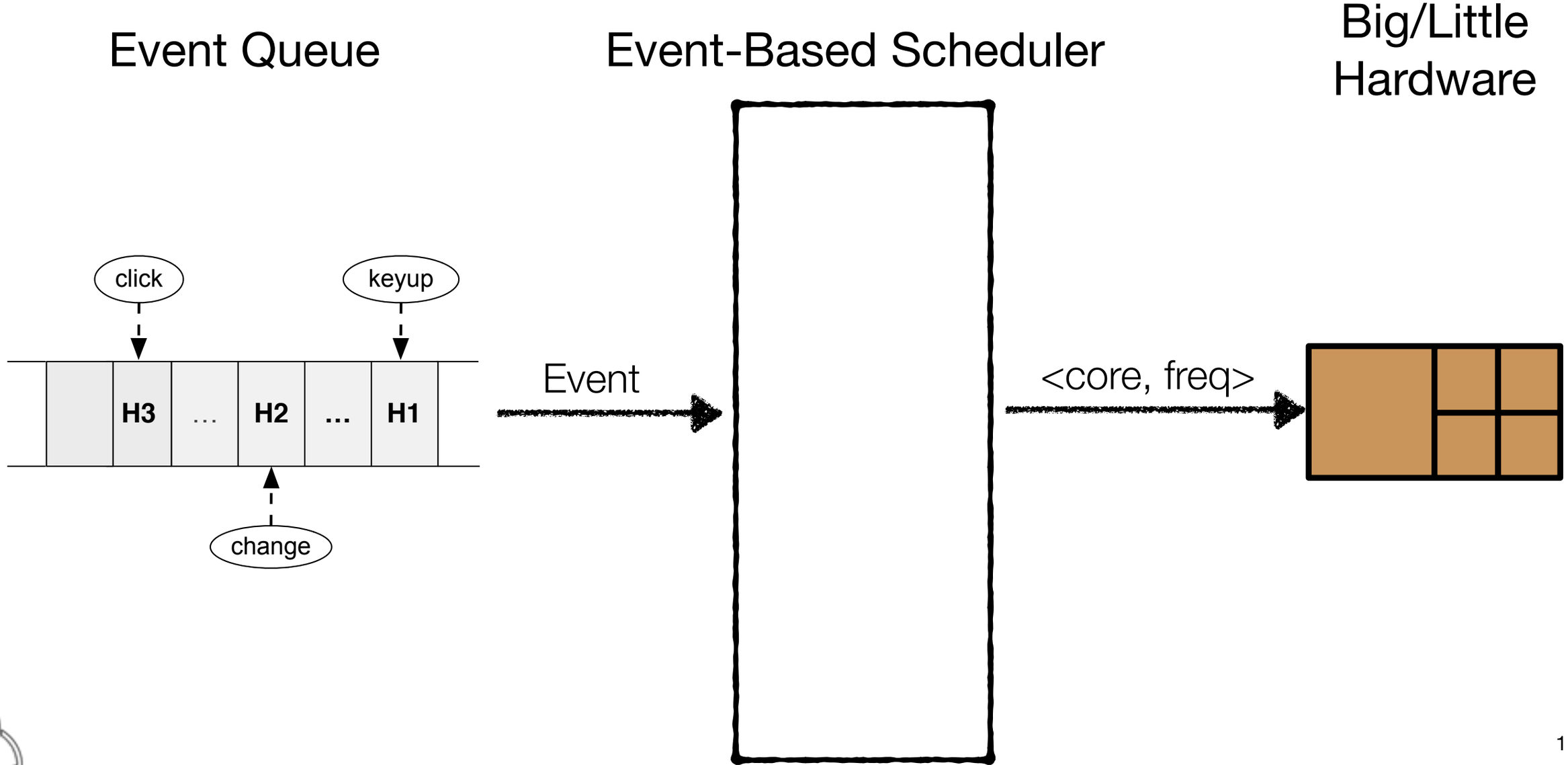
Event-Based Scheduler (EBS) Overview

- **Goal**: Find the most energy-efficient architectural configuration that meets the QoS target for each event



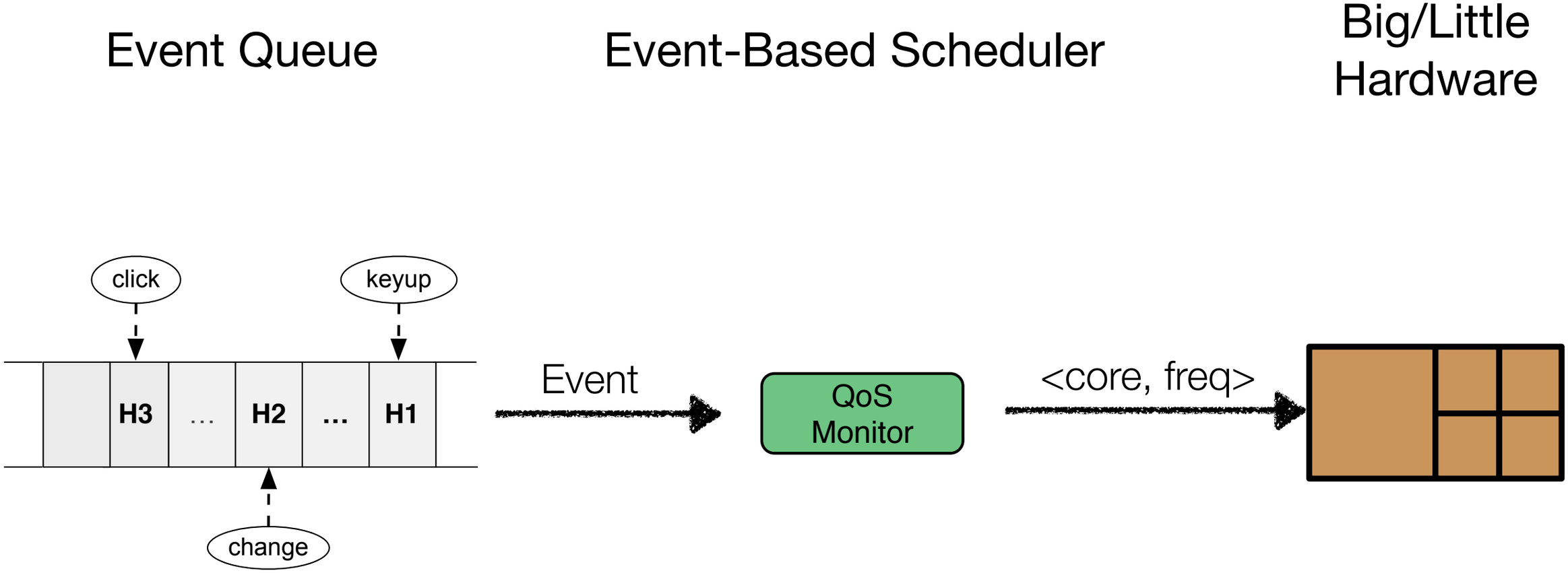
Event-Based Scheduler (EBS) Overview

- **Goal**: Find the most energy-efficient architectural configuration that meets the QoS target for each event



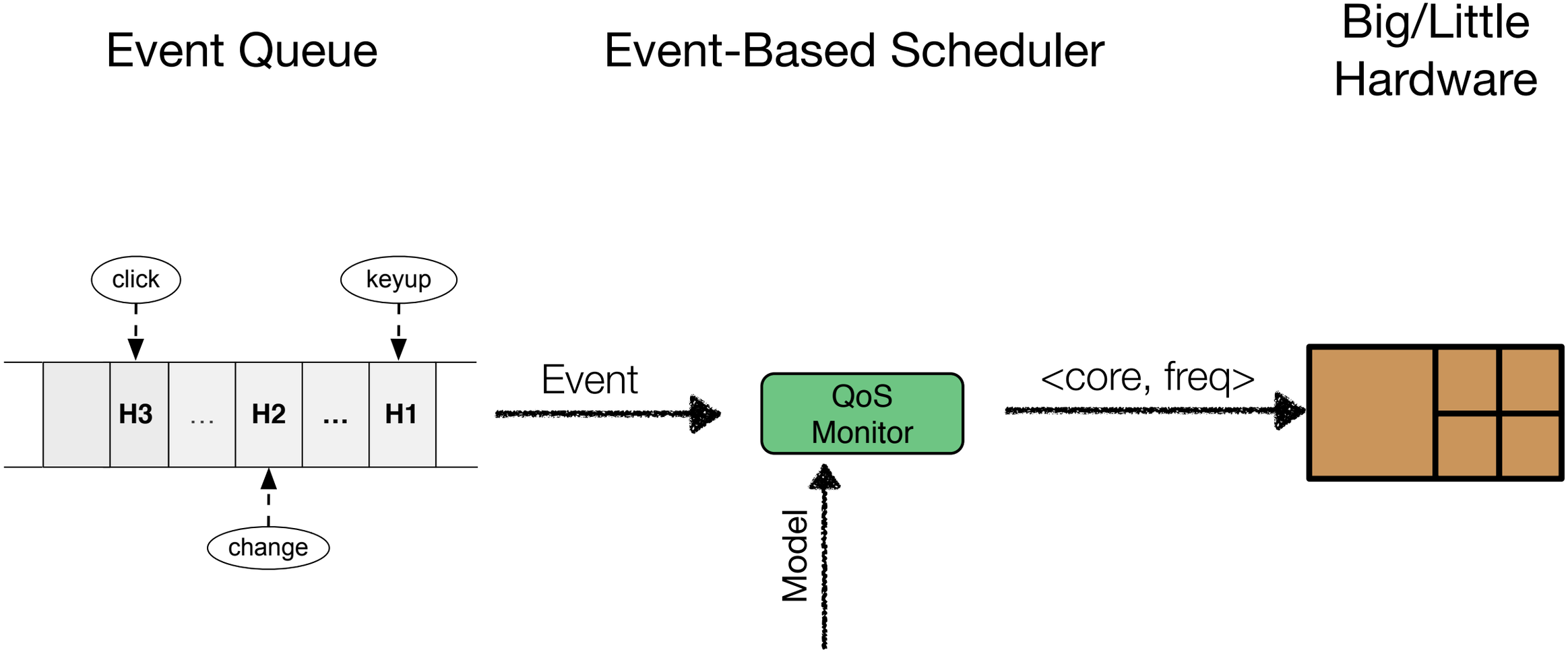
Event-Based Scheduler (EBS) Overview

- **Goal**: Find the most energy-efficient architectural configuration that meets the QoS target for each event



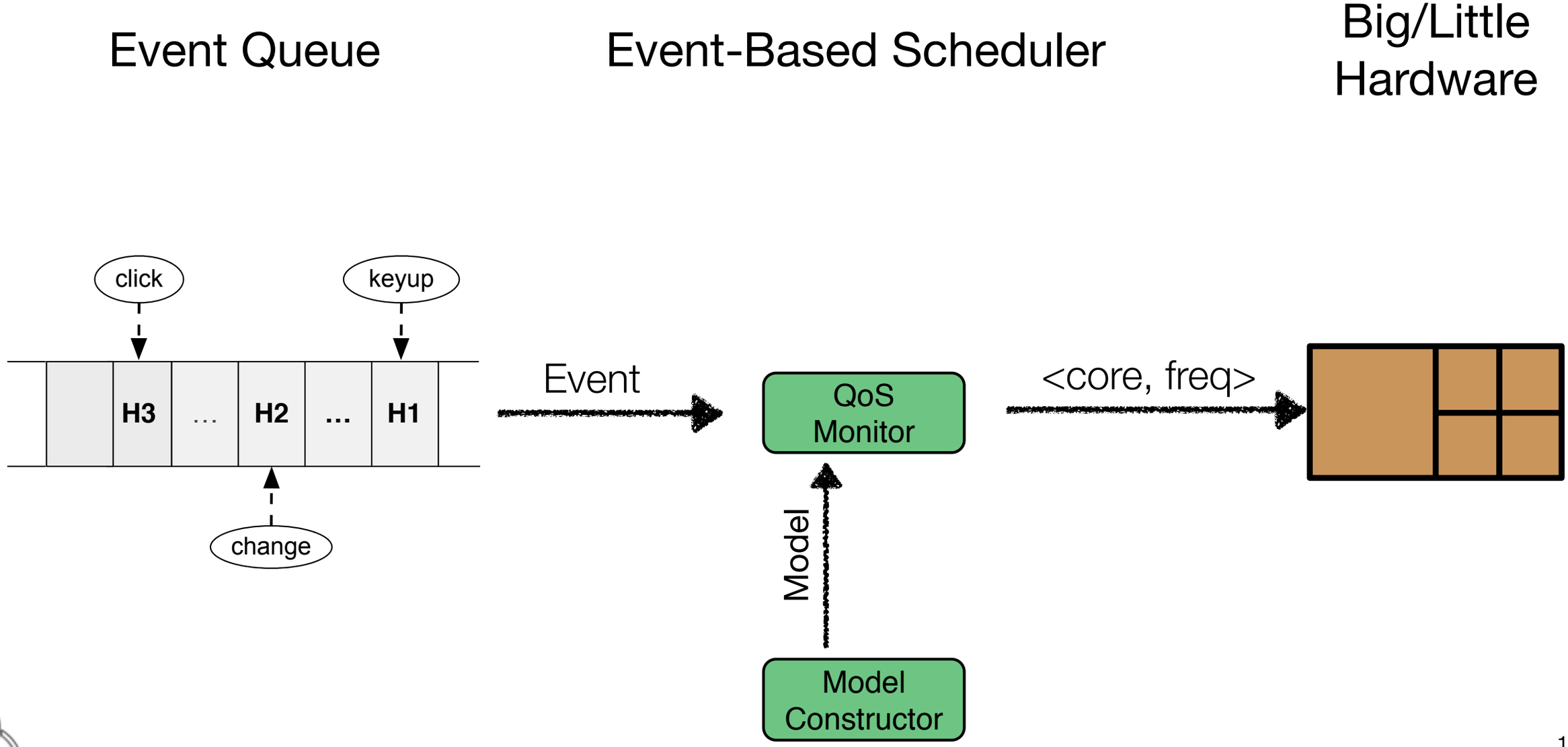
Event-Based Scheduler (EBS) Overview

- **Goal**: Find the most energy-efficient architectural configuration that meets the QoS target for each event



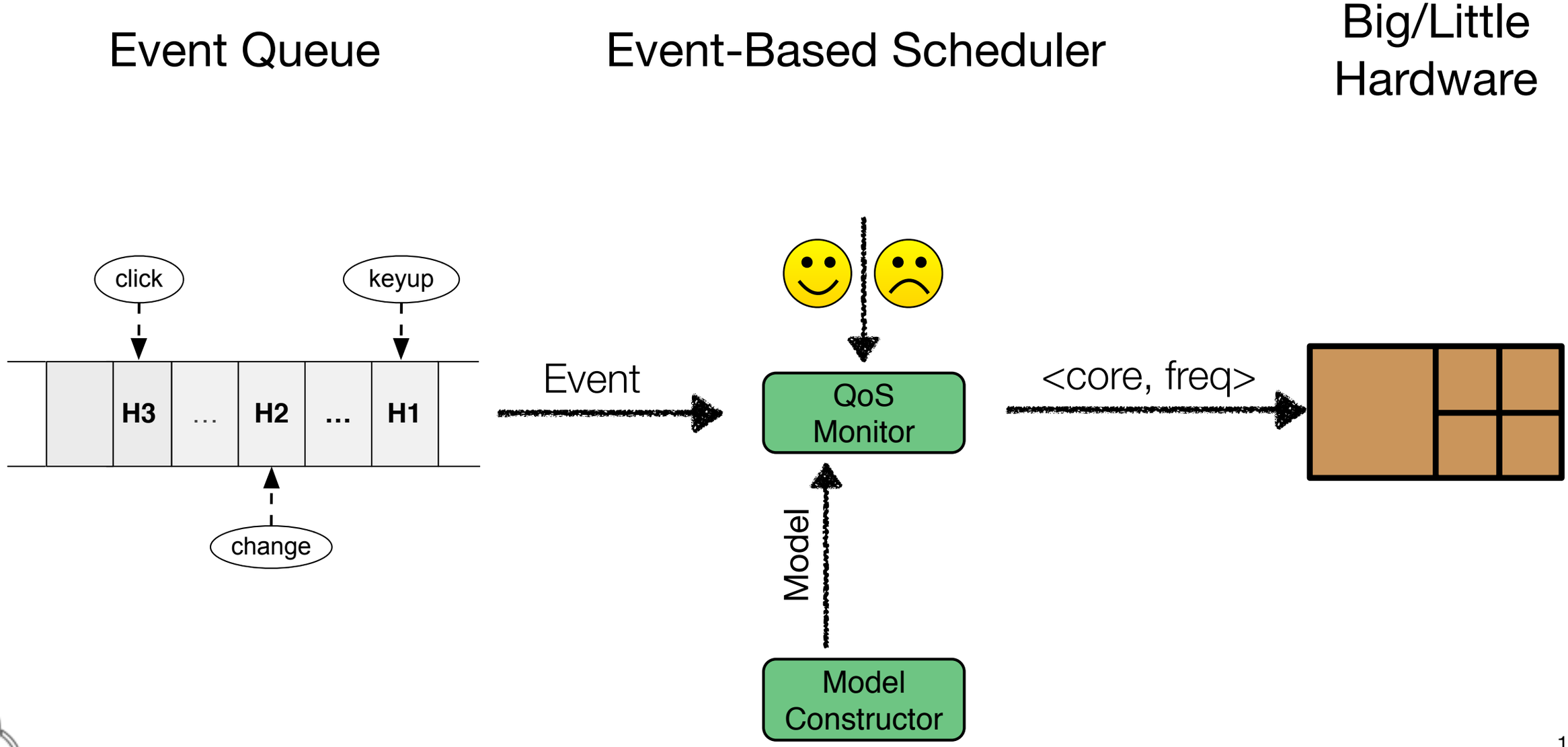
Event-Based Scheduler (EBS) Overview

- **Goal**: Find the most energy-efficient architectural configuration that meets the QoS target for each event



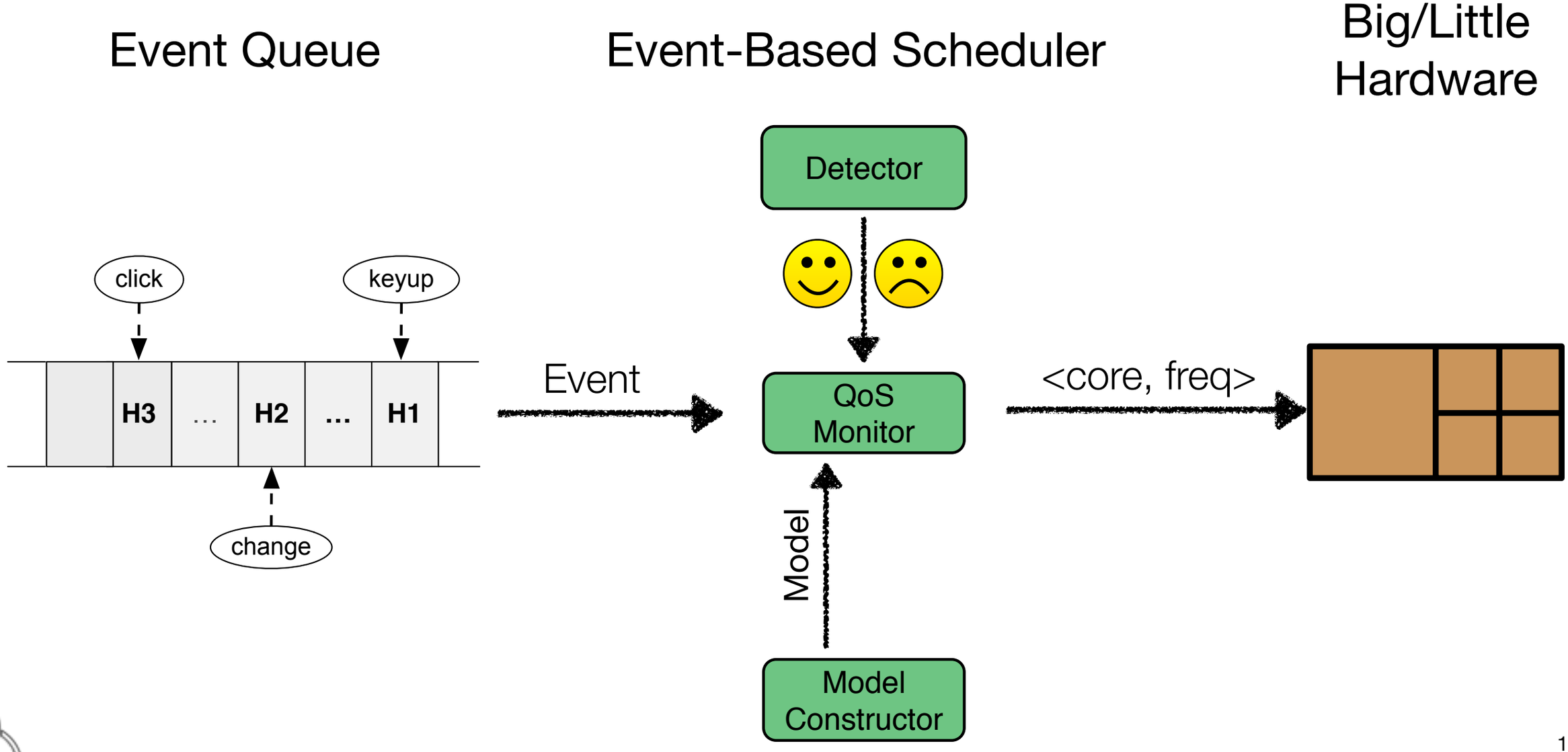
Event-Based Scheduler (EBS) Overview

- **Goal**: Find the most energy-efficient architectural configuration that meets the QoS target for each event



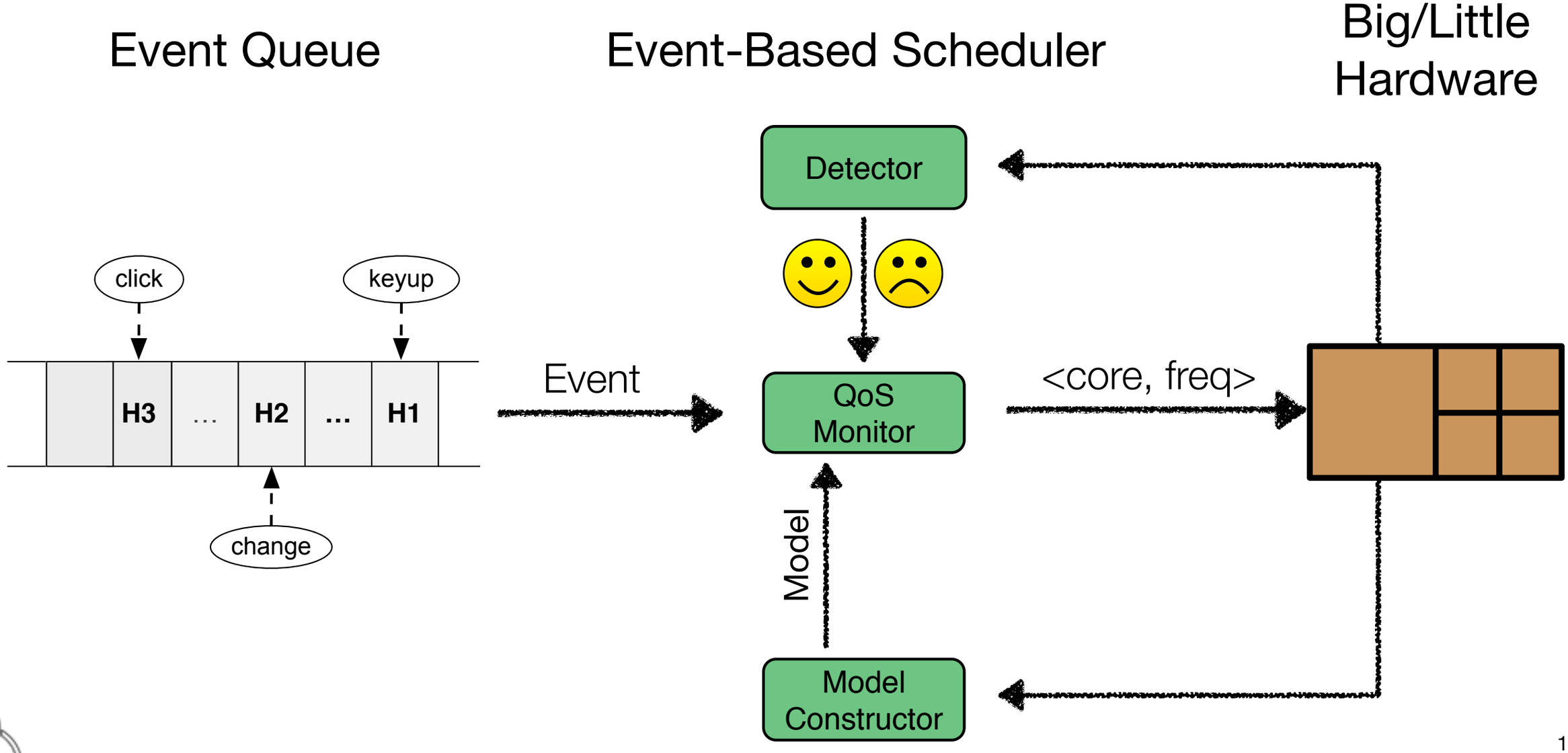
Event-Based Scheduler (EBS) Overview

- **Goal**: Find the most energy-efficient architectural configuration that meets the QoS target for each event



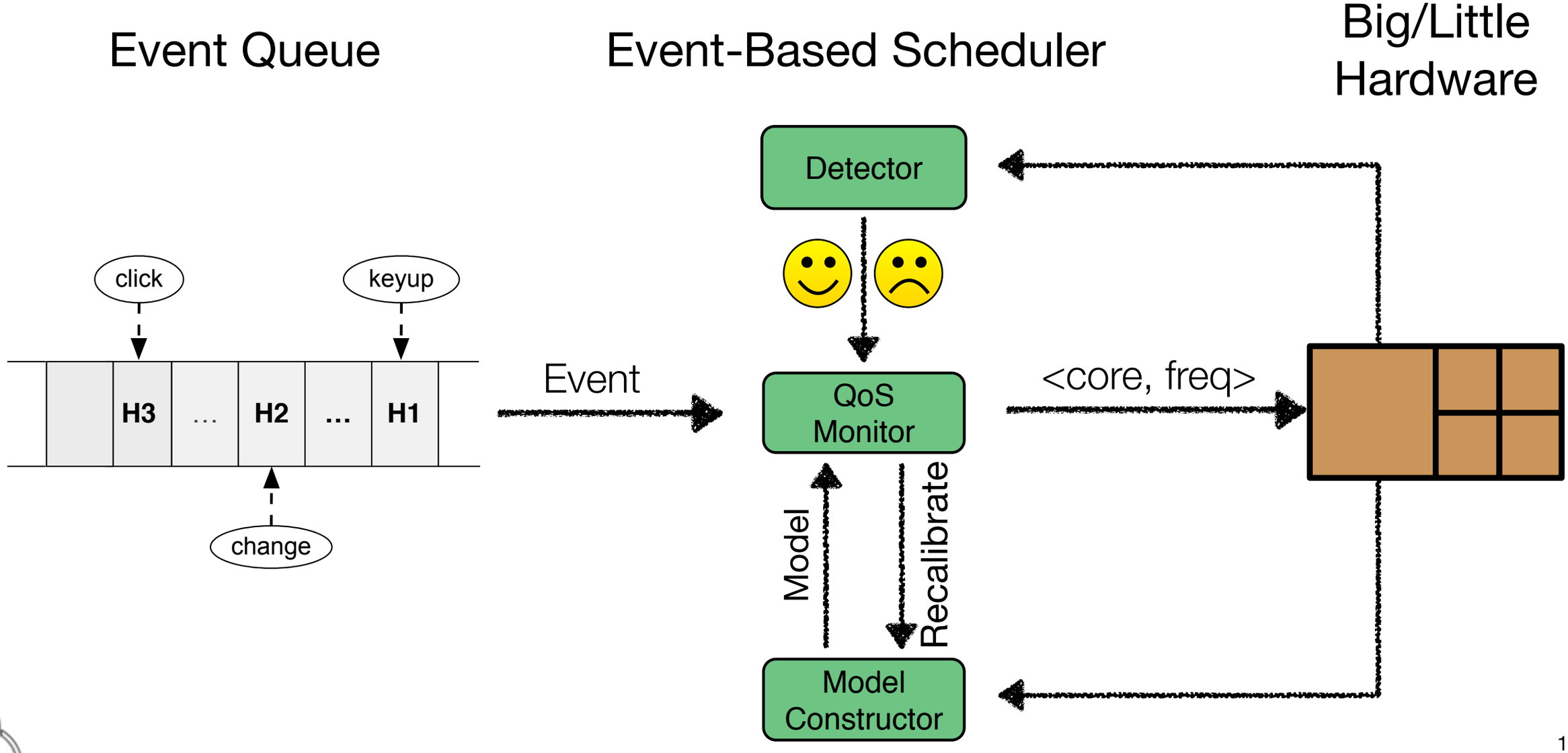
Event-Based Scheduler (EBS) Overview

- **Goal**: Find the most energy-efficient architectural configuration that meets the QoS target for each event



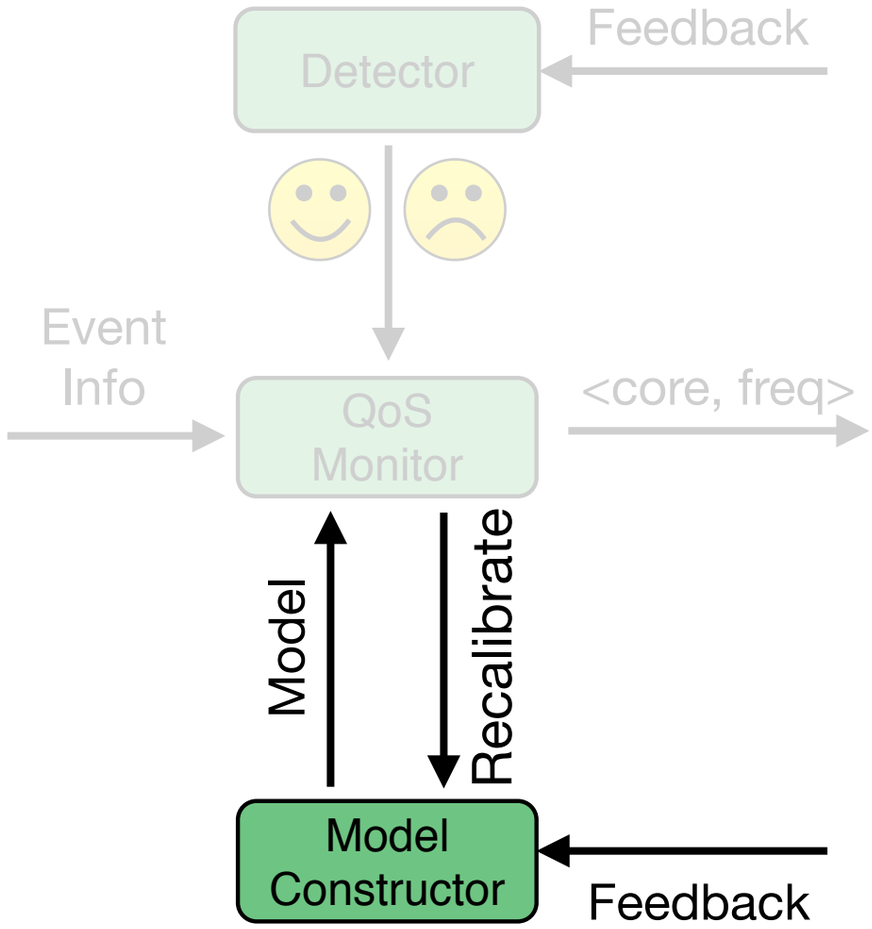
Event-Based Scheduler (EBS) Overview

- **Goal**: Find the most energy-efficient architectural configuration that meets the QoS target for each event



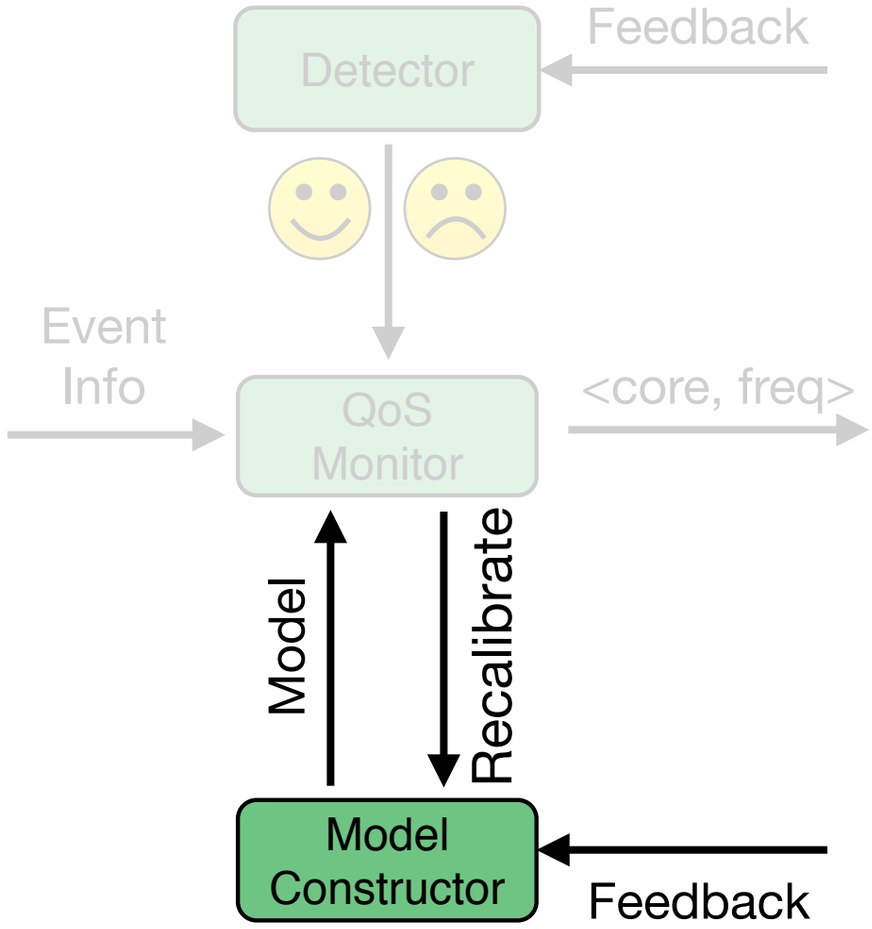
How the Model Constructor Works

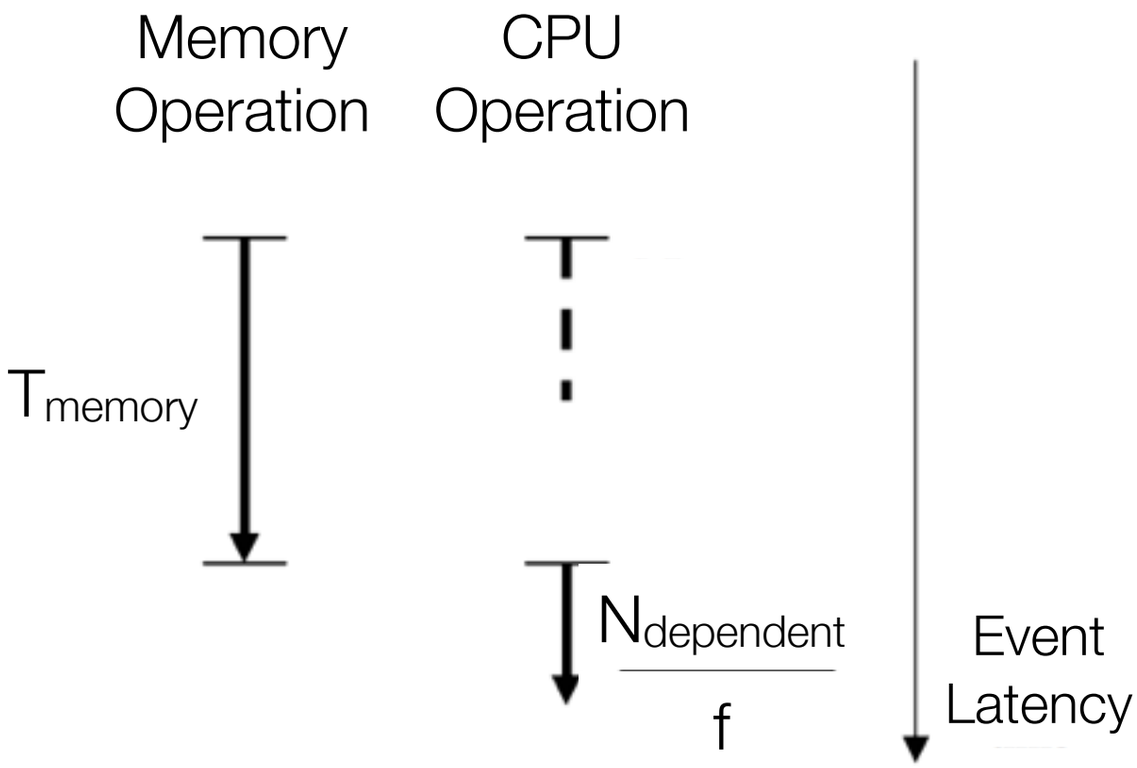
- **Goal**: Estimate the event latency under each <core, freq> config



How the Model Constructor Works

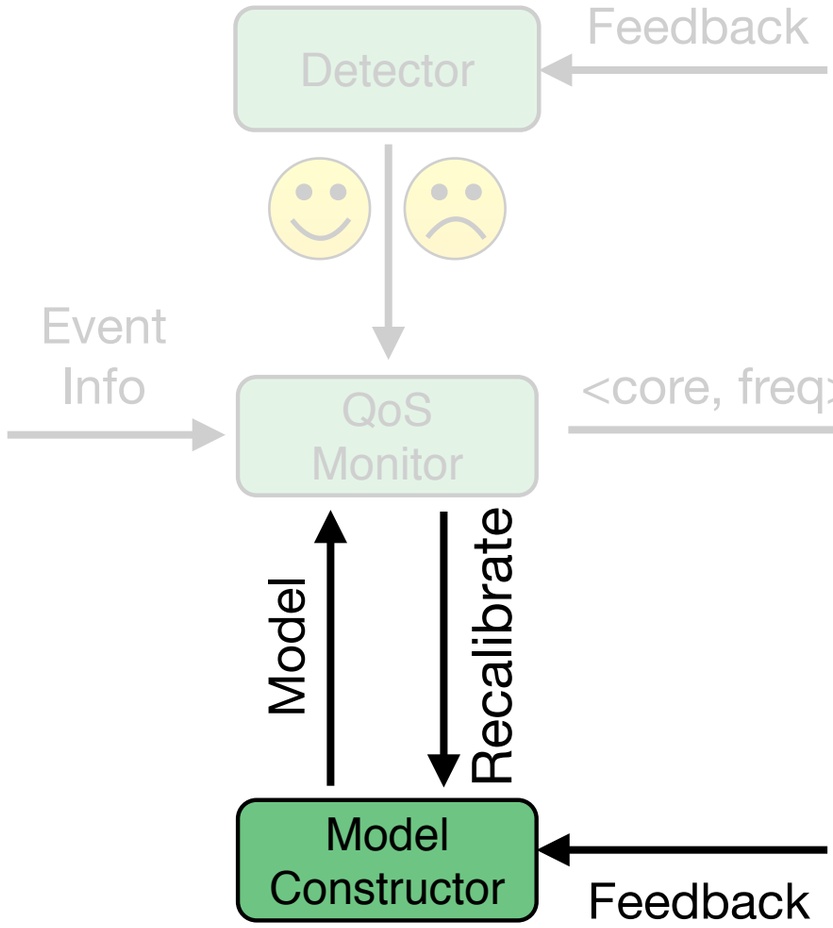
- **Goal**: Estimate the event latency under each <core, freq> config

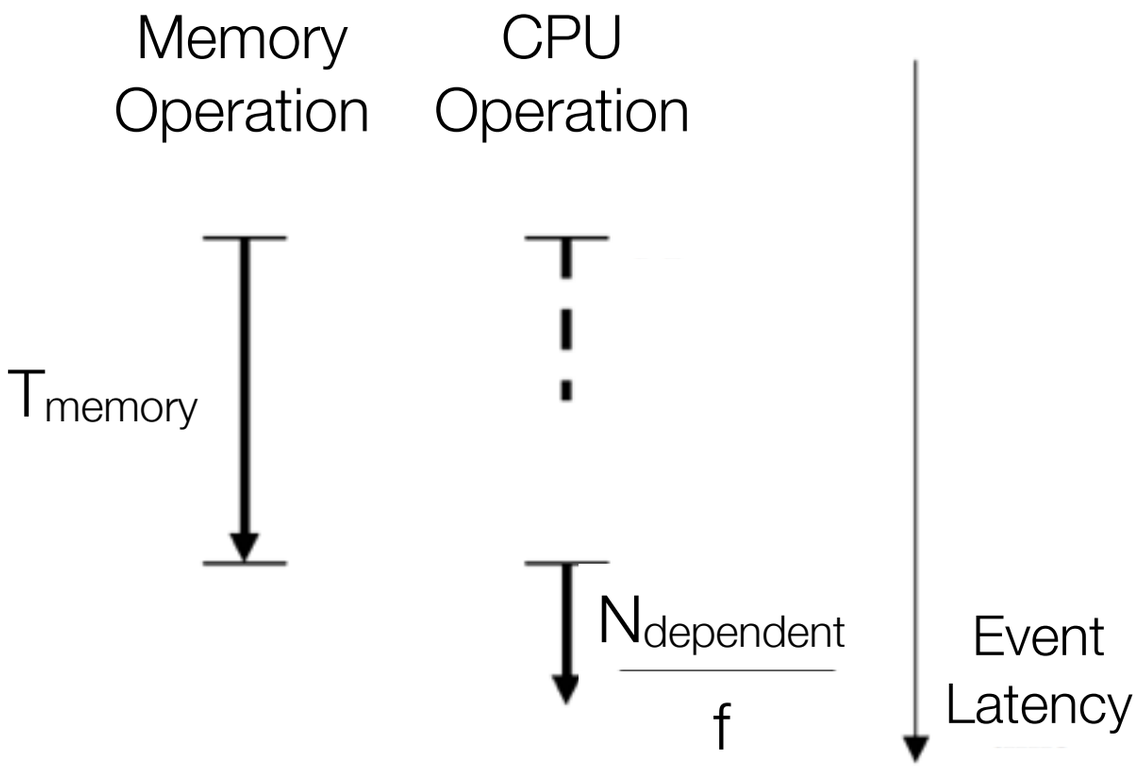




How the Model Constructor Works

- **Goal**: Estimate the event latency under each <core, freq> config



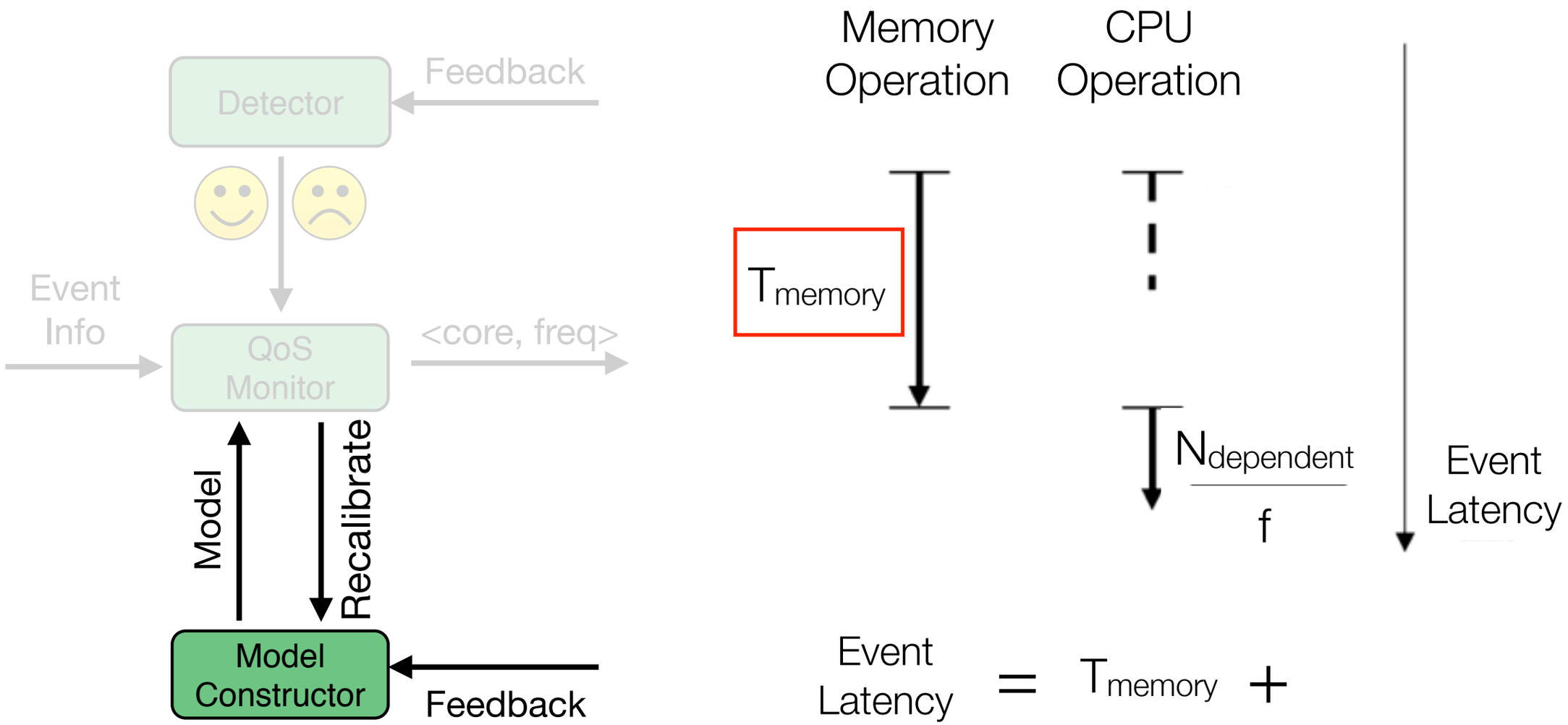


Event Latency =

Xie, et al., Compile-Time Dynamic Voltage Scaling Settings: Opportunities and Limits, PLDI'03

How the Model Constructor Works

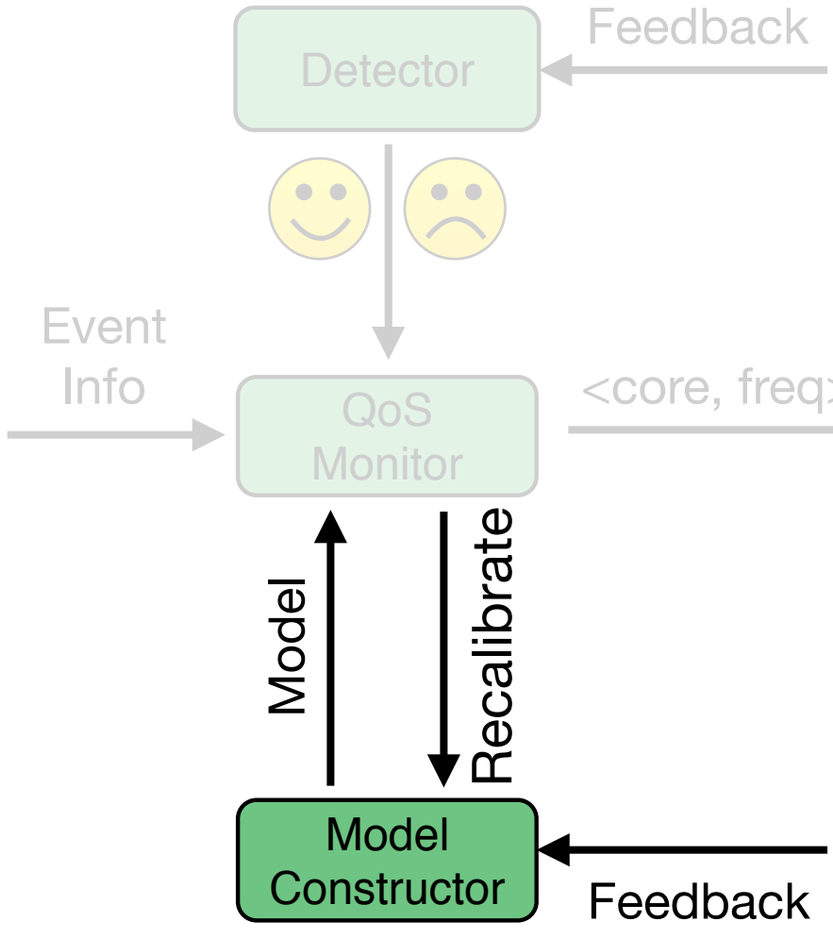
- **Goal**: Estimate the event latency under each <core, freq> config

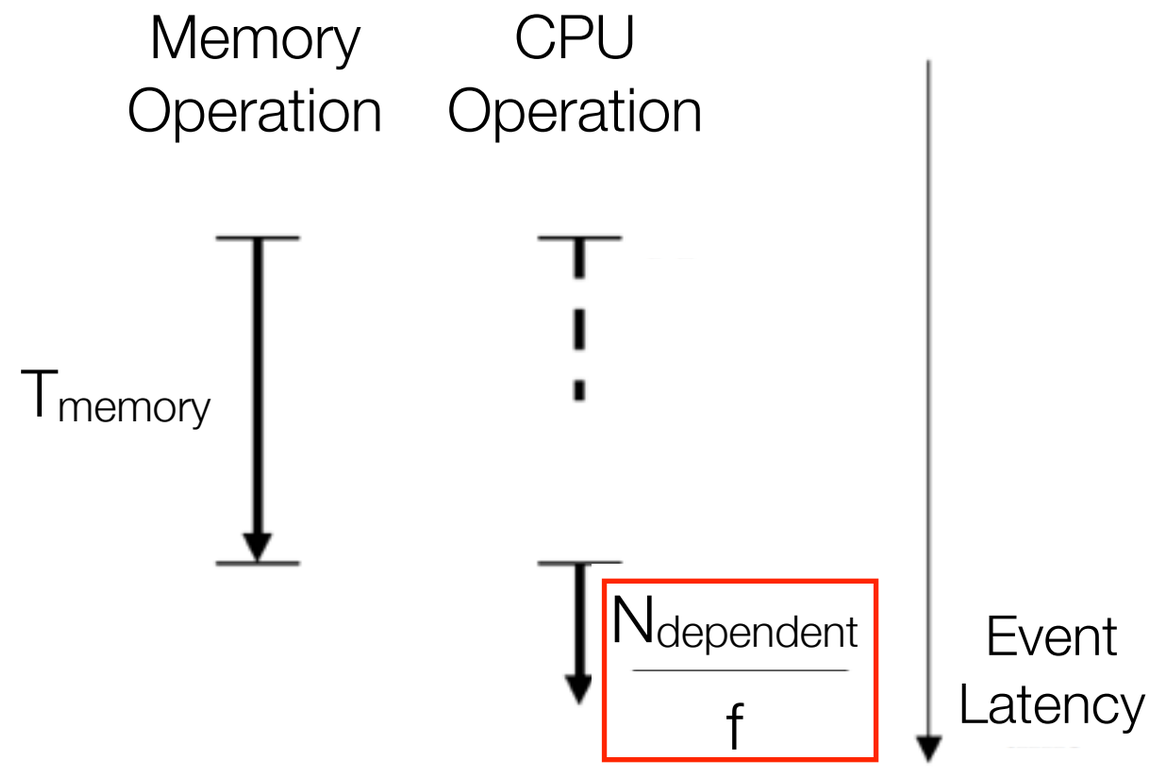


Xie, et al., Compile-Time Dynamic Voltage Scaling Settings: Opportunities and Limits, PLDI'03

How the Model Constructor Works

- **Goal**: Estimate the event latency under each <core, freq> config

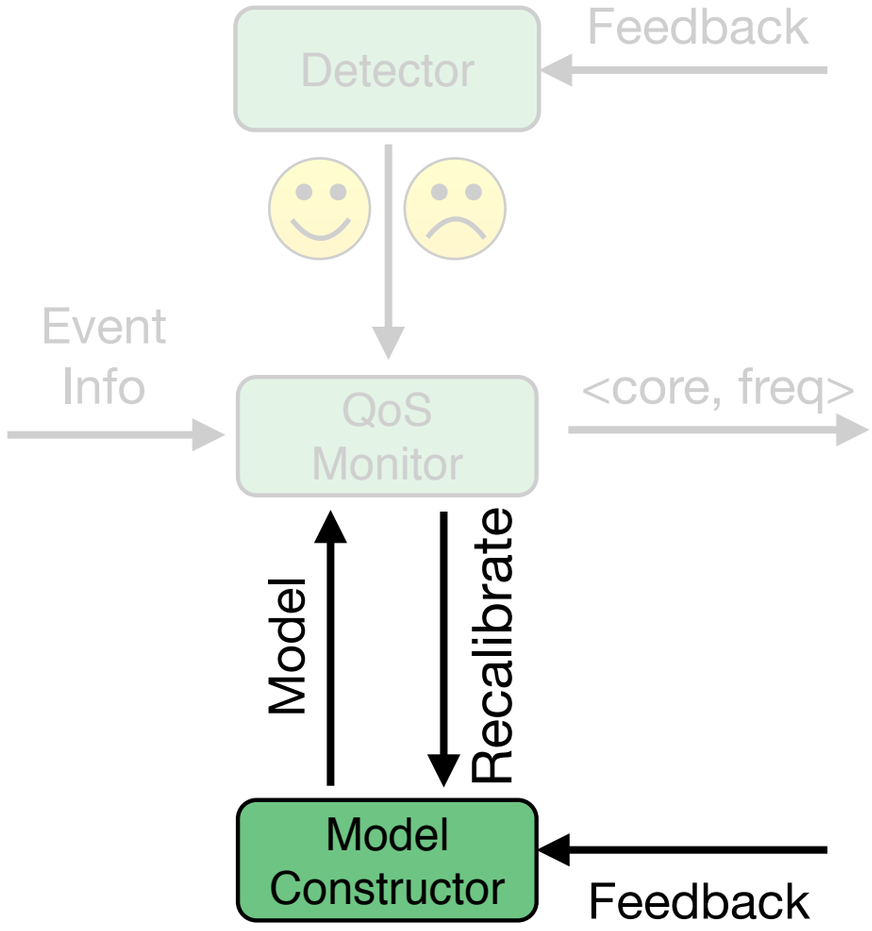


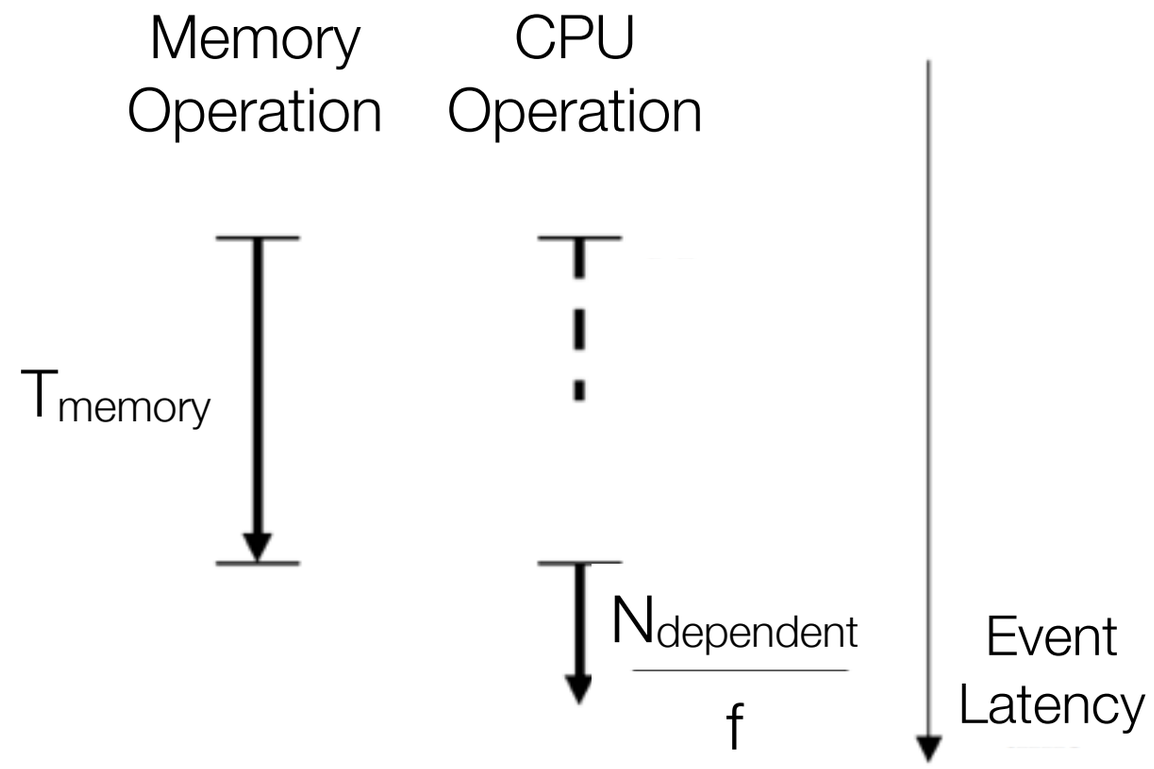


$$\text{Event Latency} = T_{memory} + N_{dependent} / f$$

How the Model Constructor Works

- **Goal**: Estimate the event latency under each <core, freq> config

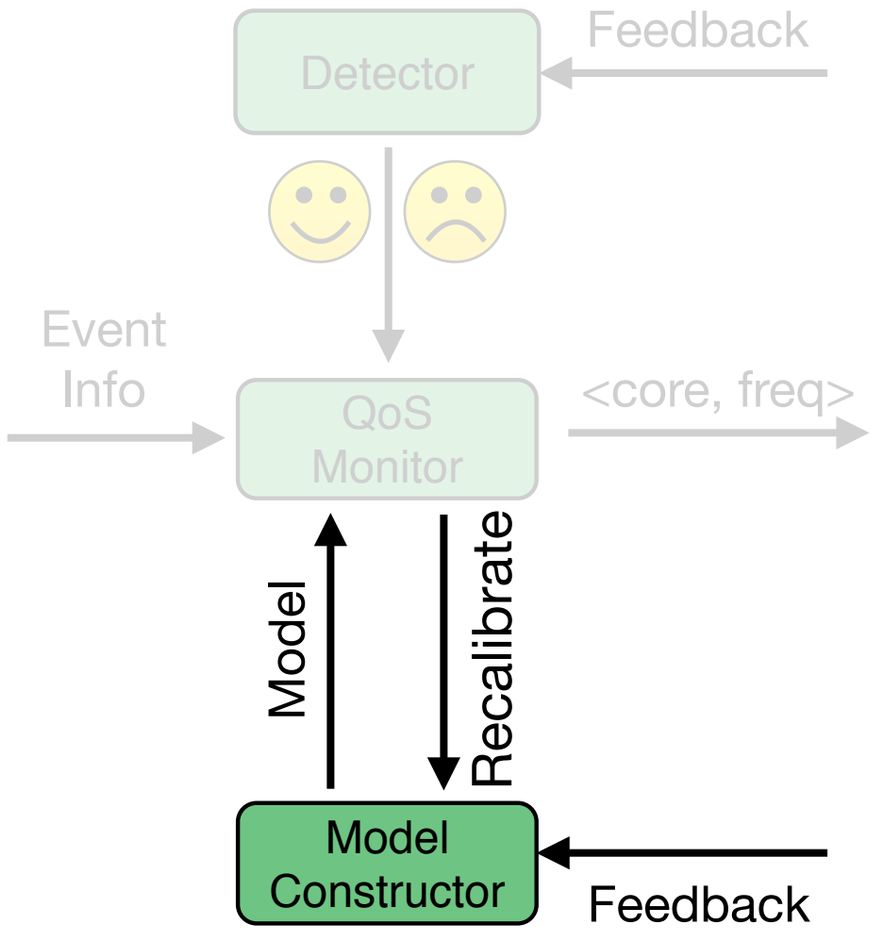


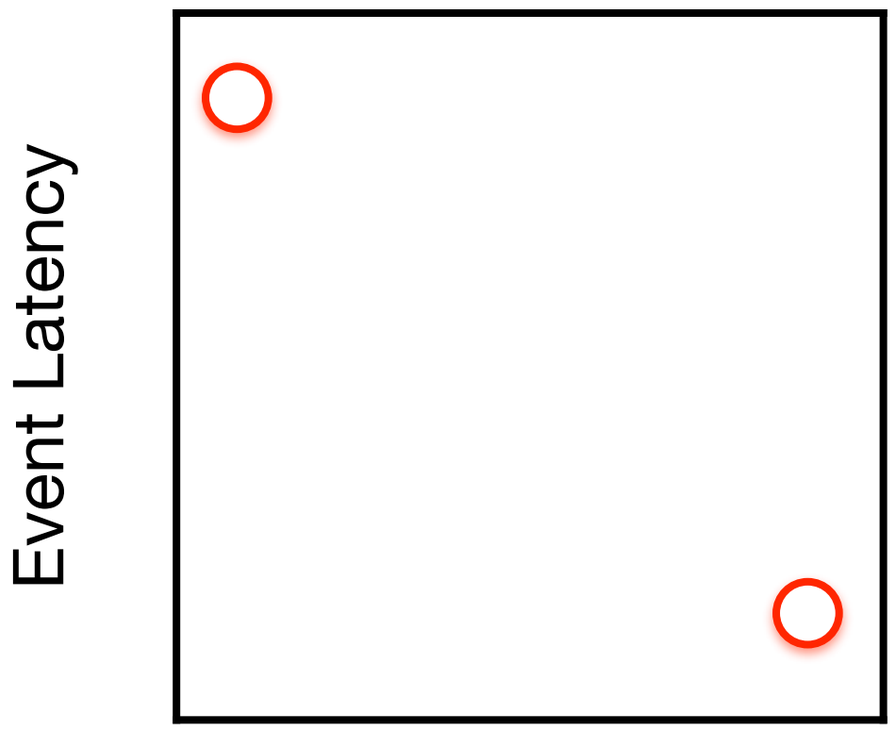


$$\text{Event Latency} = \boxed{T_{memory}} + \boxed{N_{dependent}} / f$$

How the Model Constructor Works

- **Goal**: Estimate the event latency under each <core, freq> config

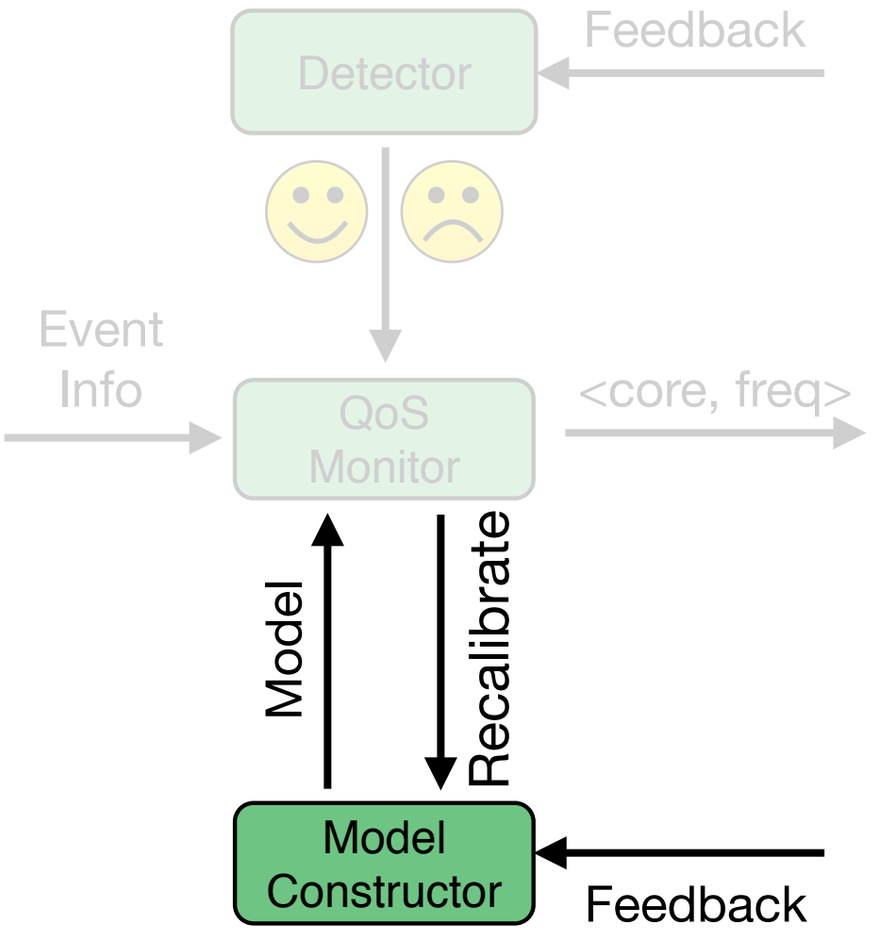


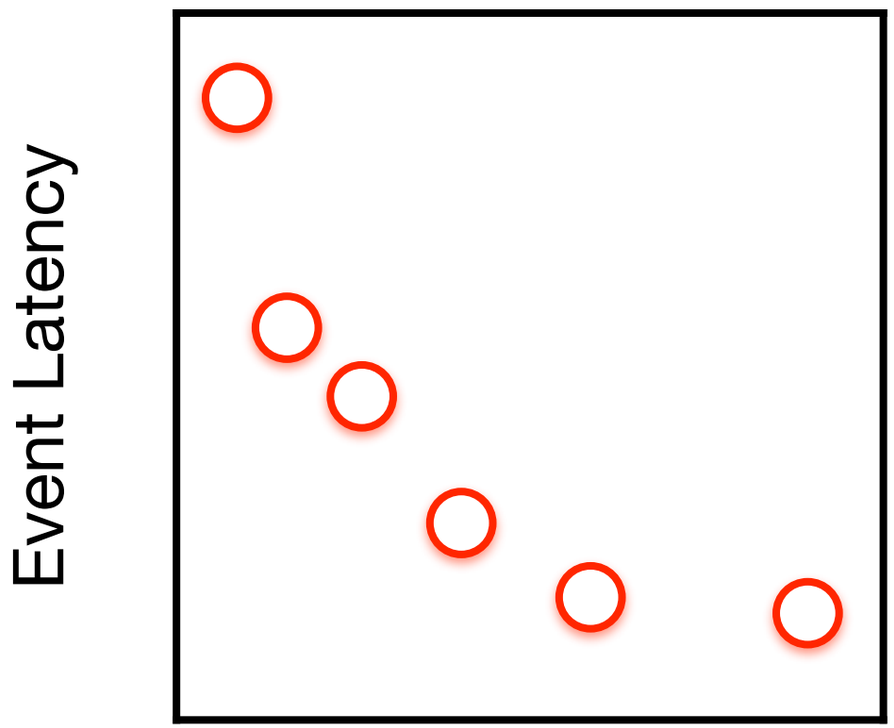


$$\text{Event Latency} = T_{memory} + N_{dependent} / f$$

How the Model Constructor Works

- **Goal**: Estimate the event latency under each <core, freq> config



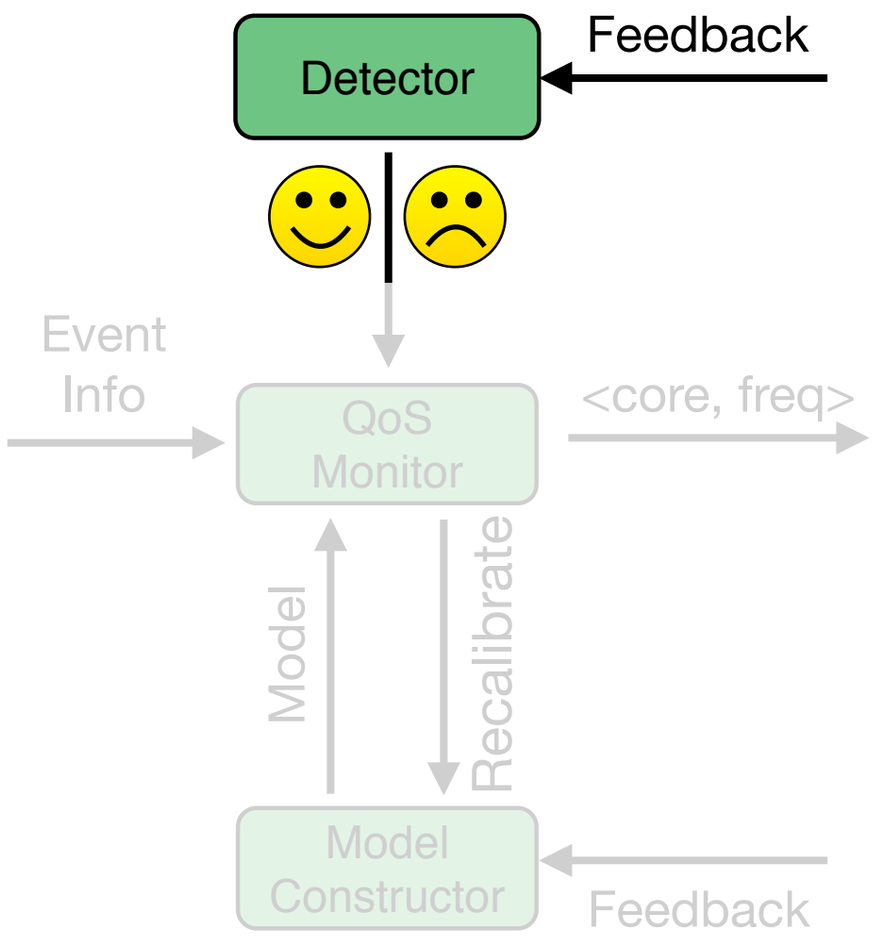


Frequency

$$\text{Event Latency} = T_{memory} + N_{dependent} / f$$

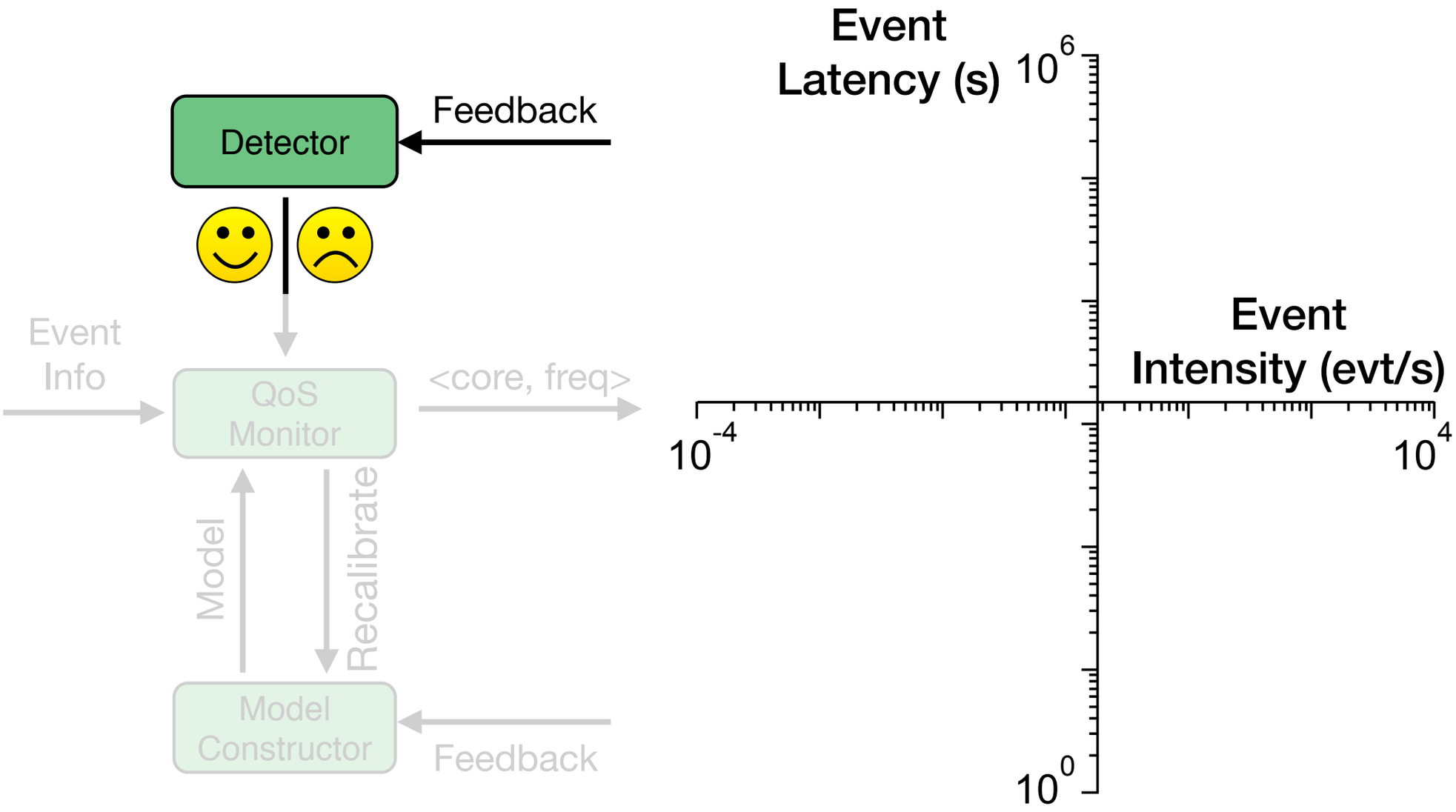
How the Detector Works

- **Goal**: Find the QoS target for each event



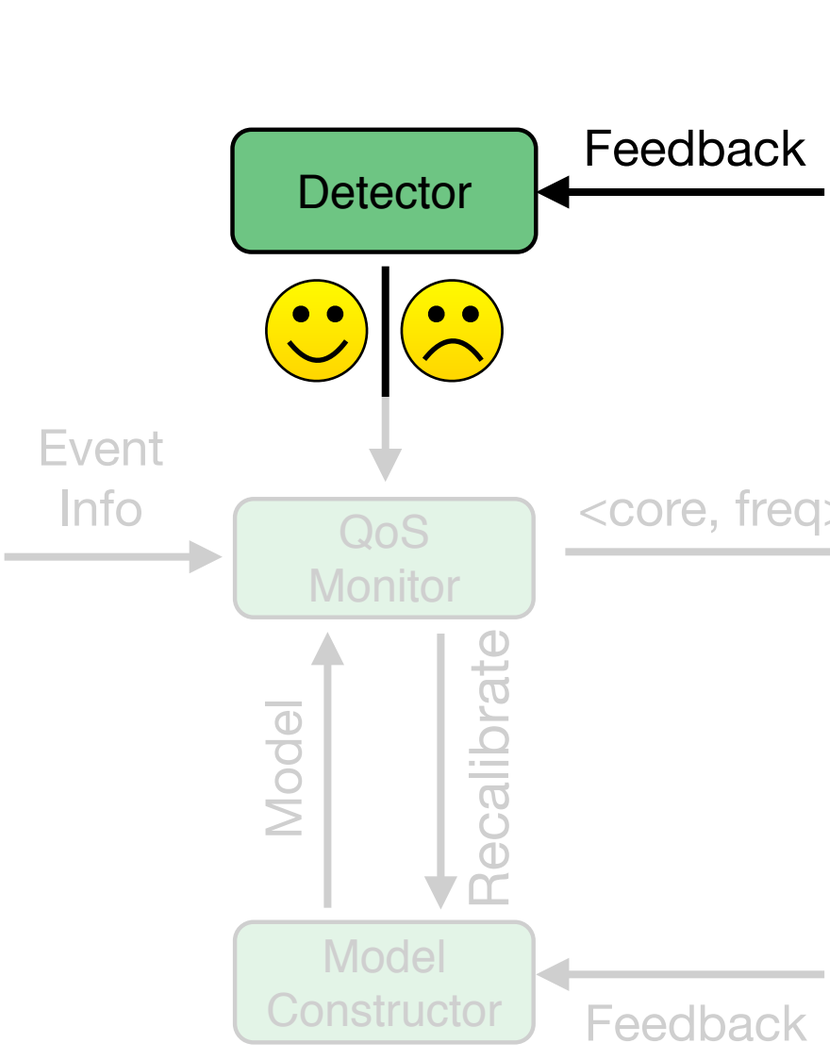
How the Detector Works

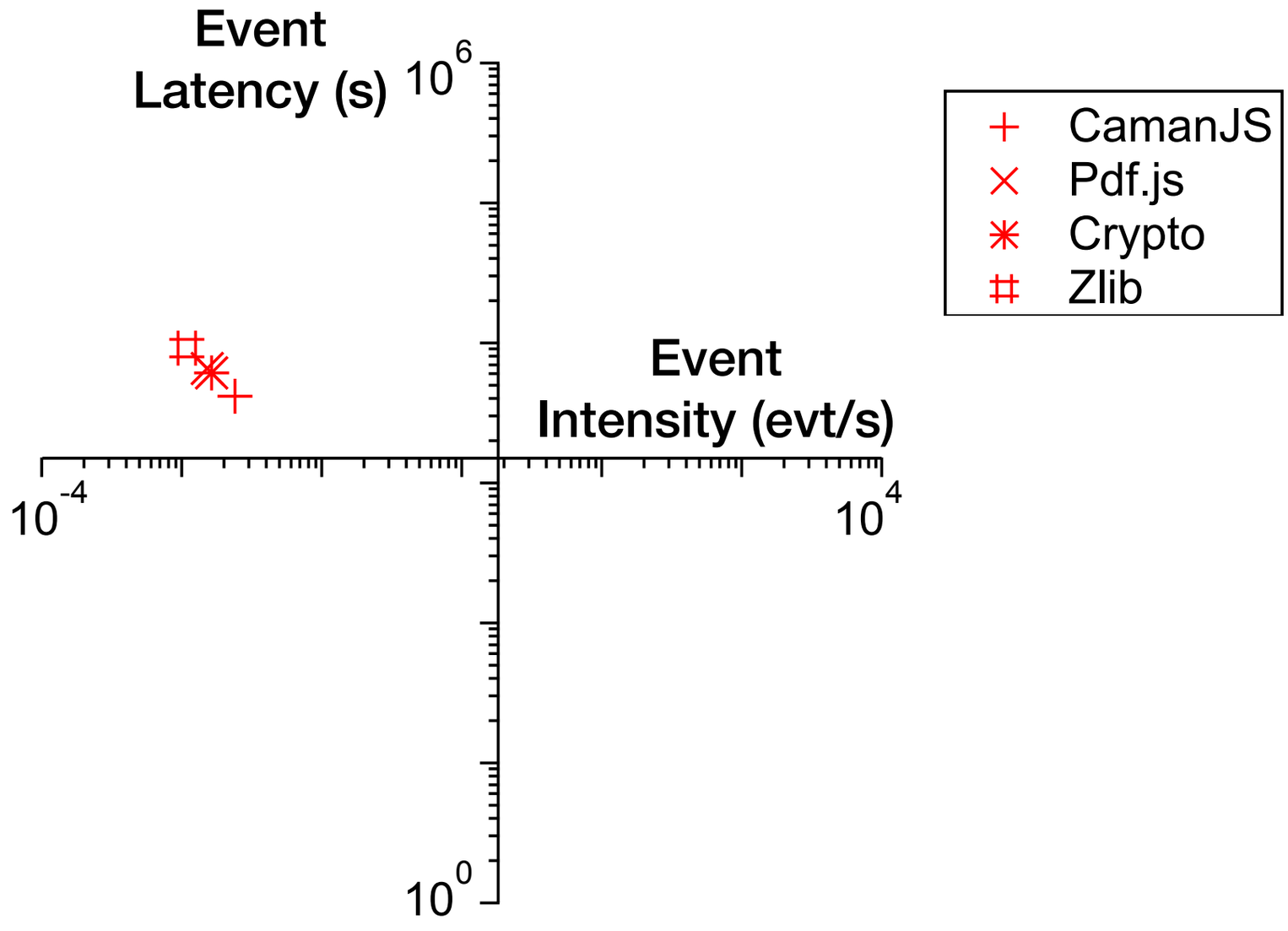
- **Goal**: Find the QoS target for each event



How the Detector Works

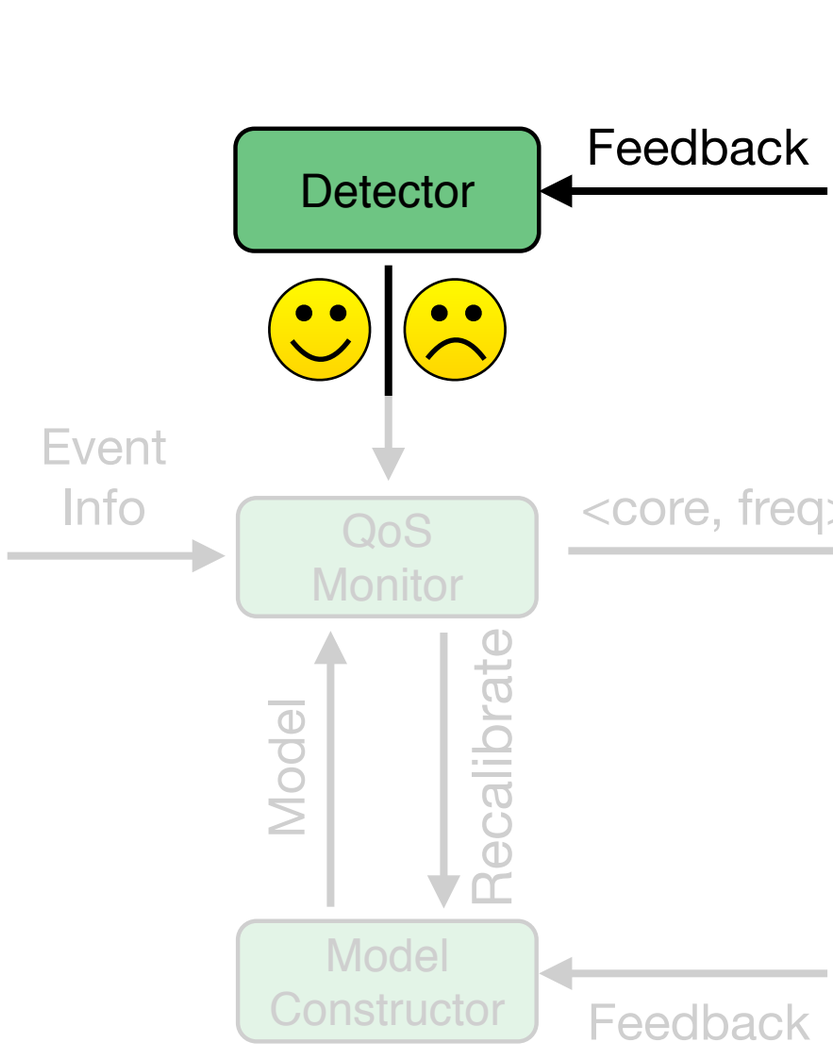
- **Goal**: Find the QoS target for each event

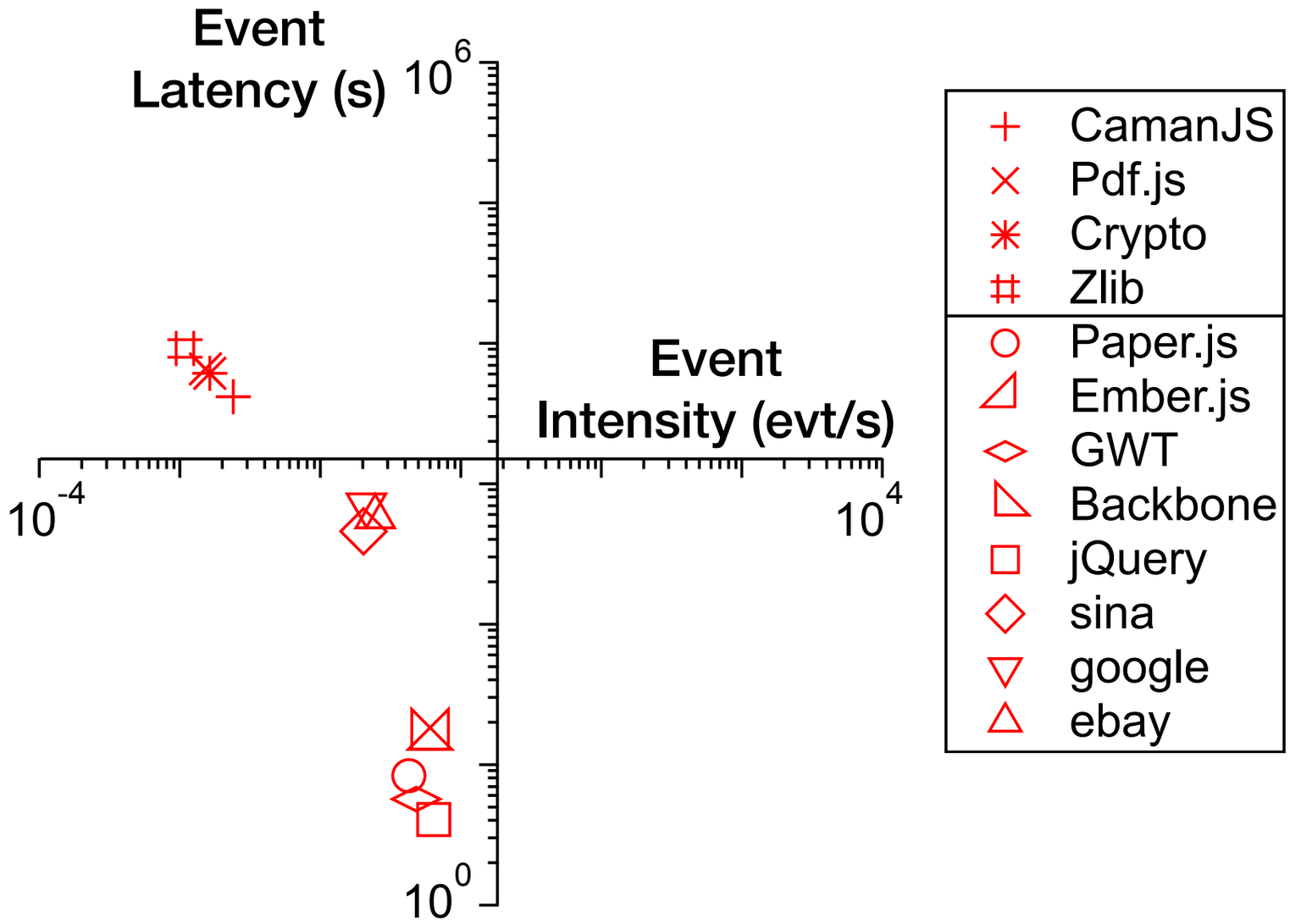




How the Detector Works

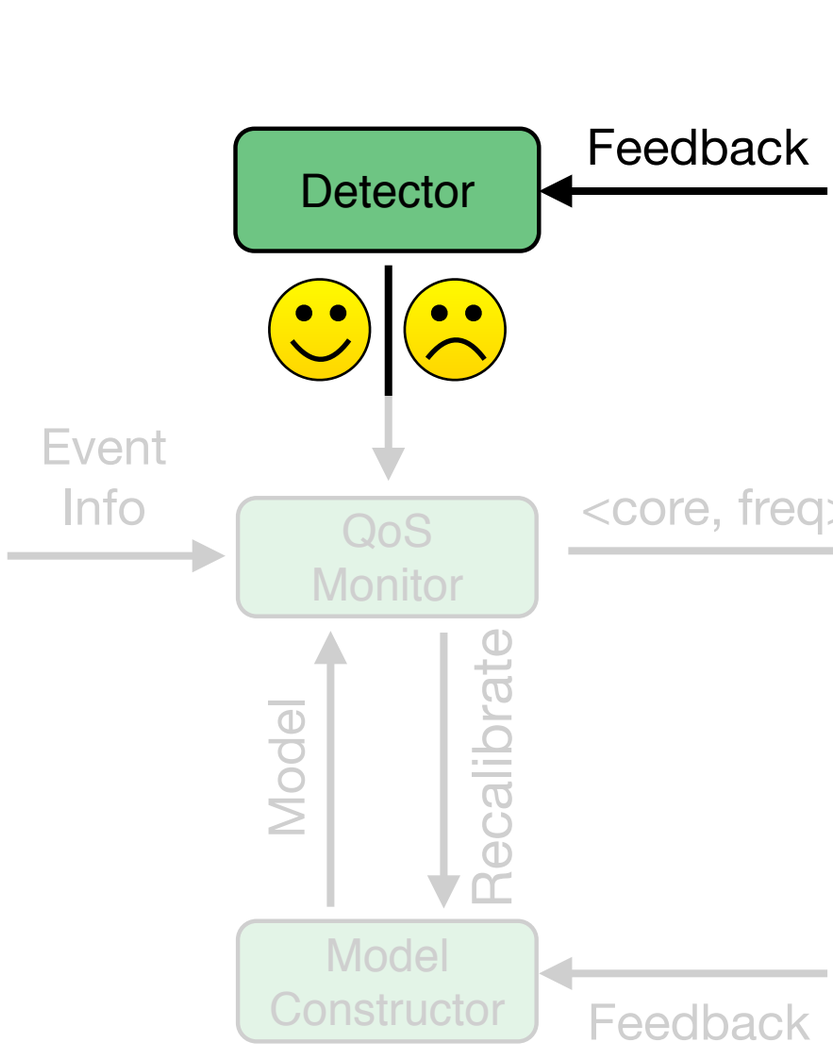
- **Goal**: Find the QoS target for each event

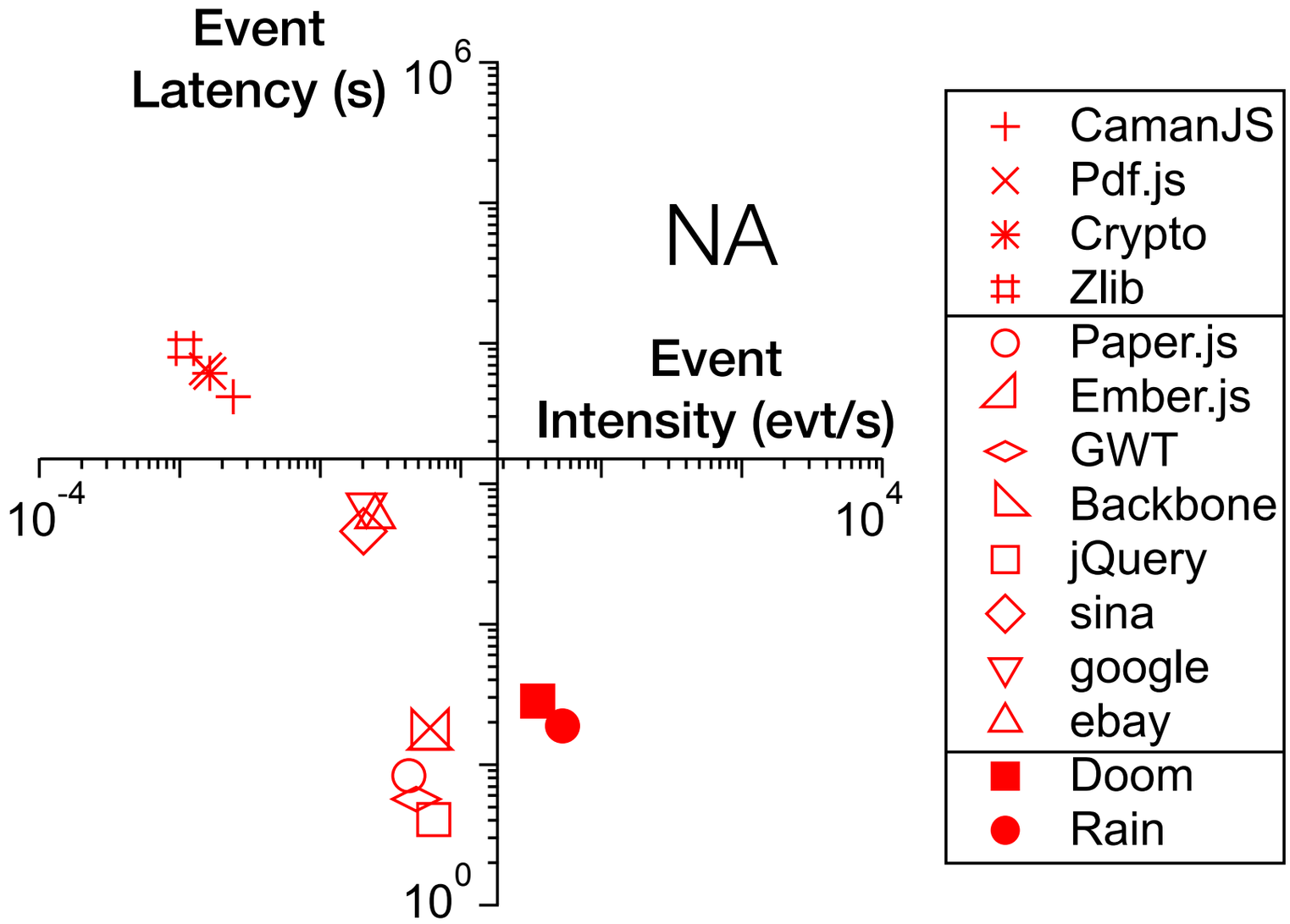




How the Detector Works

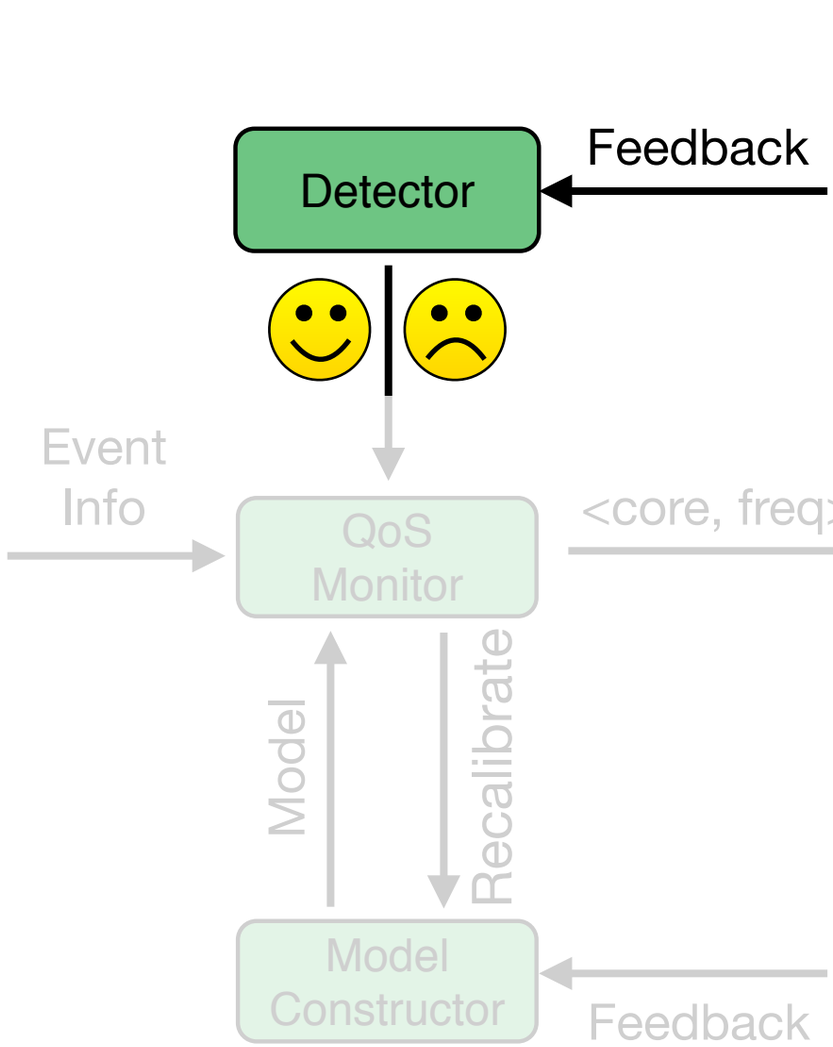
- **Goal**: Find the QoS target for each event

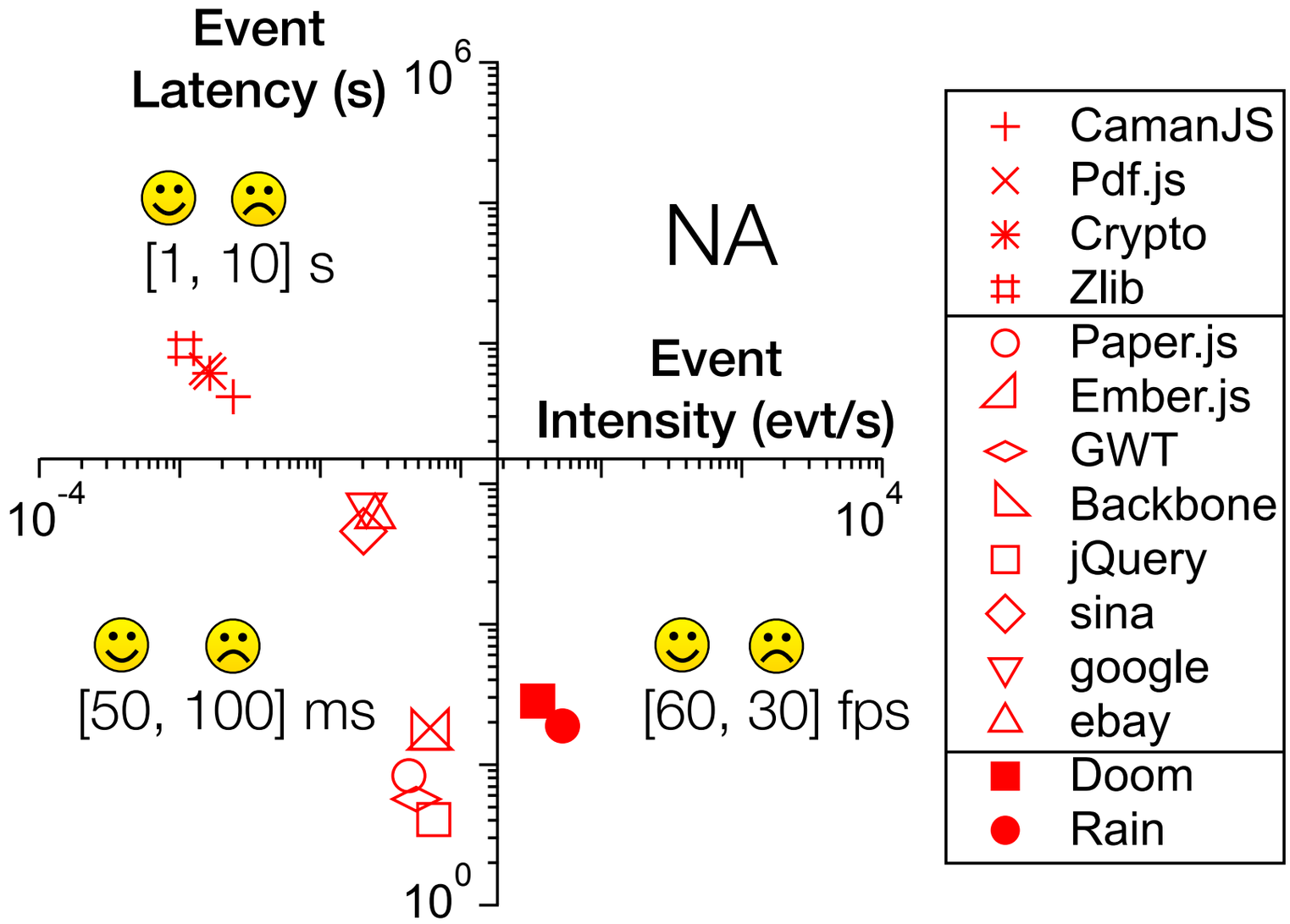




How the Detector Works

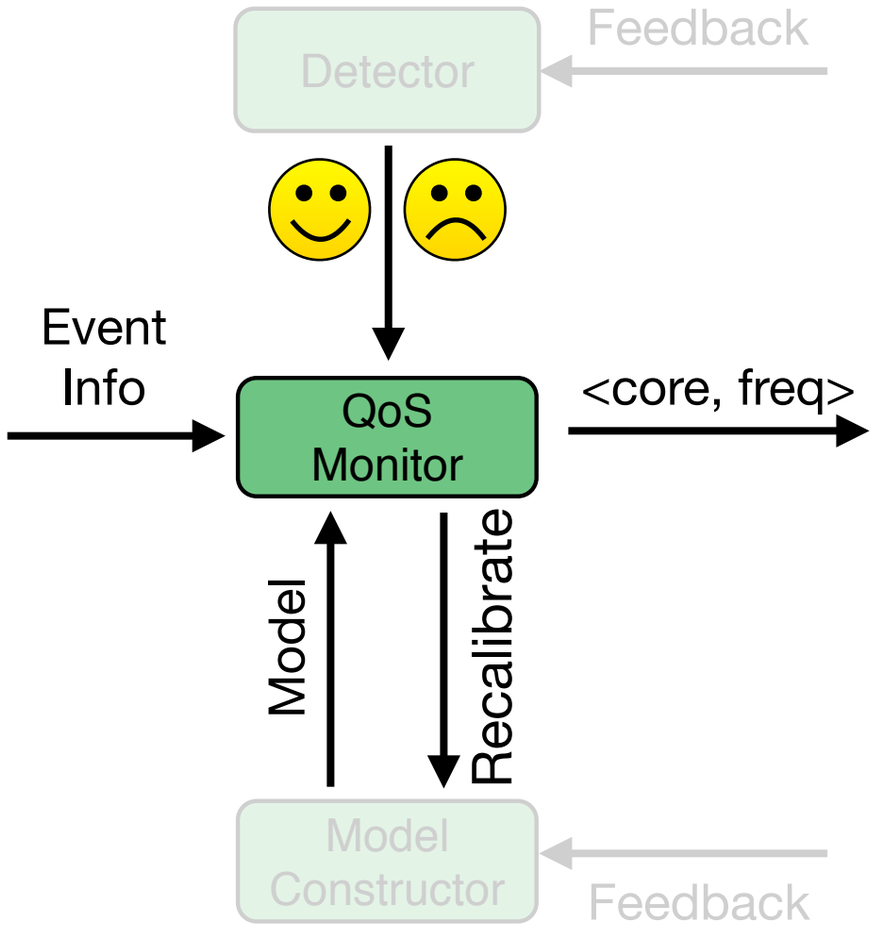
- **Goal**: Find the QoS target for each event





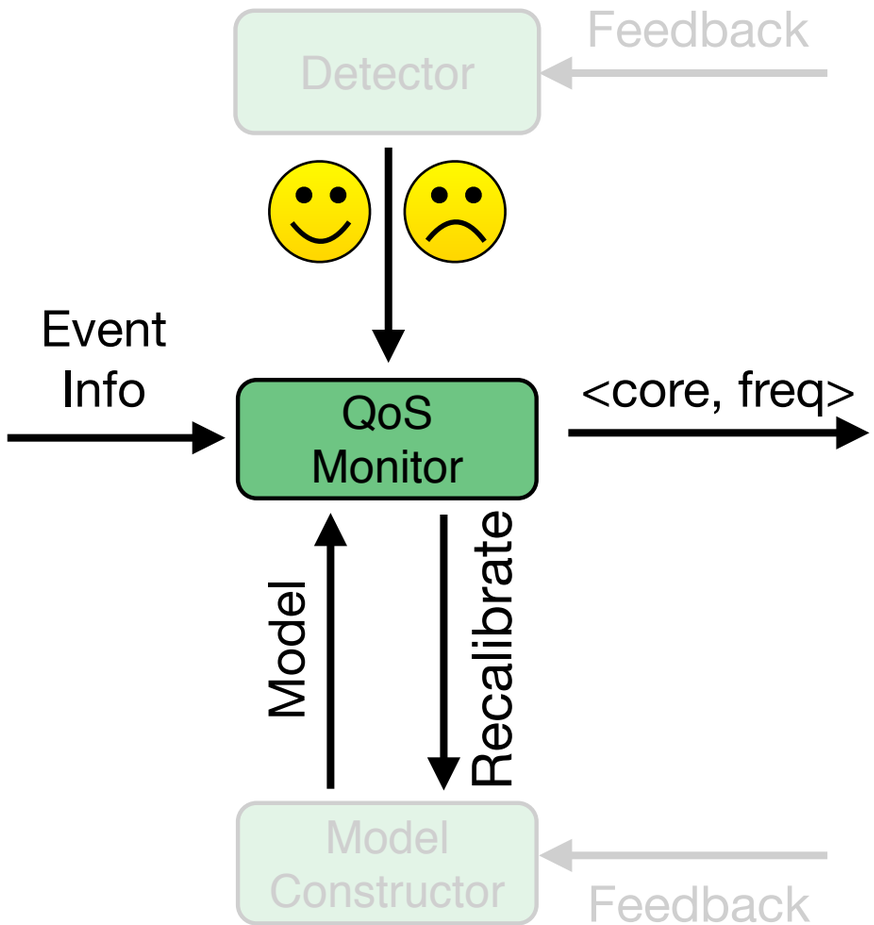
How the QoS Monitor Works

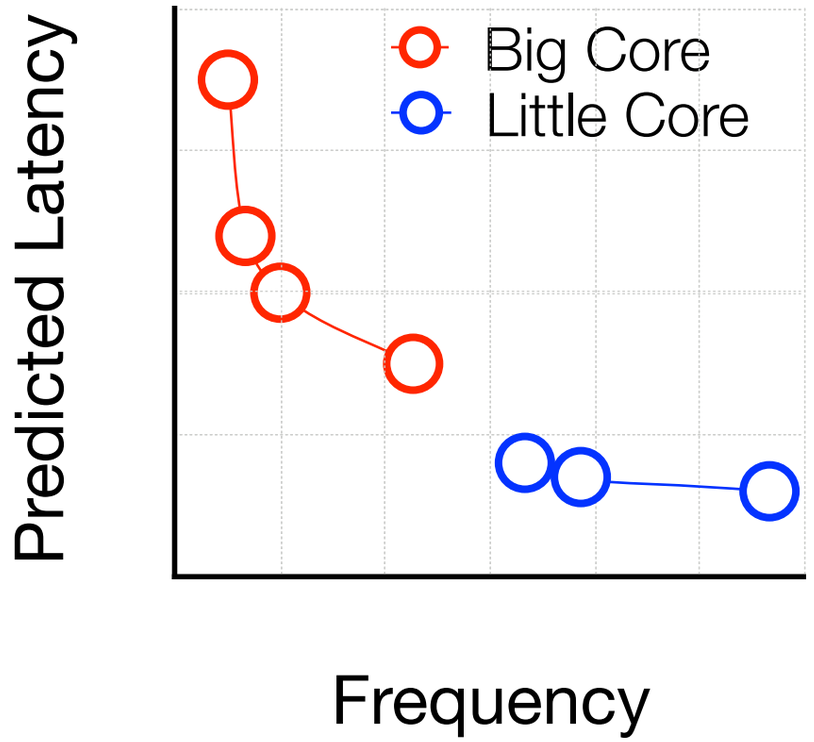
- **Goal**: Predict the <core, frequency> for the next event



How the QoS Monitor Works

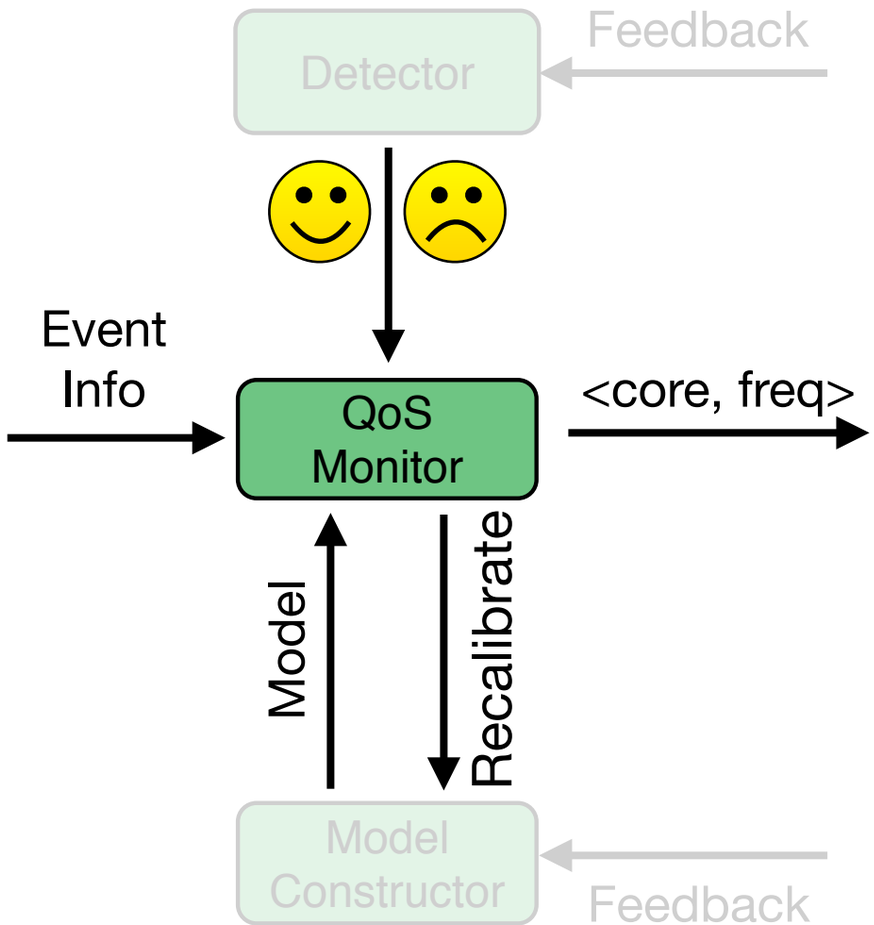
- **Goal**: Predict the <core, frequency> for the next event

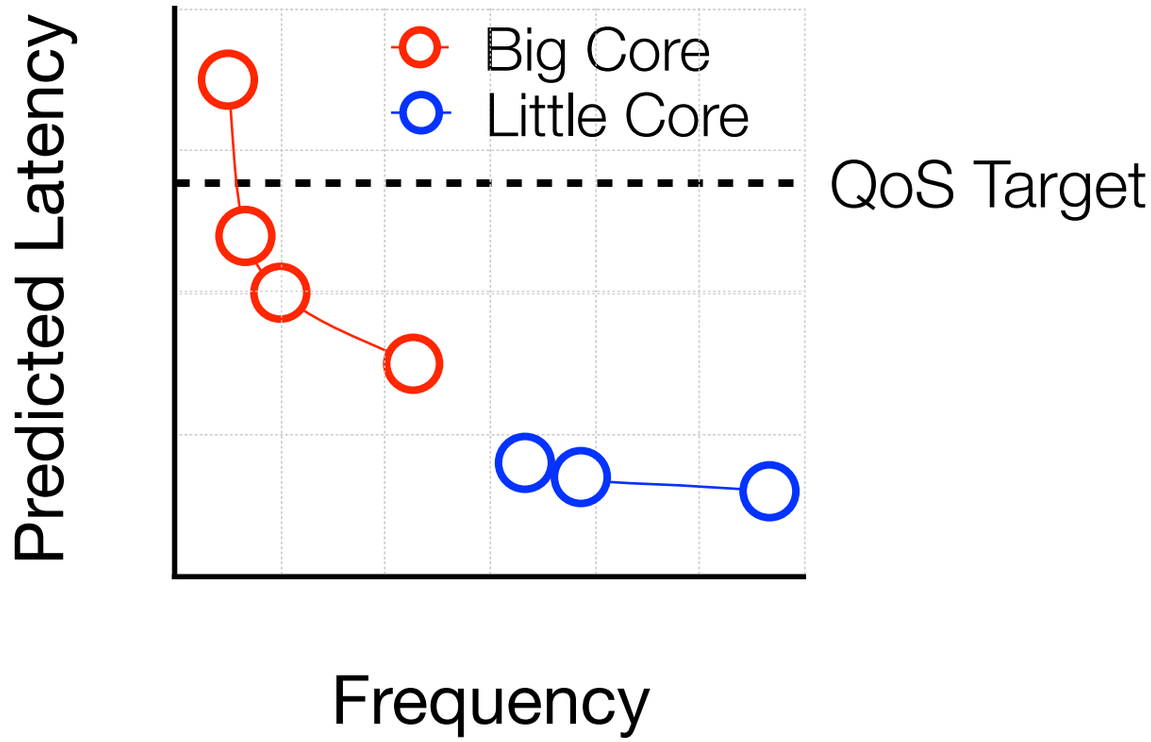




How the QoS Monitor Works

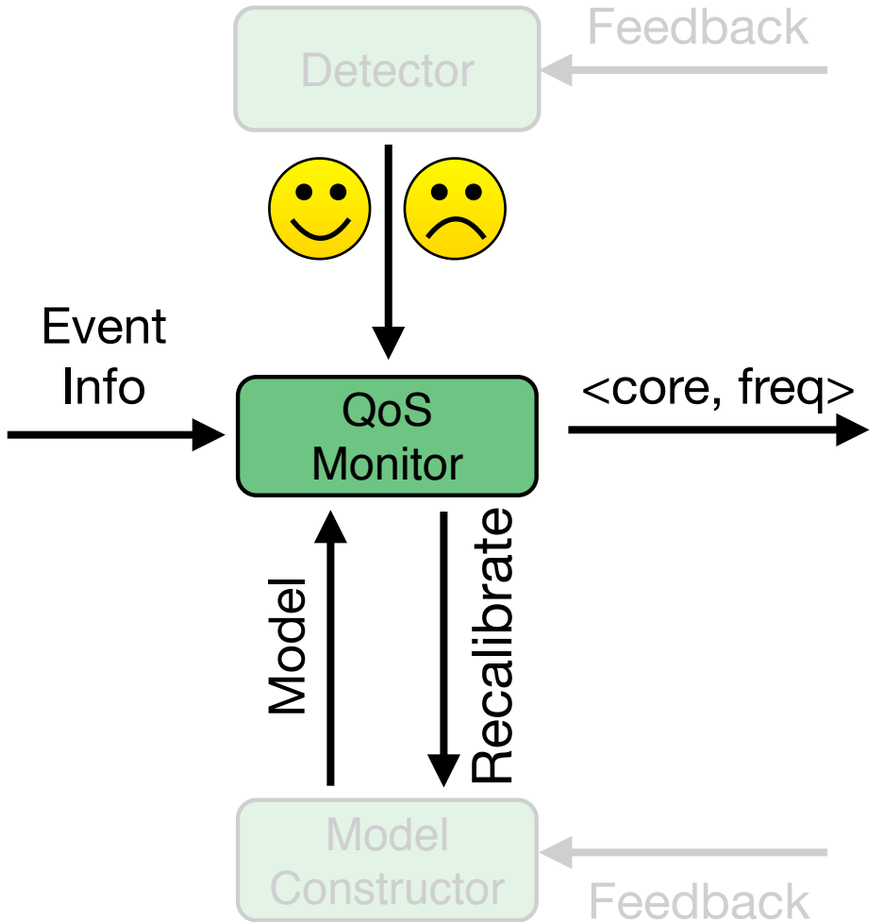
- **Goal**: Predict the <core, frequency> for the next event

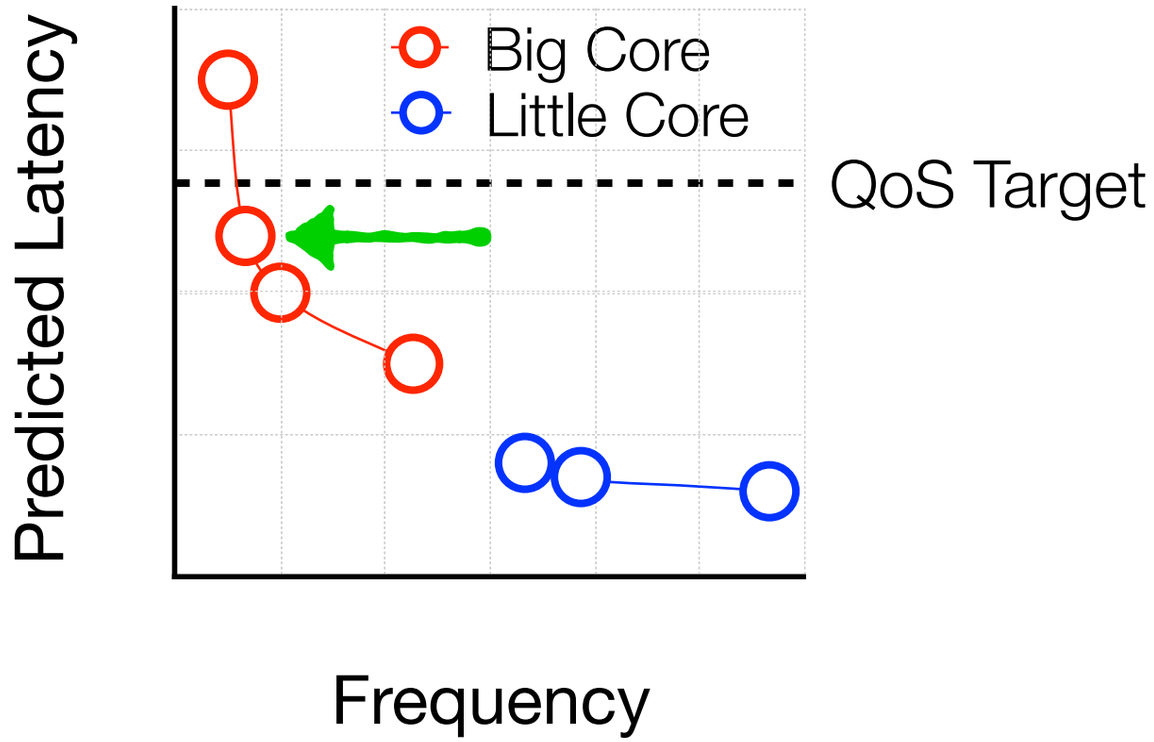




How the QoS Monitor Works

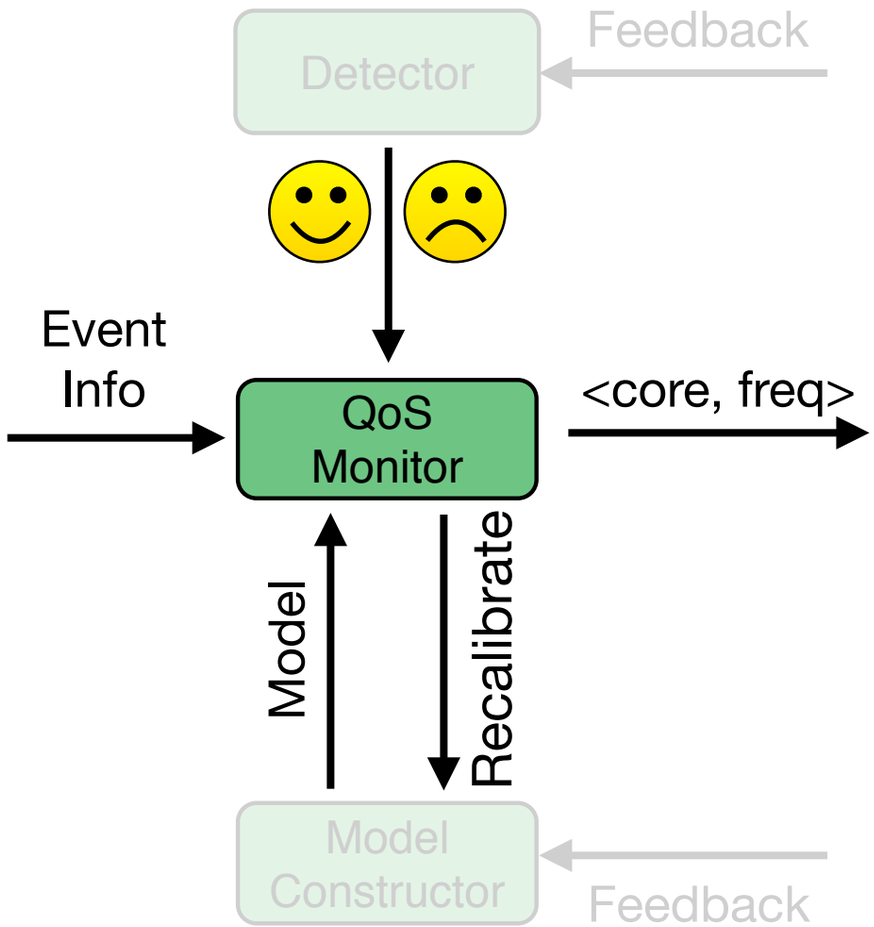
- **Goal**: Predict the <core, frequency> for the next event

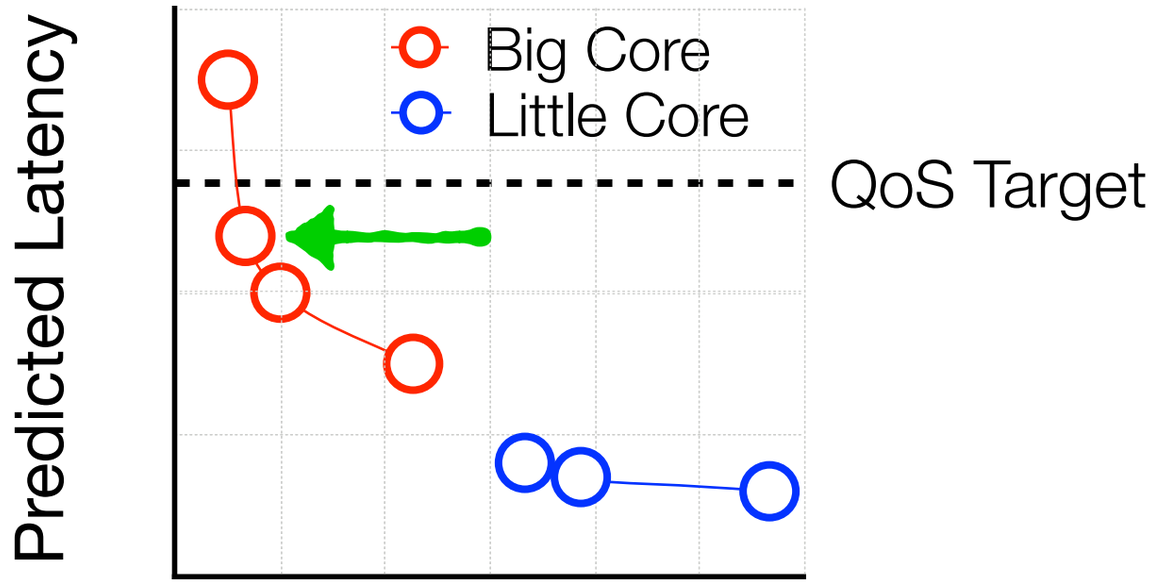




How the QoS Monitor Works

- **Goal**: Predict the <core, frequency> for the next event

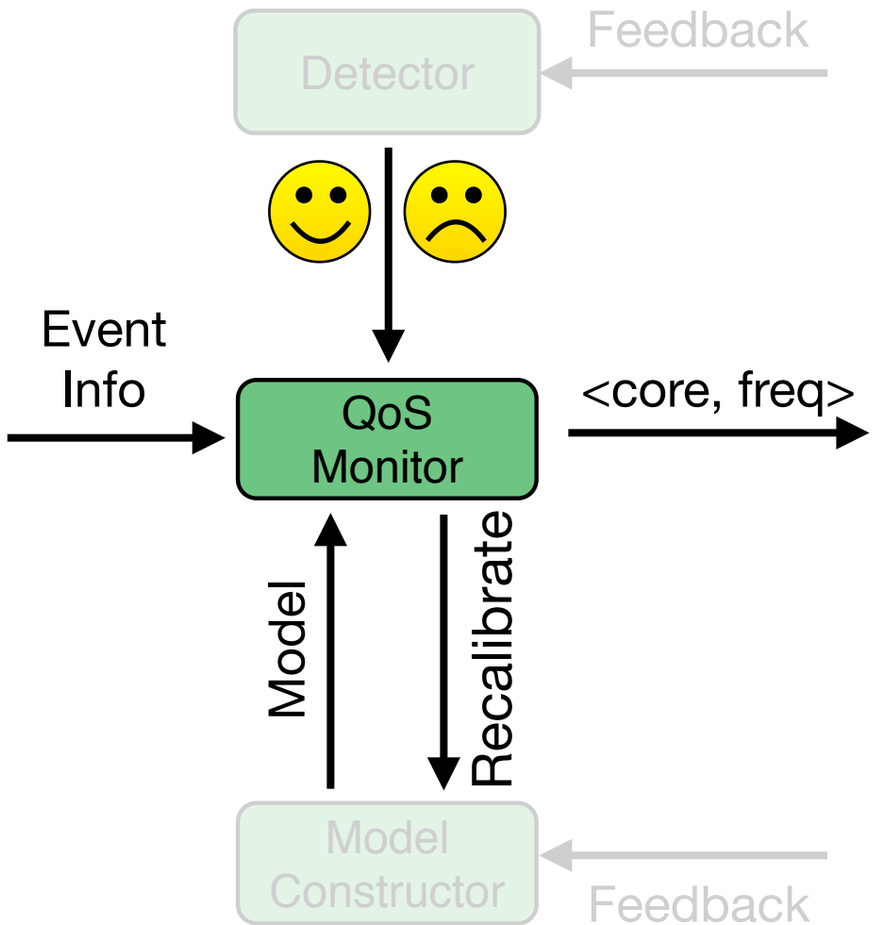


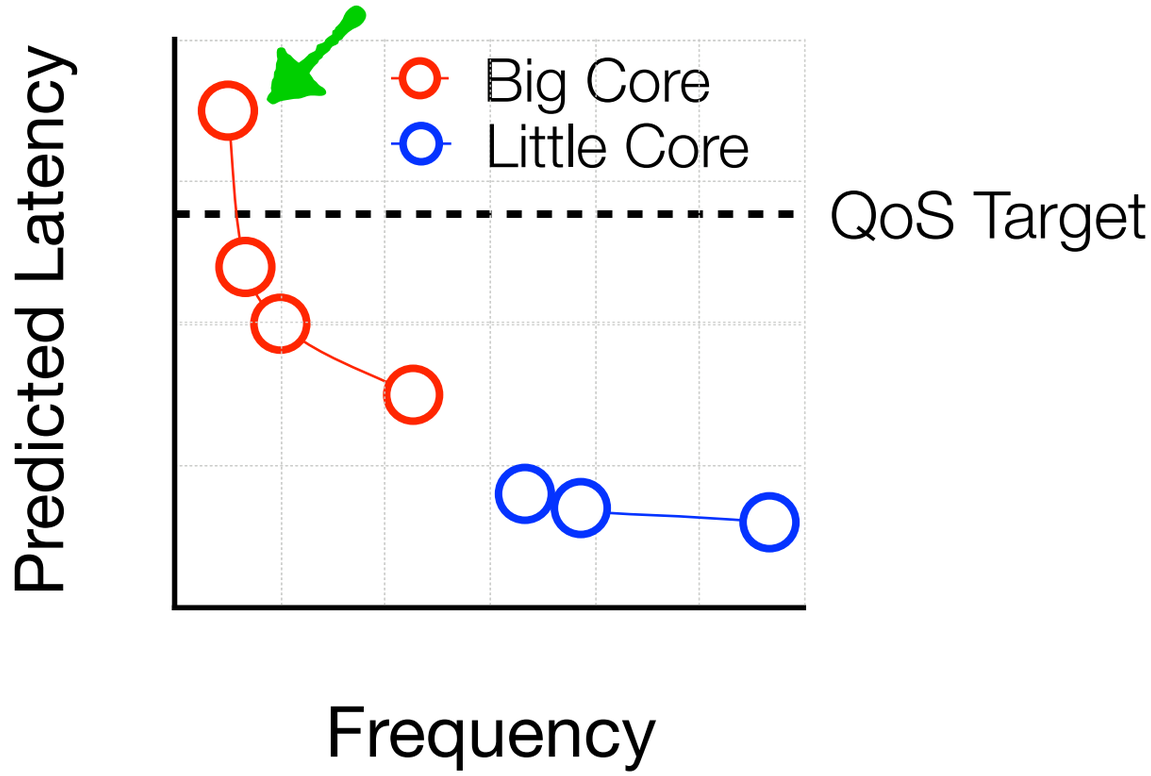


- Scheduling overhead (~120 us):
  - Core migration
  - Frequency switching

How the QoS Monitor Works

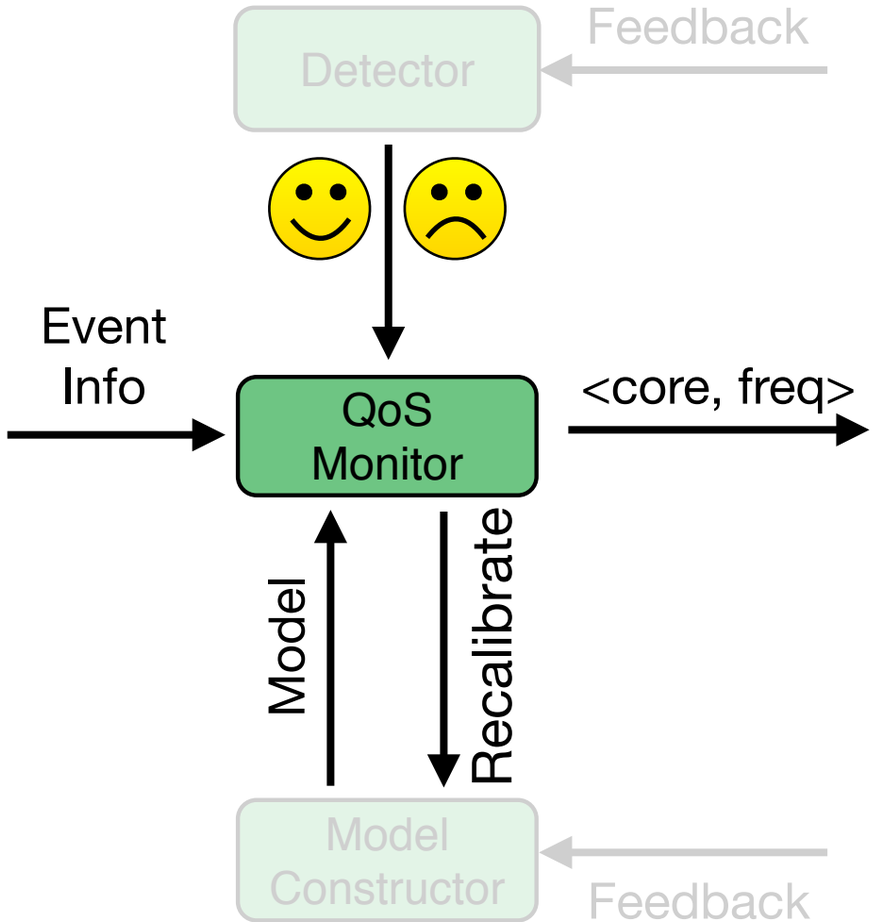
- **Goal**: Predict the <core, frequency> for the next event

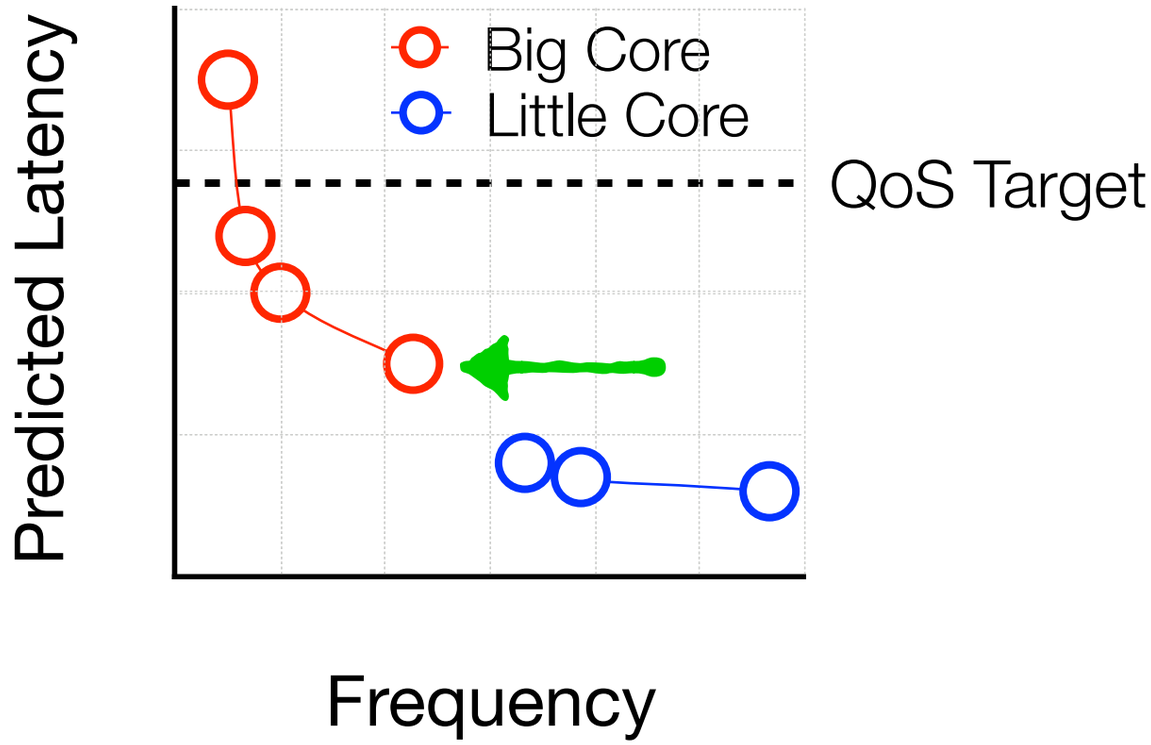




How the QoS Monitor Works

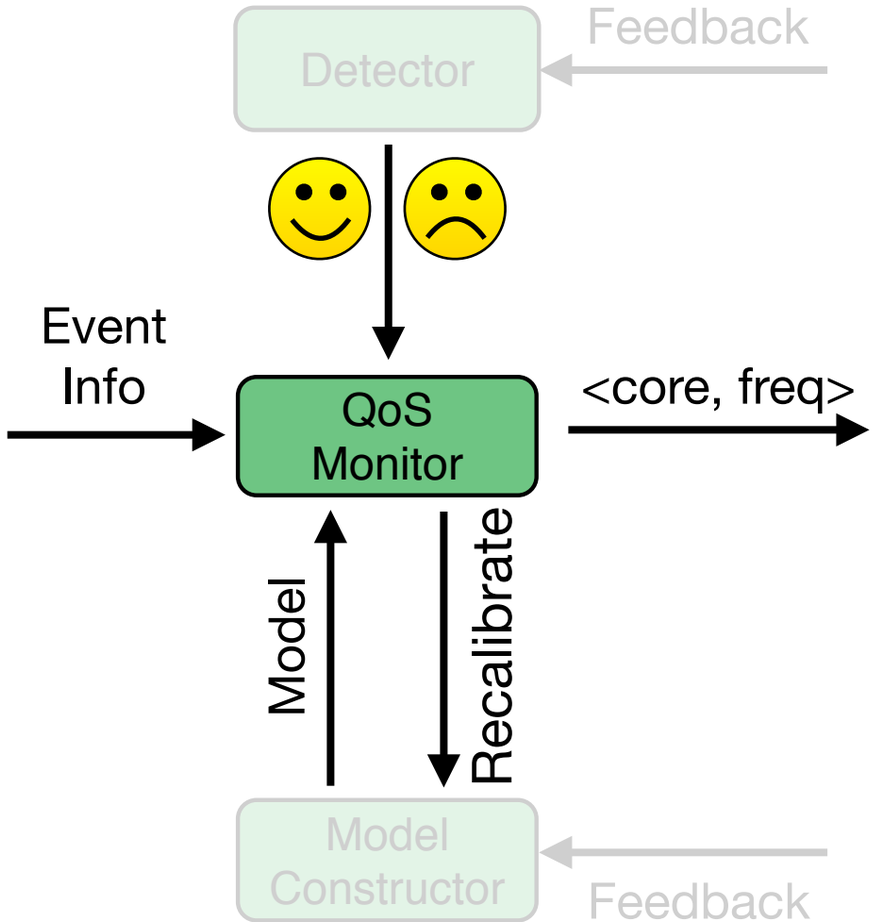
- **Goal**: Predict the <core, frequency> for the next event

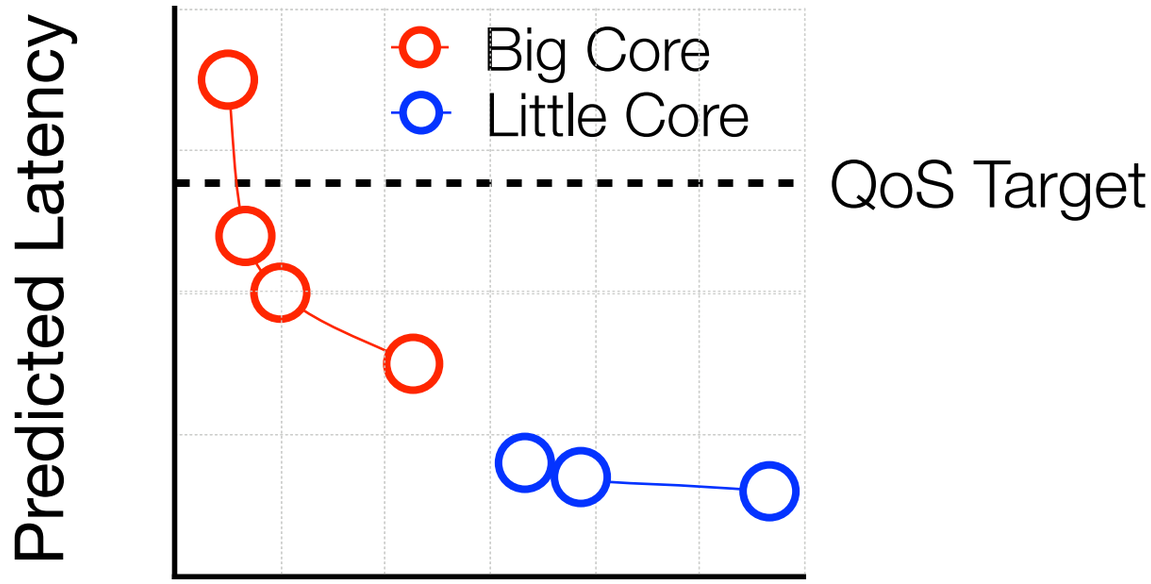




How the QoS Monitor Works

- **Goal**: Predict the <core, frequency> for the next event

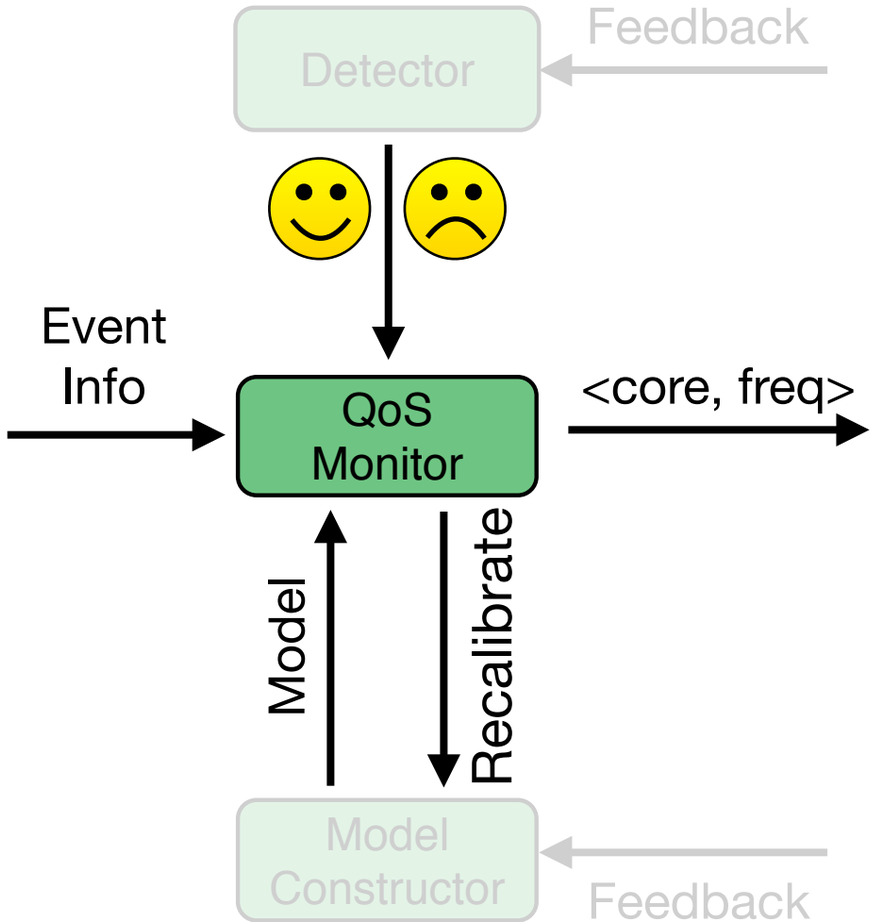


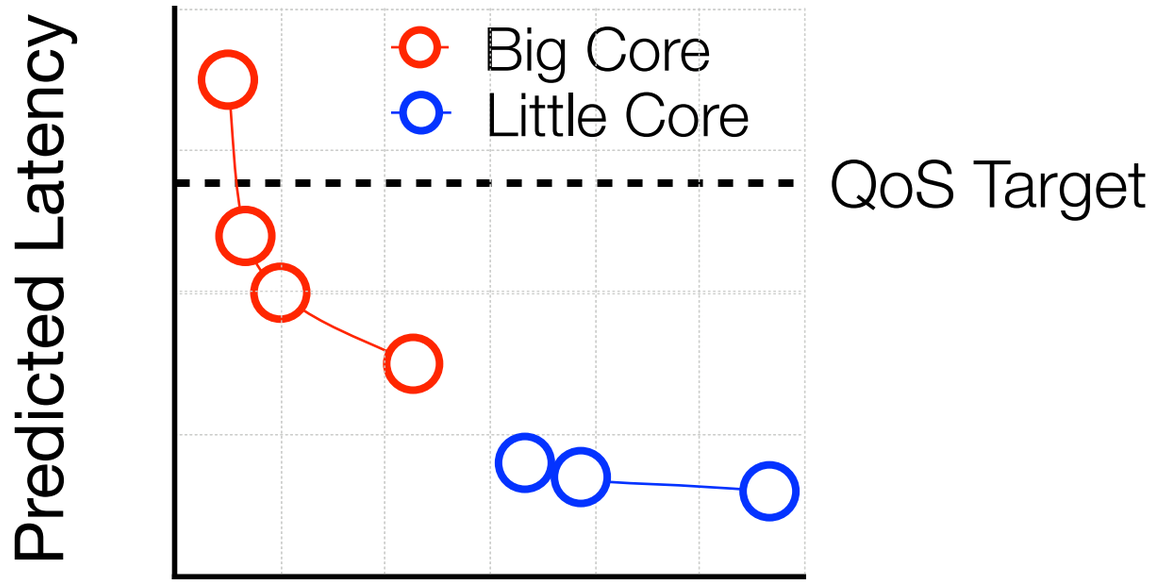


- Fine-tune the execution upon over-prediction or under-prediction

How the QoS Monitor Works

- **Goal**: Predict the <core, frequency> for the next event





- Fine-tune the execution upon over-prediction or under-prediction
- Recalibrate if it mispredicts too often

Evaluation Methodology

- Implemented inside Google Chromium Web browser

- Representative hardware platform
  - Exynos 5410 SoC (A15 + A7)

Evaluation Methodology

- Baseline Mechanisms
  - Highest performance (**Perf**) — Standard to guarantee responsiveness
  - Minimal energy (**Energy**) — Minimize energy consumption
  - Interactive governor (**Interactive**) — Android default
  - On-demand governor (**Ondemand**)

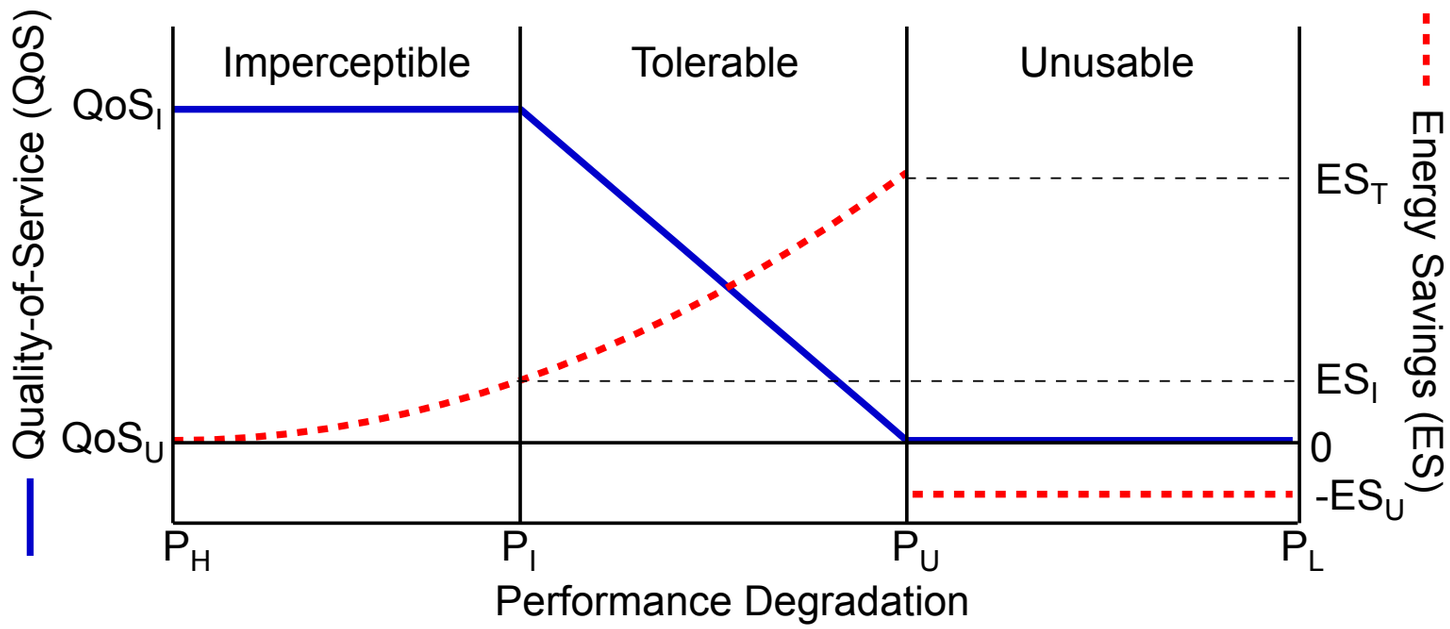
Evaluation Methodology

- Baseline Mechanisms
  - Highest performance (**Perf**) — Standard to guarantee responsiveness
  - Minimal energy (**Energy**) — Minimize energy consumption
  - Interactive governor (**Interactive**) — Android default
  - On-demand governor (**Ondemand**)

Evaluation Methodology

- Baseline Mechanisms
  - Highest performance (**Perf**) — Standard to guarantee responsiveness
  - Minimal energy (**Energy**) — Minimize energy consumption
  - Interactive governor (**Interactive**) — Android default
  - On-demand governor (**Ondemand**)

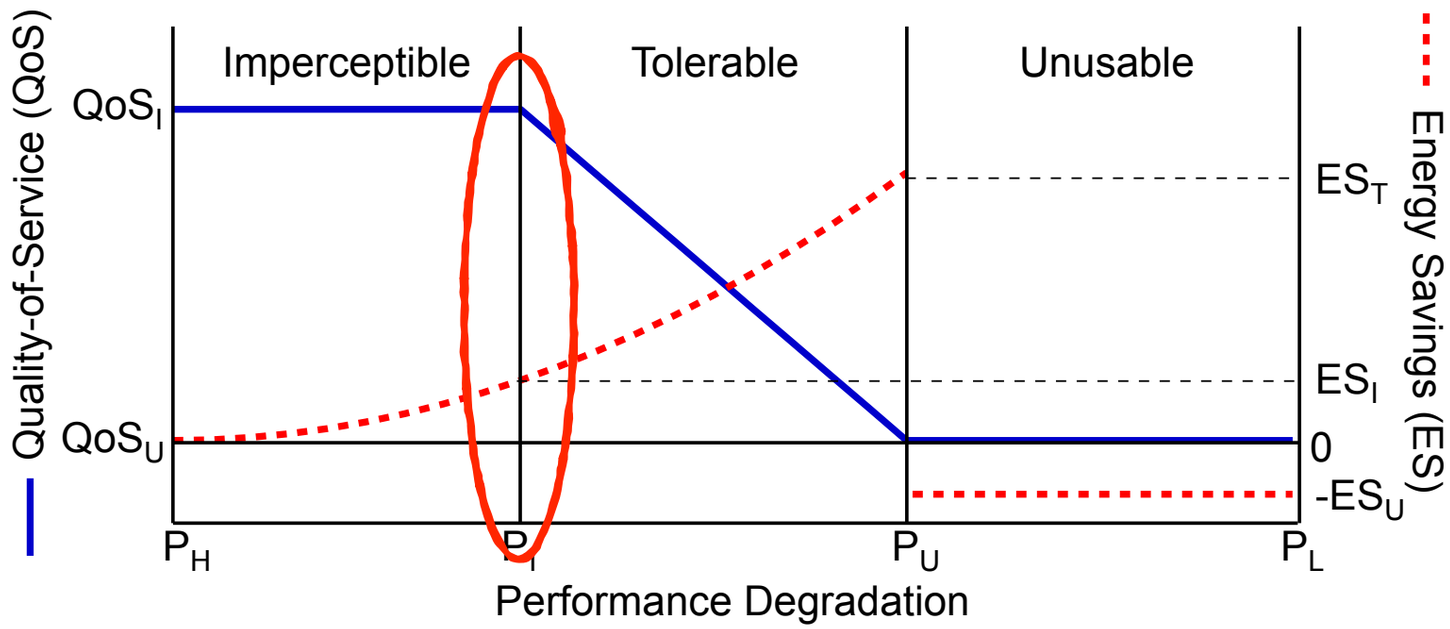
- Scheduling Scenarios



Evaluation Methodology

- Baseline Mechanisms
  - Highest performance (**Perf**) — Standard to guarantee responsiveness
  - Minimal energy (**Energy**) — Minimize energy consumption
  - Interactive governor (**Interactive**) — Android default
  - On-demand governor (**Ondemand**)

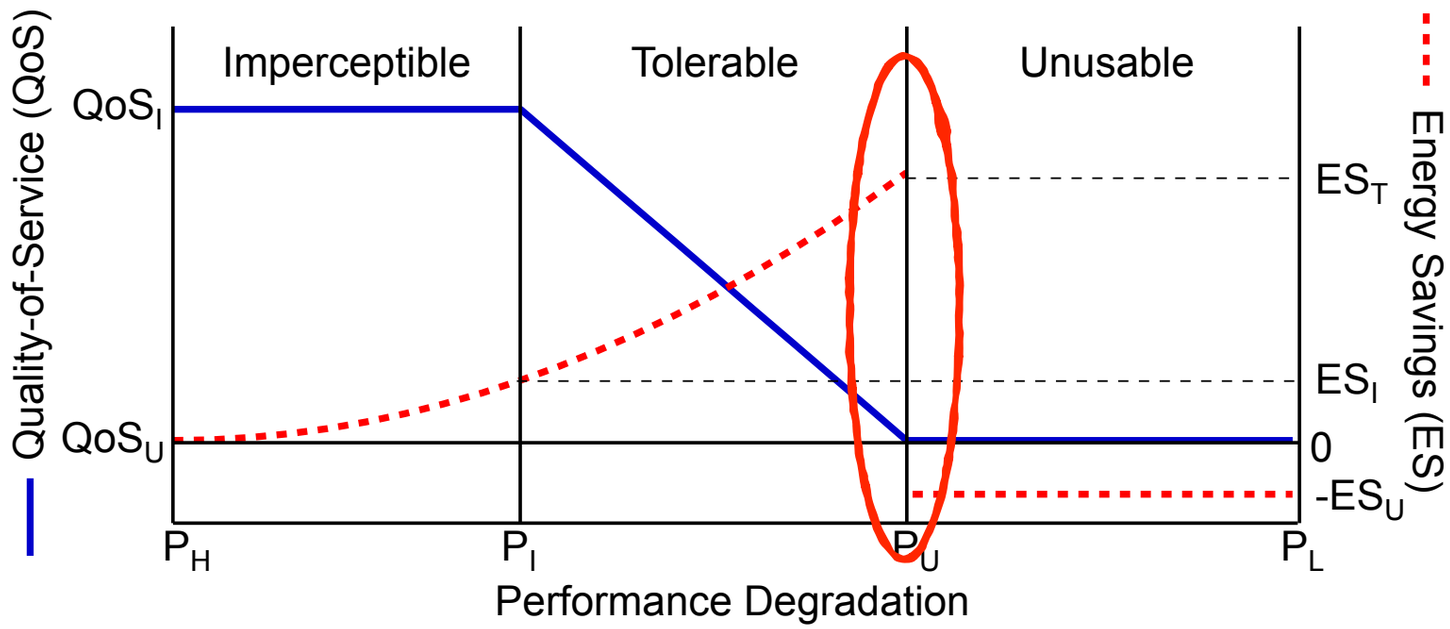
- Scheduling Scenarios
  - Scheduling for imperceptibility



Evaluation Methodology

- Baseline Mechanisms
  - Highest performance (**Perf**) — Standard to guarantee responsiveness
  - Minimal energy (**Energy**) — Minimize energy consumption
  - Interactive governor (**Interactive**) — Android default
  - On-demand governor (**Ondemand**)

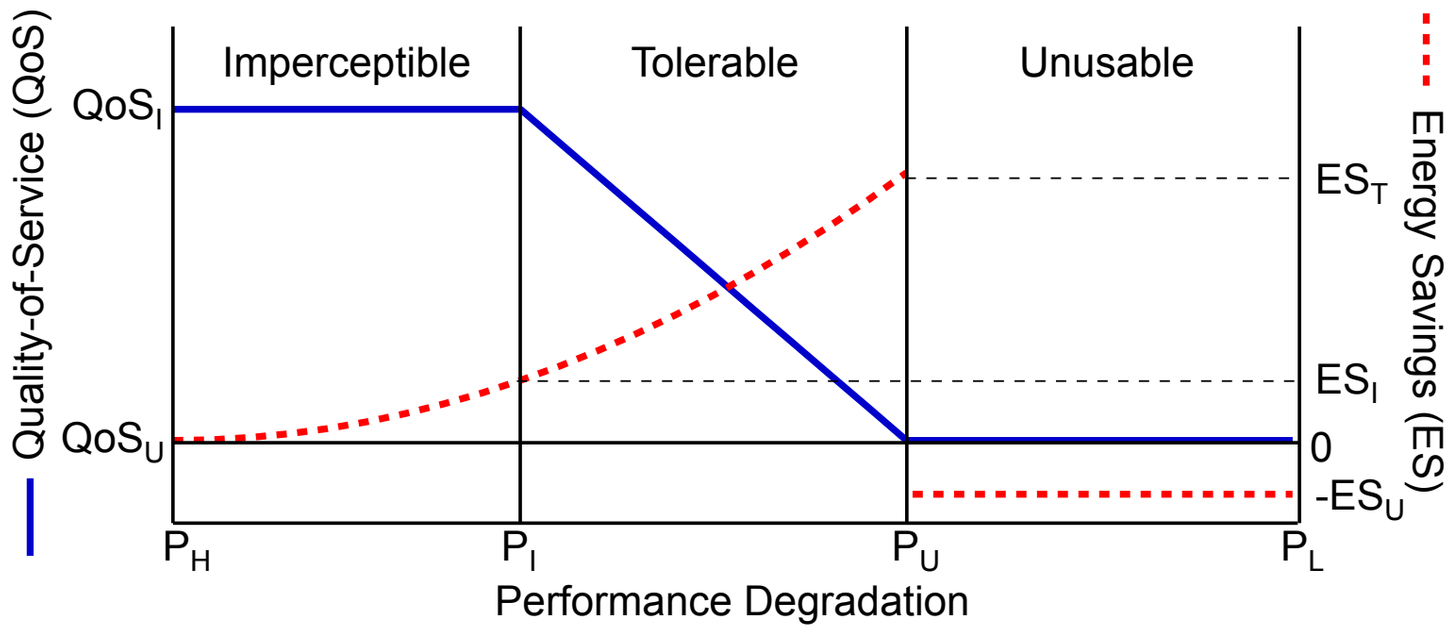
- Scheduling Scenarios
  - Scheduling for imperceptibility
  - Scheduling for tolerability



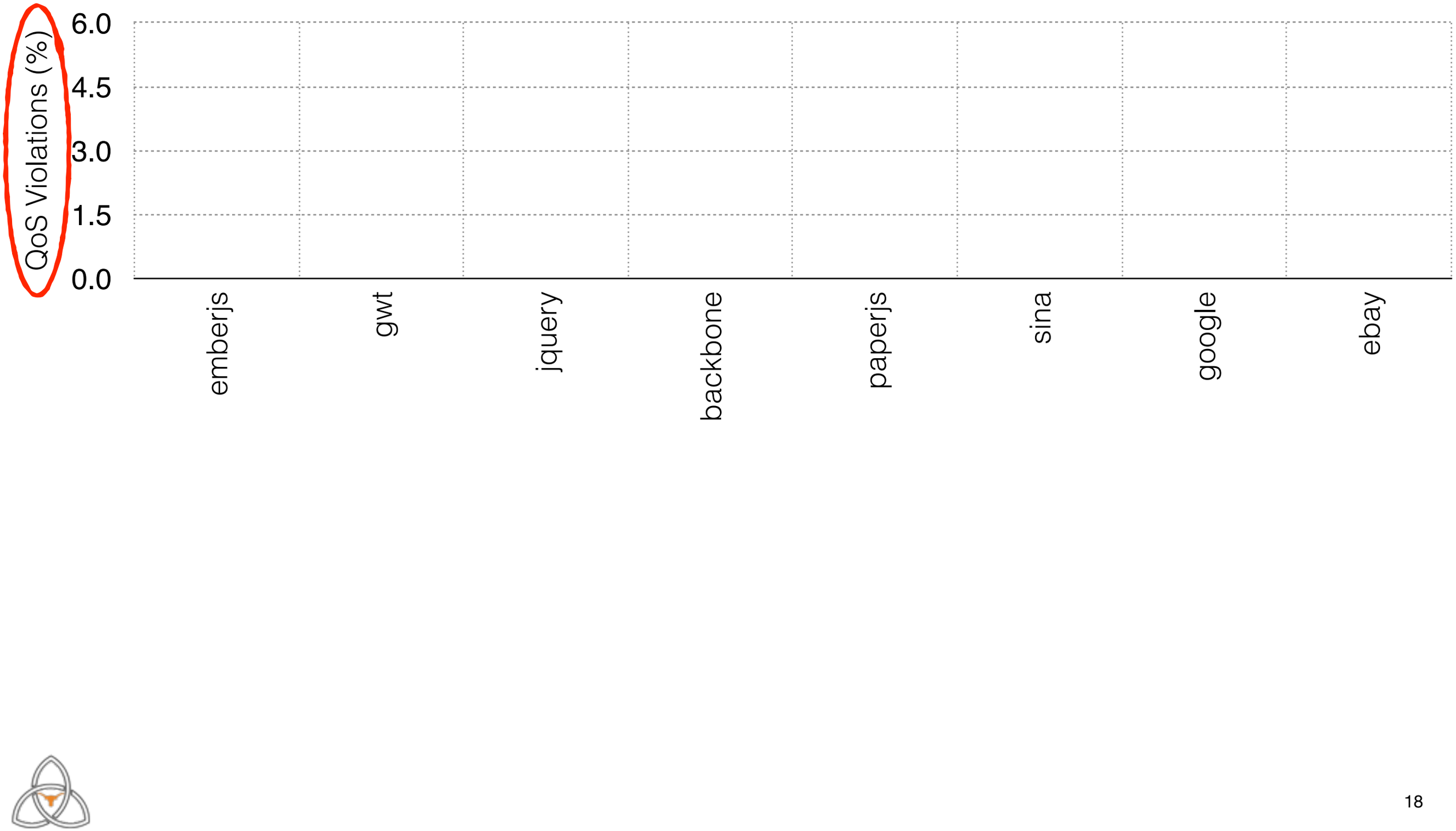
Evaluation Methodology

- Baseline Mechanisms
  - Highest performance (**Perf**) — Standard to guarantee responsiveness
  - Minimal energy (**Energy**) — Minimize energy consumption
  - Interactive governor (**Interactive**) — Android default
  - On-demand governor (**Ondemand**)

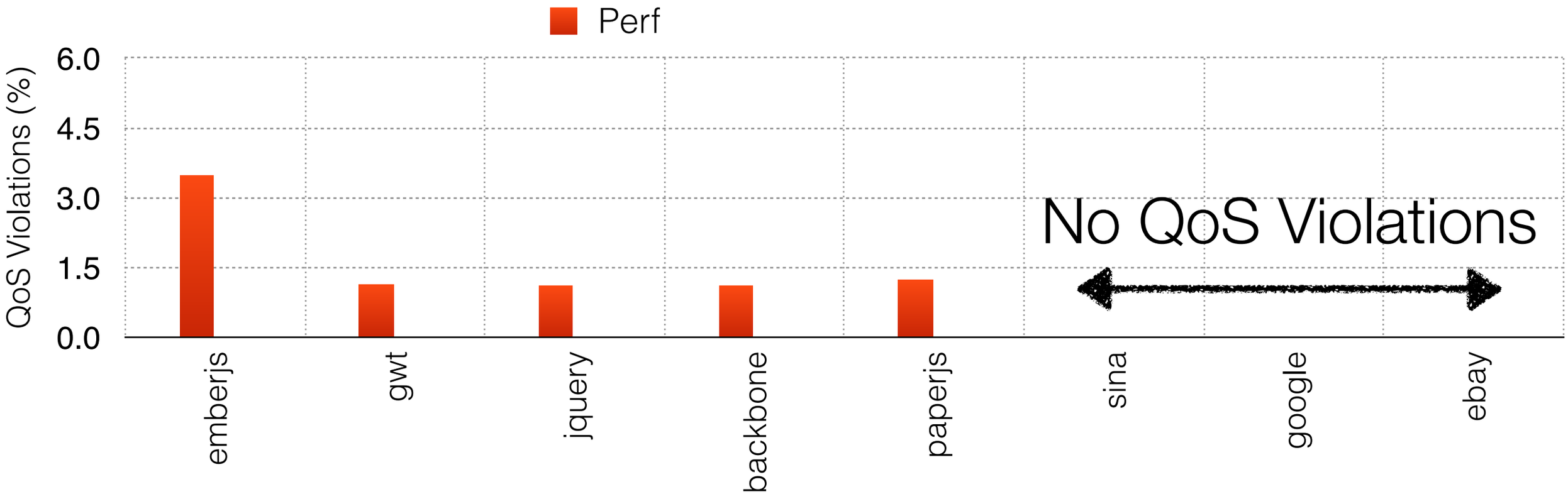
- Scheduling Scenarios
  - Scheduling for imperceptibility
  - Scheduling for tolerability



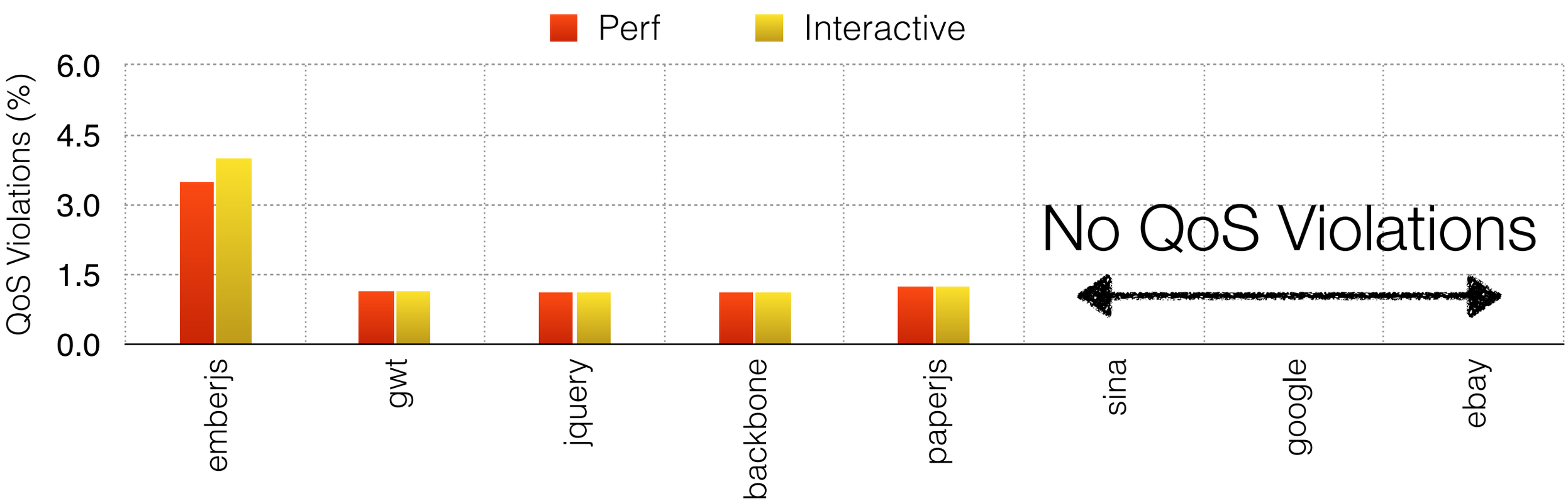
Evaluation Results



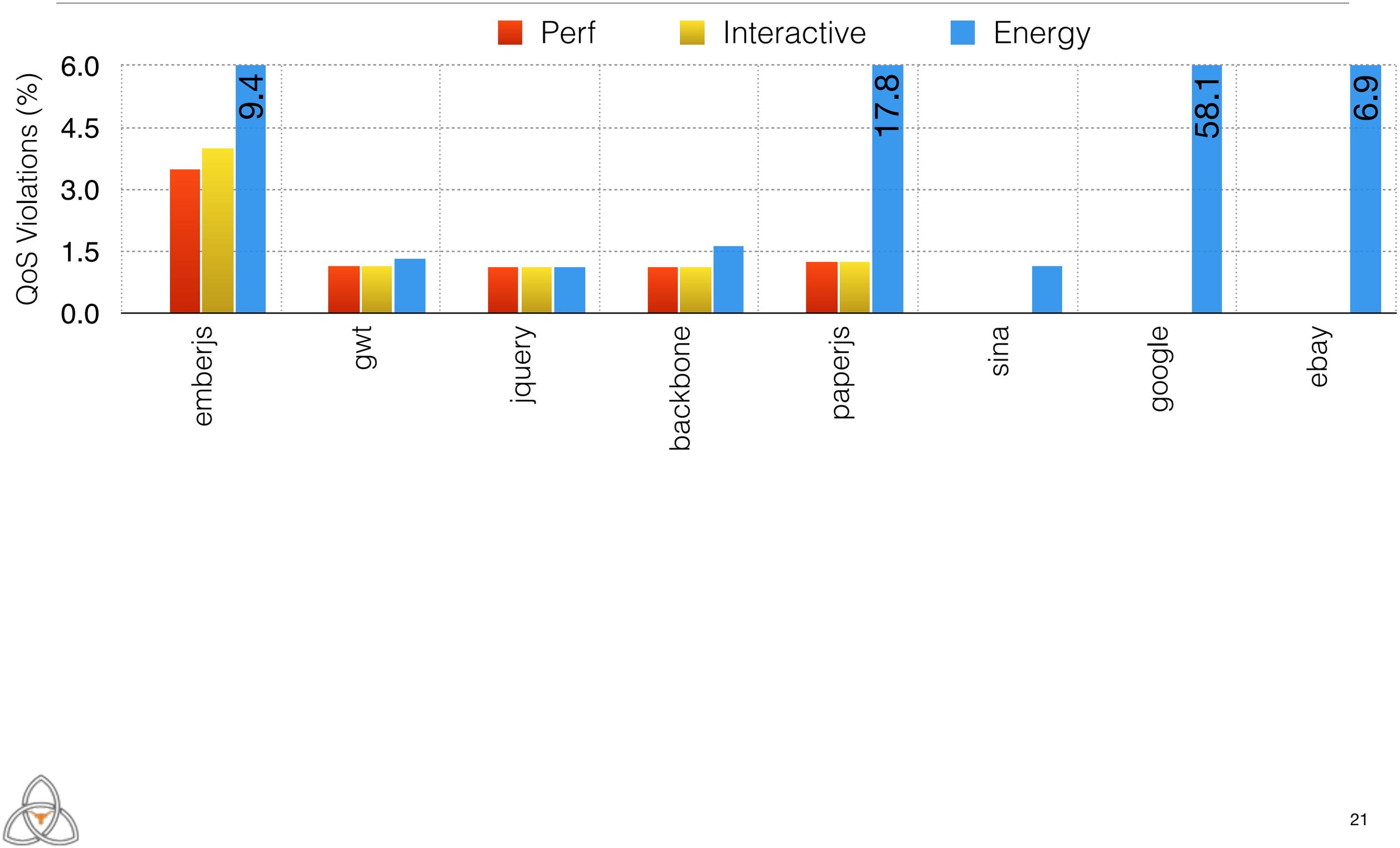
Evaluation Results



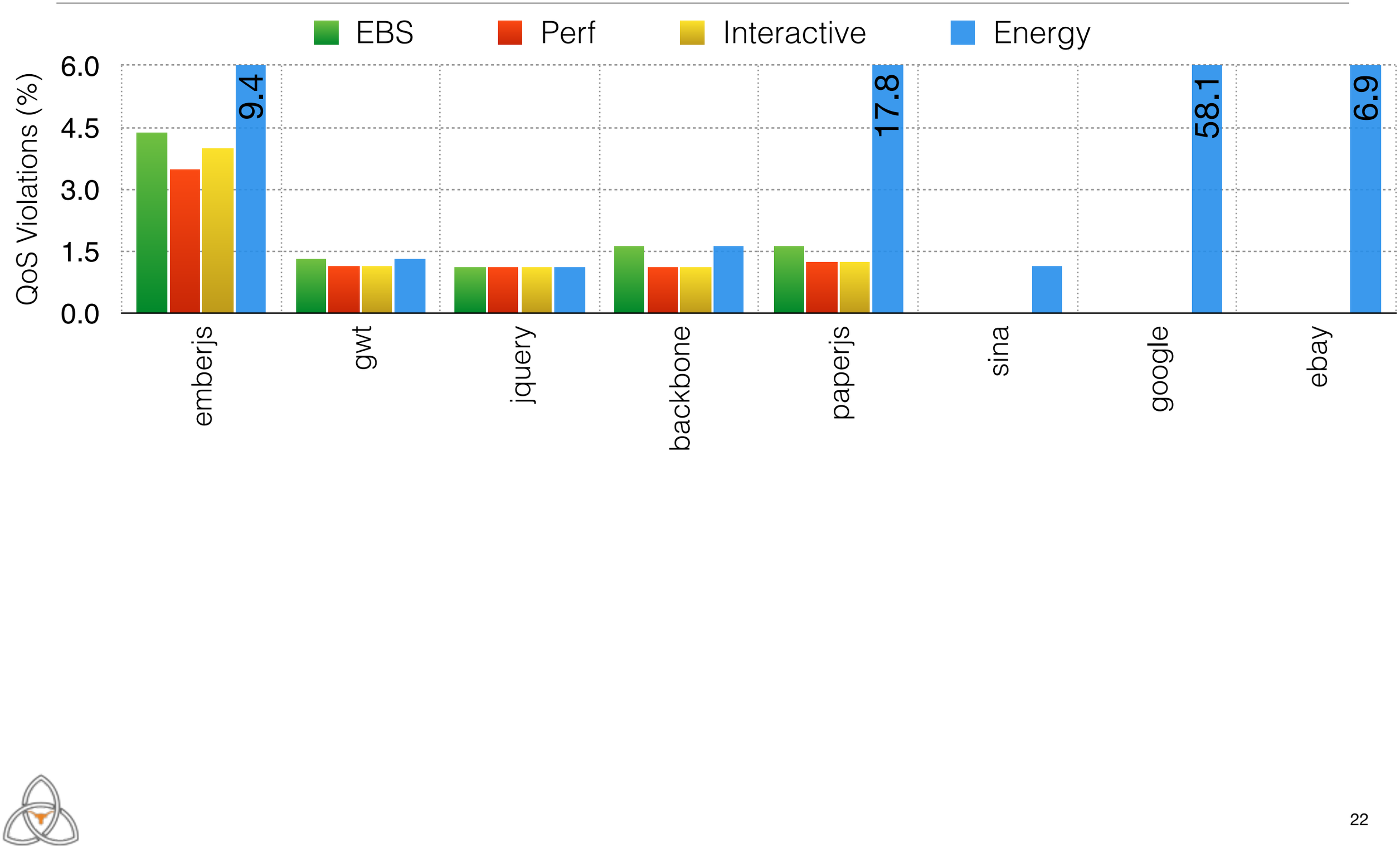
Evaluation Results



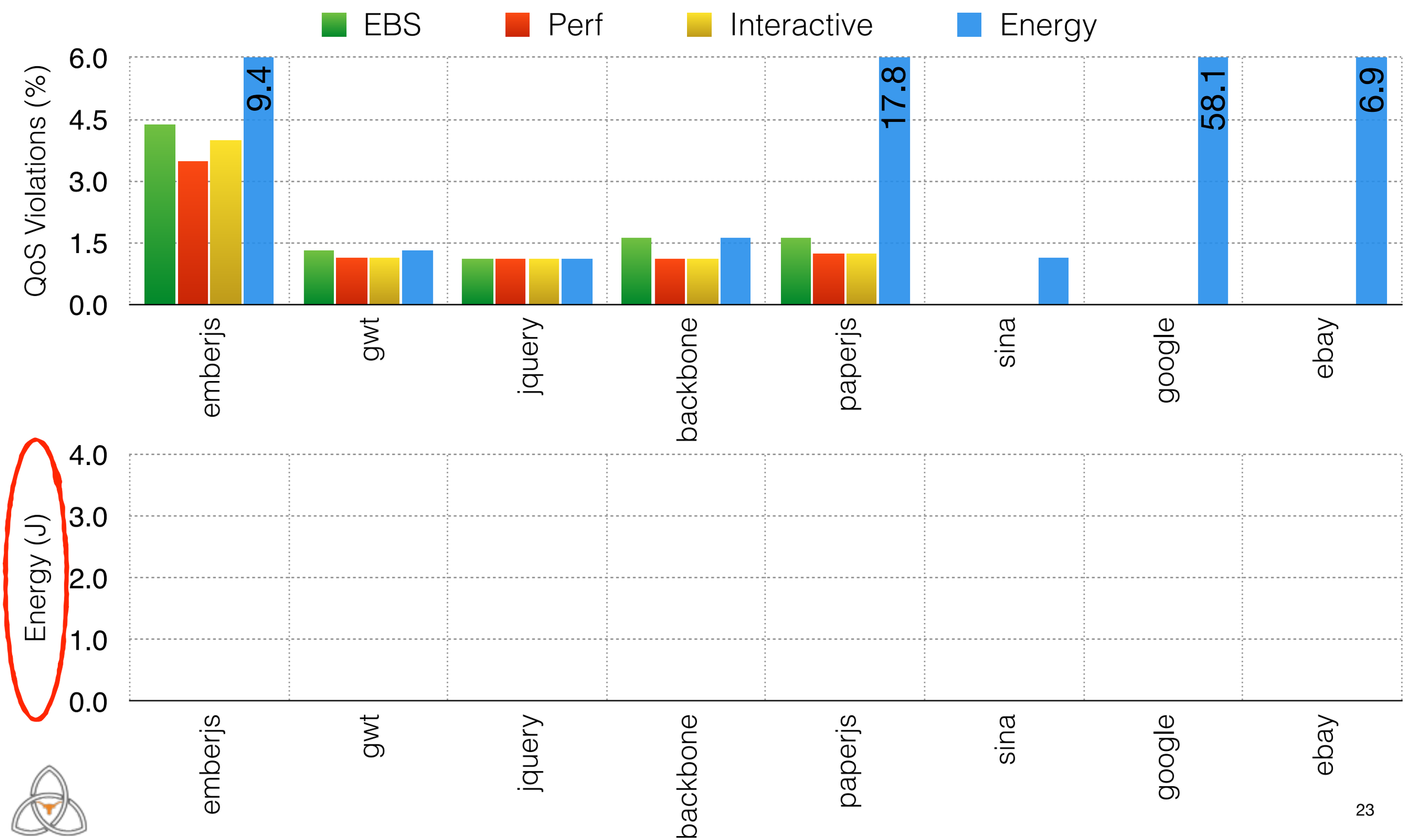
Evaluation Results



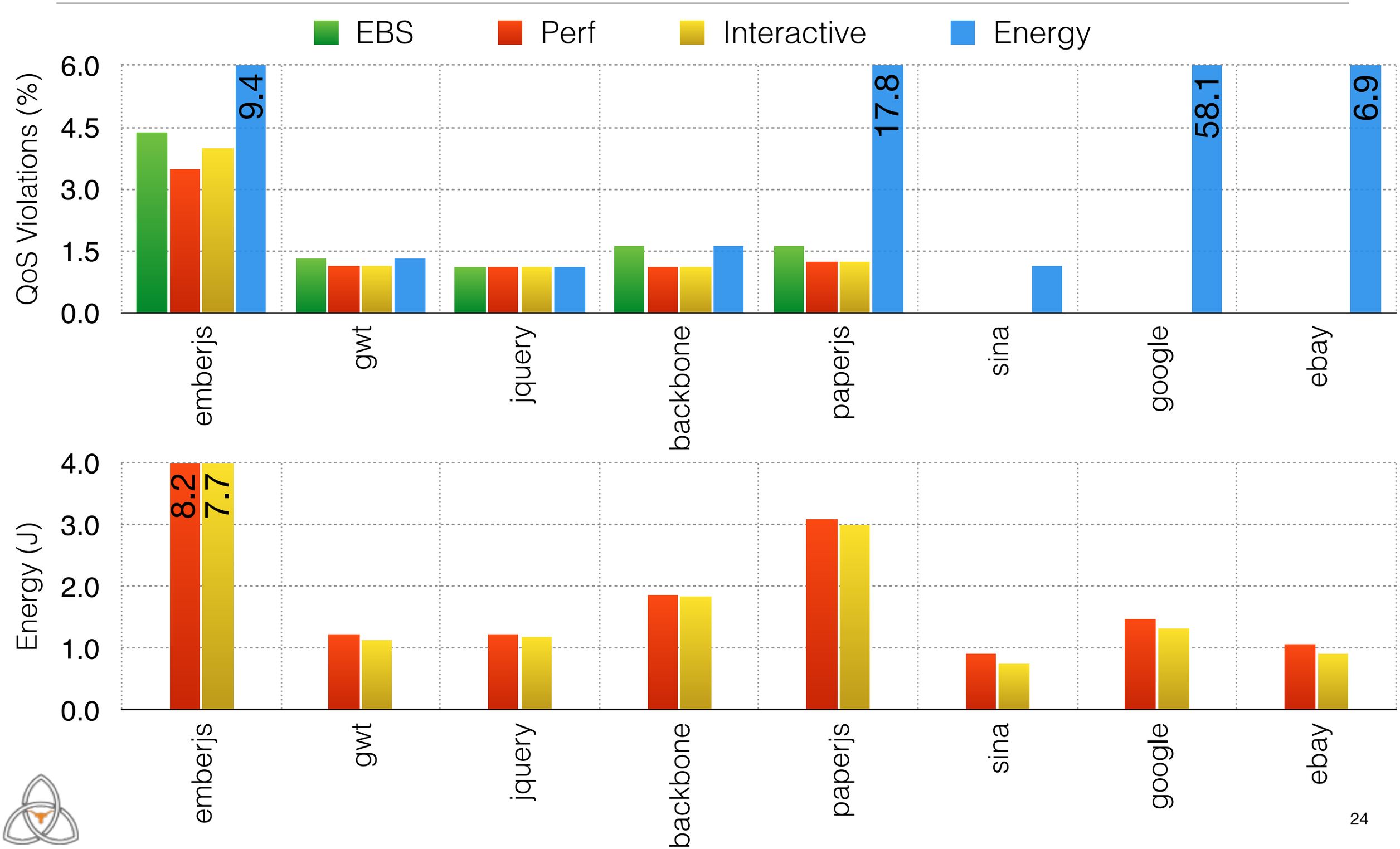
Evaluation Results



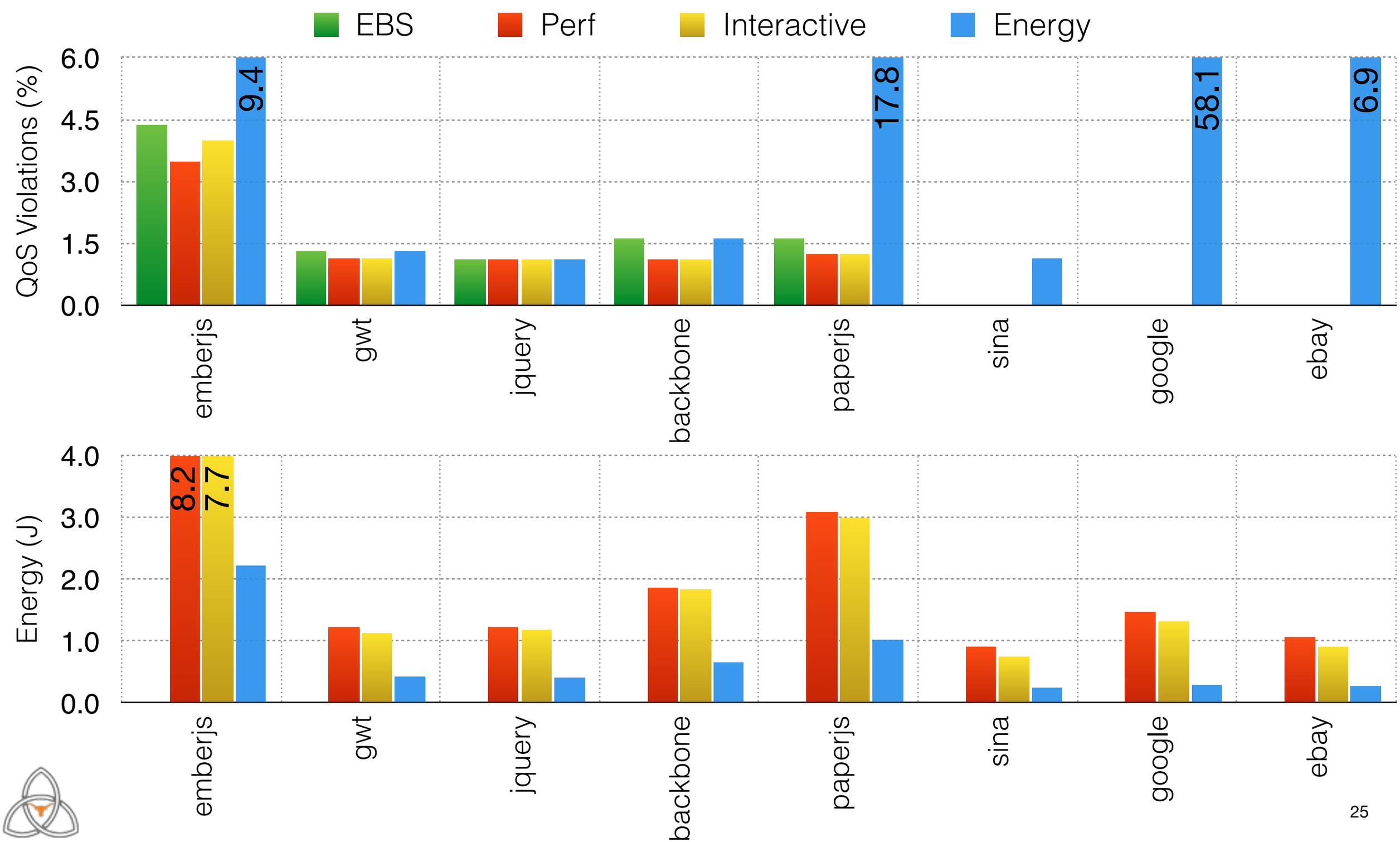
Evaluation Results



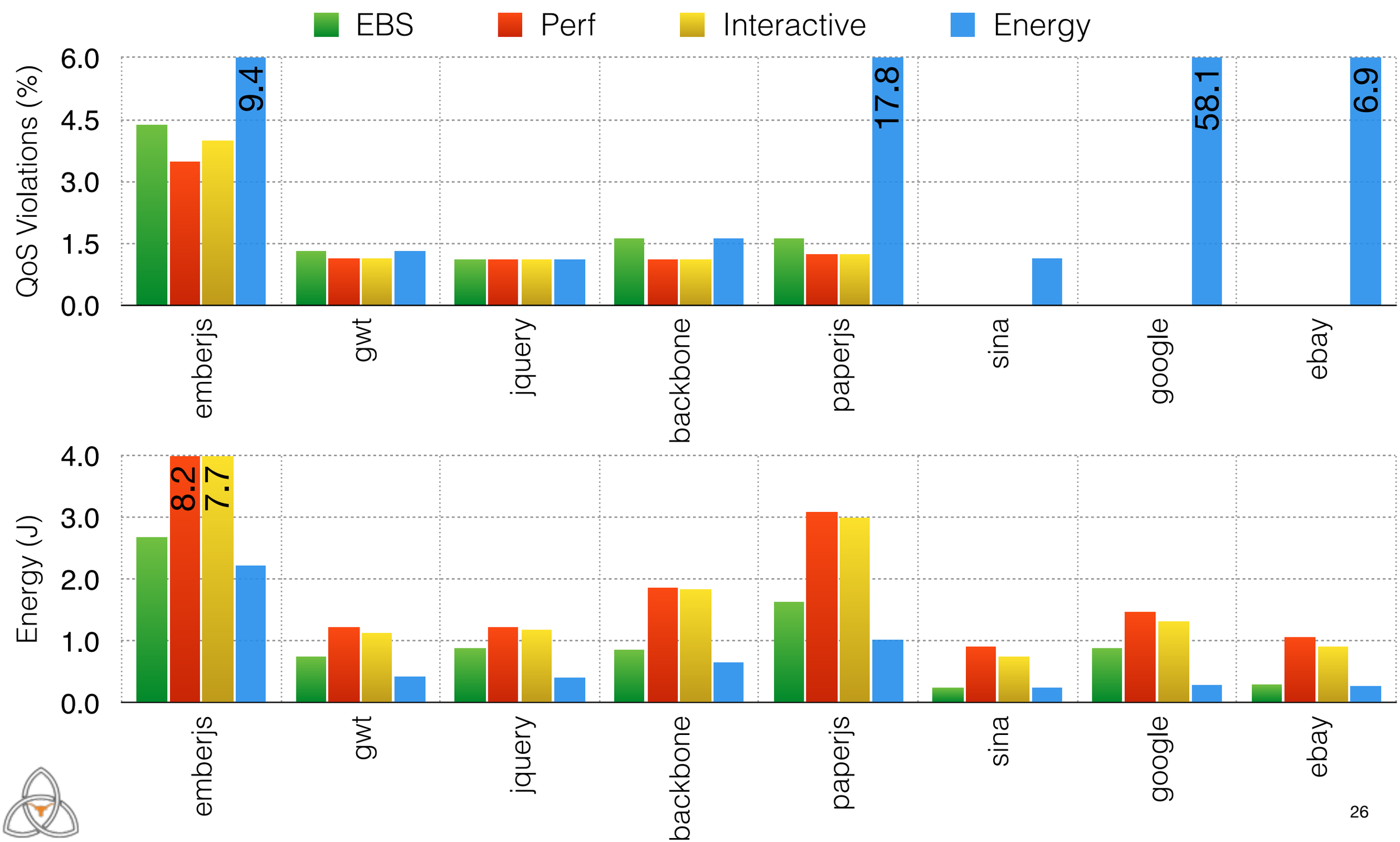
Evaluation Results



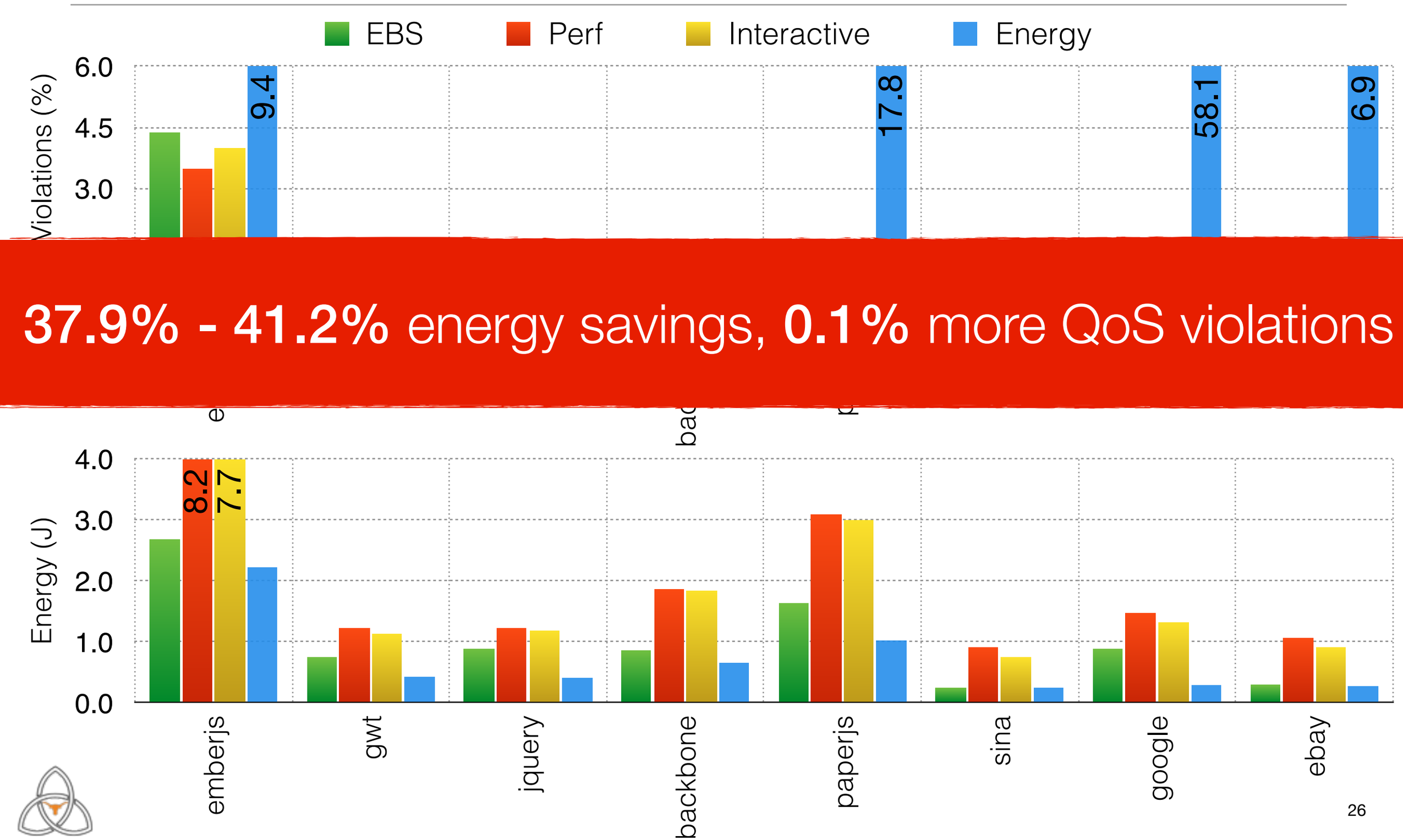
Evaluation Results



Evaluation Results



Evaluation Results

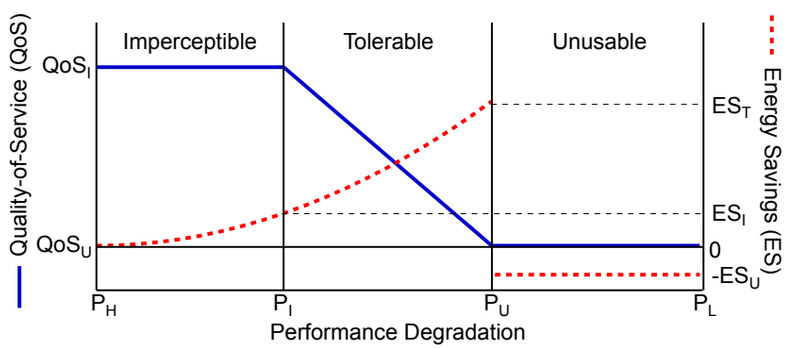


37.9% - 41.2% energy savings, **0.1%** more QoS violations

What Have We Learnt So Far

- User **interactivity** and **human perceptibility** are important because it has economic ramifications and impacts user device choice, as well as end-user satisfaction

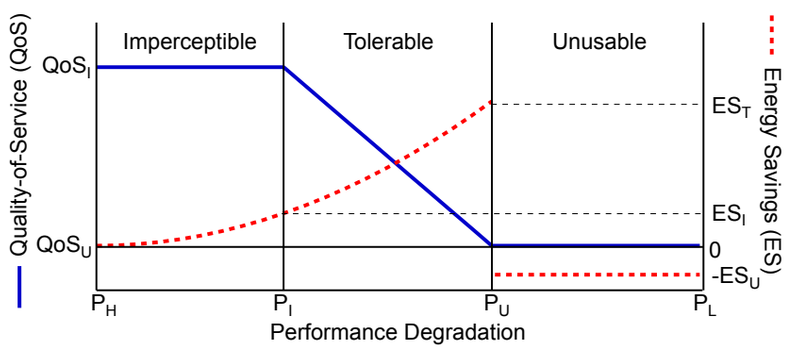
What Have We Learnt So Far



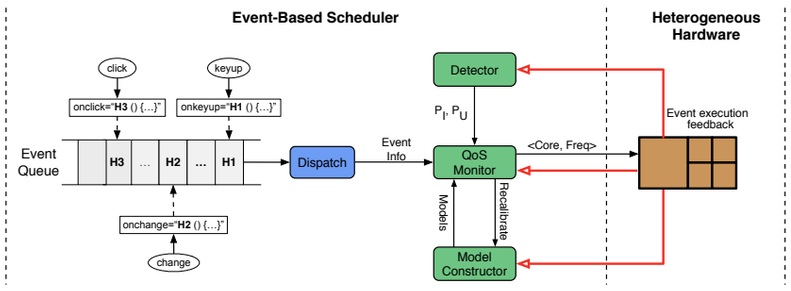
- User **interactivity** and **human perceptibility** are important because it has economic ramifications and impacts user device choice, as well as end-user satisfaction
- Need a systematic way of understanding and QoS-energy trade-offs. To that end we advocate **eQoS** to strike a balance between user satisfaction and energy reduction

What Have We Learnt So Far

- User **interactivity** and **human perceptibility** are important because it has economic ramifications and impacts user device choice, as well as end-user satisfaction



- Need a systematic way of understanding and QoS-energy trade-offs. To that end we advocate **eQoS** to strike a balance between user satisfaction and energy reduction



- Develop the event-based scheduling (**EBS**) mechanism that exploits the fundamental event-driven execution pattern of mobile applications to achieve better eQoS

Mobile is One Example

Web

Mobile

Mobile is One Example

Web

Mobile

Sensor networks

Cloud

Internet-of-Things

Mobile is One Example

Web

Mobile

Sensor networks

Cloud

Internet-of-Things

Event-based processing is a fundamental computation pattern.

Mobile is One Example

Web | Mobile | Sensor networks | Cloud | Internet-of-Things

Event-based processing is a fundamental computation pattern.

- Another talk @ **Best of CAL session**
- Wednesday after the keynote
- How computer architects can improve Web technologies

Thank you