

Energy-Efficient Mobile Web Computing

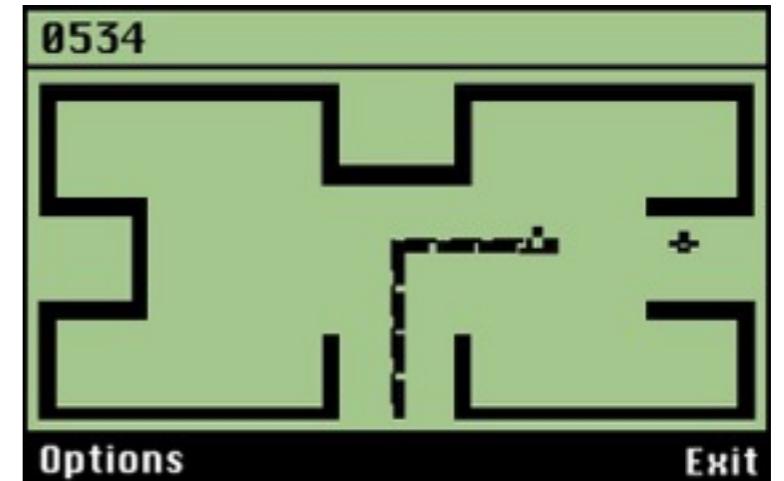
Yuhao Zhu
UT Austin
Advisor: Vijay Janapa Reddi



Call
Text



Call
Text



The (in)famous
“snake game”

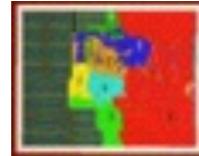


Architects Make Mobile Processors Faster

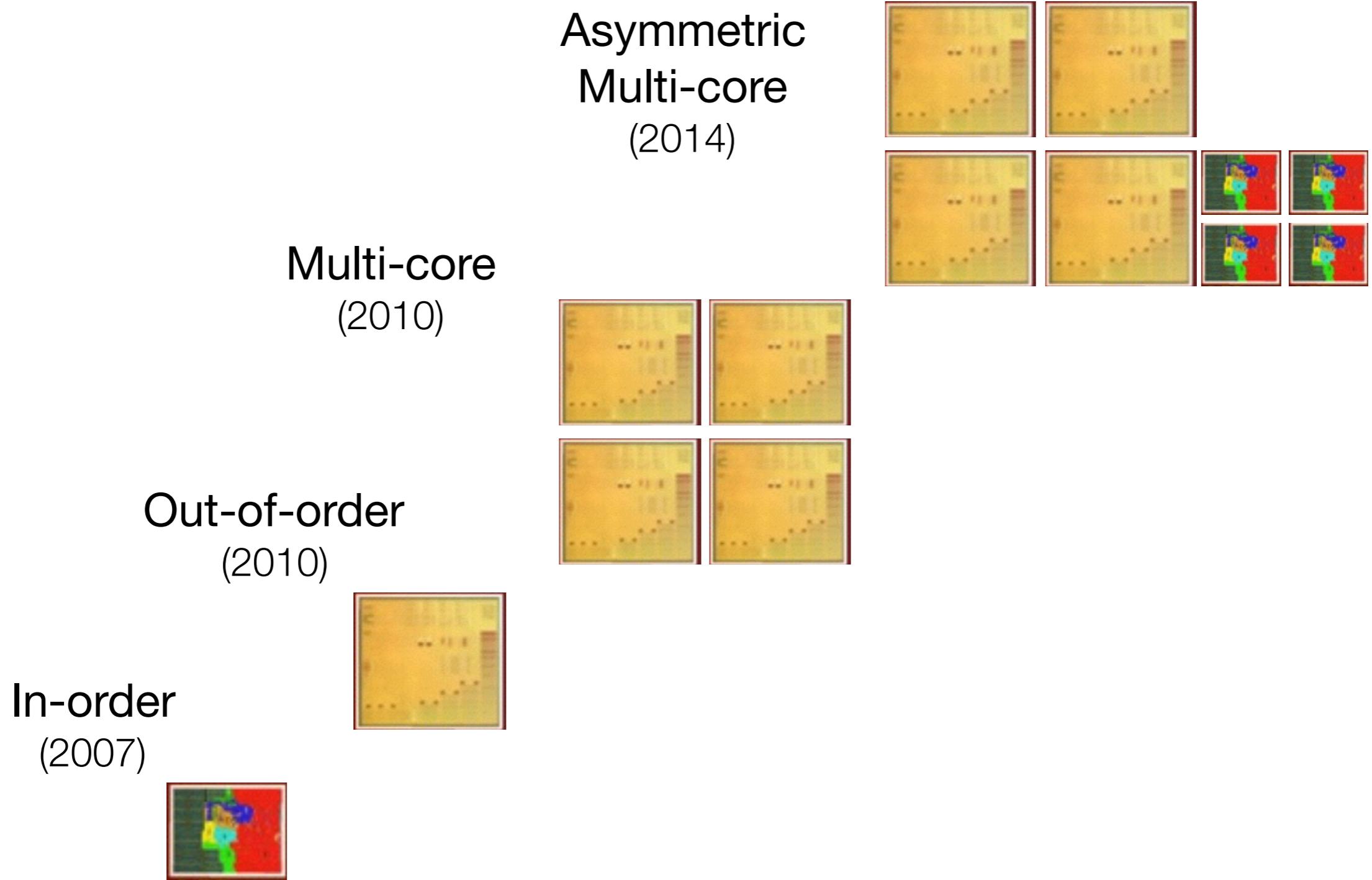


Architects Make Mobile Processors Faster

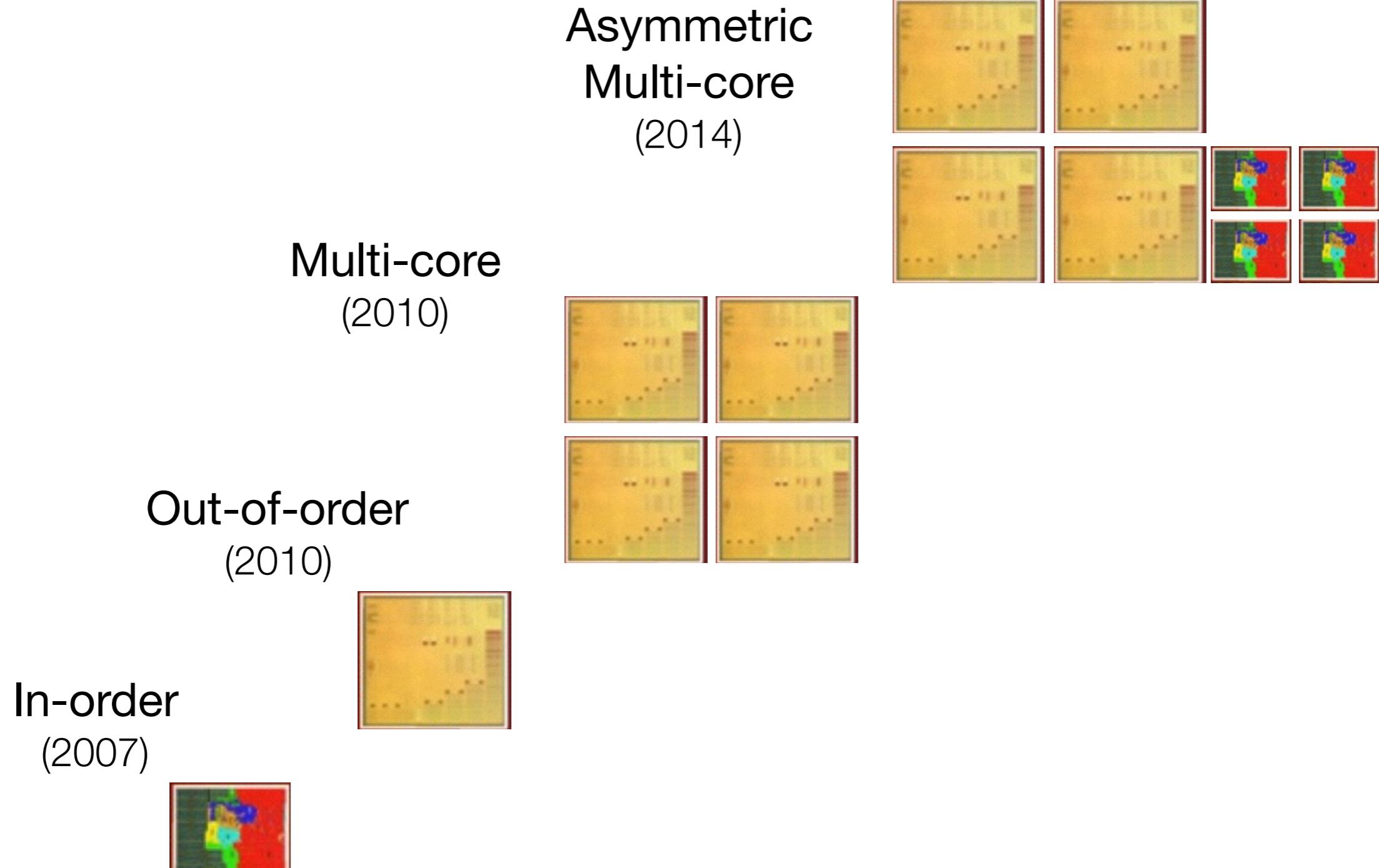
In-order
(2007)



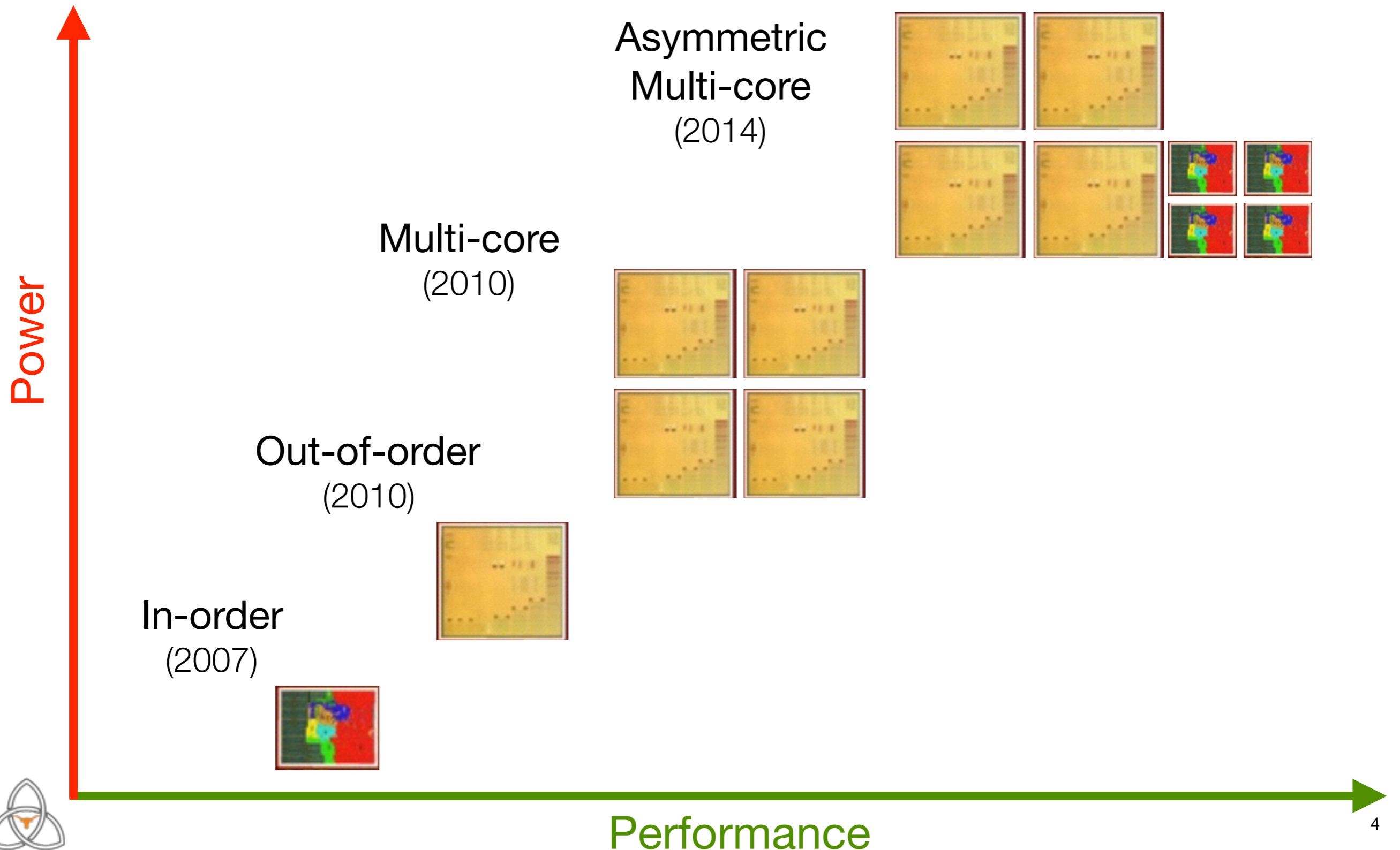
Architects Make Mobile Processors Faster



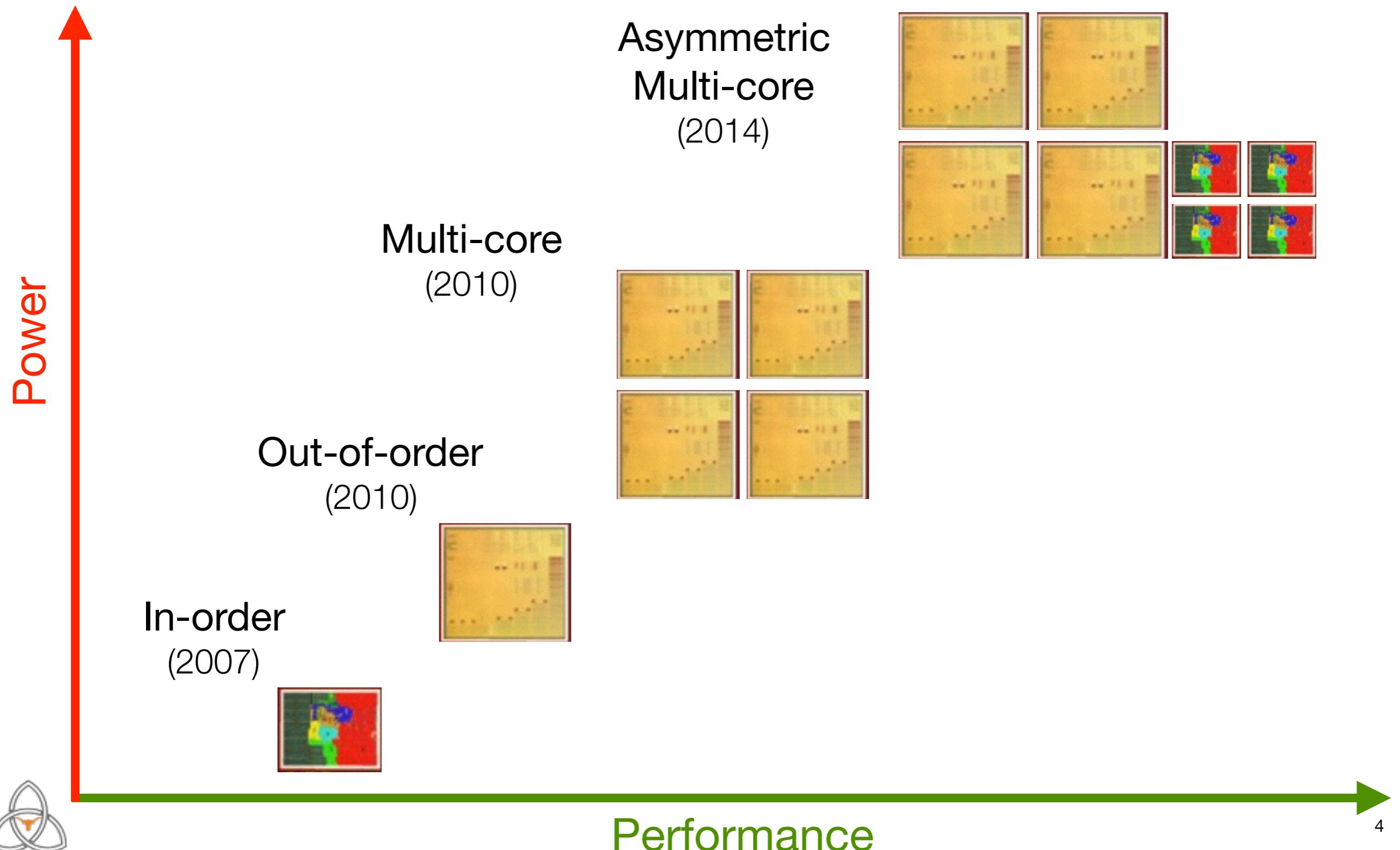
Architects Make Mobile Processors Faster



Architects Make Mobile Processors Faster



Architects Make Mobile Processors Faster At the Expense of Excessive Power



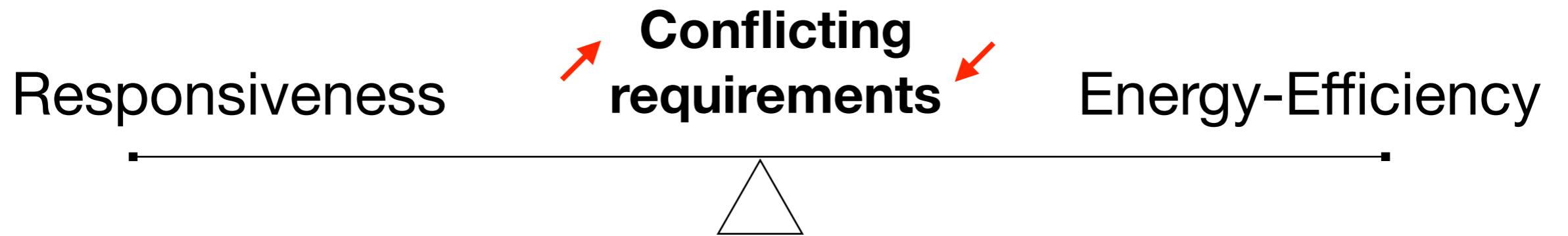
Responsiveness



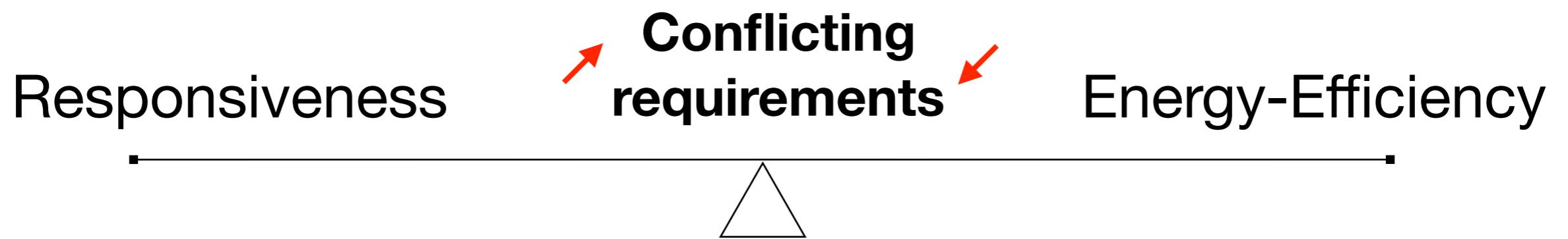
Responsiveness

Energy-Efficiency





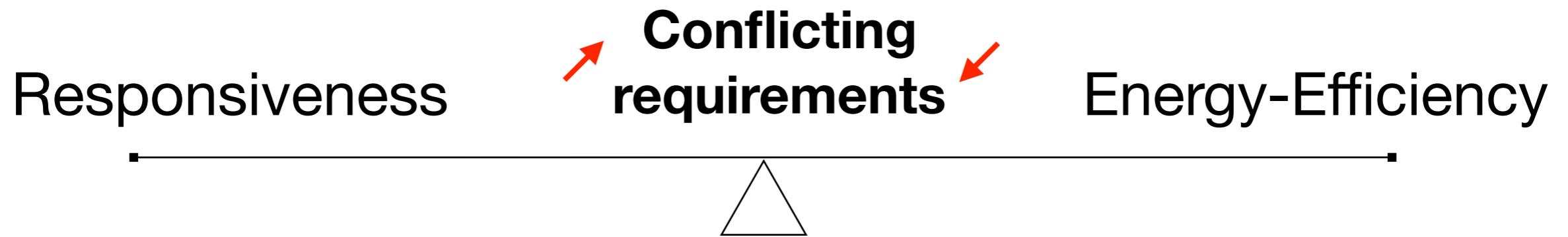
Thesis Statement



A mobile computing system that
satisfies user QoS requirements
on a mobile energy budget



Thesis Statement



A mobile computing system that
satisfies user QoS requirements
on a mobile energy budget
for the mobile Web





HTML



CSS



JS



HTML



CSS



JS



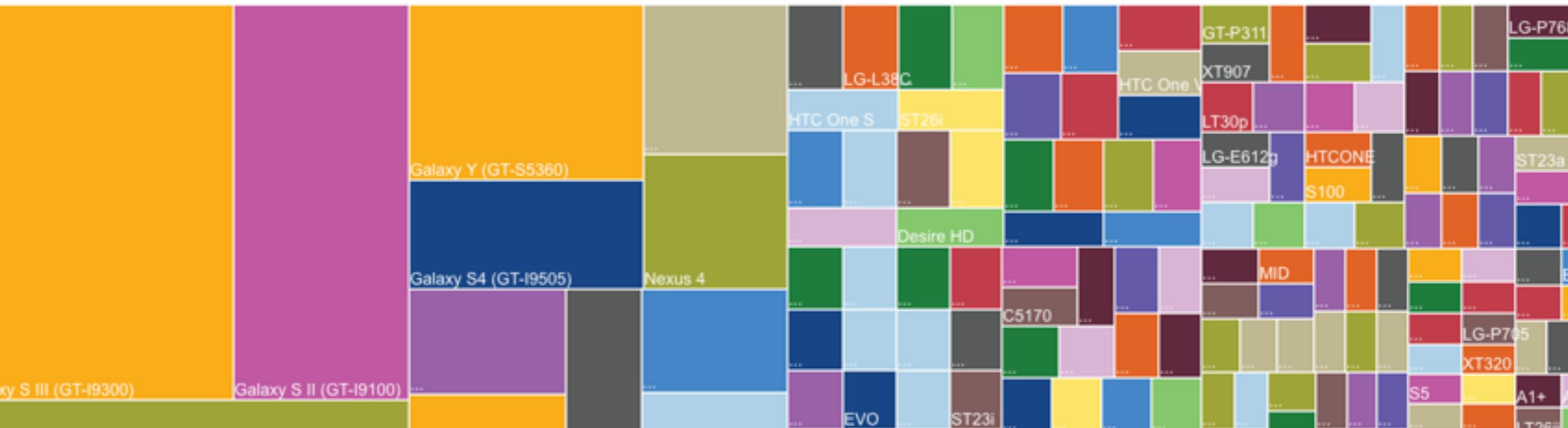
HTML



CSS

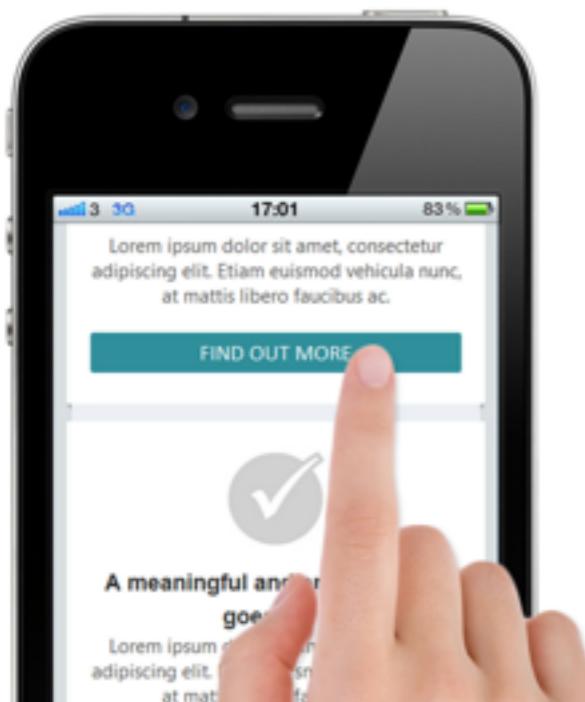


JS



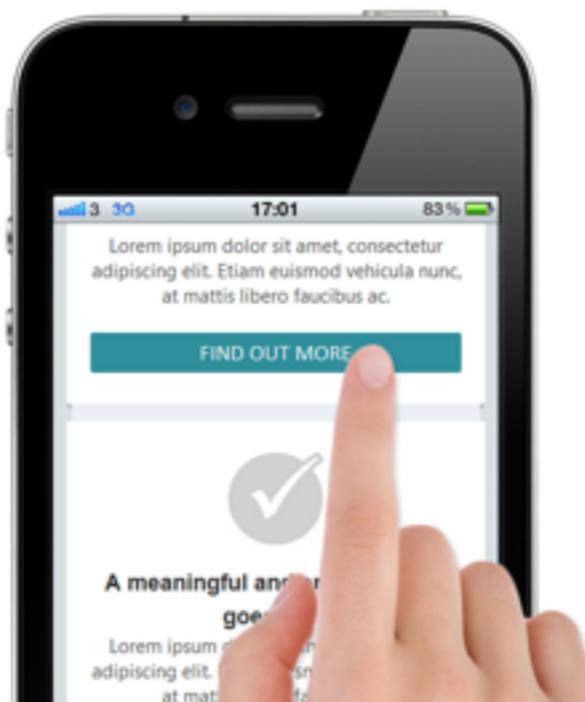
Achieving Mobile Web Performance

Mobile Client



Achieving Mobile Web Performance

Mobile
Client

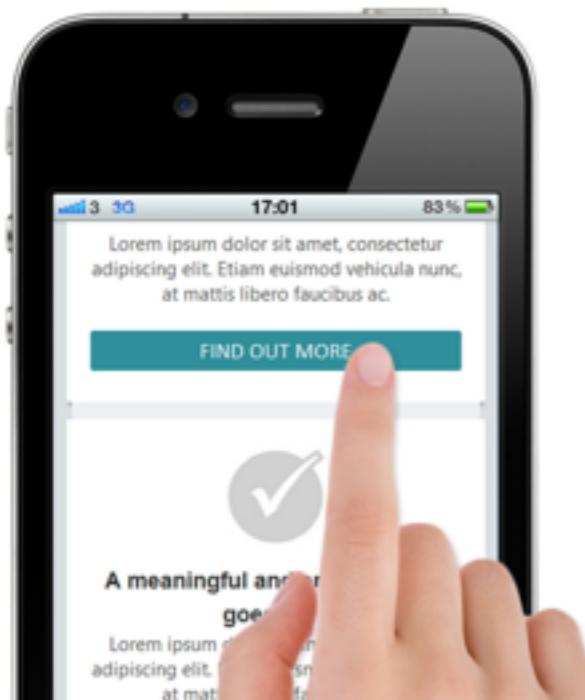


Cloud
Web Servers

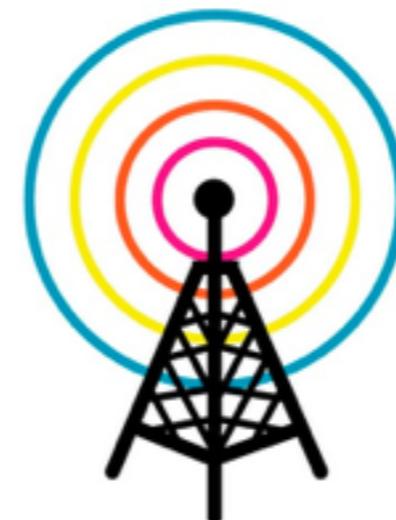


Achieving Mobile Web Performance

Mobile
Client



Cellular
Network

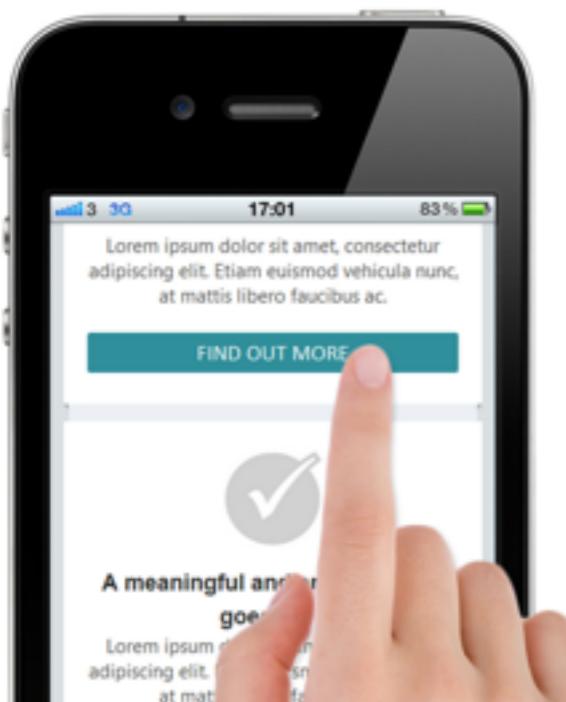


Cloud
Web Servers

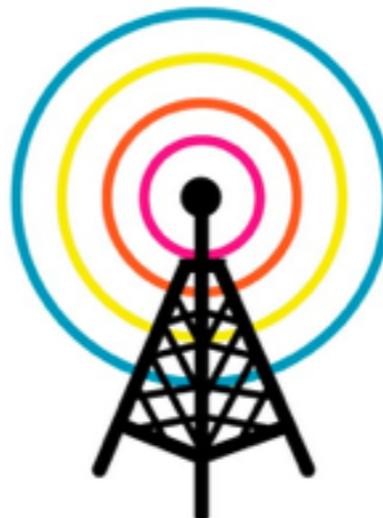


Achieving Mobile Web Performance

Mobile
Client



Cellular
Network

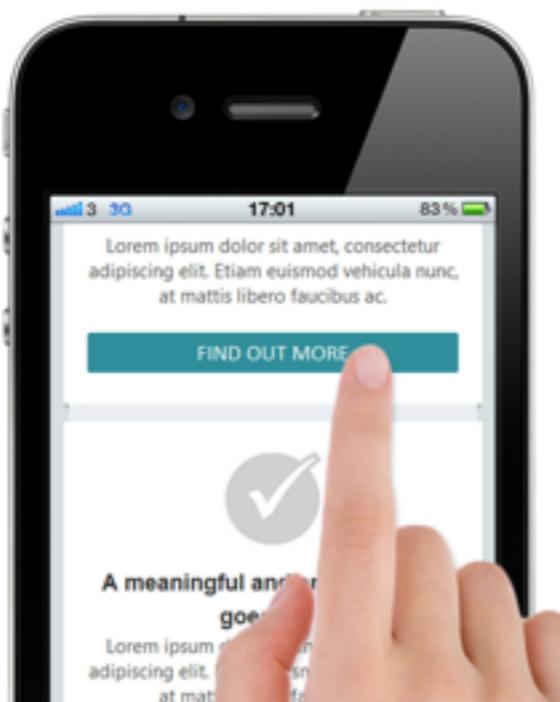


Cloud
Web Servers

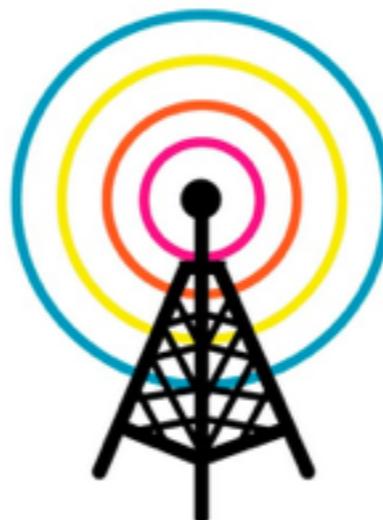


Achieving Mobile Web Performance

Mobile Client



Cellular Network



Cloud Web Servers



[MICRO 2015] (Top Picks
Honorable Mention)

Achieving Mobile Web Performance

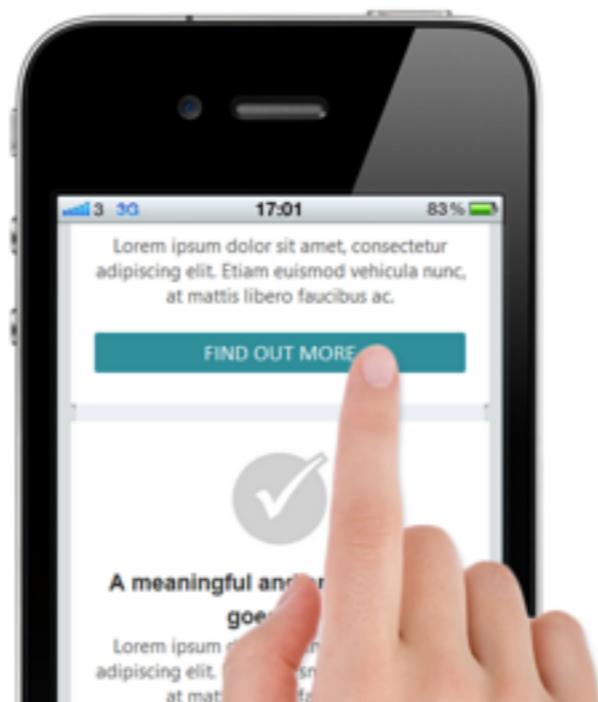
Mobile
Client

Cellular
Network

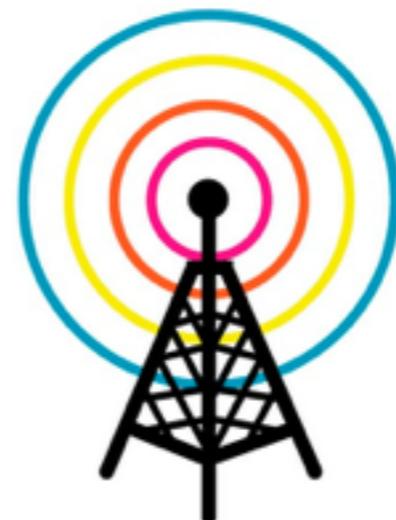


Achieving Mobile Web Performance

Mobile
Client

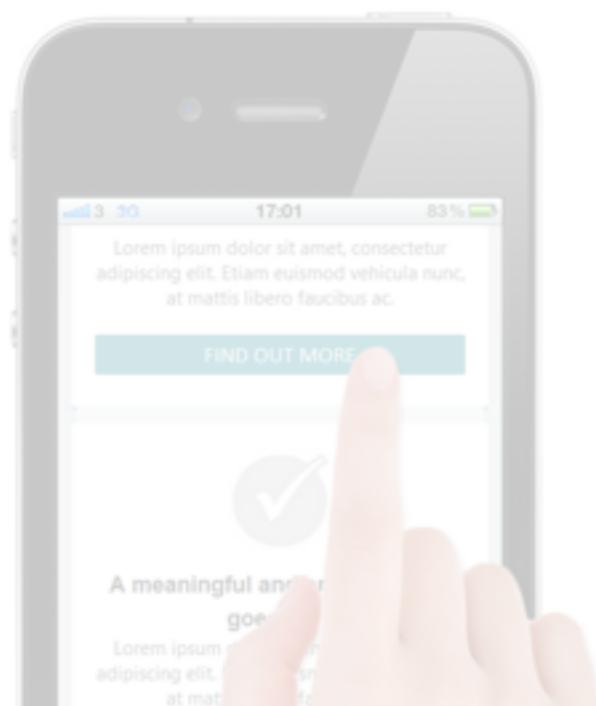


Cellular
Network

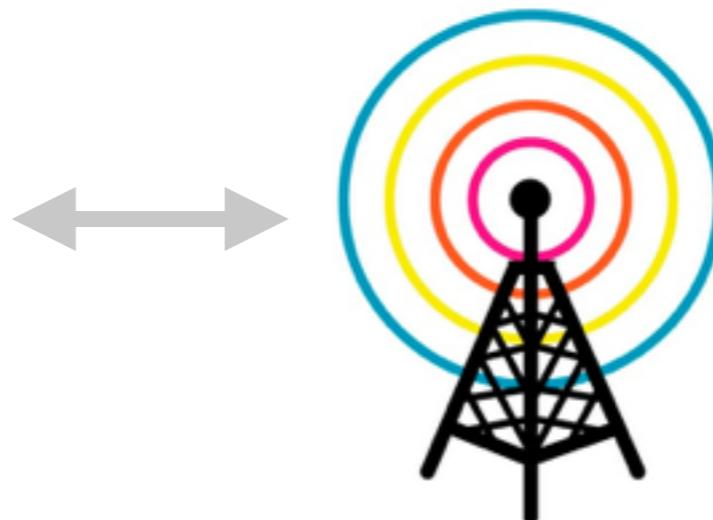


Isn't Responsiveness a **Network** Issue?

Mobile
Client



Cellular
Network



Isn't Responsiveness a Network Issue?

[HotMobile'11, WWW'12], 100+ citations

Why are Web Browsers Slow on Smartphones?

¹Zhen Wang, ²Felix Xiaozhu Lin, ^{1,2}Lin Zhong, and ³Mansoor Chishtie

¹Dept. of ECE and ²Dept of CS, Rice University, Houston, TX 77005 ³Texas Instruments, Dallas, TX

ABSTRACT

We report the first work that examines the internals of web browsers on smartphones, using the WebKit codebase, two generations of Android smartphones, and webpages visited by 25 smartphone users over three months. We make many surprising findings. First, over half of the webpages visited by smartphone users are not optimized for mobile devices. This highlights the importance of client-based optimization and the limitation of prior work that only studies mobile webpages. Second, while prior work suggests that several compute-intensive operations should be the focus of optimization, our measurement and analysis show that their improvement will only lead to marginal performance gain with existing webpages. Furthermore, we find that resource loading, ignored by all except one prior work, contributes most to the browser delay. While our results agree with a recent network study showing that network round-trip time is a major problem, we further demonstrate how the internals of the browser and operating system contribute to the browser delay and therefore reveal new opportunities for optimization.

1. Introduction

As one of the most important applications on smartphones, the

ble using techniques employed in prior work.

We make the following key findings. (i) Improvement on compute-intensive operations suggested by prior work such as *style formatting*, *layout calculation* [2-4], and *JavaScript execution* [1] will lead to marginal improvement in browser performance on smartphones. (ii) Instead, *resource loading* is the key to browser performance on smartphones. Resource loading is the process that resources, usually files of various types needed by opening a webpage, are acquired by the smartphone from the web server. In contrast, prior work [2-4] assume resource loading contributes negligible delay, which is not true with smartphones. (iii) Given a resource, the delay of resource loading is determined by the network condition, the browser loading procedure and the processing power of the smartphone. Our results agree with the findings from [1] that long network RTT is detrimental to the browser performance. We further find that improvement in network bandwidth will not improve browser performance much beyond typical 3G network. Finally, by comparing the behaviors of two smartphones, Google Nexus One (N1) and HTC Dream (G1), we observe a more powerful hardware, e.g. N1, will reduce the browser delay mainly by accelerating OS services and network stack, instead of the compute-intensive operations suggested by prior work.

Our findings not only shed light into the behavior of web

Isn't Responsiveness a Network Issue?

[HotMobile'11, WWW'12], 100+ citations

Why are Web Browsers Slow on Smartphones?

¹Zhen Wang, ²Felix Xiaozhu Lin, ^{1,2}Lin Zhong, and ³Mansoor Chishtie

¹Dept. of ECE and ²Dept of CS, Rice University, Houston, TX 77005 ³Texas Instruments, Dallas, TX

Resource loading is the bottleneck

Isn't Responsiveness a Network Issue?

[HotMobile'11, WWW'12], 100+ citations

Why are Web Browsers Slow on Smartphones?

¹Zhen Wang, ²Felix Xiaozhu Lin, ^{1,2}Lin Zhong, and ³Mansoor Chishtie

¹Dept. of ECE and ²Dept of CS, Rice University, Houston, TX 77005 ³Texas Instruments, Dallas, TX

Resource loading is the bottleneck

Client compute doesn't matter much

Isn't Responsiveness a Network Issue?

[HotMobile'11, WWW'12], 100+ citations

Why are Web Browsers Slow on Smartphones?

¹Zhen Wang, ²Felix Xiaozhu Lin, ^{1,2}Lin Zhong, and ³Mansoor Chishtie

¹Dept. of ECE and ²Dept of CS, Rice University, Houston, TX 77005 ³Texas Instruments, Dallas, TX

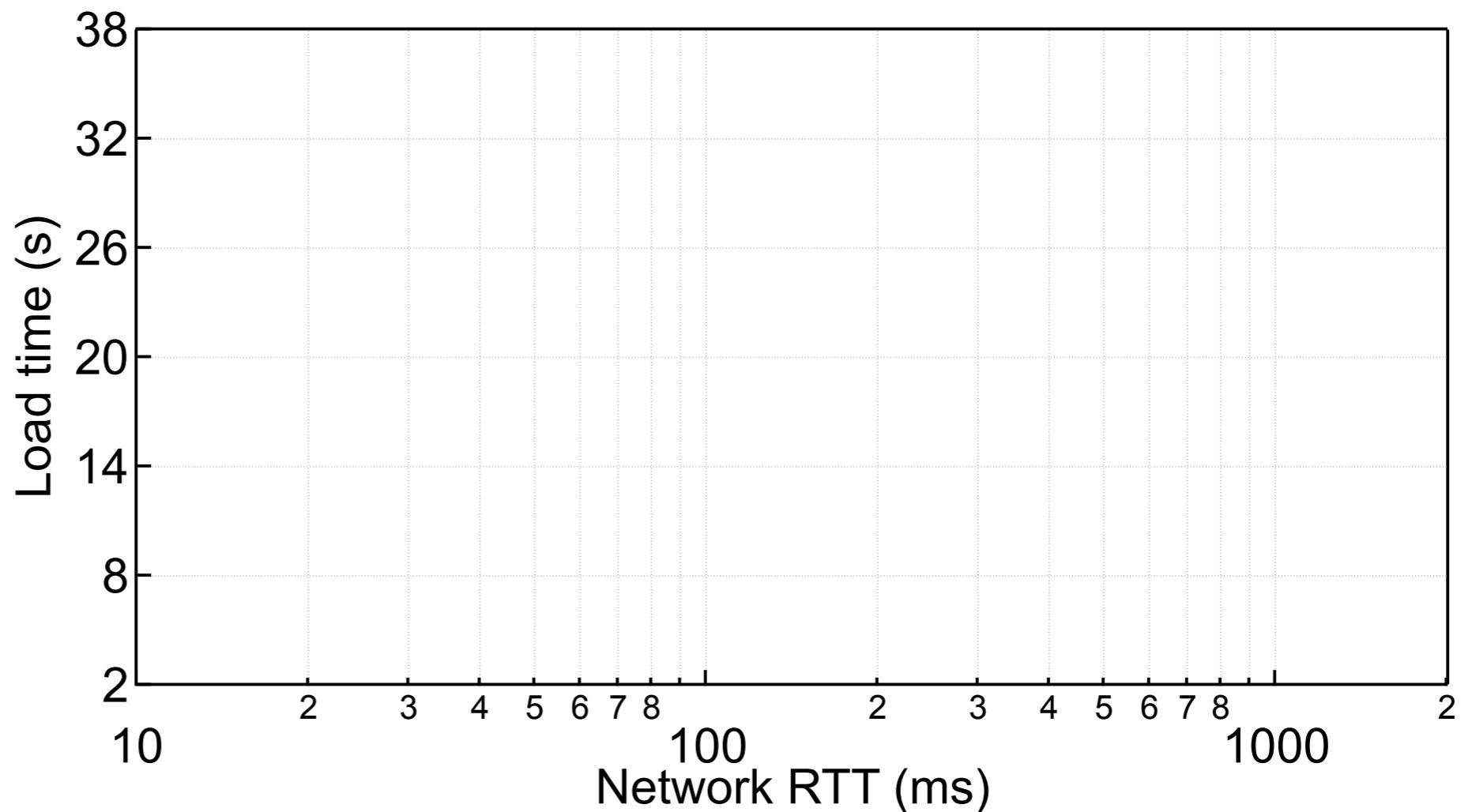
Resource loading is the bottleneck

Client compute doesn't matter much

Conclusions circa 2010!

Isn't Responsiveness a **Network** Issue?

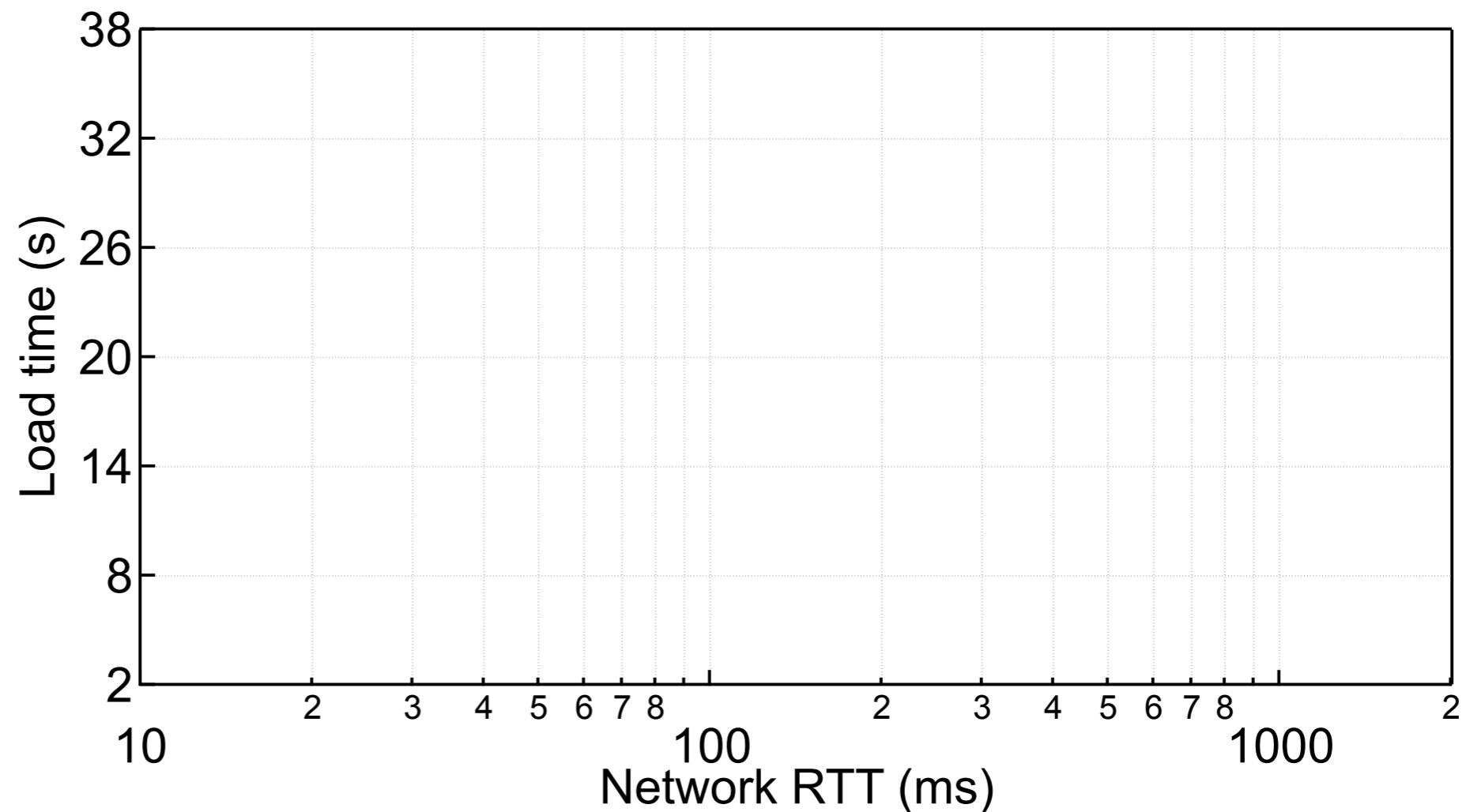
A Year 2015 Experiment!



Isn't Responsiveness a **Network** Issue?

A Year 2015 Experiment!

- ▶ Samsung Galaxy S4 smartphone.
- ▶ Hot webpages from Alexa¹.
- ▶ Time measured using Navigation Timing API².

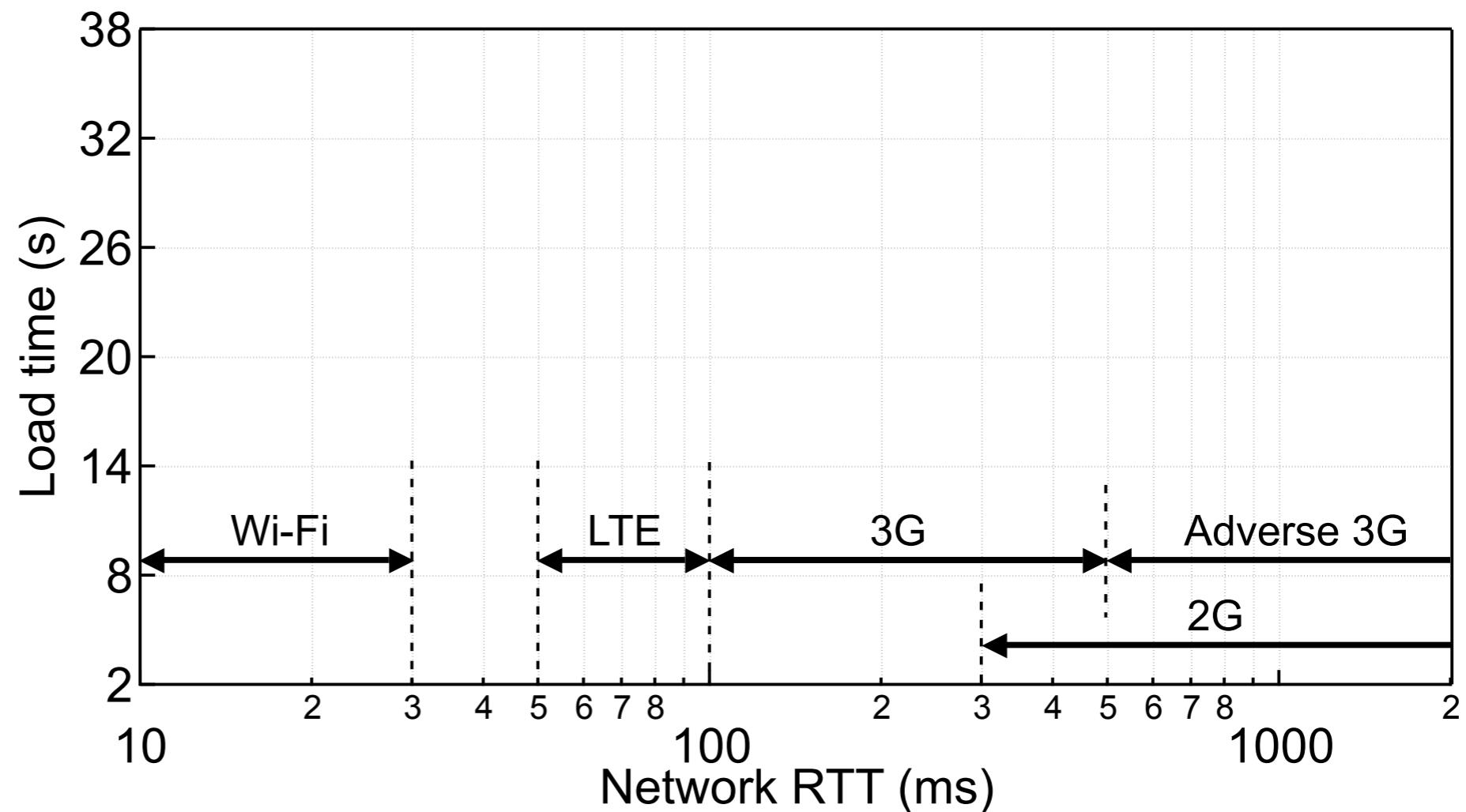


1. <http://www.alexa.com/>
2. <https://www.w3.org/TR/navigation-timing-2/>

Isn't Responsiveness a **Network** Issue?

A Year 2015 Experiment!

- ▶ Samsung Galaxy S4 smartphone.
- ▶ Hot webpages from Alexa¹.
- ▶ Time measured using Navigation Timing API².

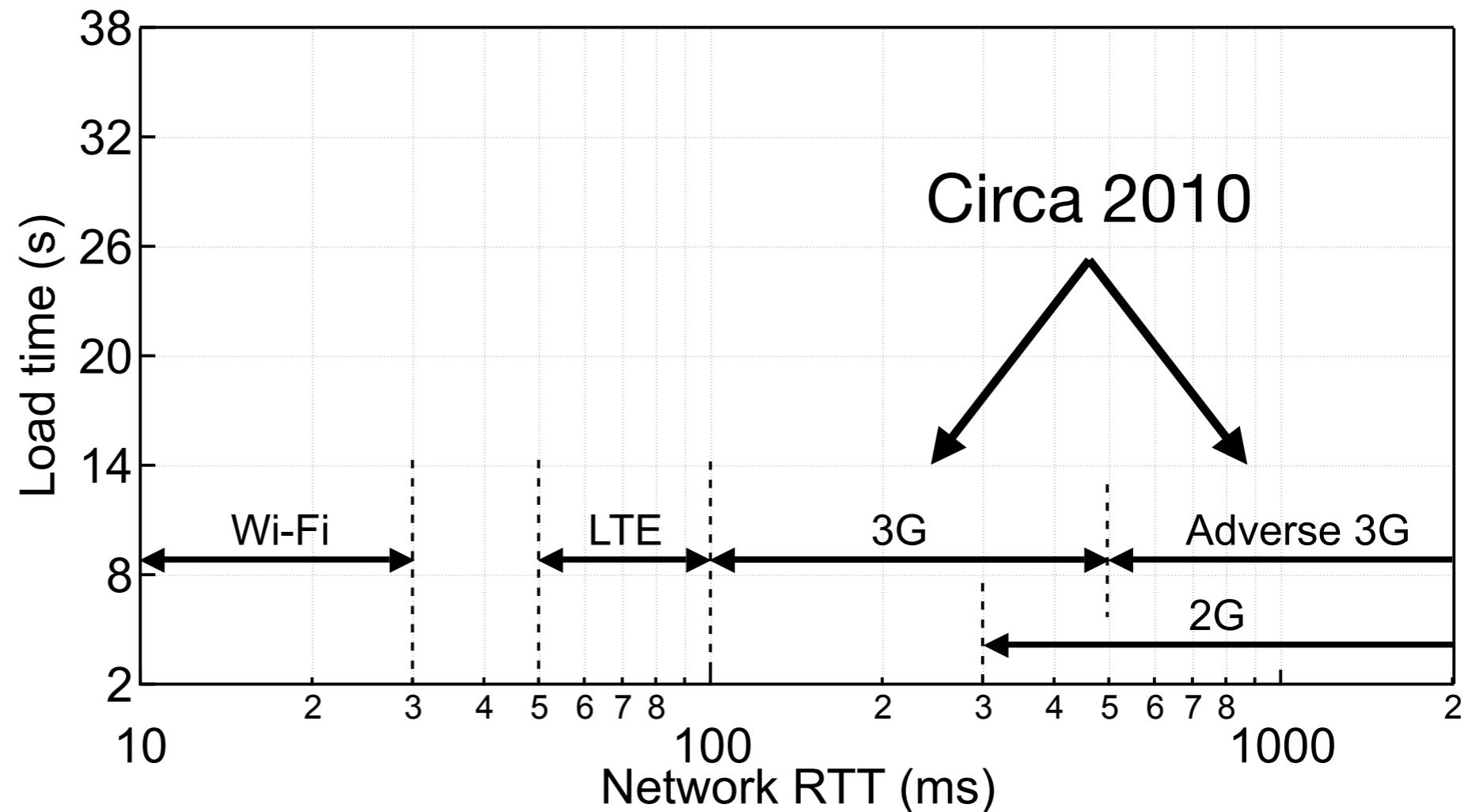


1. <http://www.alexa.com/>
2. <https://www.w3.org/TR/navigation-timing-2/>

Isn't Responsiveness a **Network** Issue?

A Year 2015 Experiment!

- ▶ Samsung Galaxy S4 smartphone.
- ▶ Hot webpages from Alexa¹.
- ▶ Time measured using Navigation Timing API².

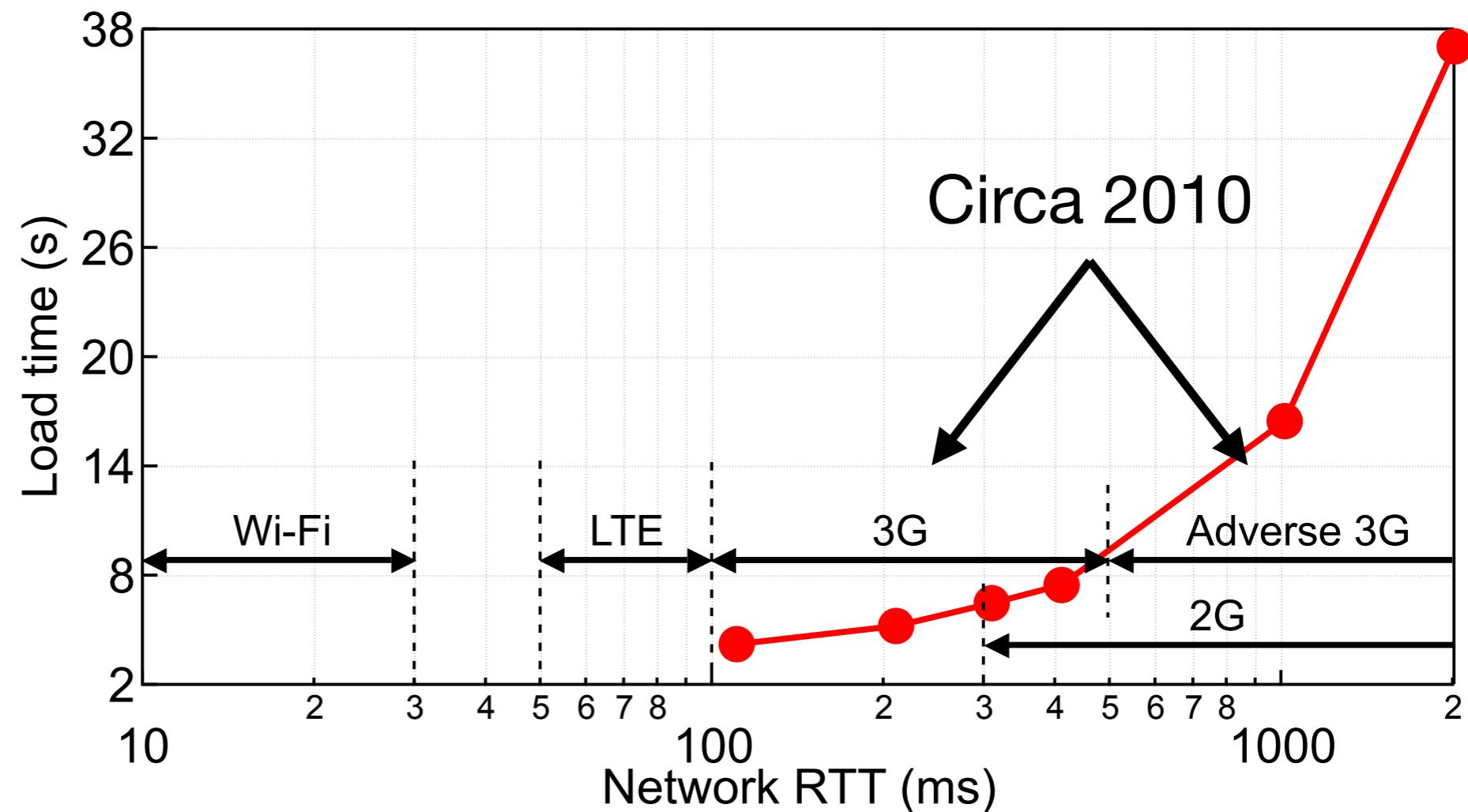


1. <http://www.alexa.com/>
2. <https://www.w3.org/TR/navigation-timing-2/>

Isn't Responsiveness a **Network** Issue?

A Year 2015 Experiment!

- ▶ Samsung Galaxy S4 smartphone.
- ▶ Hot webpages from Alexa¹.
- ▶ Time measured using Navigation Timing API².

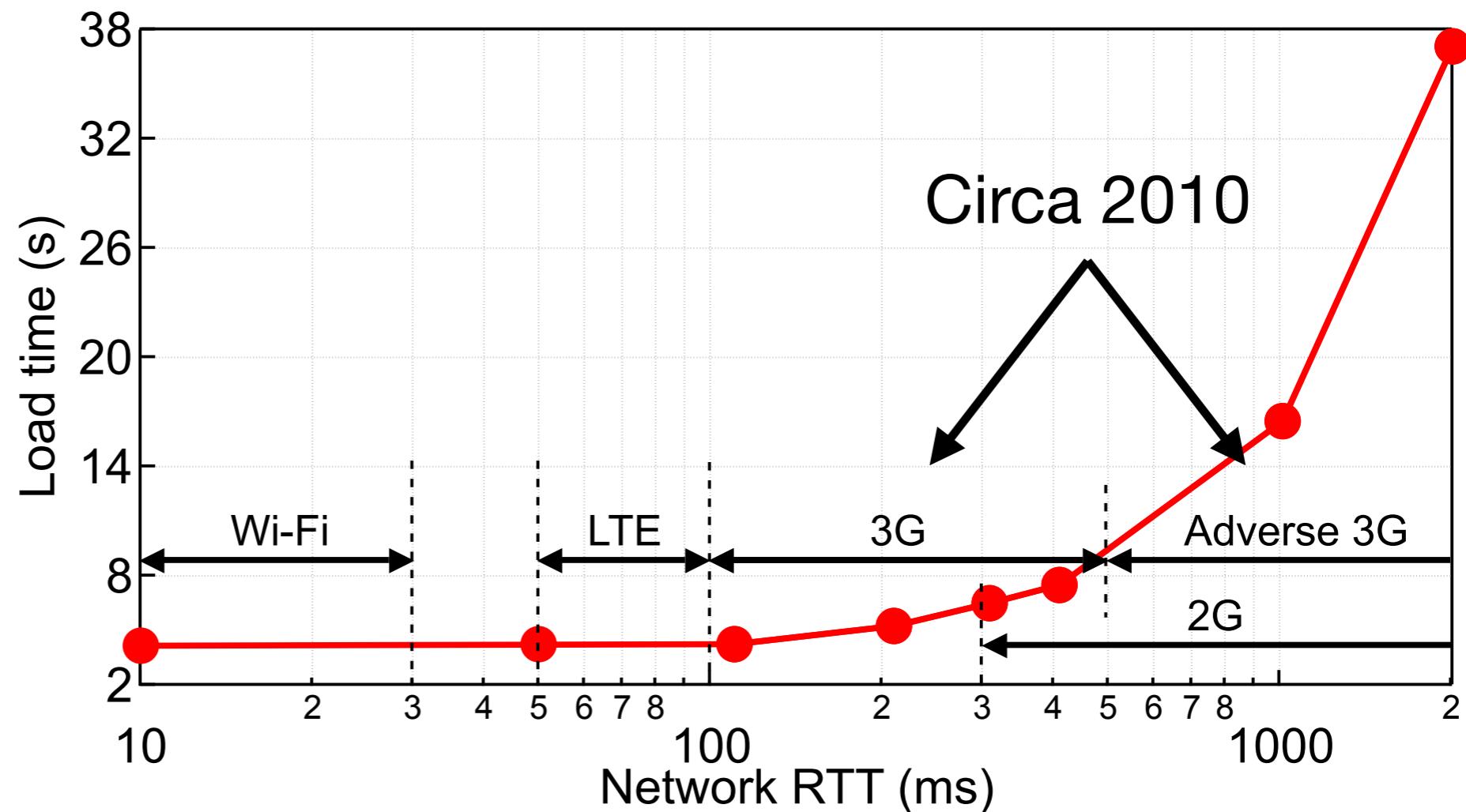


1. <http://www.alexa.com/>
2. <https://www.w3.org/TR/navigation-timing-2/>

Isn't Responsiveness a Network Issue?

A Year 2015 Experiment!

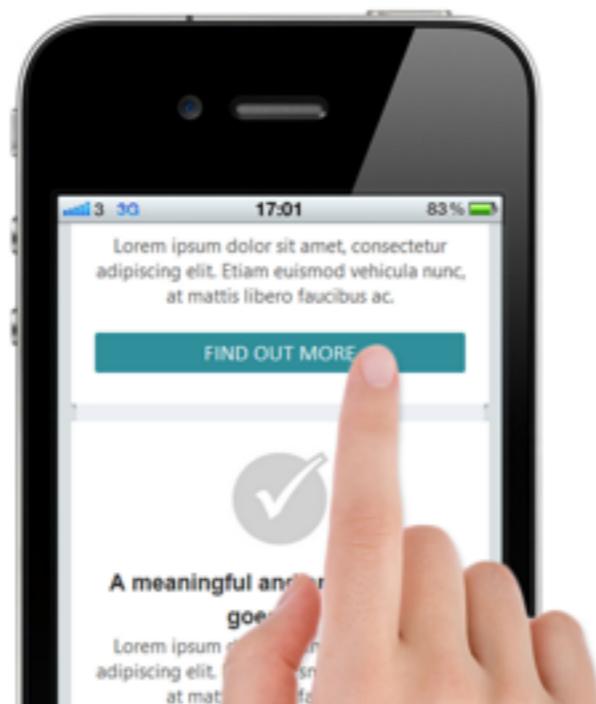
- ▶ Samsung Galaxy S4 smartphone.
- ▶ Hot webpages from Alexa¹.
- ▶ Time measured using Navigation Timing API².



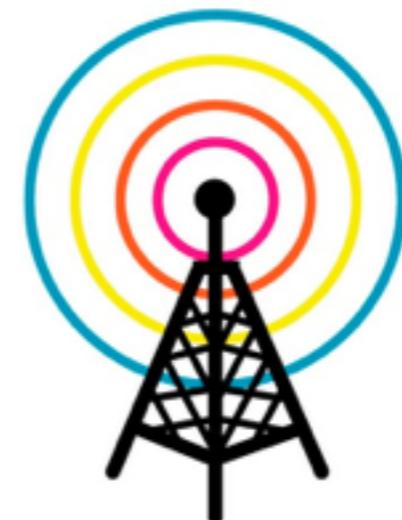
1. <http://www.alexa.com/>
2. <https://www.w3.org/TR/navigation-timing-2/>

Responsiveness is also a **Compute** Issue!

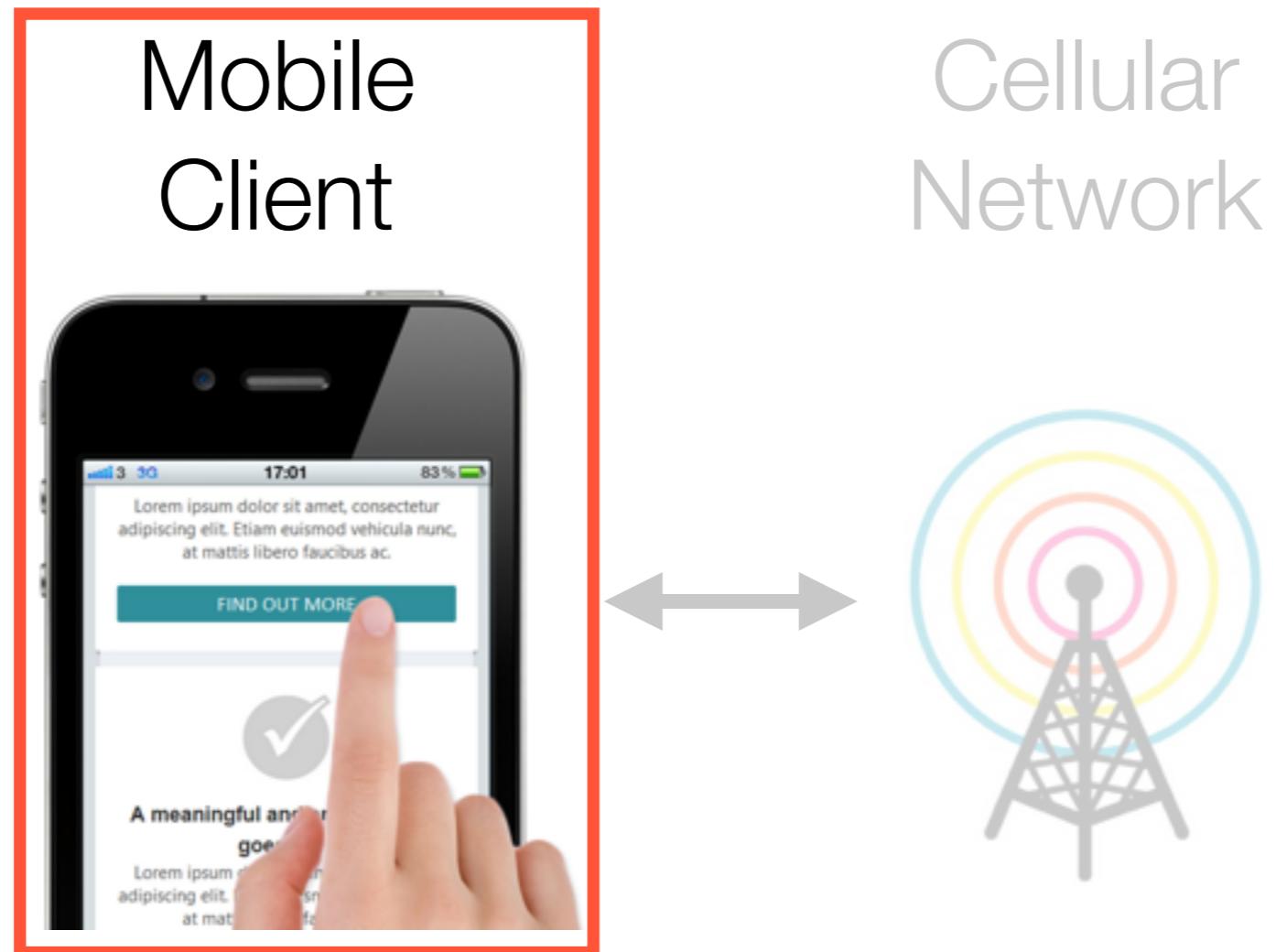
Mobile
Client



Cellular
Network



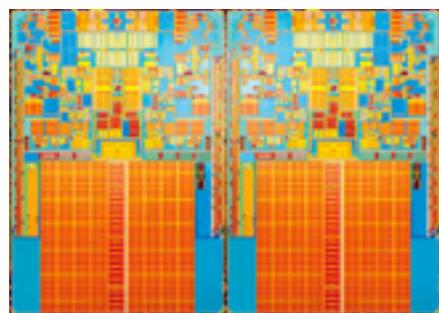
Responsiveness is also a **Compute** Issue!



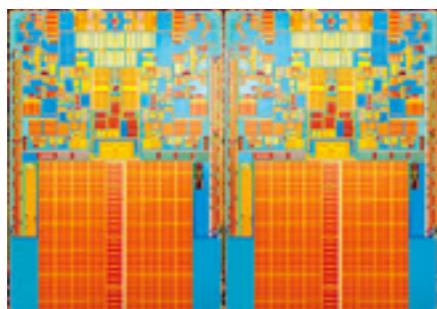
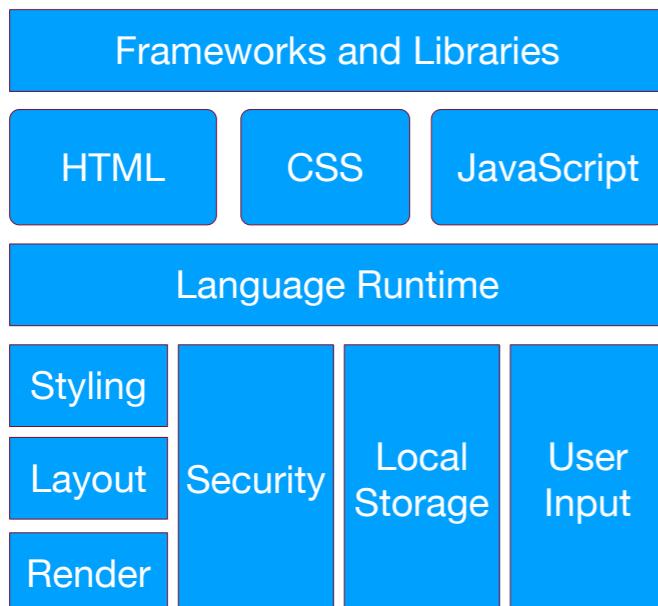
This Proposal



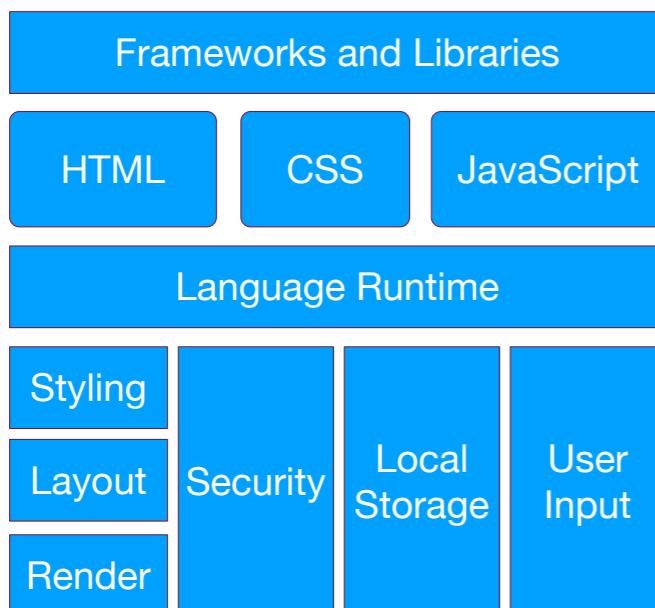
Traditional Approach



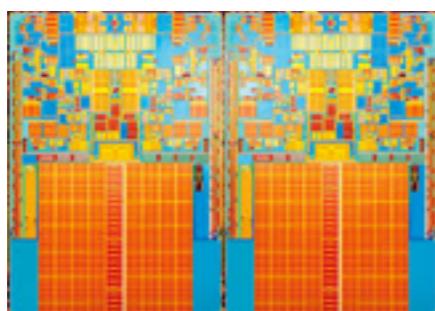
Traditional Approach



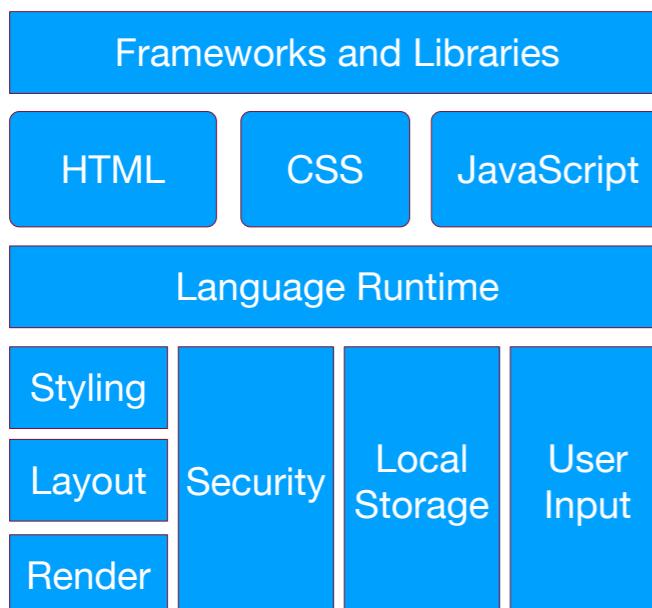
Traditional Approach



Application

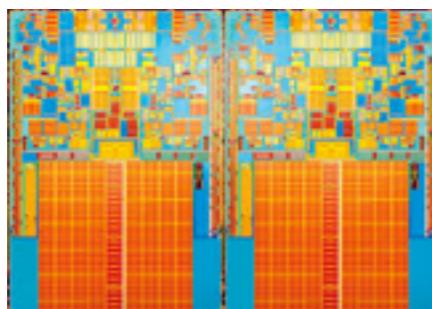


Traditional Approach

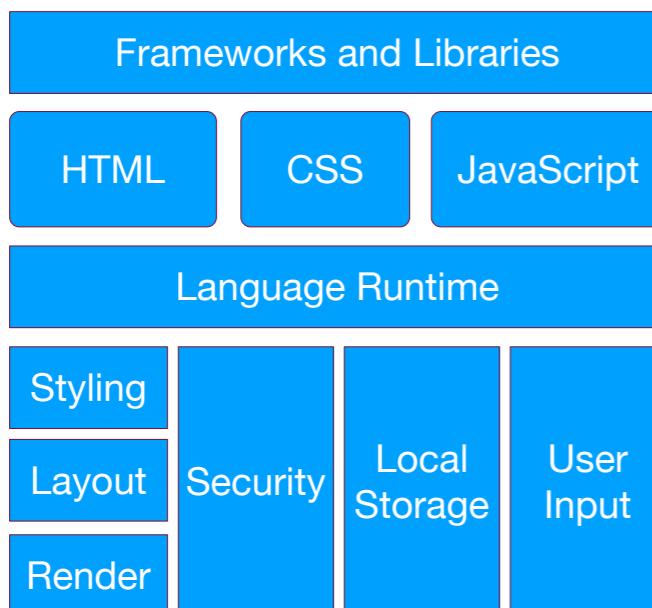


Application

► Parallelize browser computation



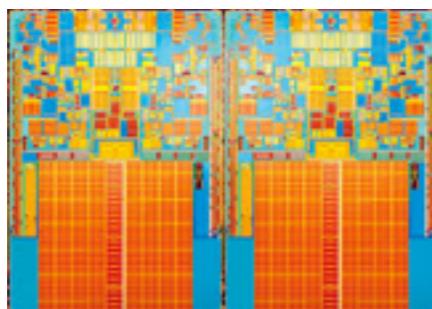
Traditional Approach



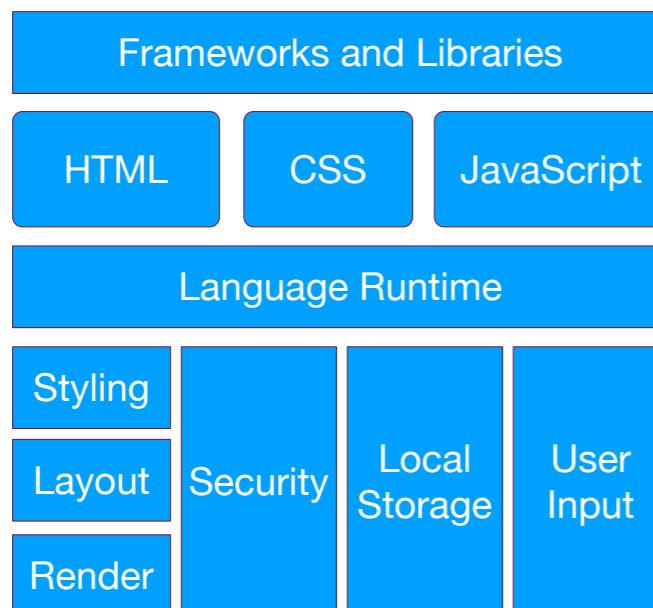
Application

► Parallelize browser computation

Architecture



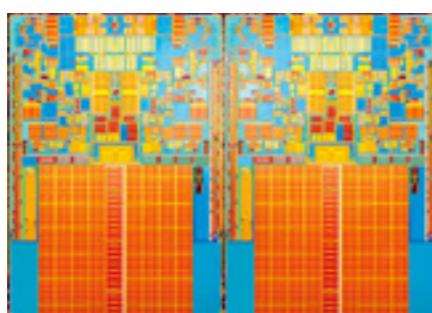
Traditional Approach



Application

- ▶ Parallelize browser computation

Architecture

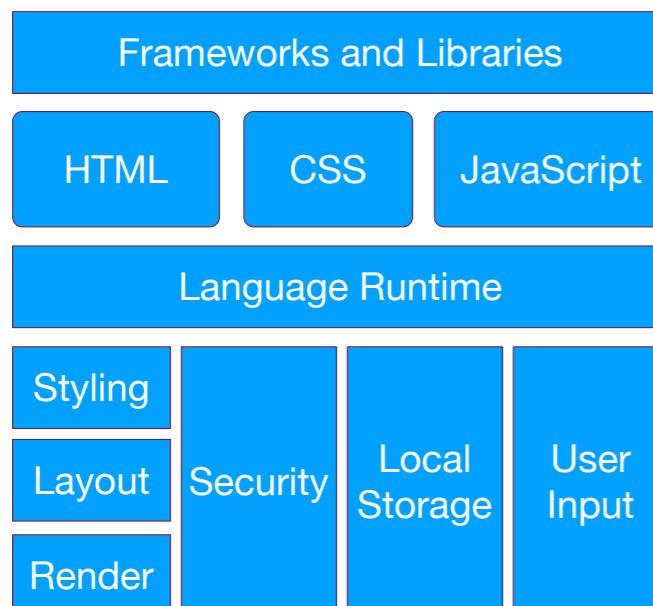


- ▶ Voltage/frequency scaling on general-purpose processors

Traditional Approach

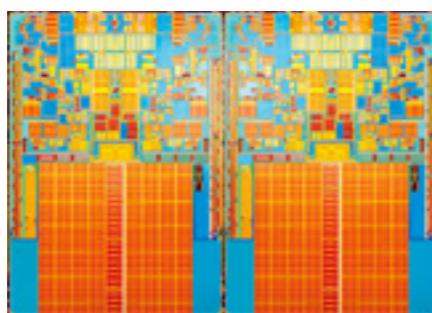


Inputs



Application

- ▶ Parallelize browser computation



Architecture

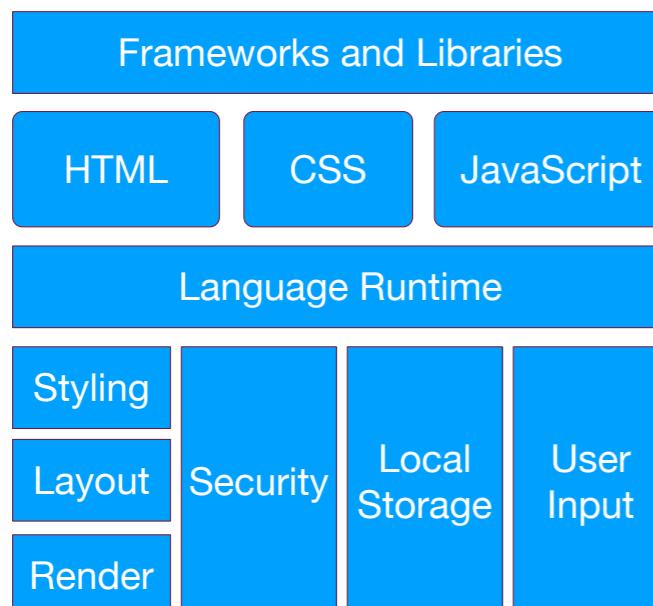
- ▶ Voltage/frequency scaling on general-purpose processors

Traditional Approach



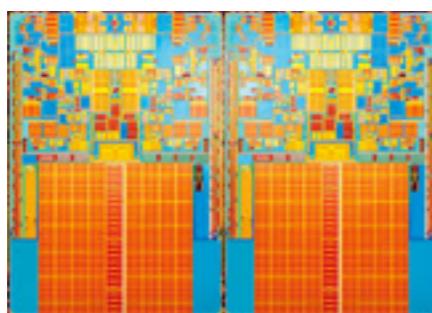
Inputs

- ▶ Ignored!



Application

- ▶ Parallelize browser computation



Architecture

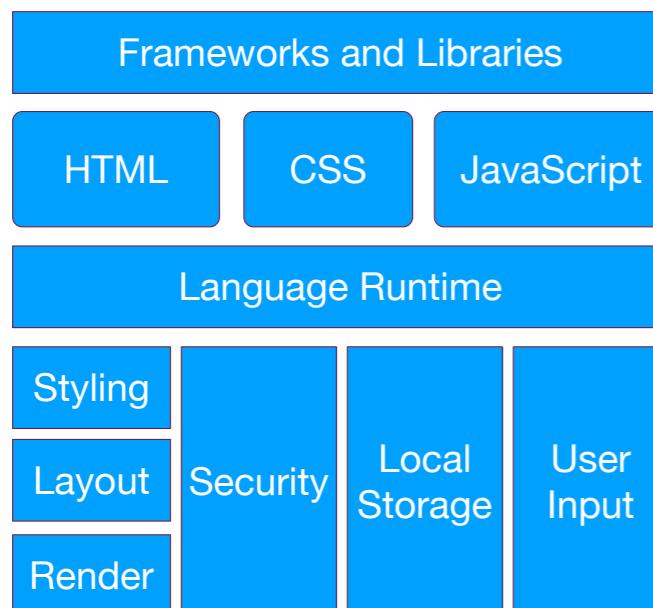
- ▶ Voltage/frequency scaling on general-purpose processors

Traditional Approach



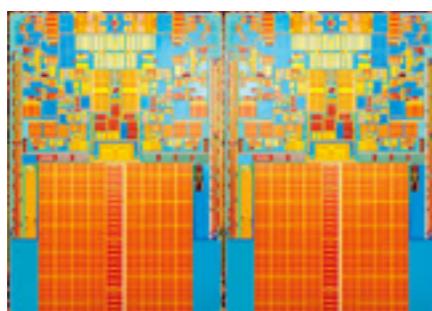
Inputs

- ▶ Ignored!



Application

- ▶ Parallelize browser computation



Architecture

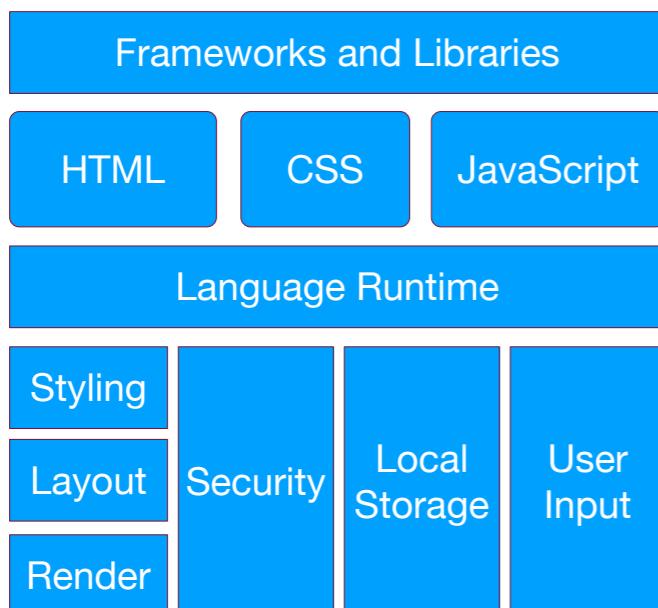
- ▶ End of Dennard Scaling!
- ▶ Diminishing return

My Approach



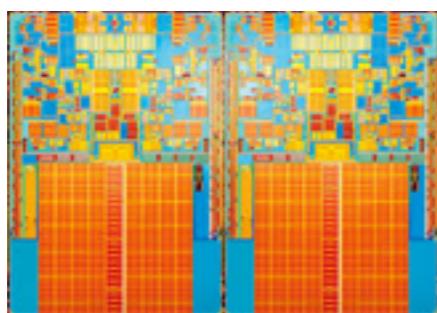
Inputs

► Ignored!



Application

► Parallelize browser computation



Architecture

WebCore

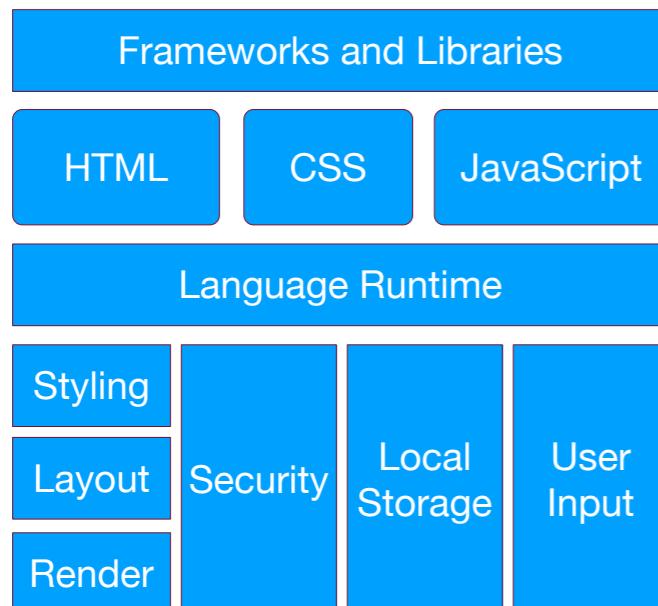
Web-specific Architecture

My Approach



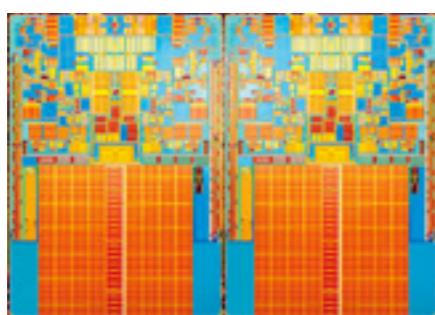
Inputs

- ▶ Lost page-level diversity
- ▶ Lost user QoS requirements



Application

- ▶ Parallelize browser computation

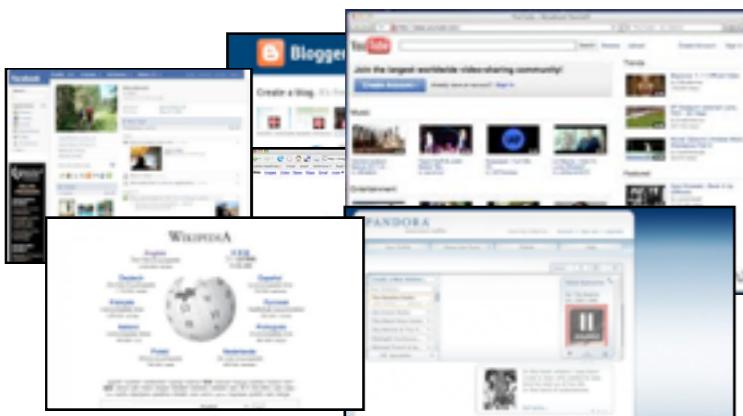


Architecture

WebCore

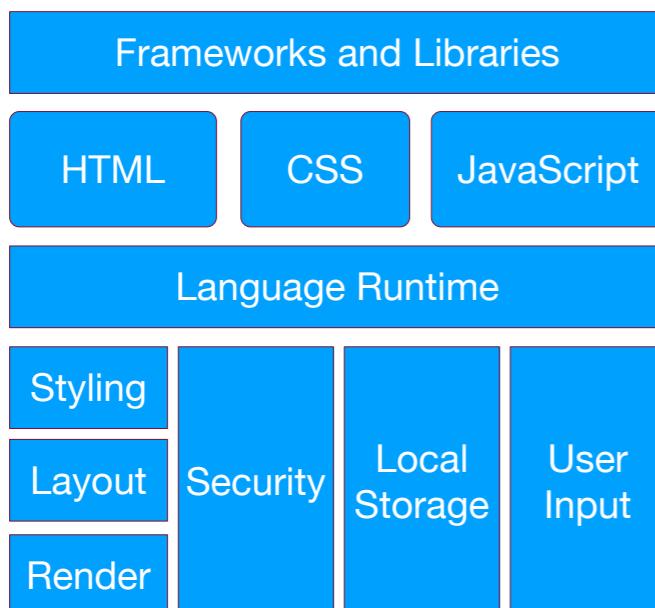
Web-specific Architecture

My Approach

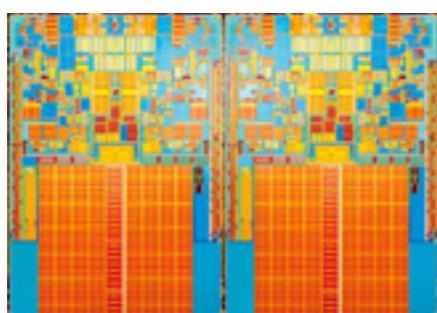


Application

- ▶ Lost page-level diversity
- ▶ Lost user QoS requirements



- ▶ Parallelize browser computation

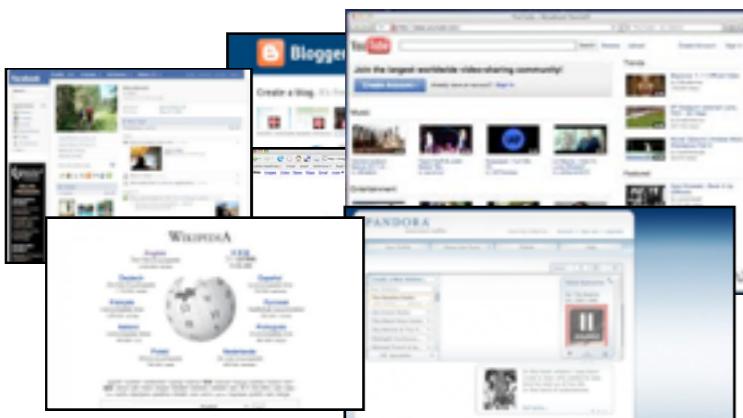


Architecture

WebCore

Web-specific Architecture

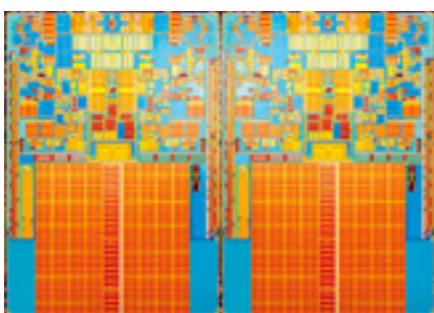
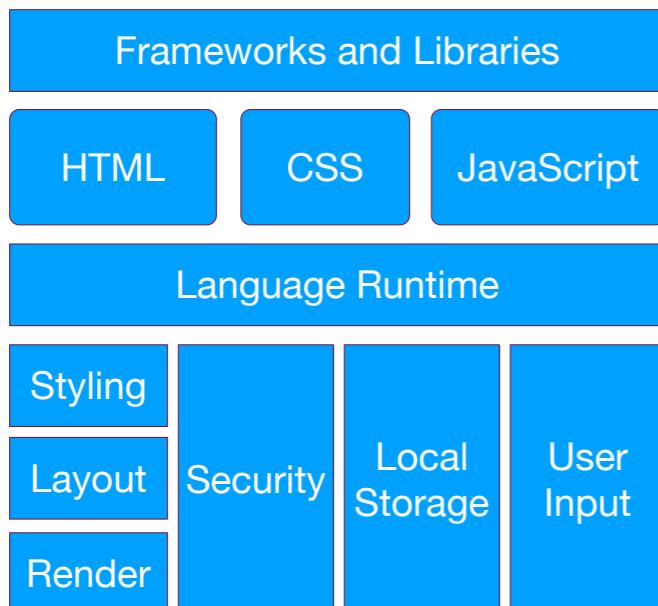
My Approach



Application

GreenWeb

QoS Language Extensions



Architecture

WebCore

Web-specific Architecture



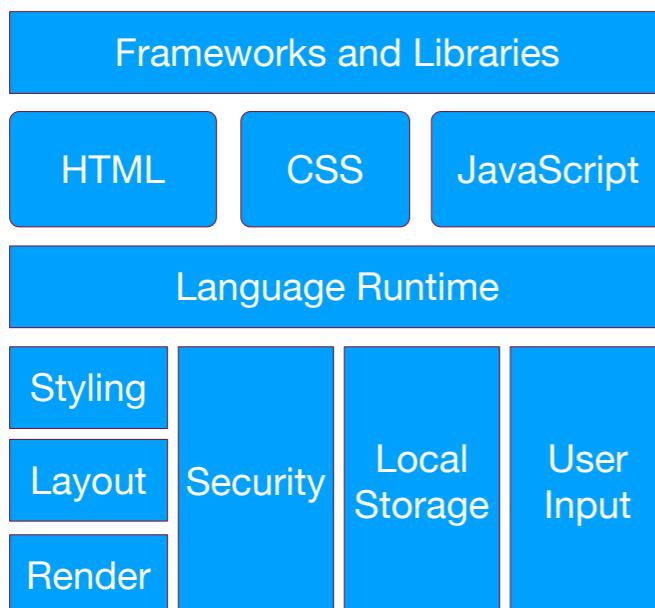
My Approach



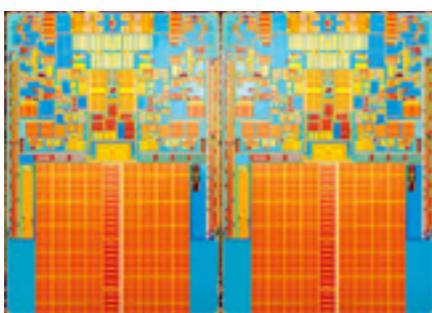
Application

GreenWeb

QoS Language Extensions



Runtime



Architecture

WebCore

Web-specific Architecture

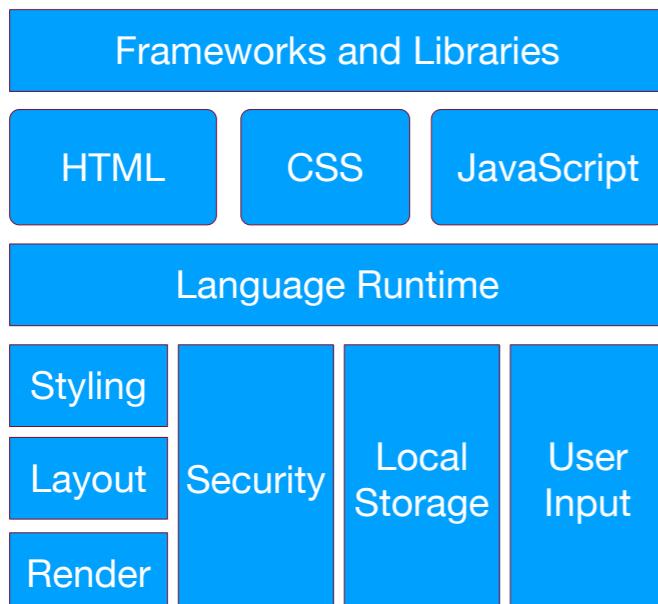
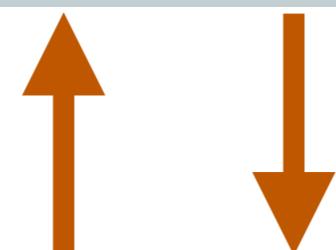


My Approach

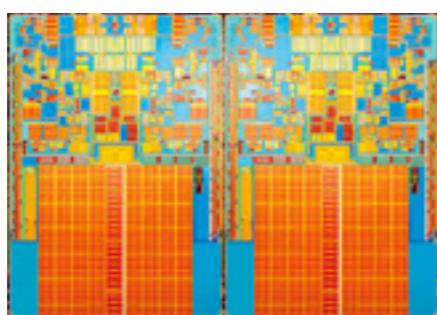


Application

GreenWeb
QoS Language Extensions



Runtime

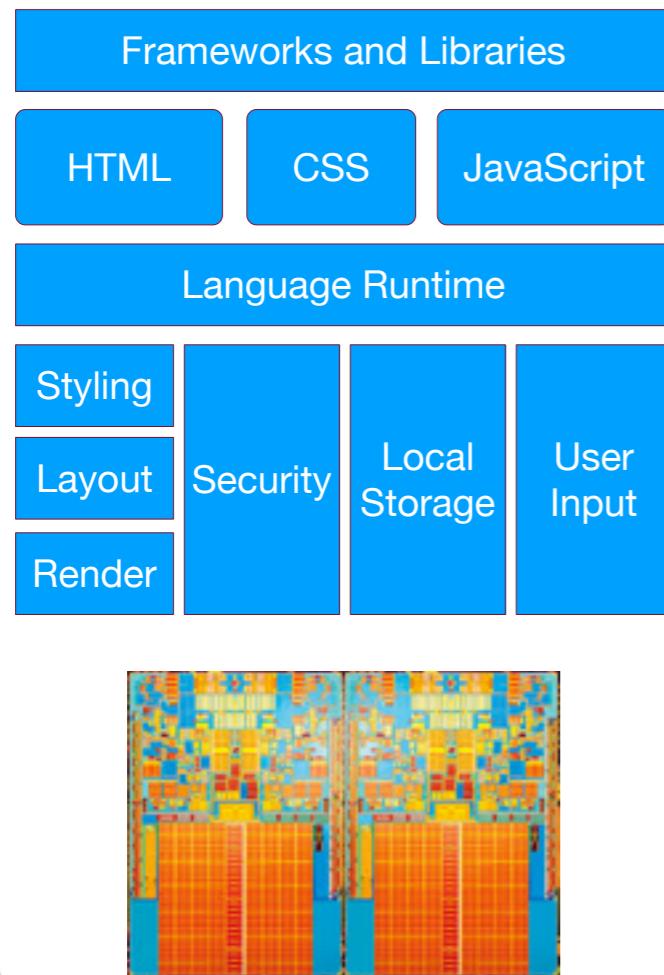


Architecture

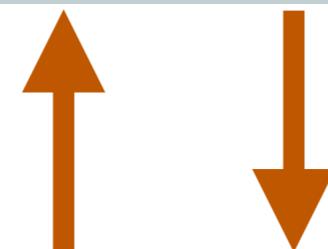
WebCore
Web-specific Architecture



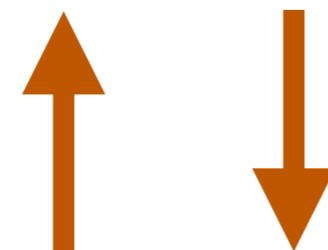
My Approach



Application



Runtime

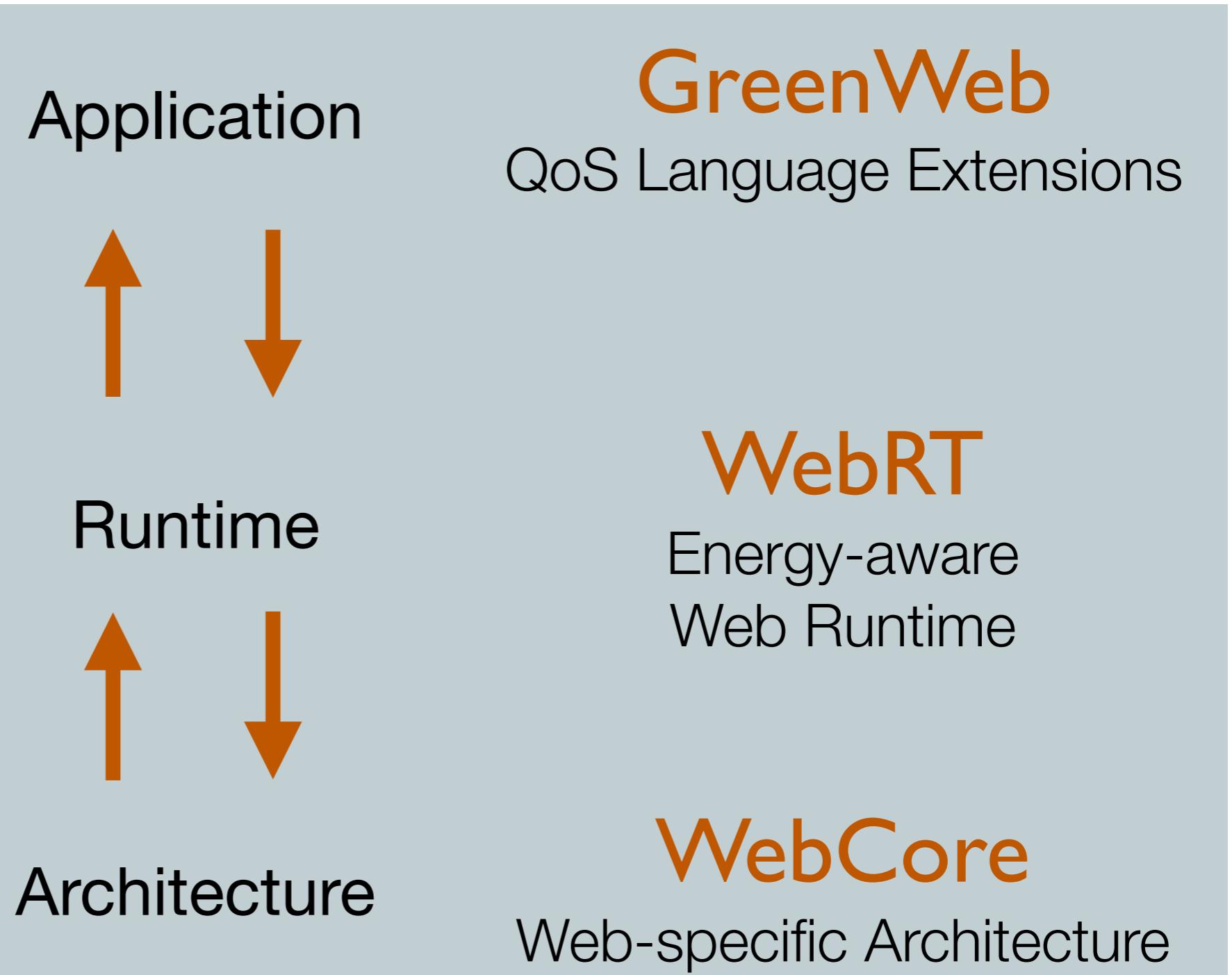
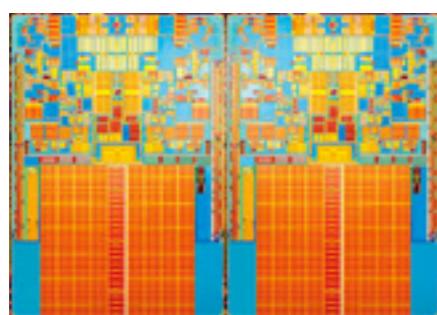
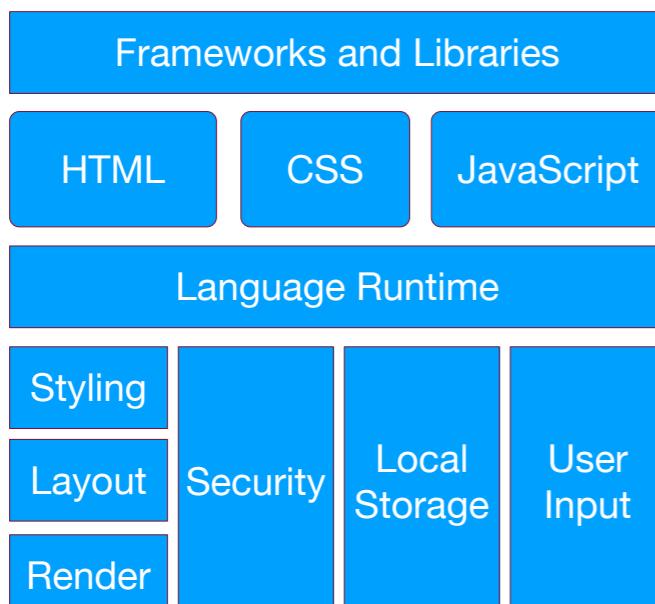


Architecture

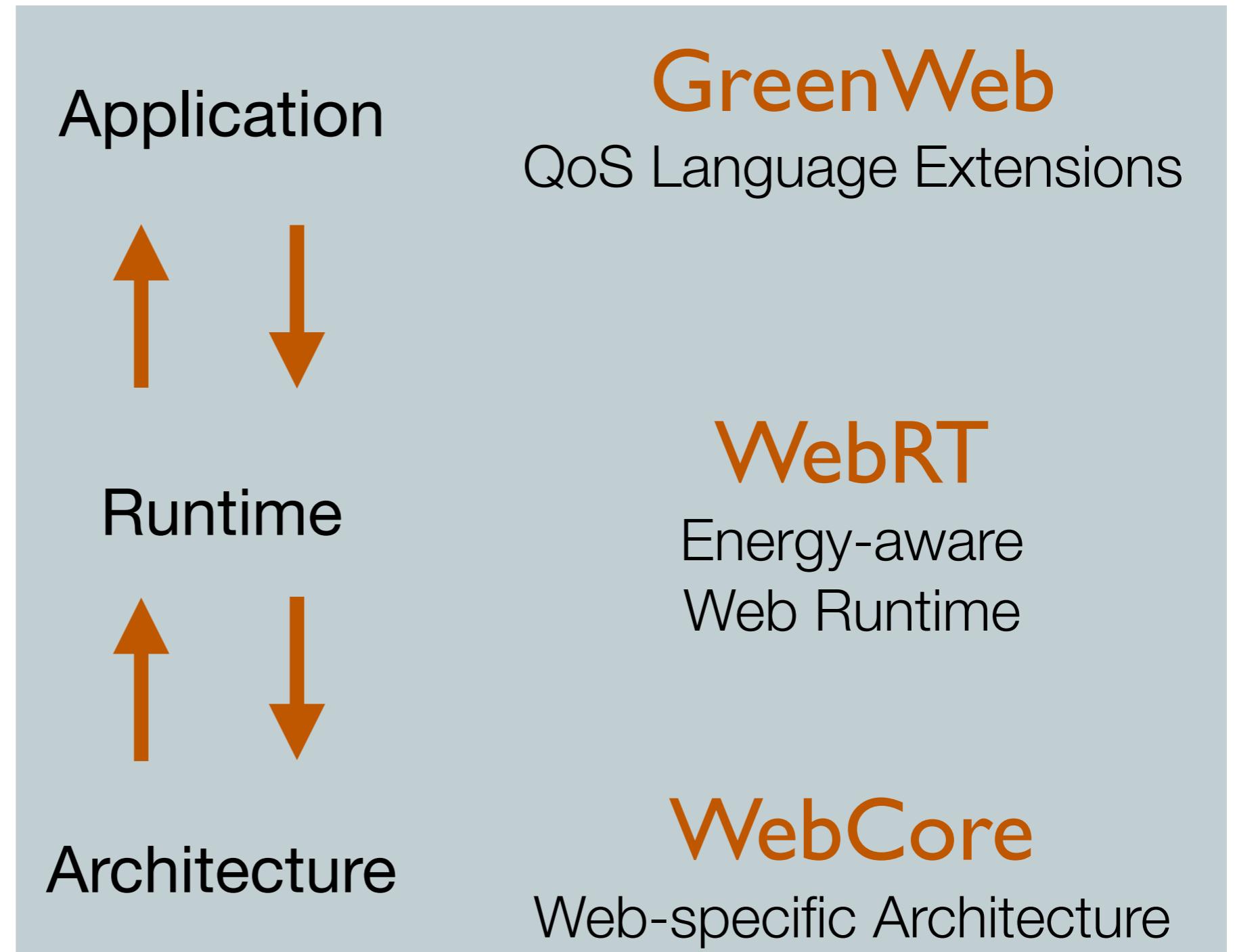
GreenWeb
QoS Language Extensions

WebCore
Web-specific Architecture

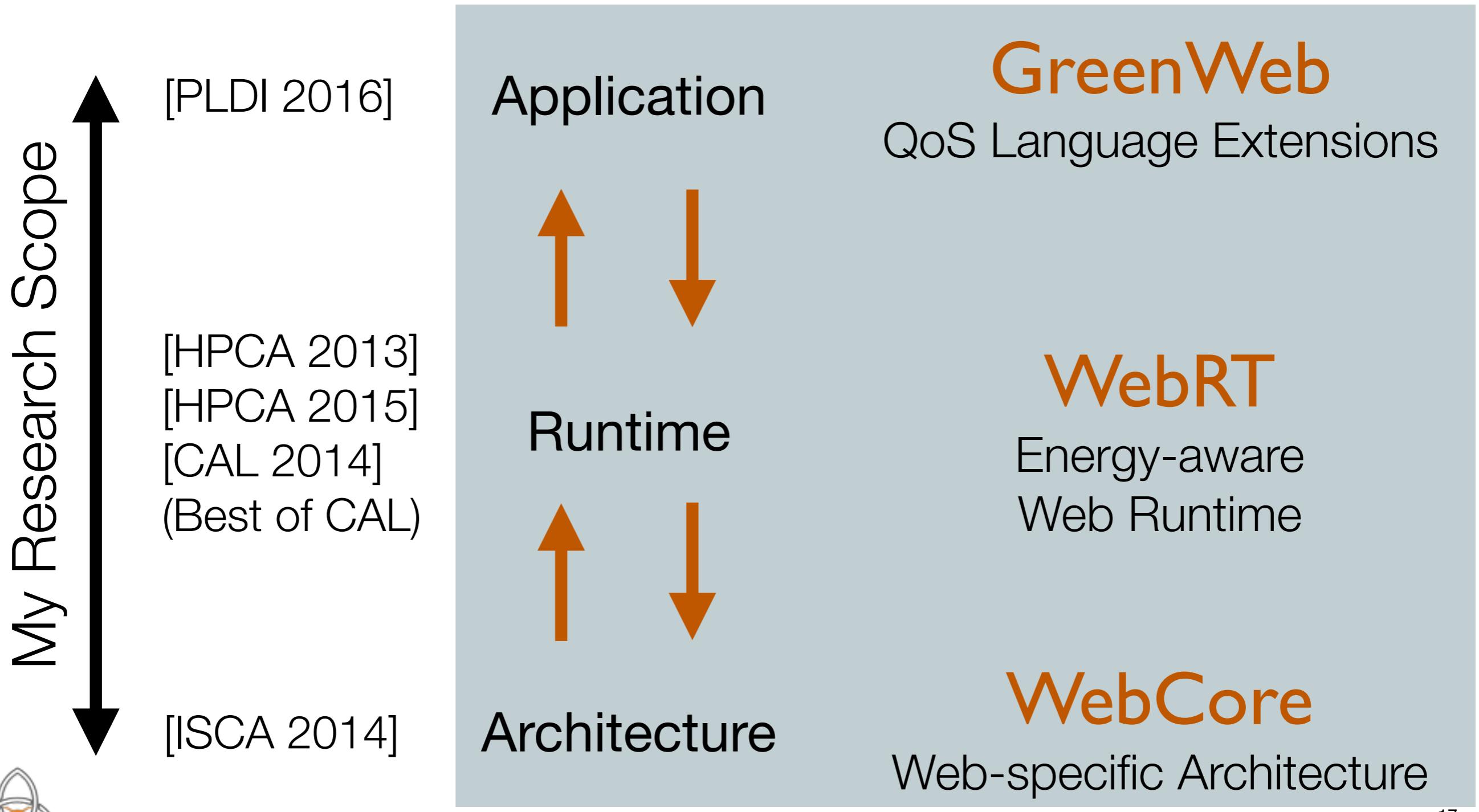
My Approach



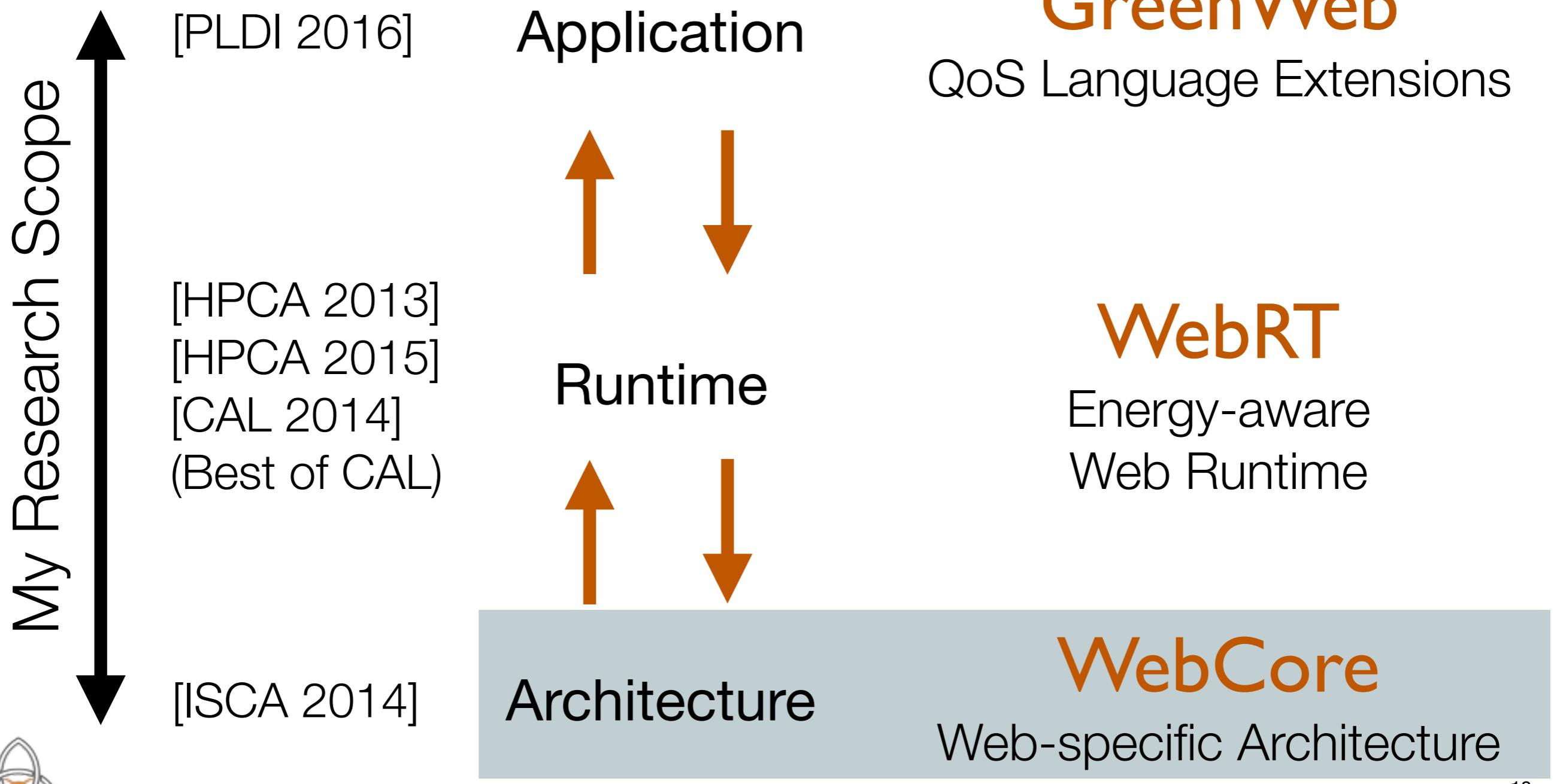
My Approach



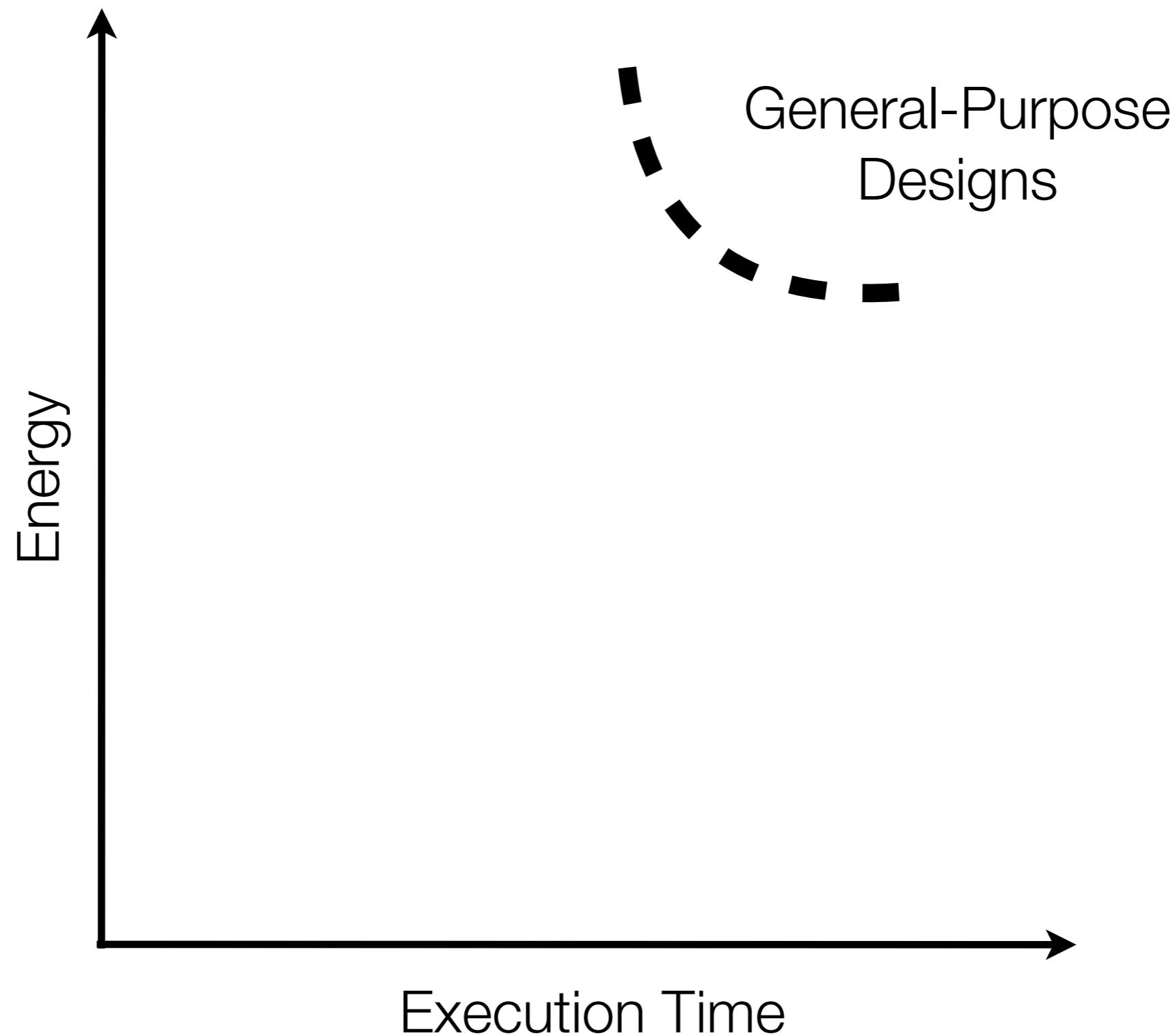
My Approach



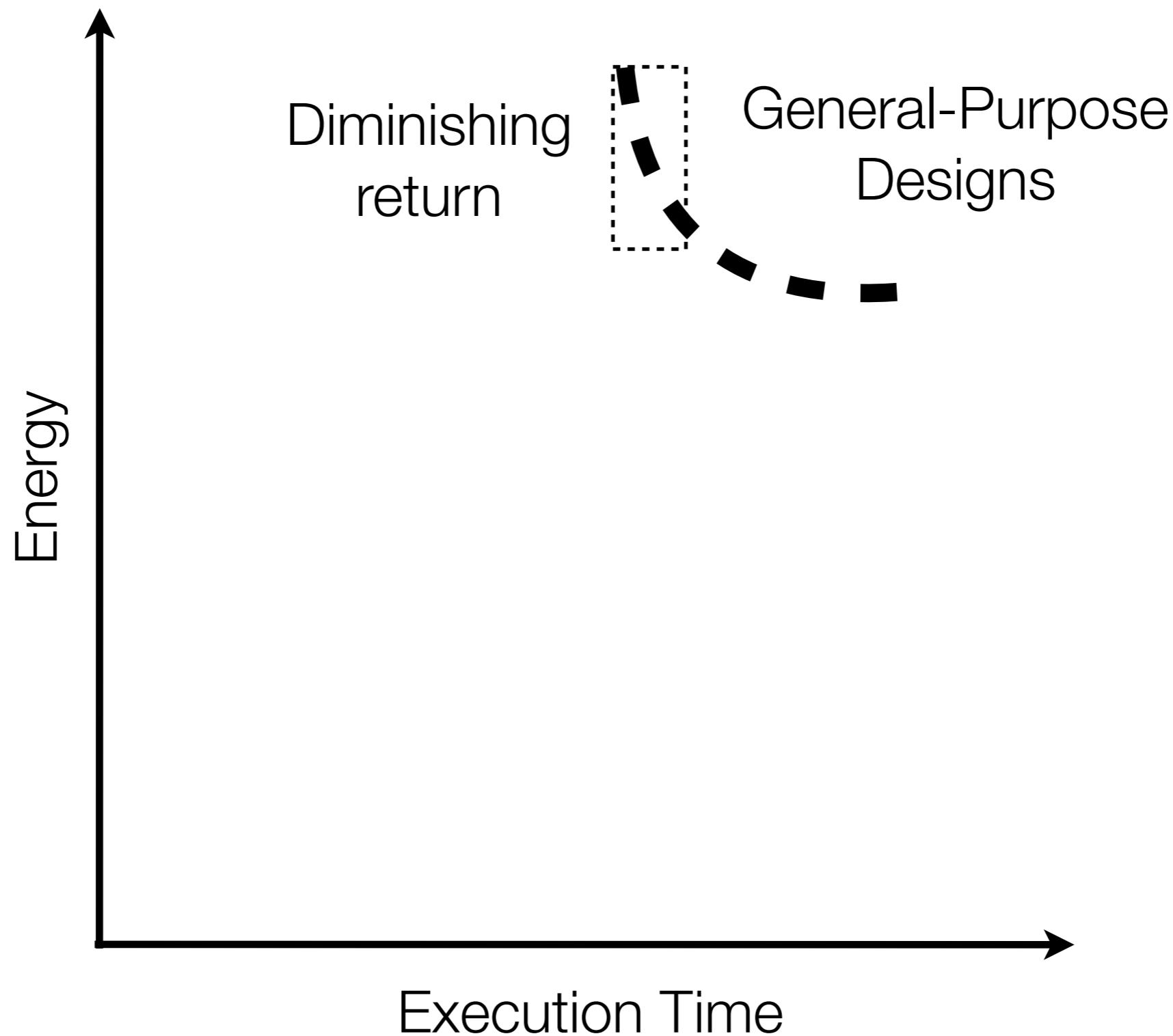
My Approach



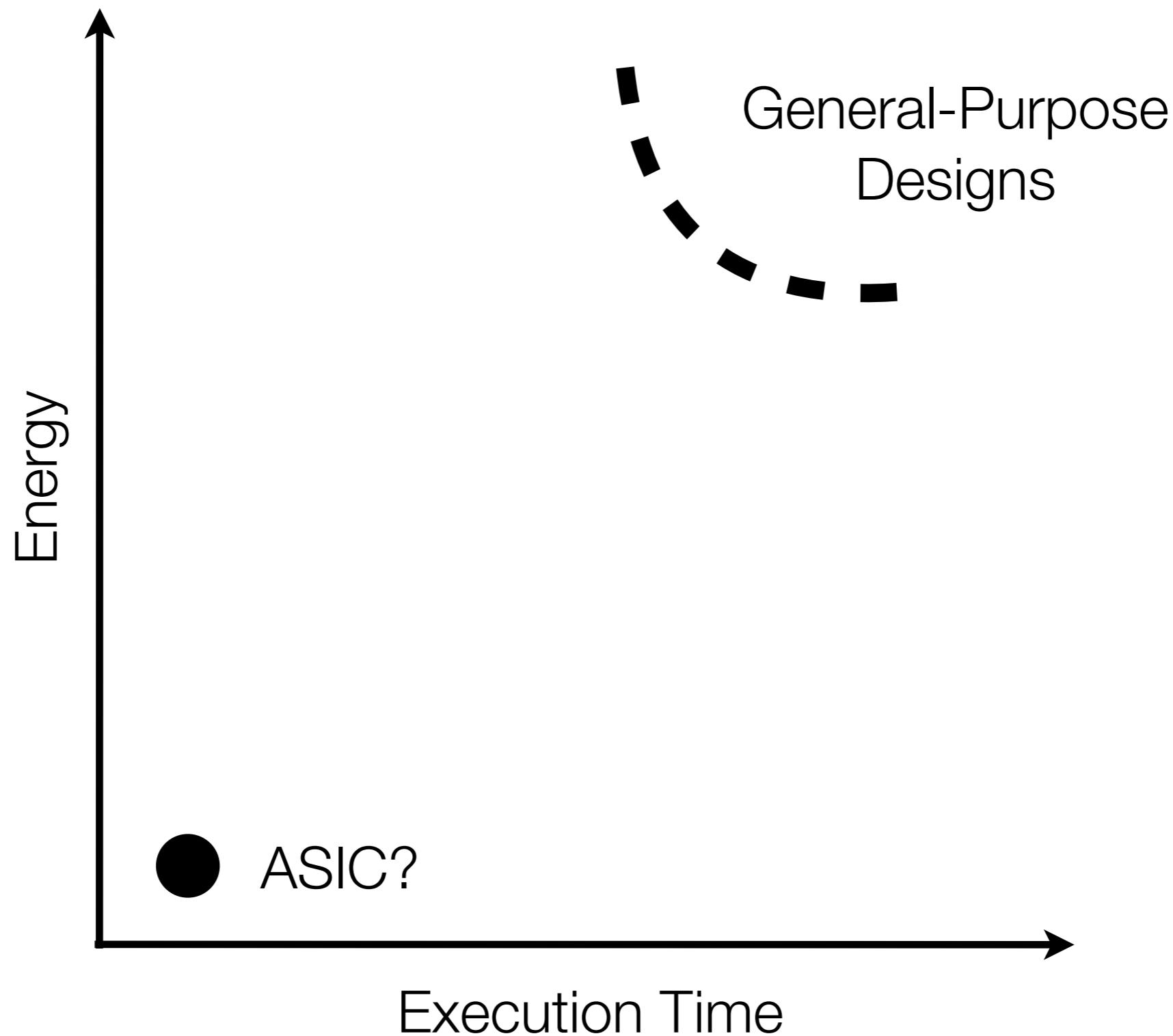
WebCore: a Web-Specific Mobile Architecture



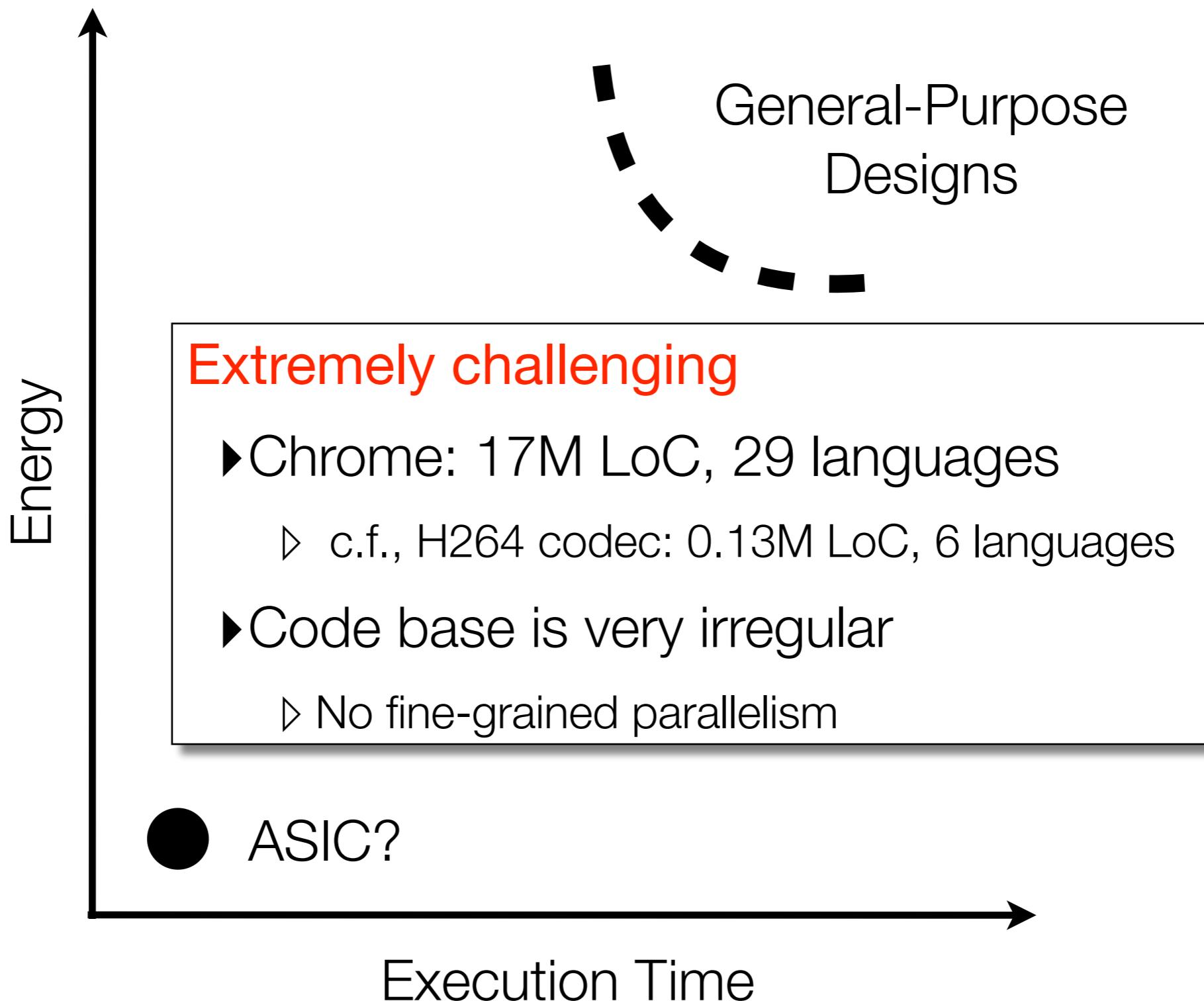
WebCore: a Web-Specific Mobile Architecture



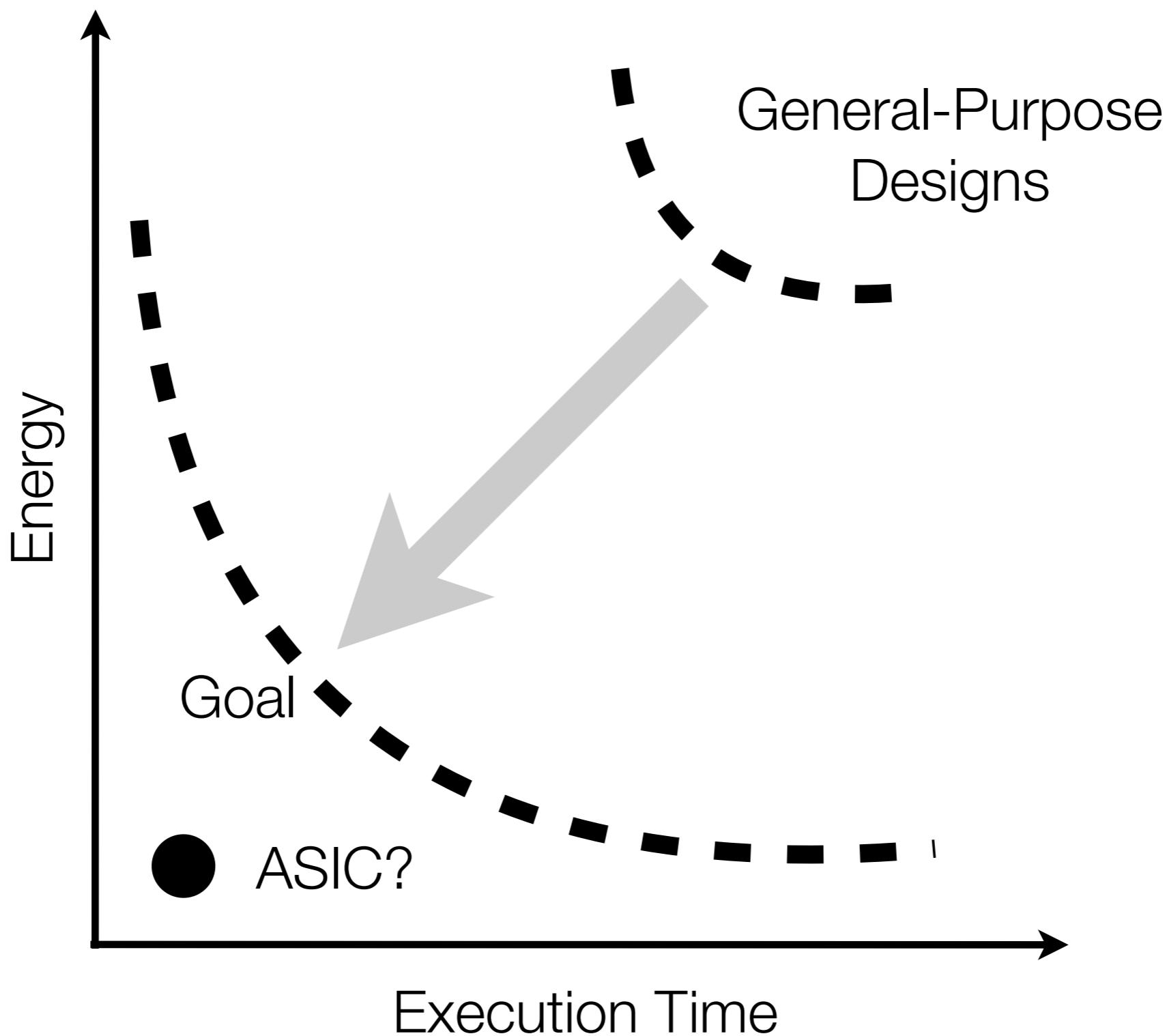
WebCore: a Web-Specific Mobile Architecture



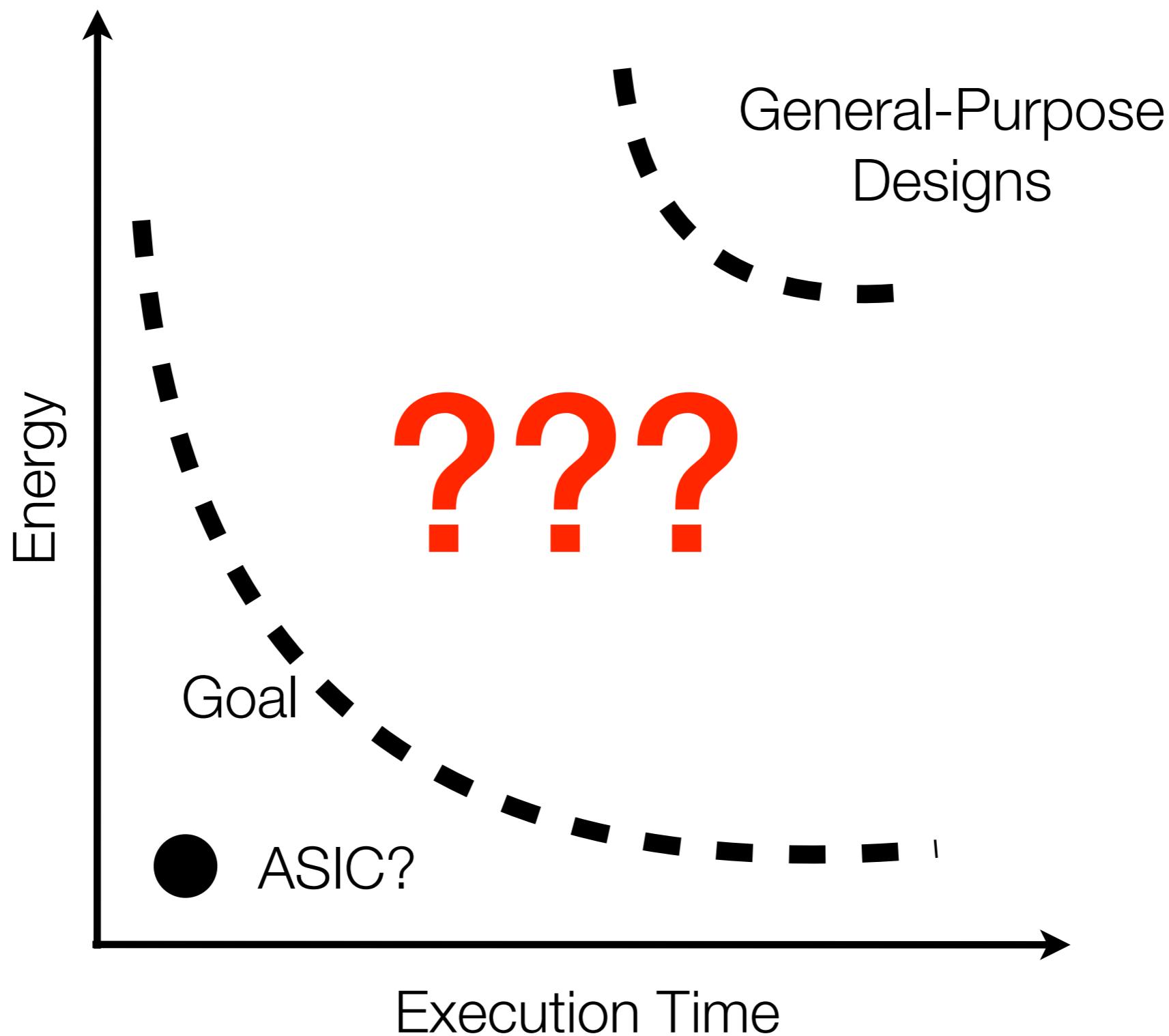
WebCore: a Web-Specific Mobile Architecture



WebCore: a Web-Specific Mobile Architecture



WebCore: a Web-Specific Mobile Architecture



WebCore Philosophy

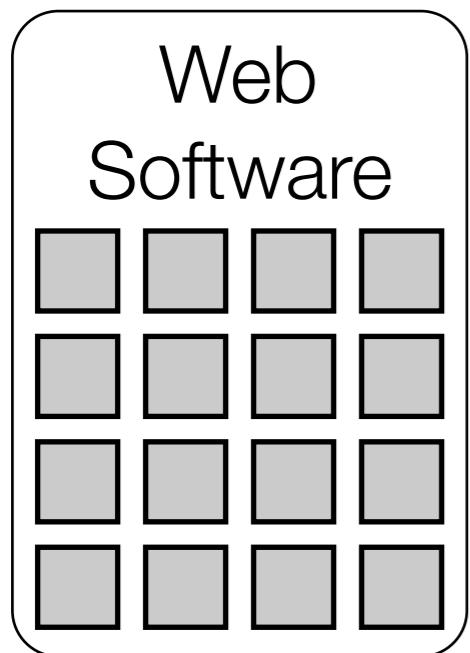
Claim: Instead of directly jumping to fully specialization, we must take it step by step



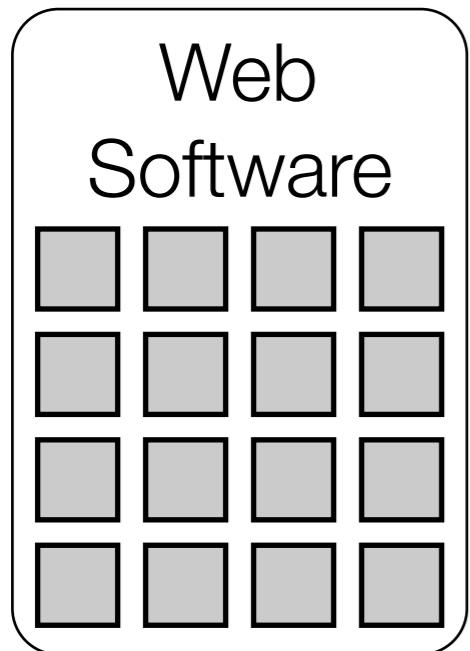
WebCore Philosophy



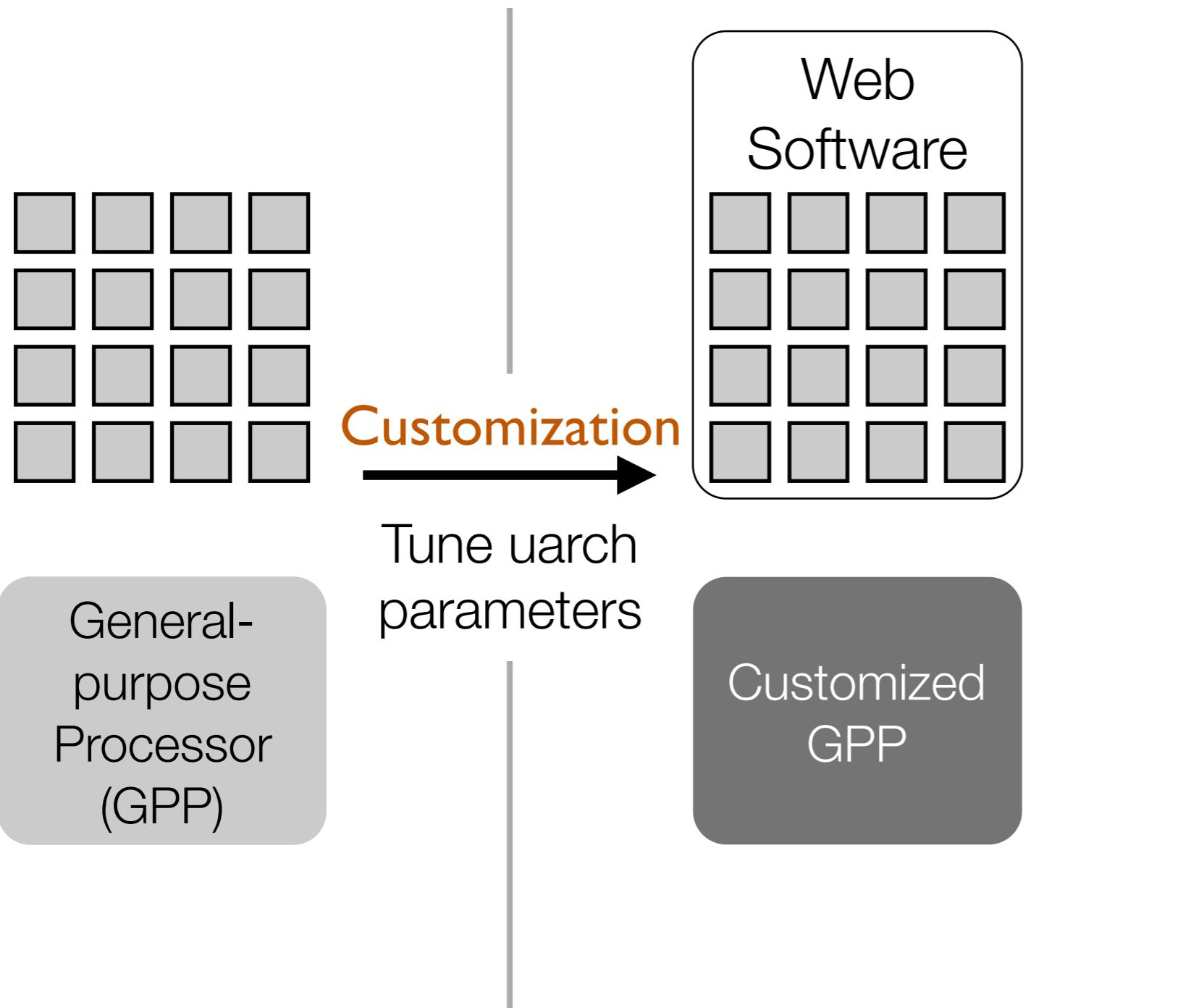
WebCore Philosophy



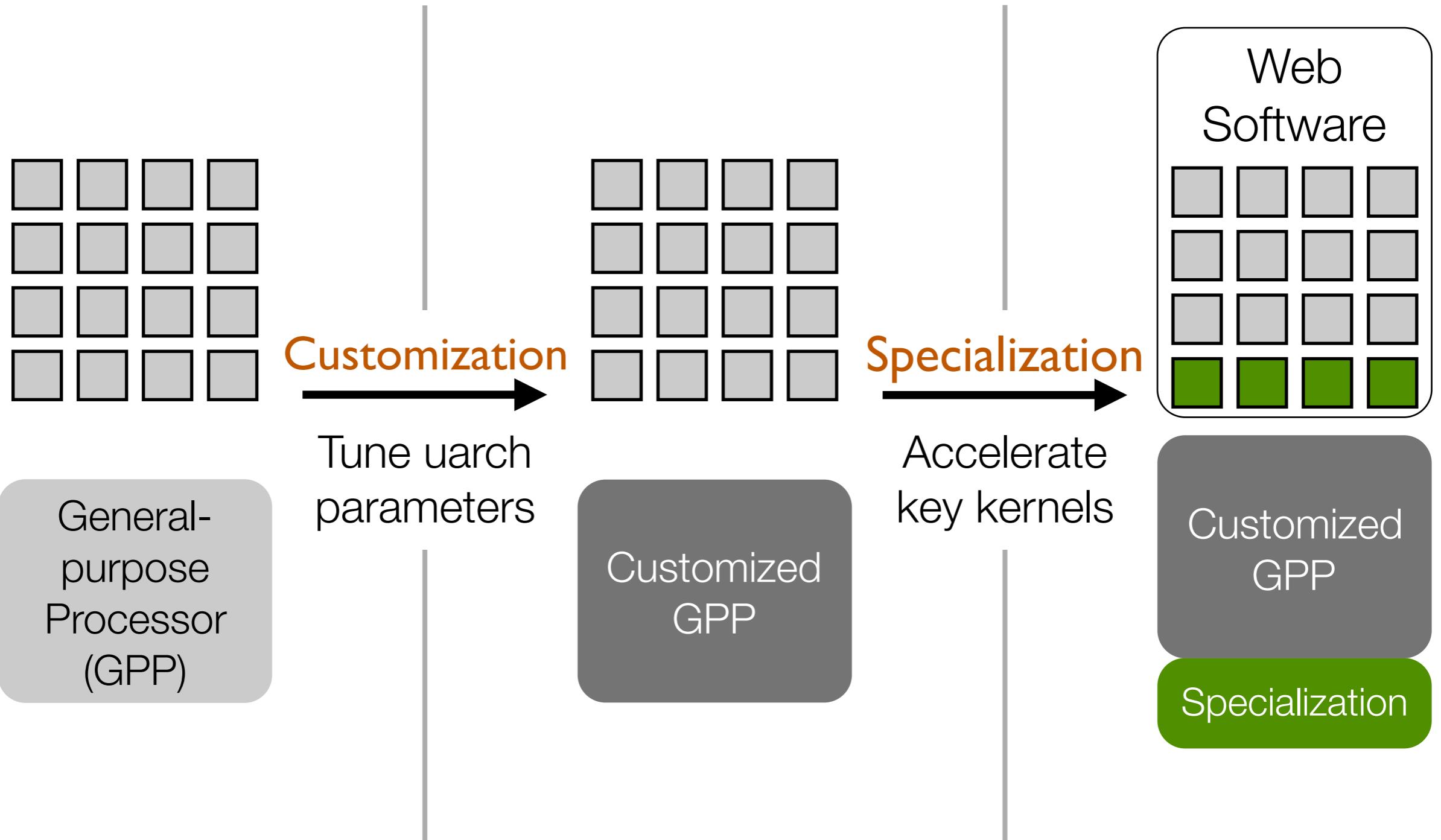
WebCore Philosophy



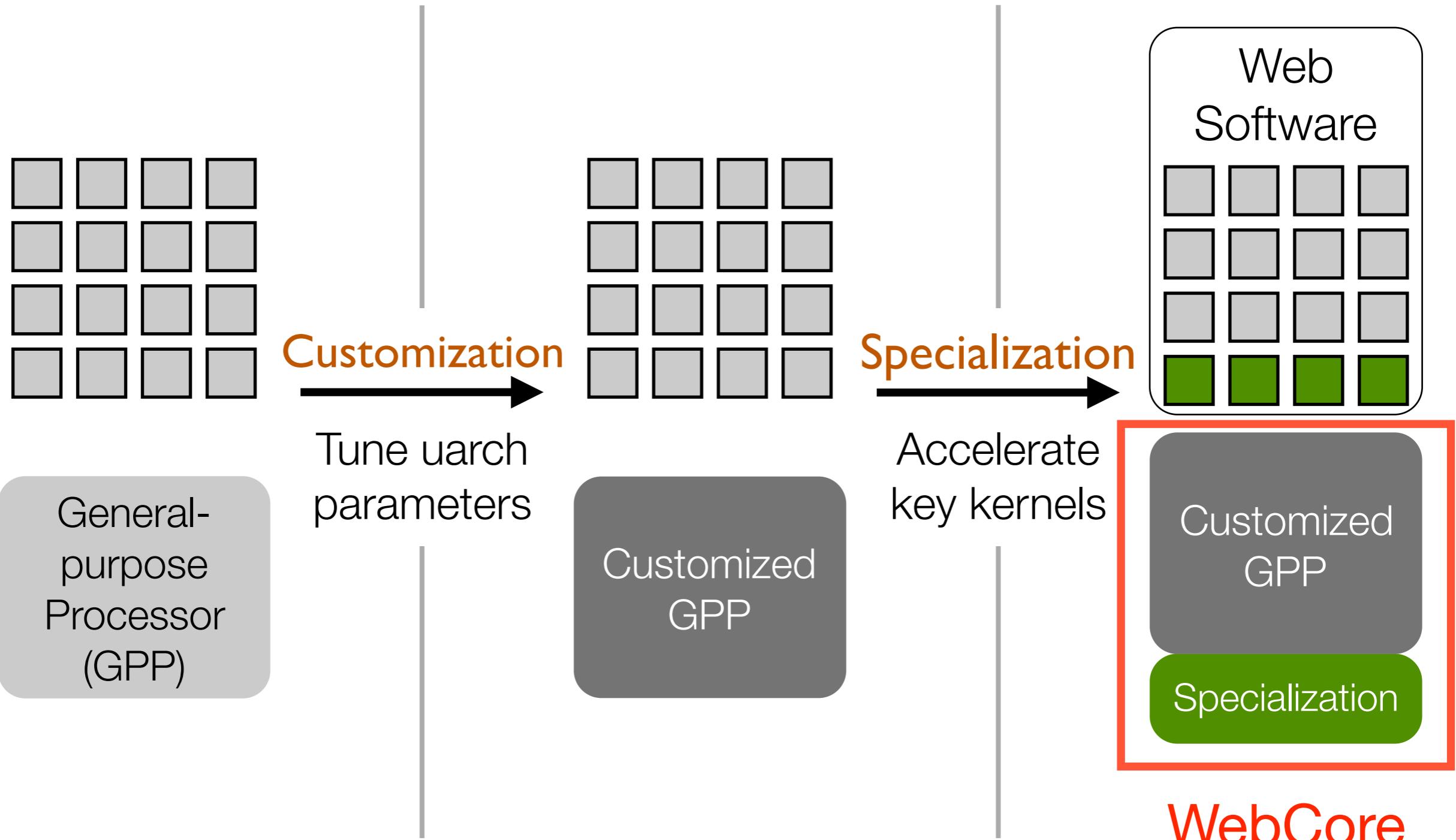
WebCore Philosophy



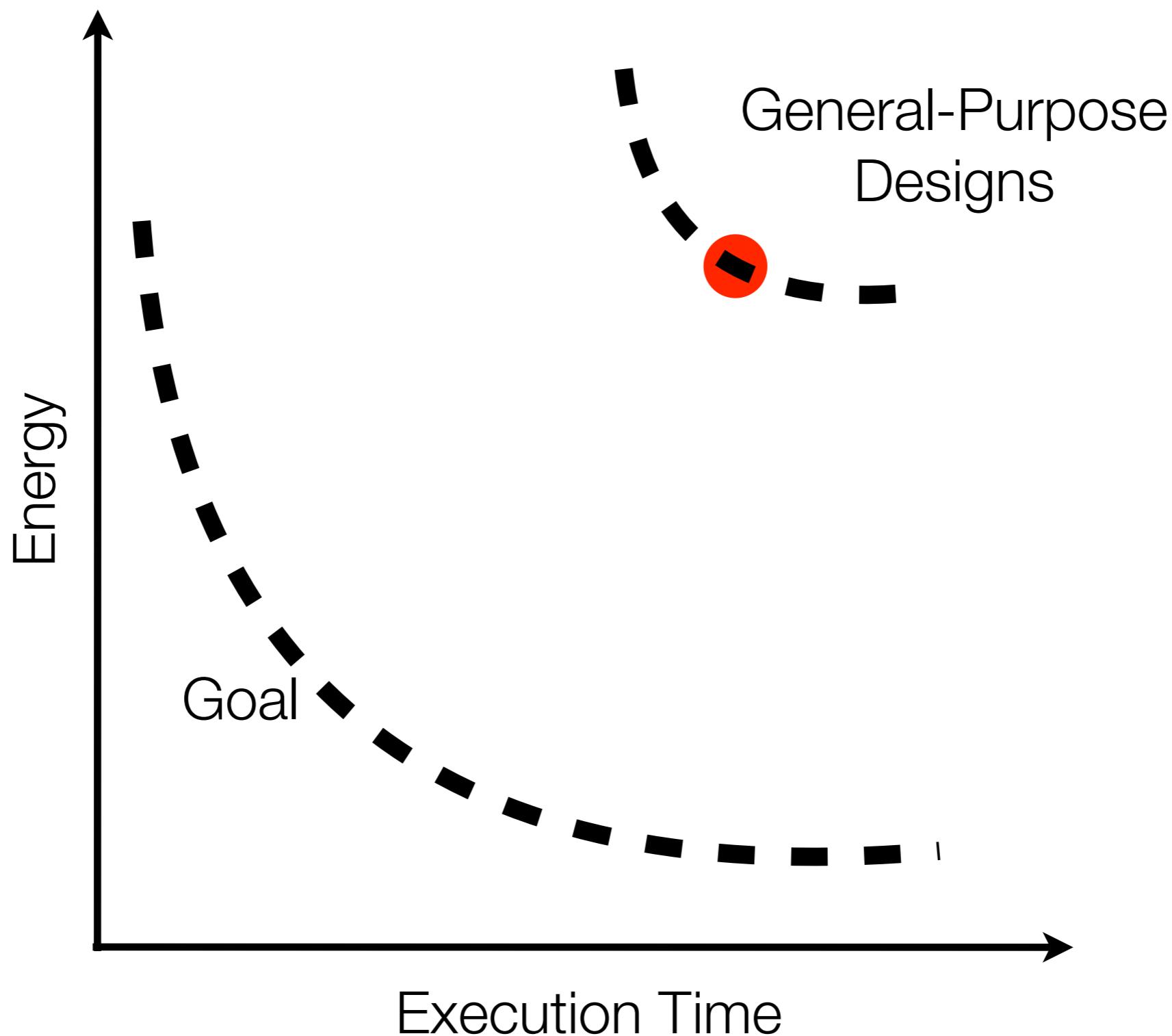
WebCore Philosophy



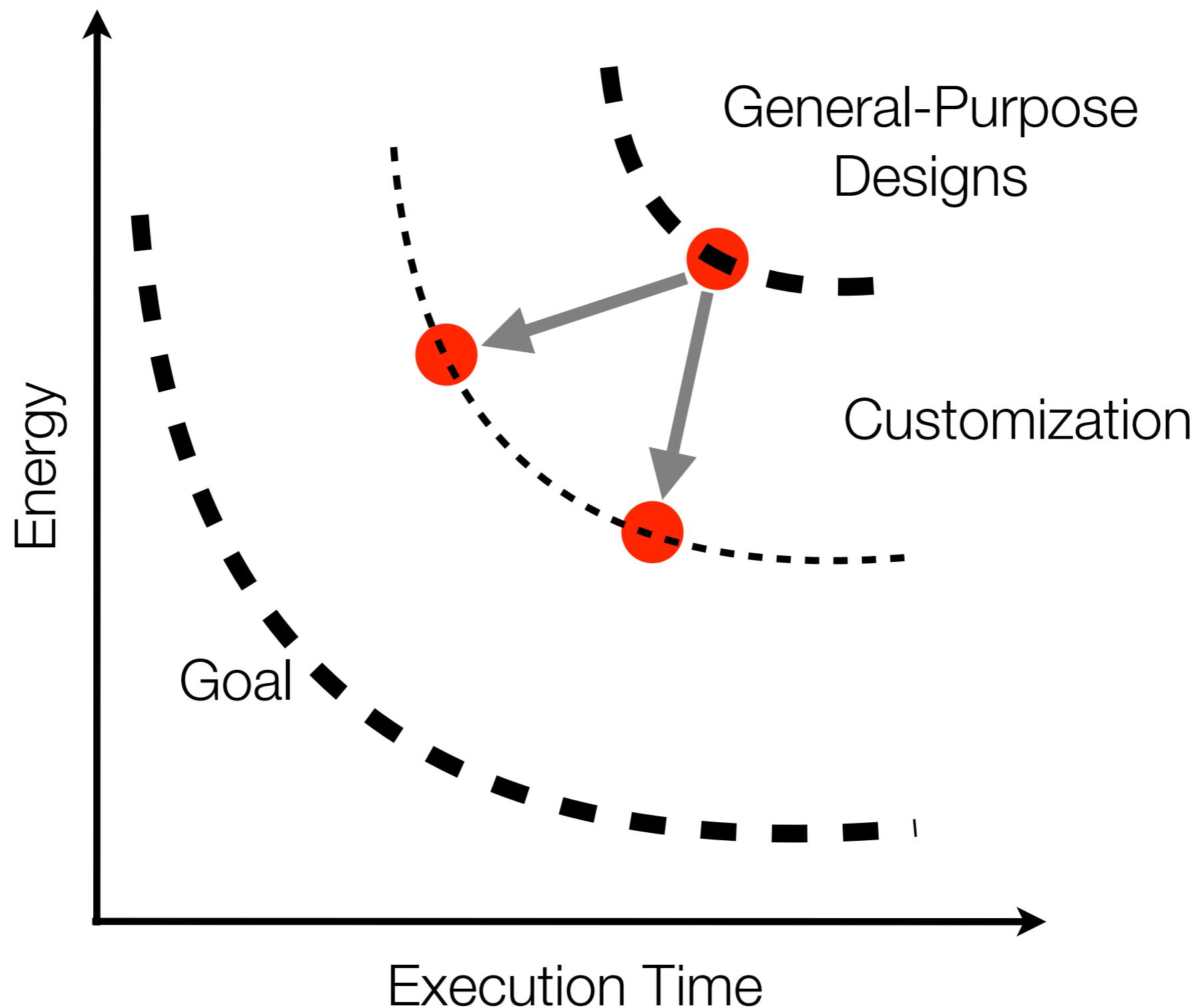
WebCore Philosophy



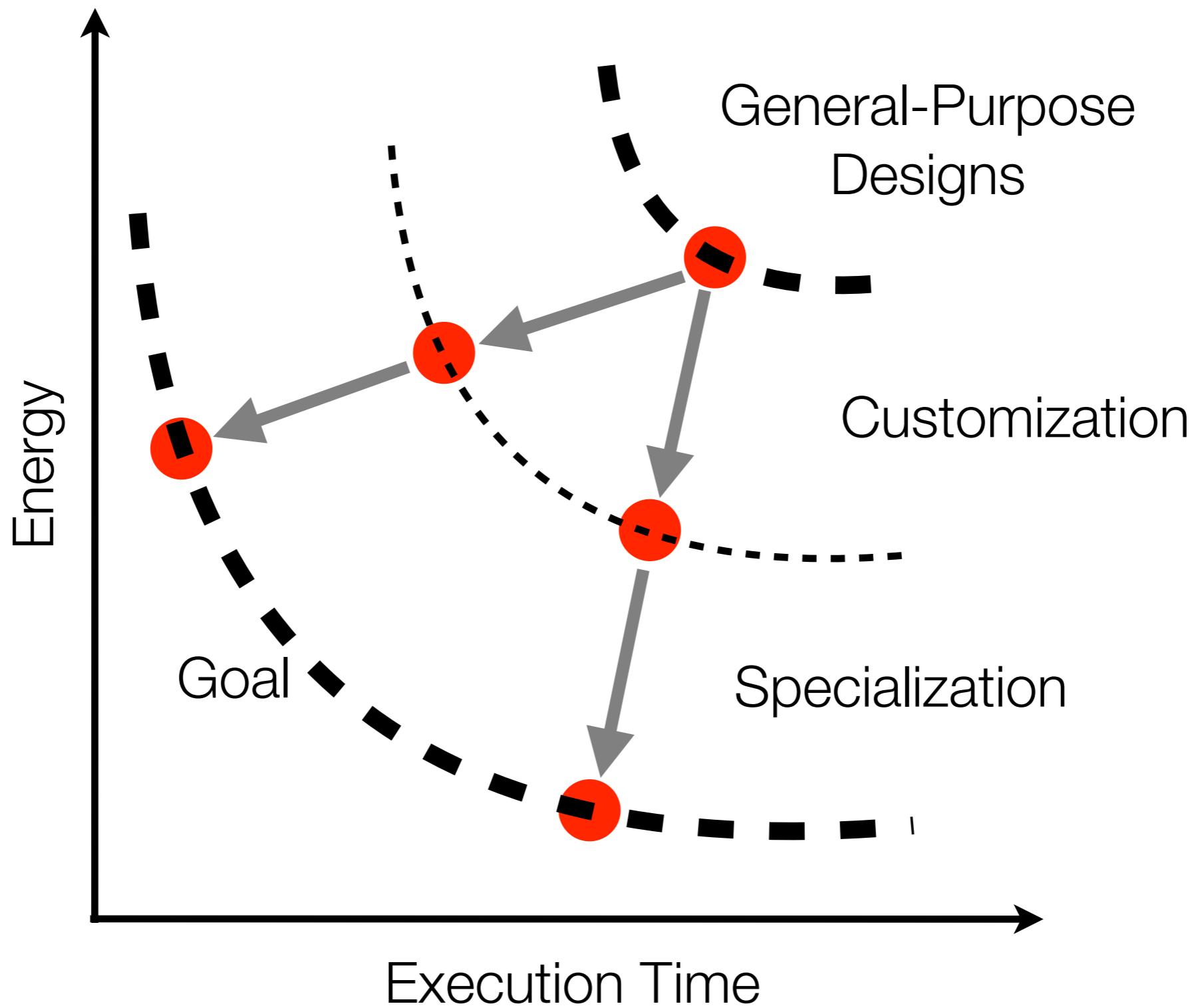
WebCore: a Web-Specific Mobile Architecture



WebCore: a Web-Specific Mobile Architecture



WebCore: a Web-Specific Mobile Architecture



Customization: Find an Ideal General Purpose Architecture for the Mobile Web



Customization: Find an Ideal General Purpose Architecture for the Mobile Web

- ▶ What is a proper general purpose baseline architecture?
 - ▷ Out-of-order (Silvermont, A15) or in-order (Saltwell, A7)?
 - ▷ Are existing general purpose mobile designs ideal?



Customization: Find an Ideal General Purpose Architecture for the Mobile Web

- ▶ What is a proper general purpose baseline architecture?
 - ▷ Out-of-order (Silvermont, A15) or in-order (Saltwell, A7)?
 - ▷ Are existing general purpose mobile designs ideal?
- ▶ Exhaustive design space exploration.



Customization: Find an Ideal General Purpose Architecture for the Mobile Web

► What is a proper

- ▷ Out-of-order (Silvermont)
- ▷ Are existing general purpose architectures good?

► Exhaustive design

Parameters	Measure	Range
Issue width	count	1,2,4
# Functional units	count	1::1::4
Load queue size	# entries	4::4::16
Store queue size	# entries	4::4::16
Branch prediction size	$\log_2(\# \text{entries})$	1::1::10
ROB size	# entries	8::8::128
# Physical registers	# entries	5::5::140
L1 I-cache size	$\log_2(\text{KB})$	3::1::8
L1 I-cache delay	cycles	1::1::3
L1 D-cache size	$\log_2(\text{KB})$	3::1::8
L1 D-cache delay	cycles	1::1::3
L2 cache size	$\log_2(\text{KB})$	7::1::10
L2 cache delay	cycles	16,32,64

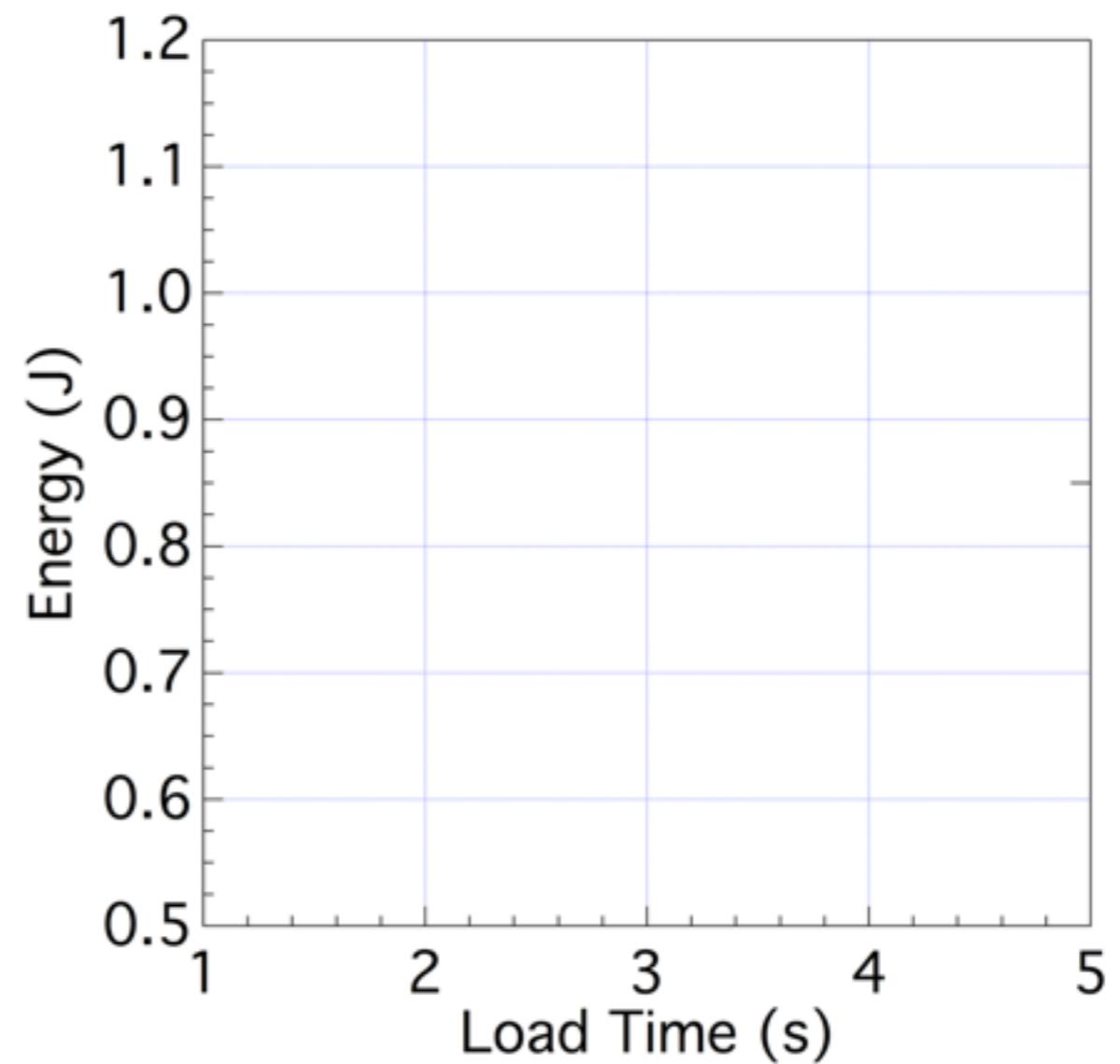


Design Space Exploration (DSE) Setup

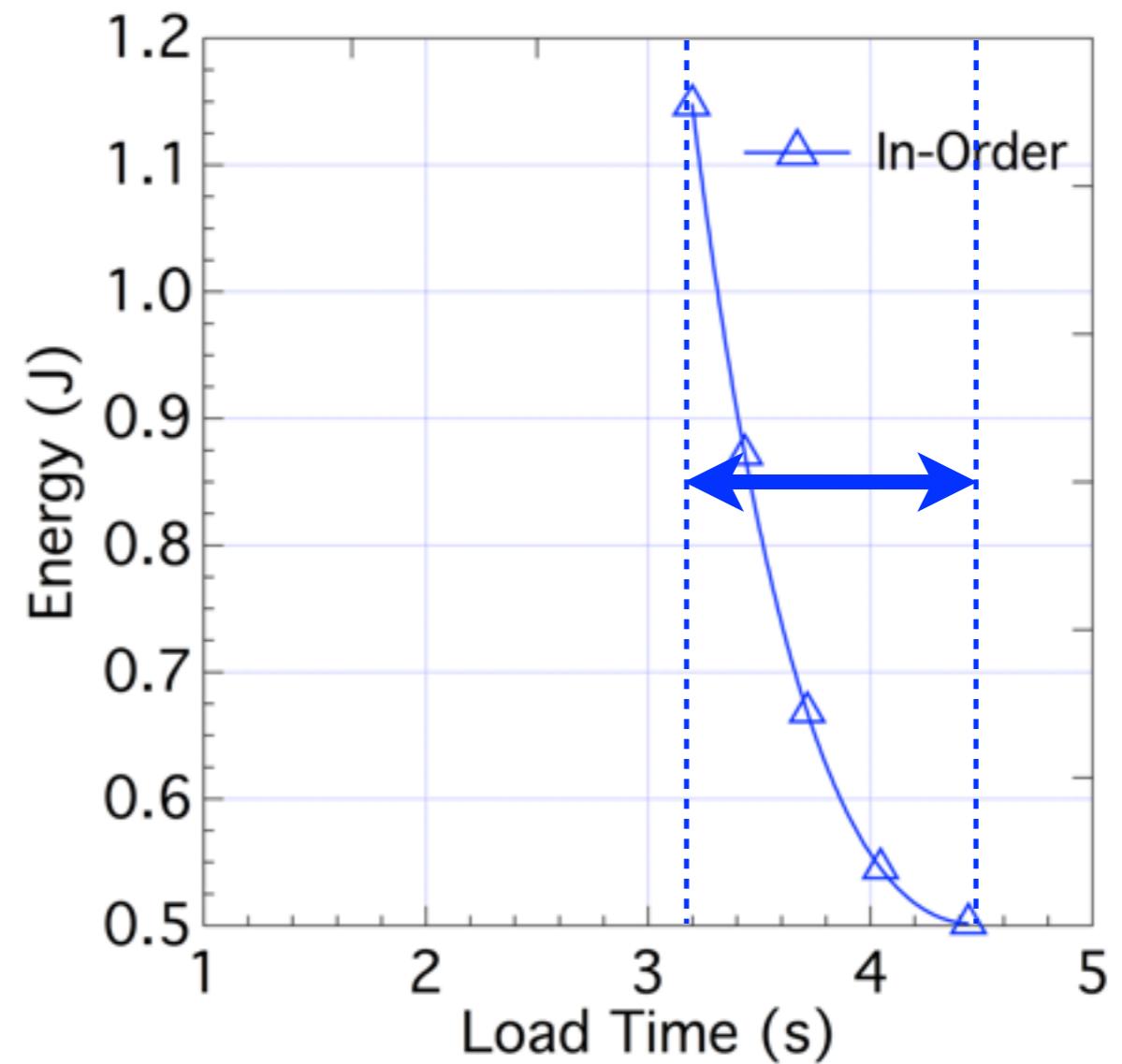
- ▶ Search space of over **3 billion design points**
 - ▷ Leverage statistical inference models to increase search speed
- ▶ Use integrated simulators
 - ▷ McPAT for Power
 - ▷ Marss86 for Performance (x86 full-system simulator)
- ▶ Chromium Web browser



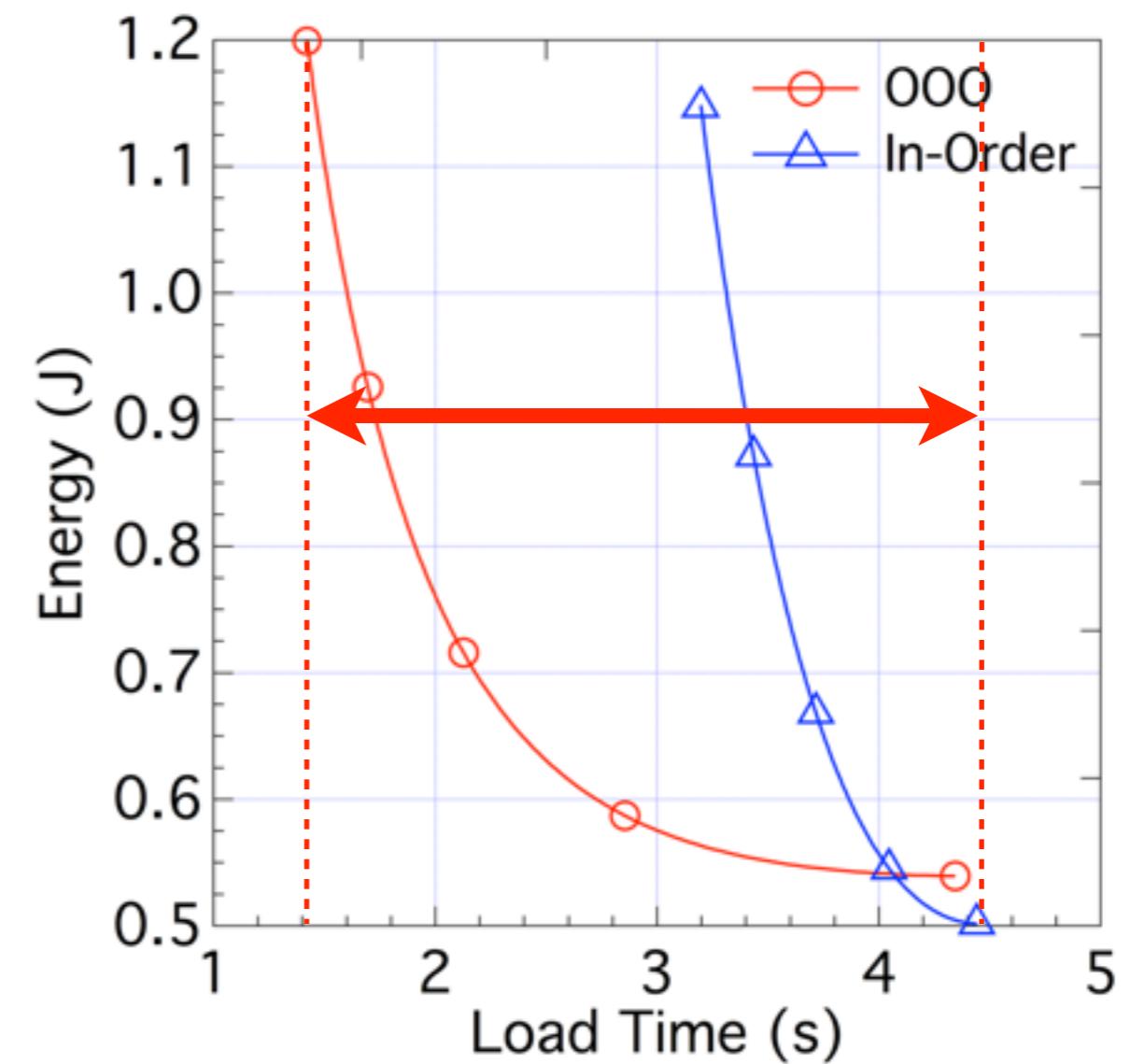
Design Space Exploration (DSE) Findings



Design Space Exploration (DSE) Findings

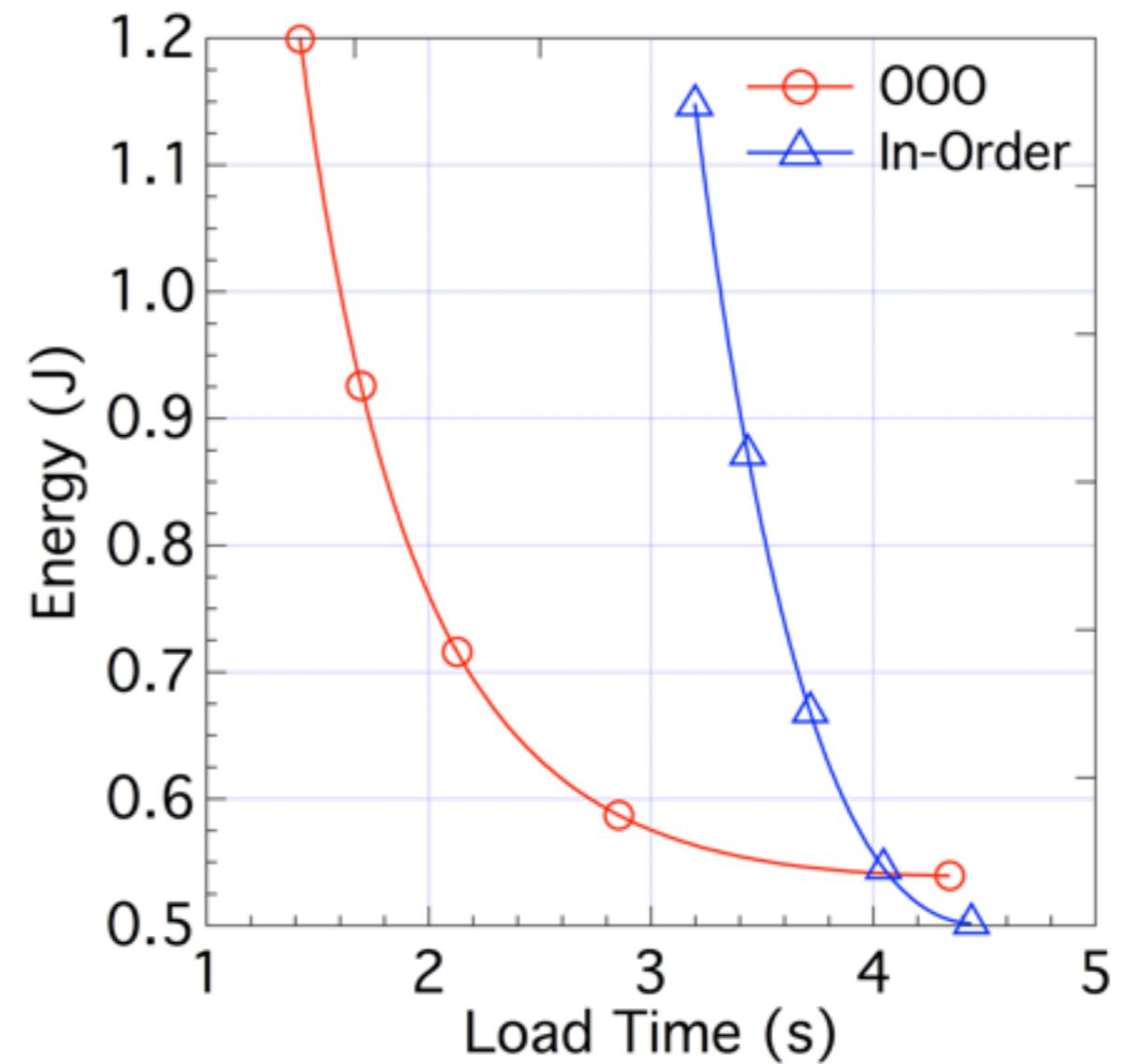


Design Space Exploration (DSE) Findings



Design Space Exploration (DSE) Findings

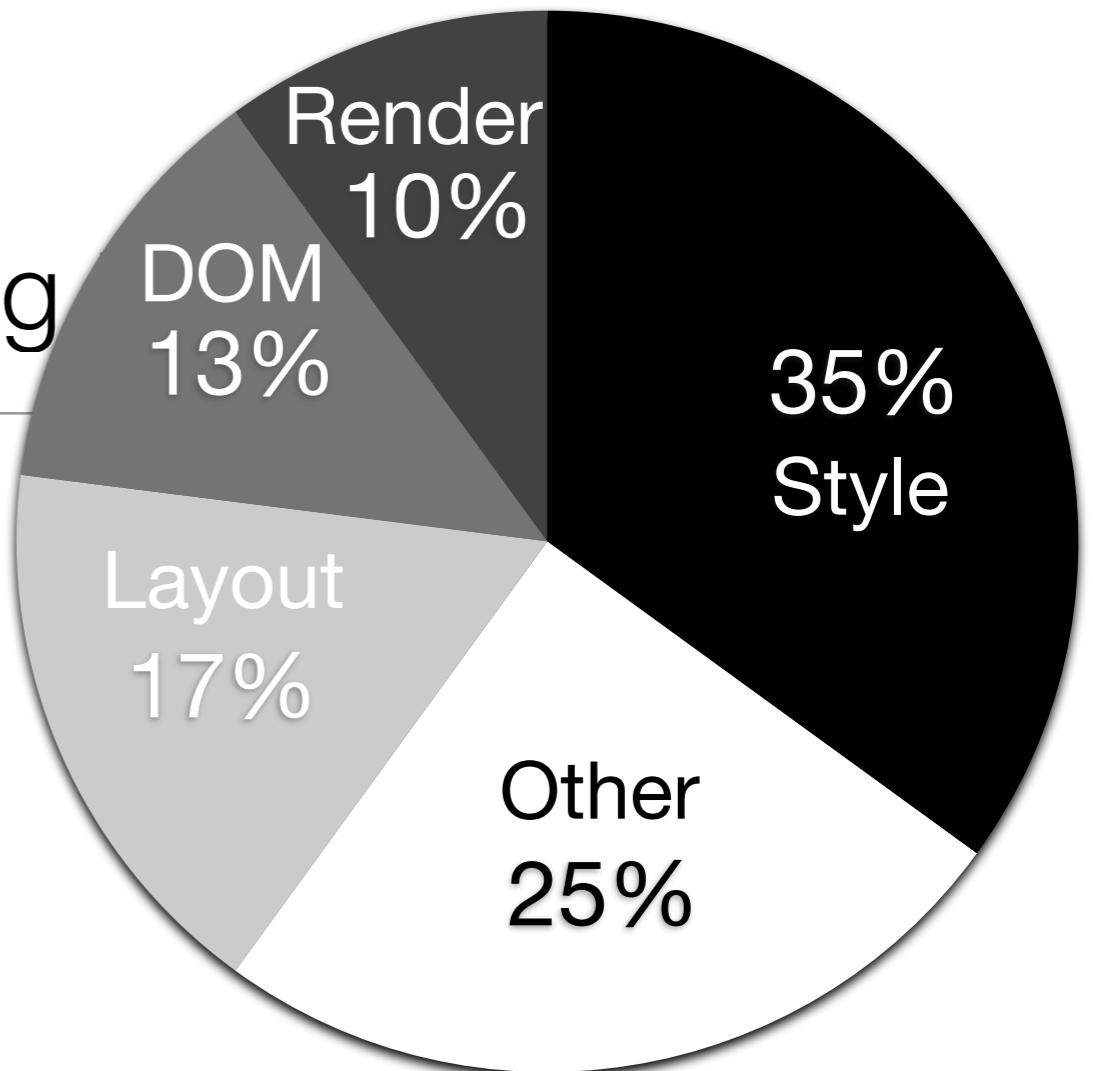
- ▶ Out-of-order designs are more flexible



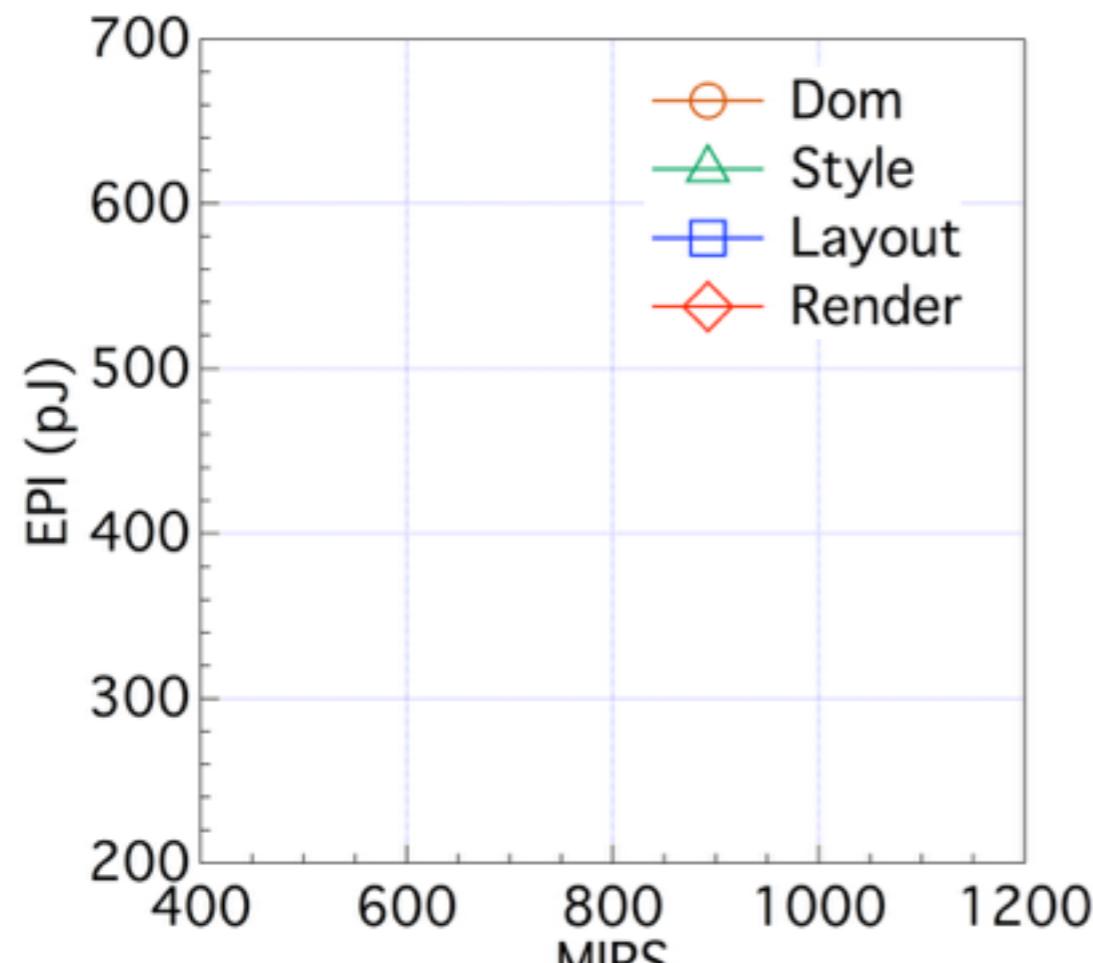
Understand the Difference Using **Kernel** Knowledge



Understand the Difference Using



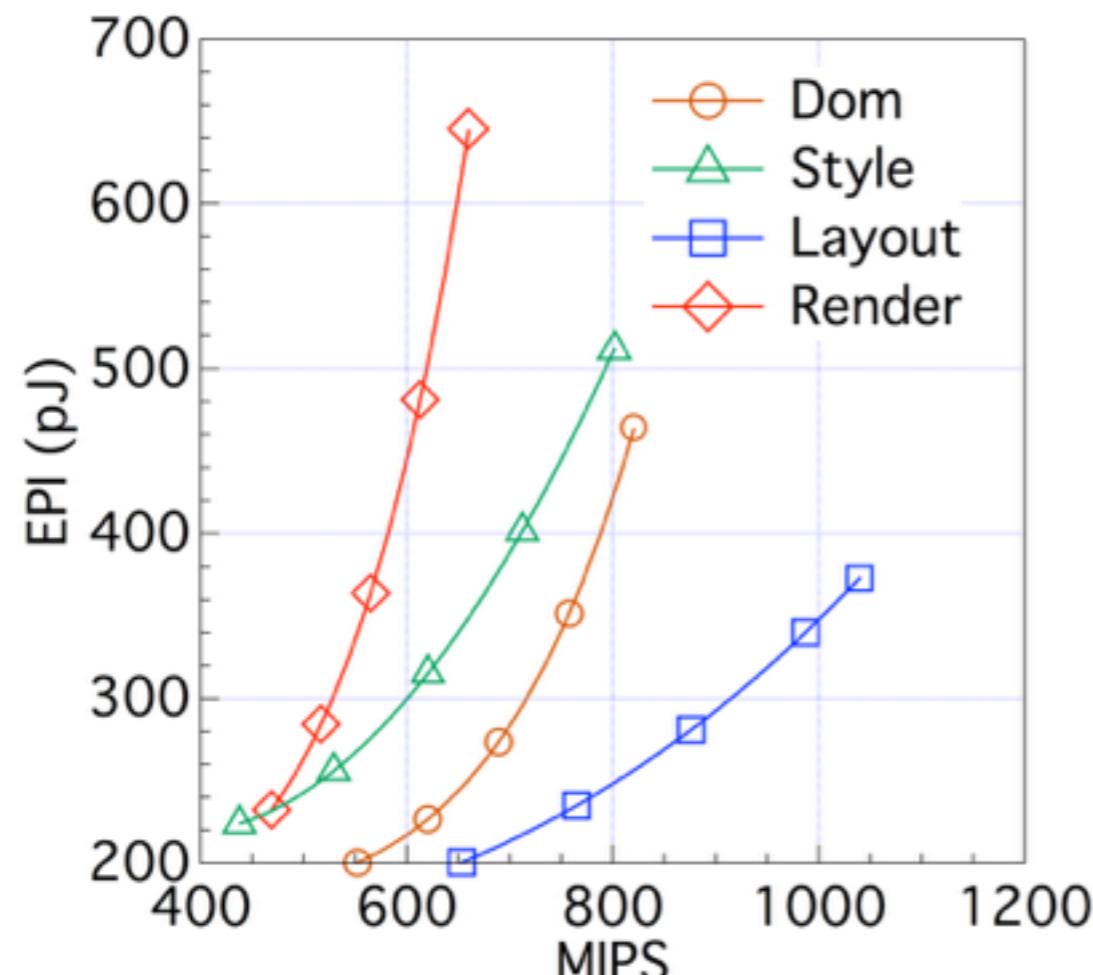
Understand the Difference Using **Kernel** Knowledge



In-order design



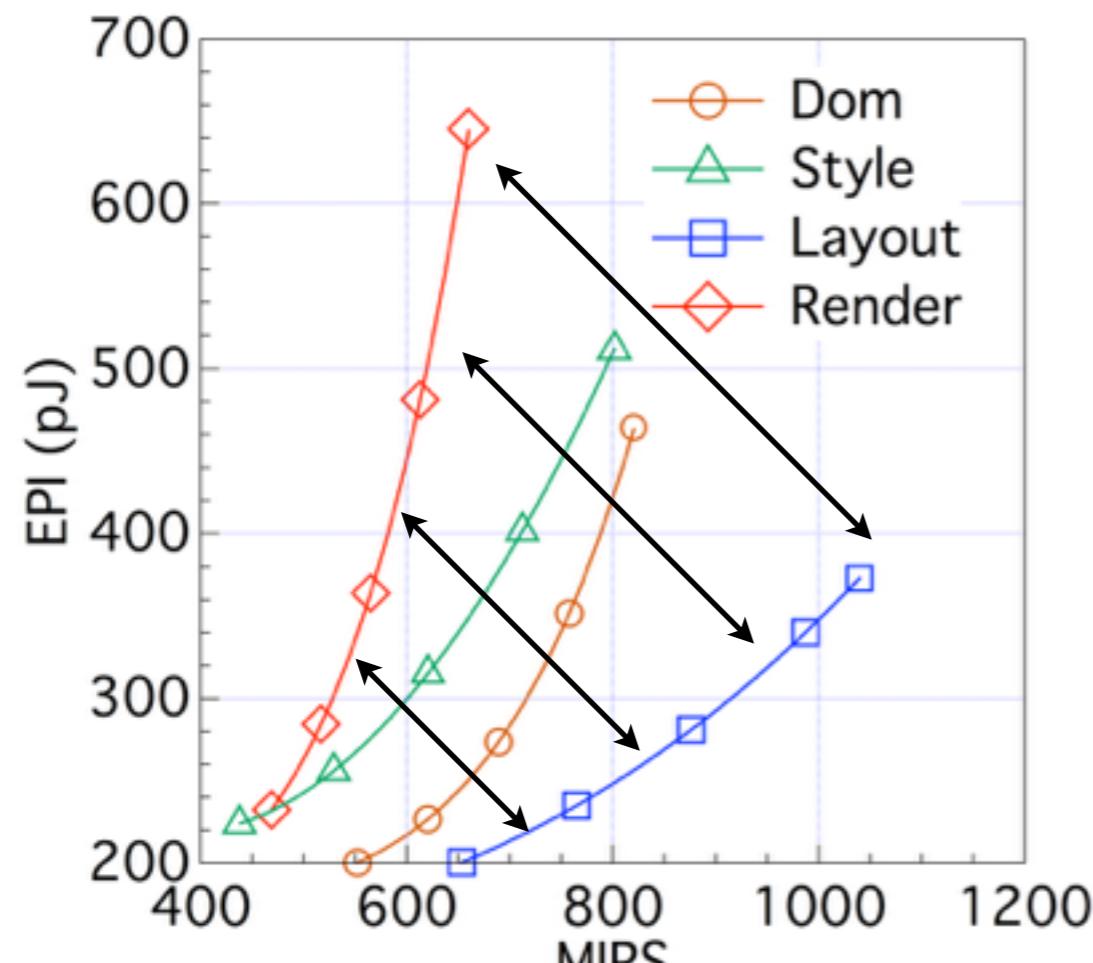
Understand the Difference Using **Kernel** Knowledge



In-order design

Understand the Difference Using **Kernel** Knowledge

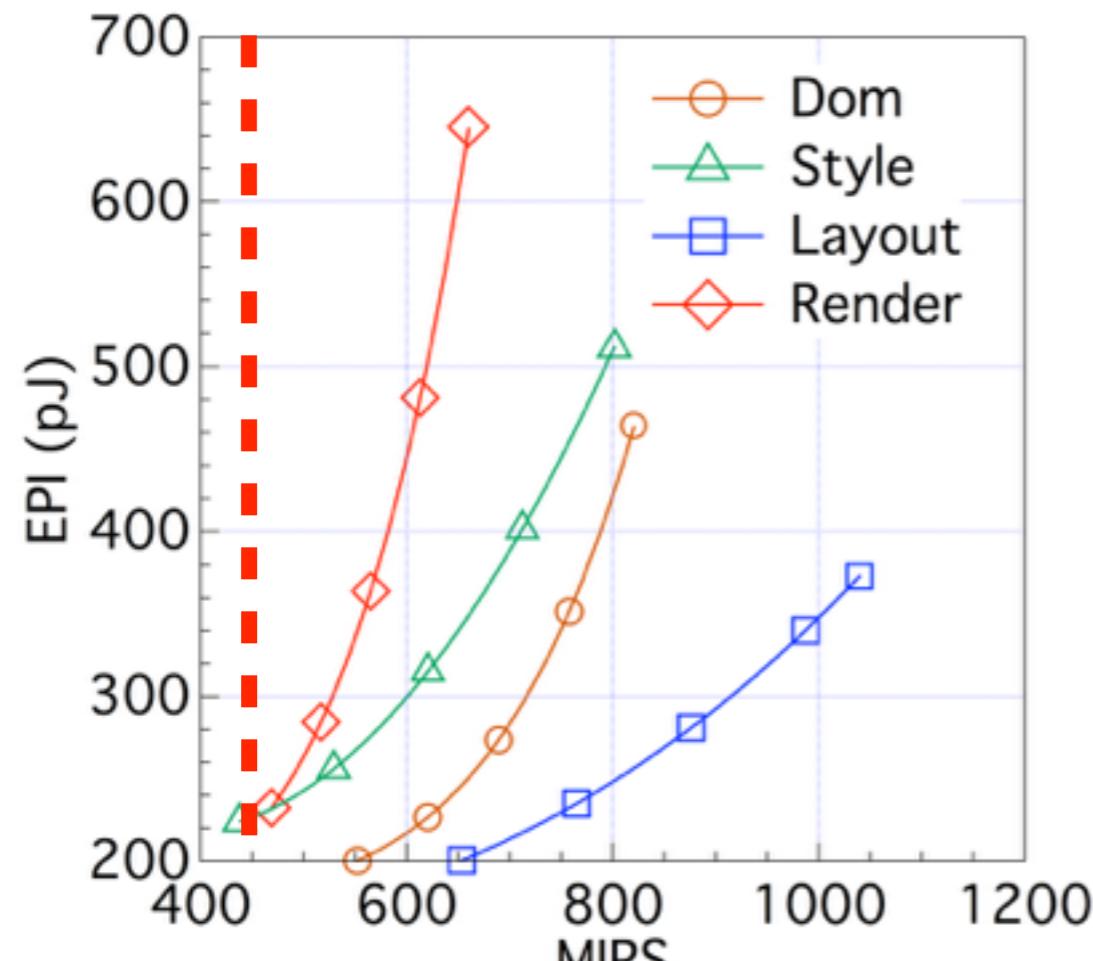
- ▶ In-order designs show strong **kernel variance**



In-order design

Understand the Difference Using **Kernel** Knowledge

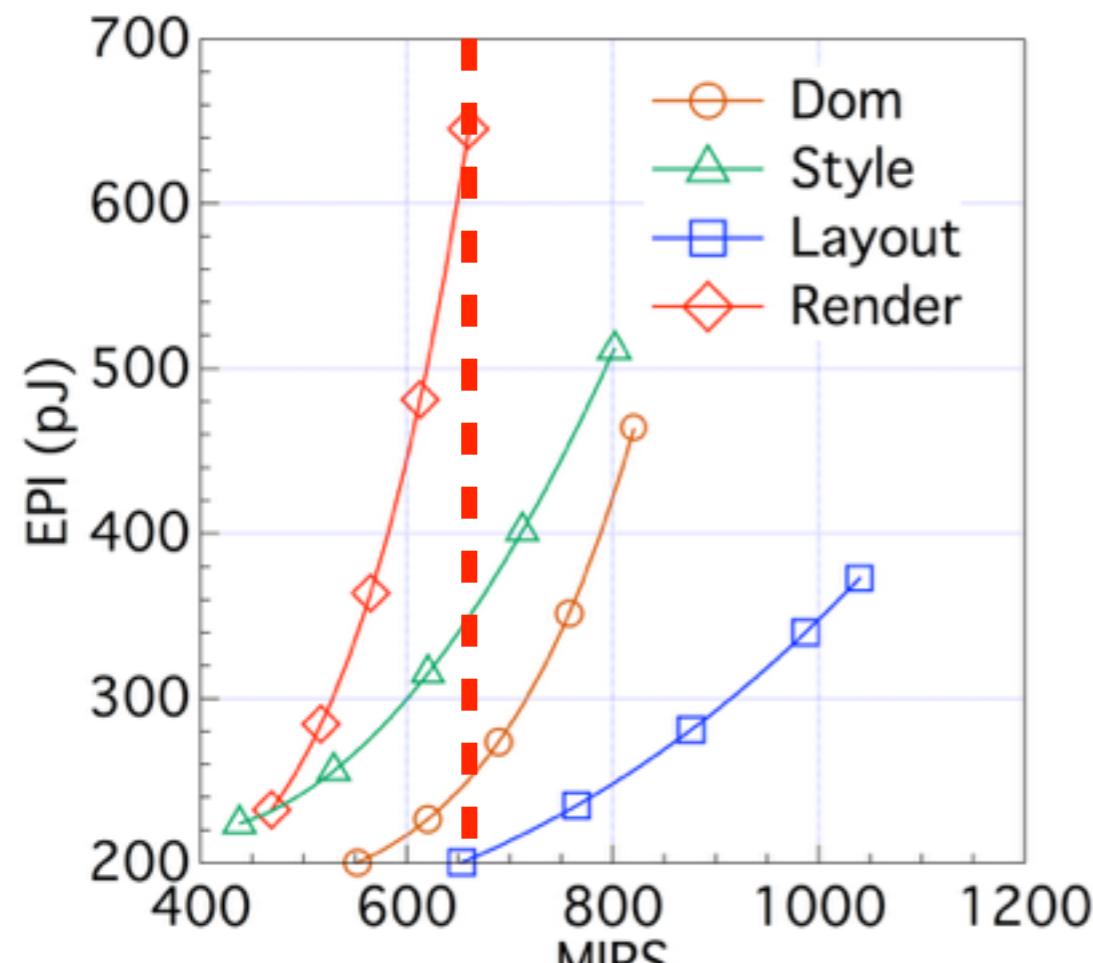
- In-order designs show strong **kernel variance**



In-order design

Understand the Difference Using **Kernel** Knowledge

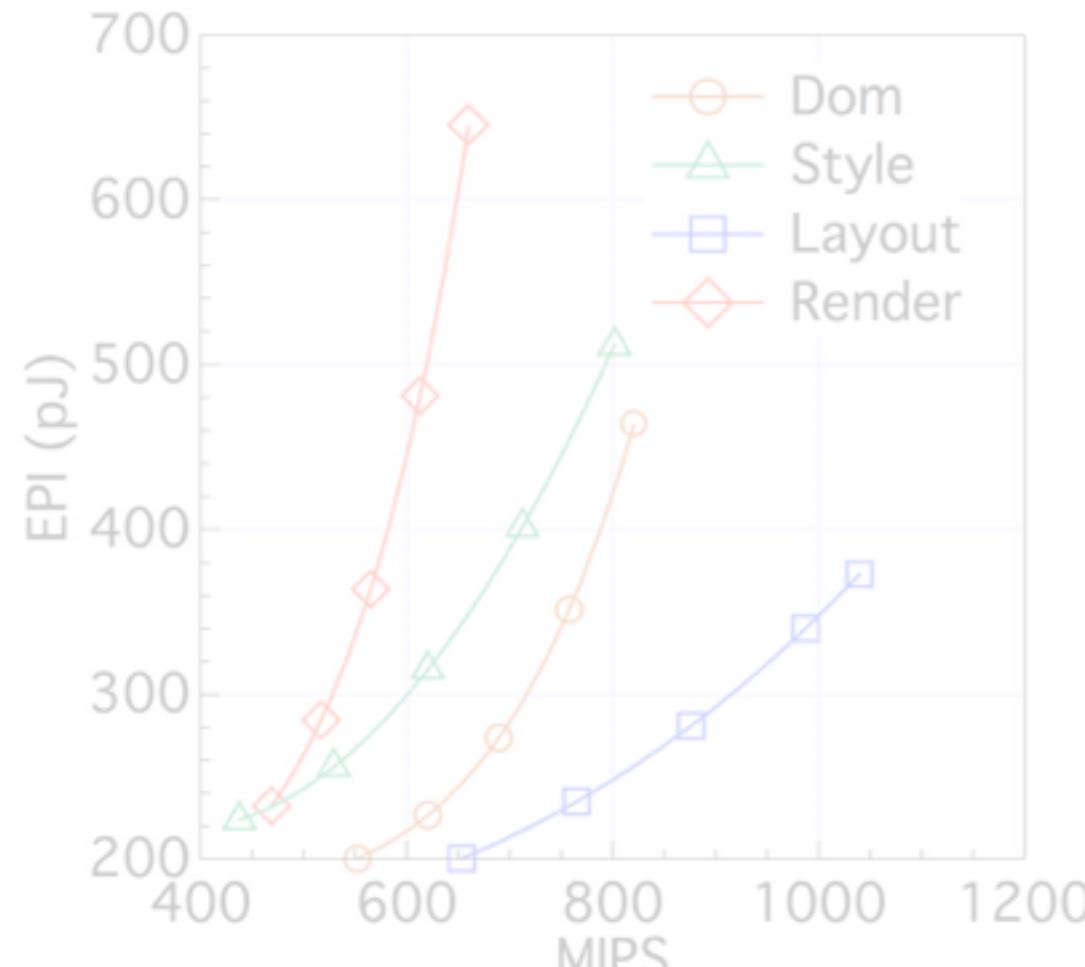
- ▶ In-order designs show strong **kernel variance**



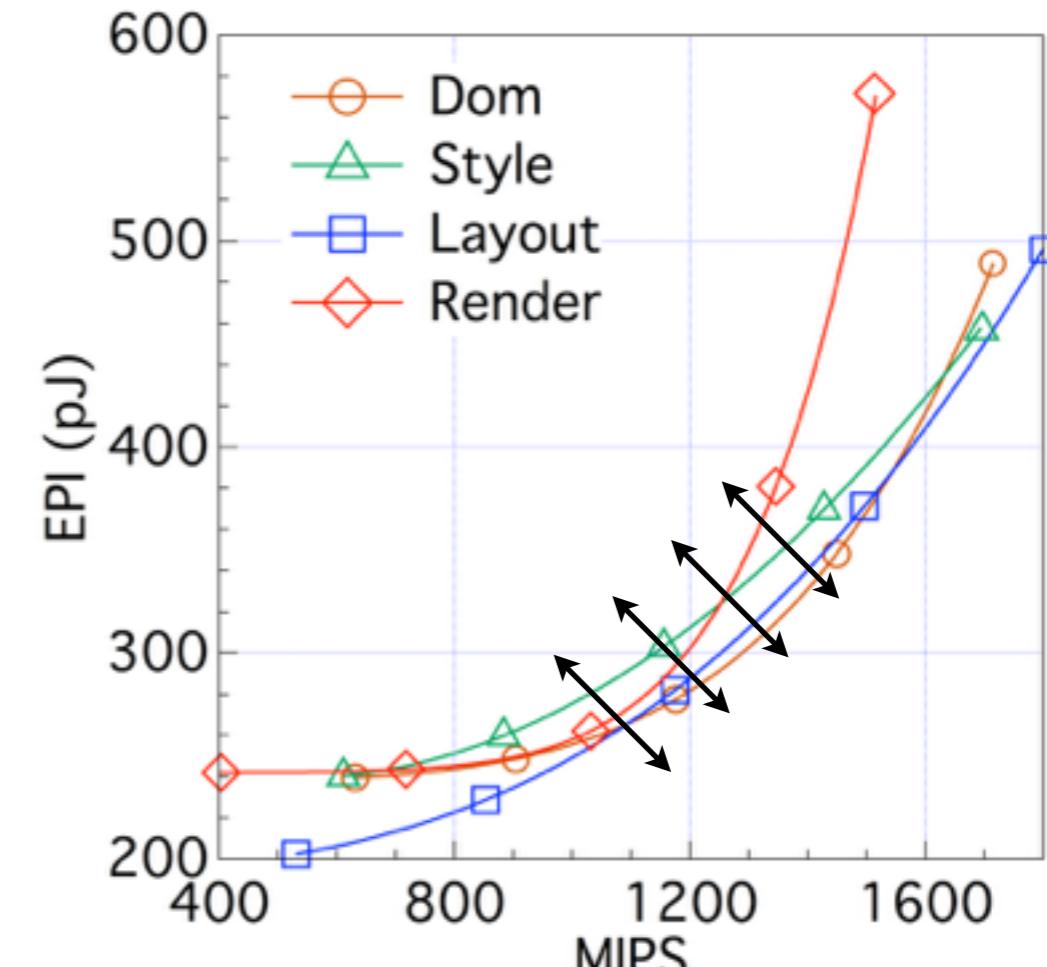
In-order design

Understand the Difference Using **Kernel** Knowledge

- In-order designs show strong **kernel variance**



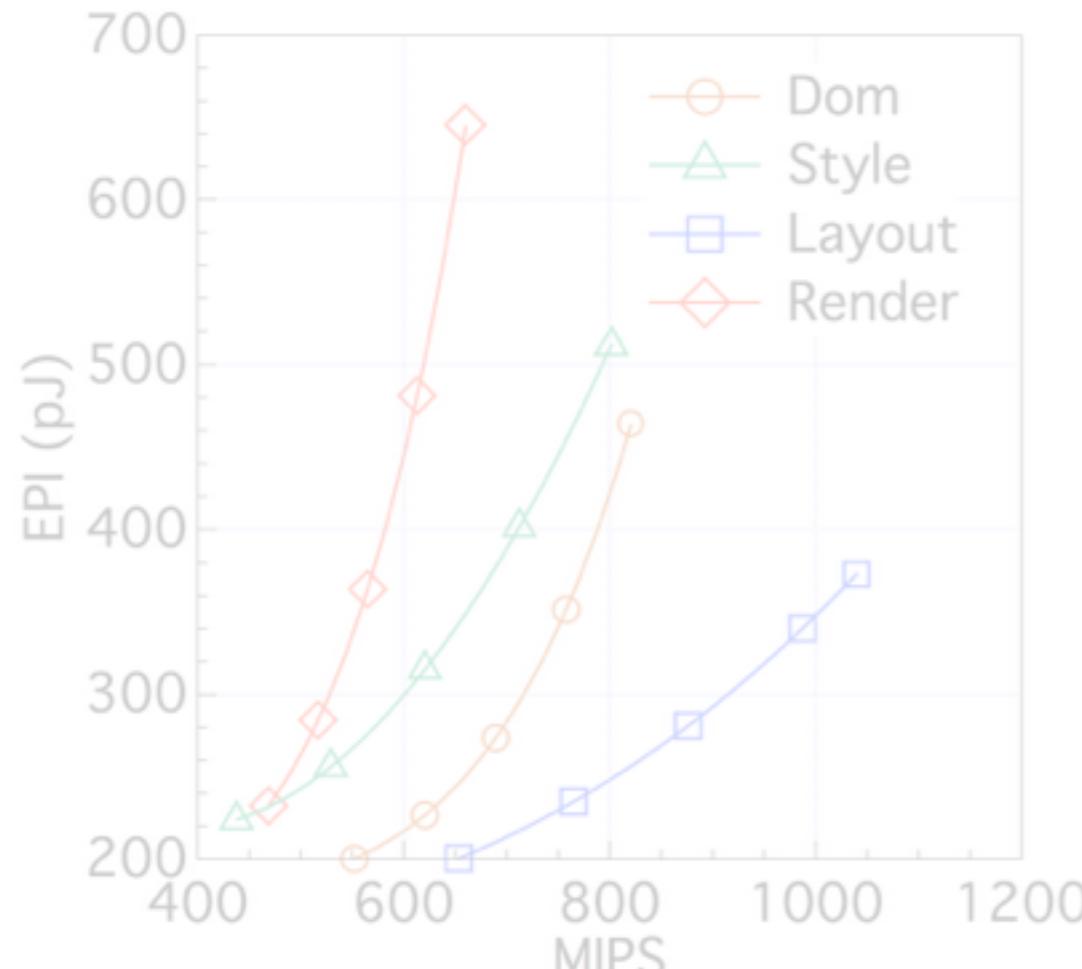
In-order design



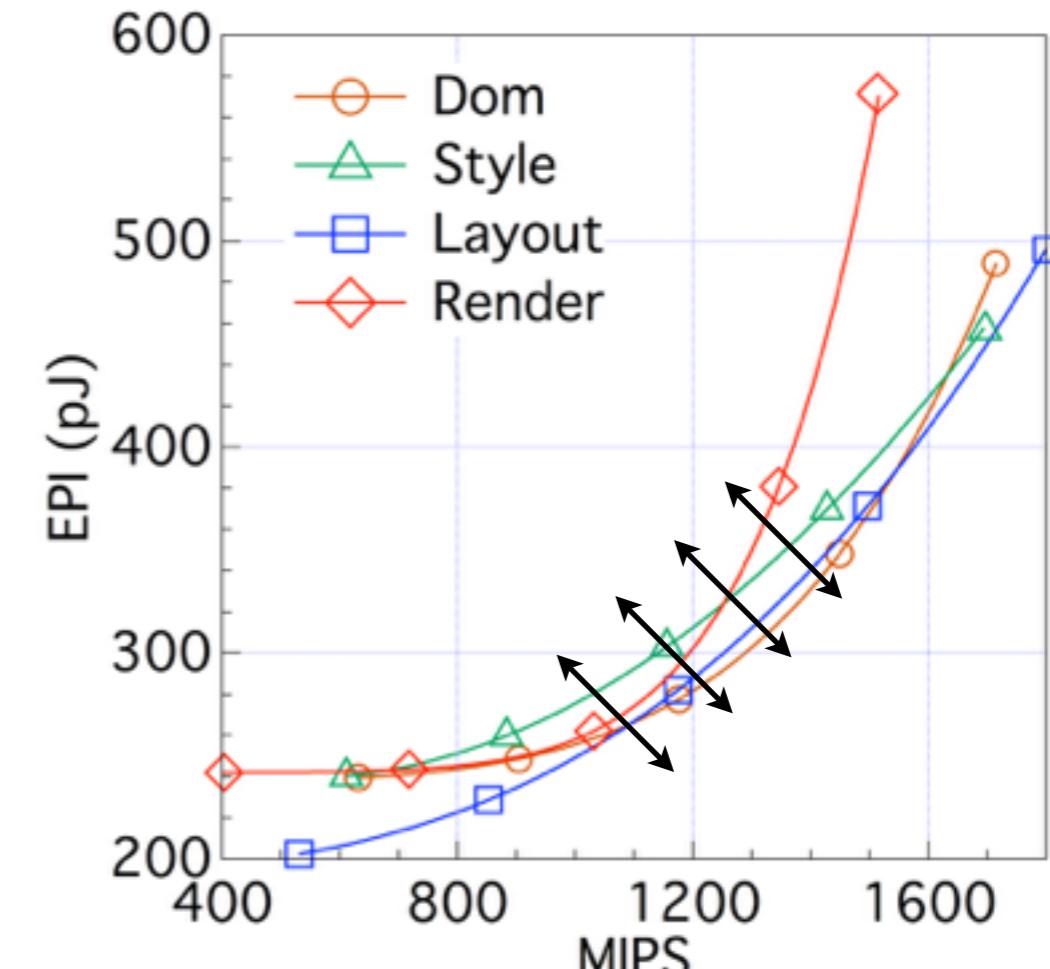
Out-of-order design

Understand the Difference Using **Kernel** Knowledge

- ▶ In-order designs show strong **kernel variance**
- ▶ An Out-of-order design can **accommodate** kernel variance

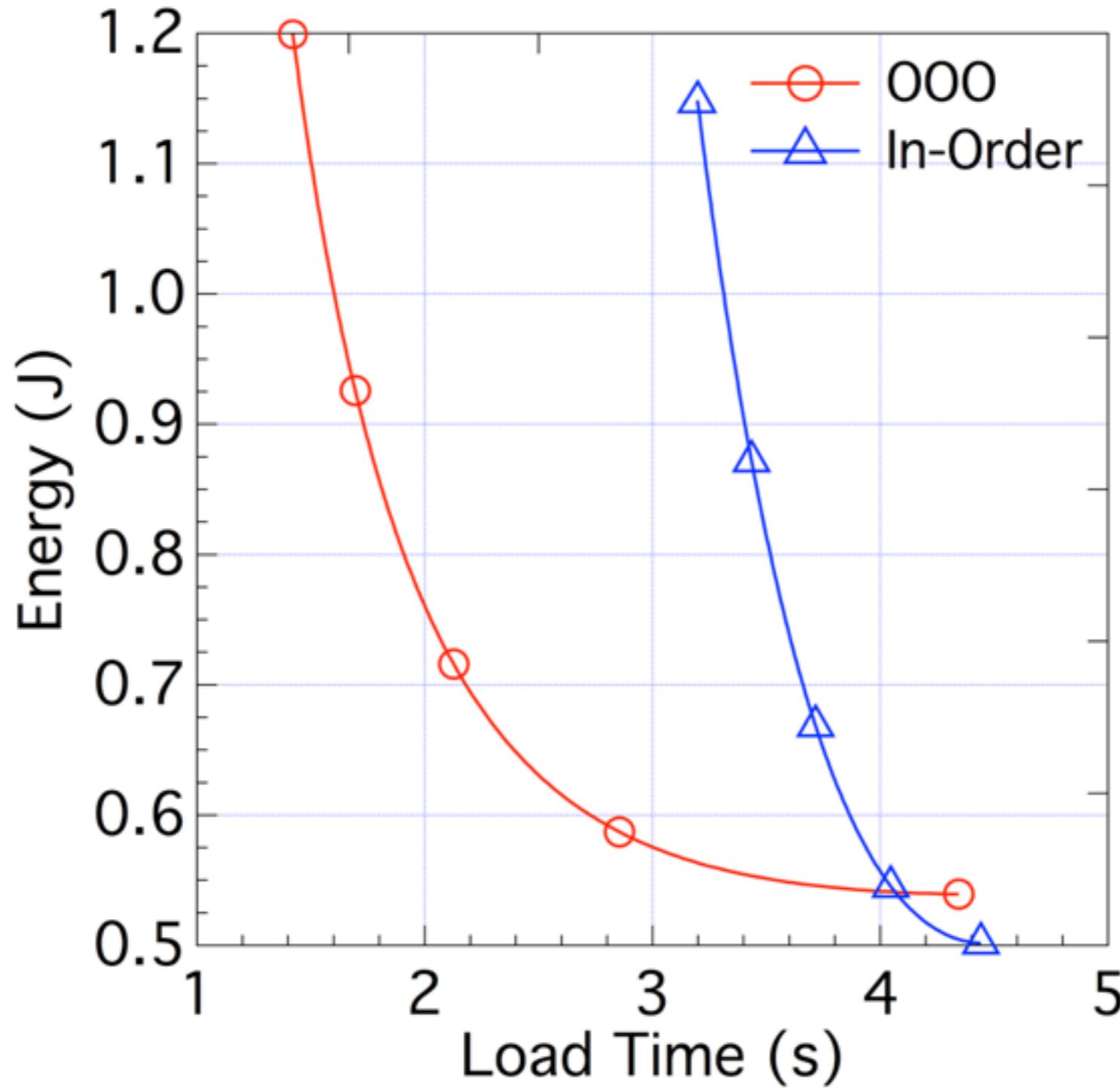


In-order design

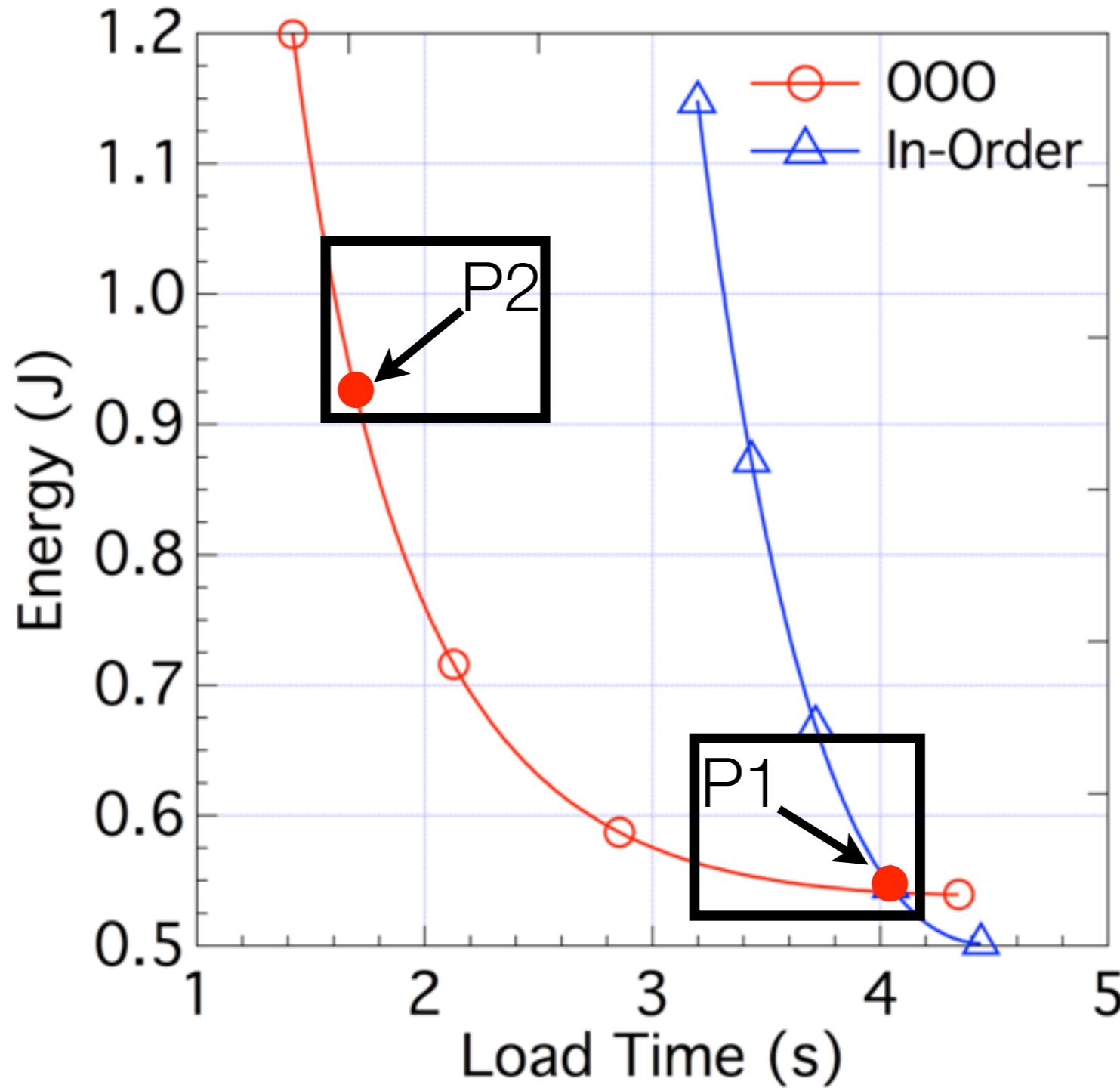


Out-of-order design

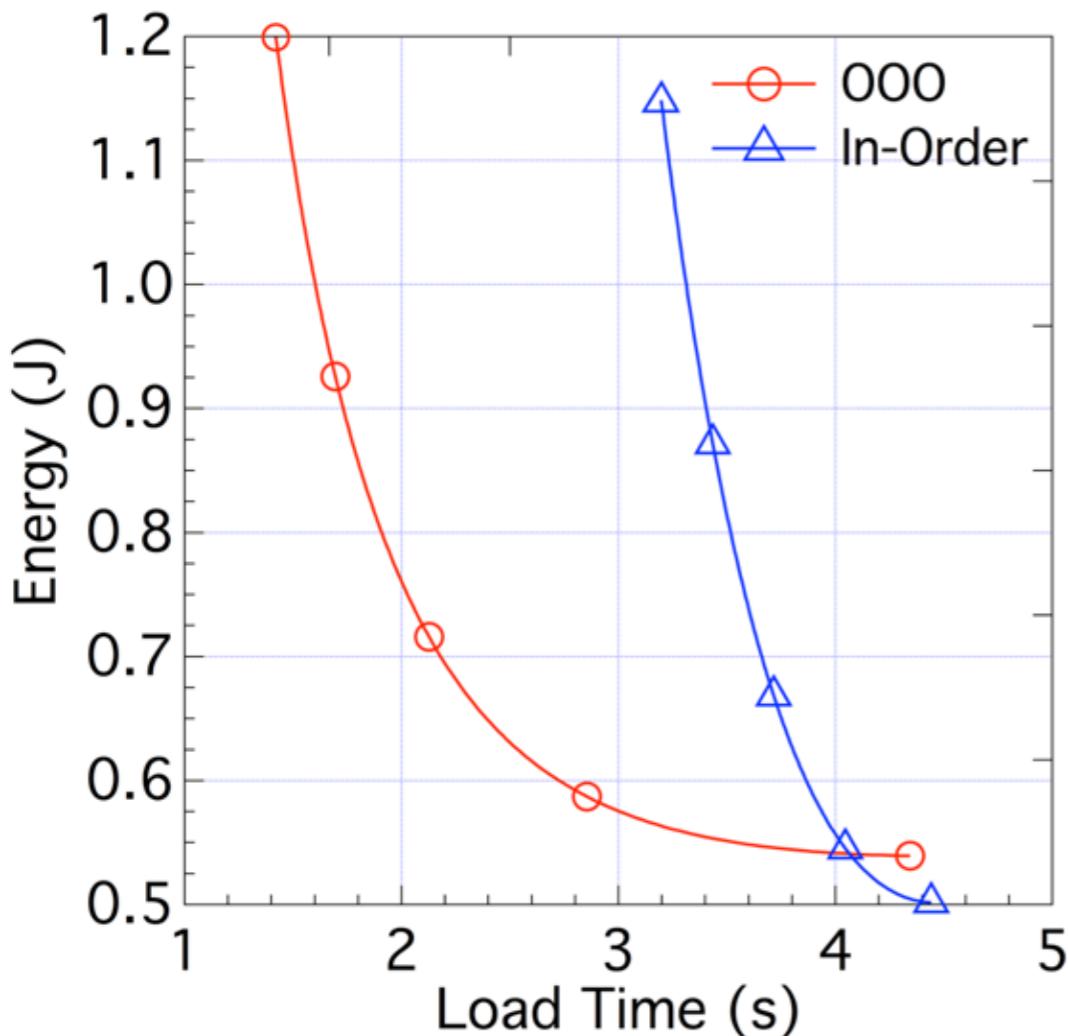
Customization: Identifying Major Sources of Energy Inefficiency



Customization: Identifying Major Sources of Energy Inefficiency

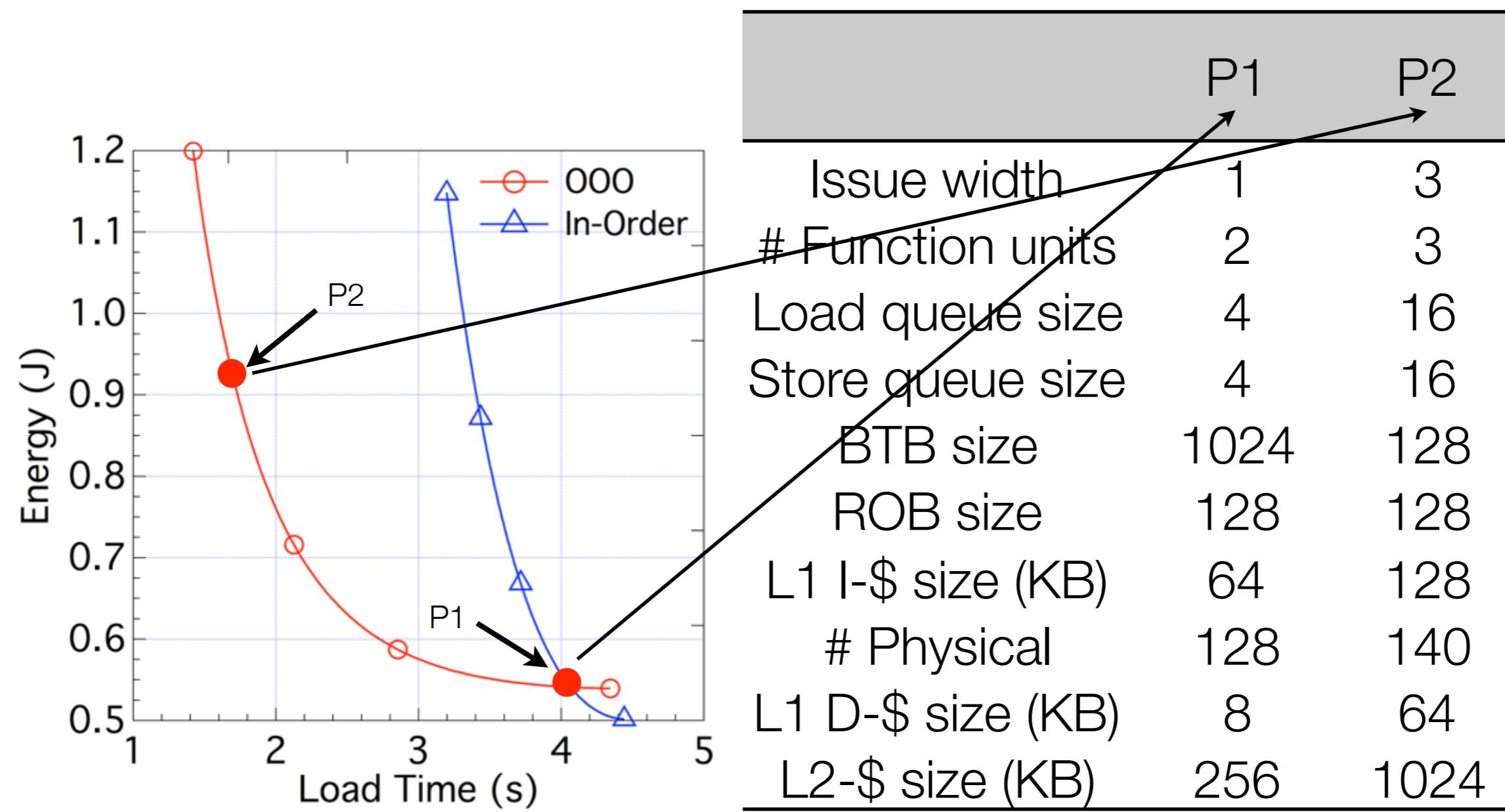


Customization: Identifying Major Sources of Energy Inefficiency

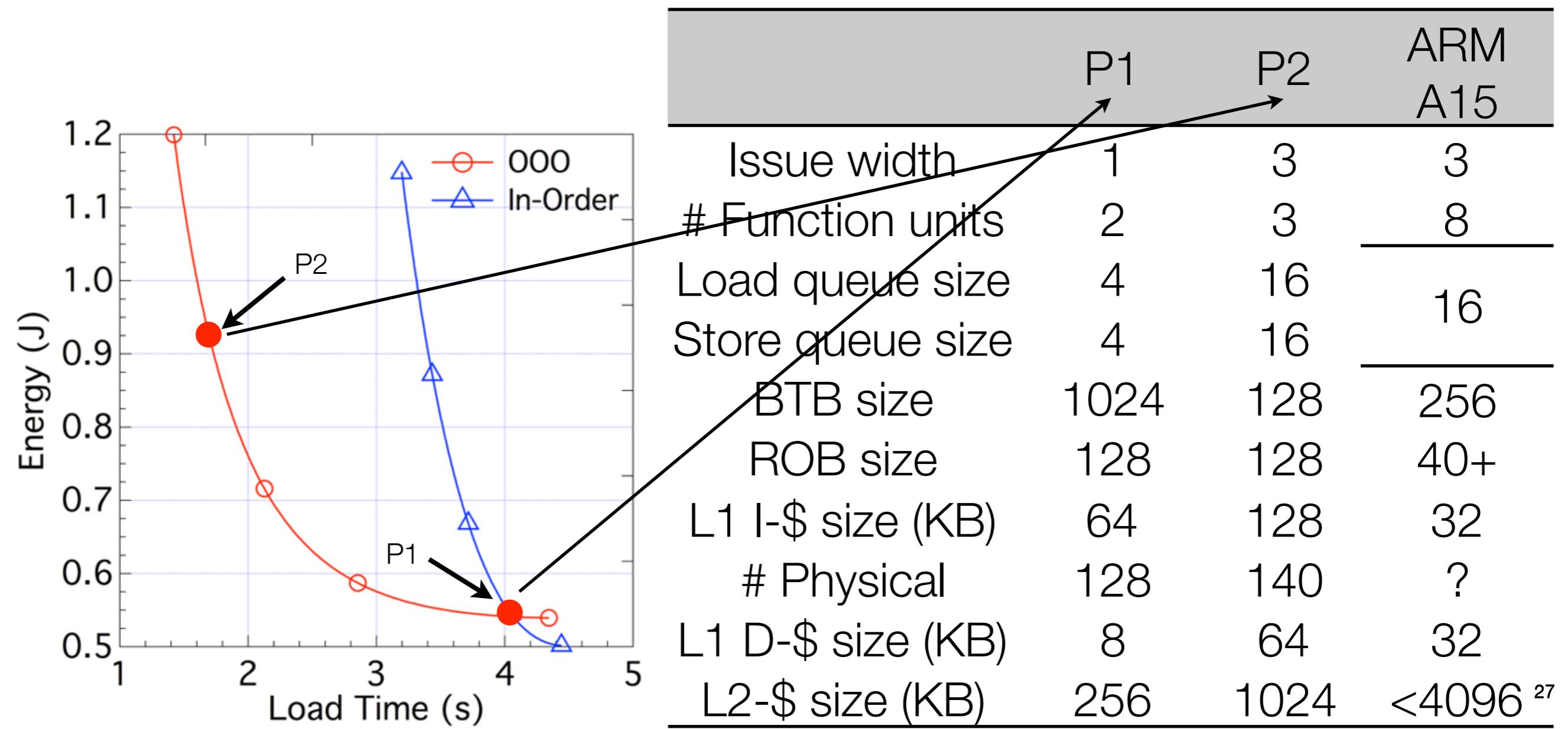


	P1	P2
Issue width	1	3
# Function units	2	3
Load queue size	4	16
Store queue size	4	16
BTB size	1024	128
ROB size	128	128
L1 I-\$ size (KB)	64	128
# Physical	128	140
L1 D-\$ size (KB)	8	64
L2-\$ size (KB)	256	1024

Customization: Identifying Major Sources of Energy Inefficiency

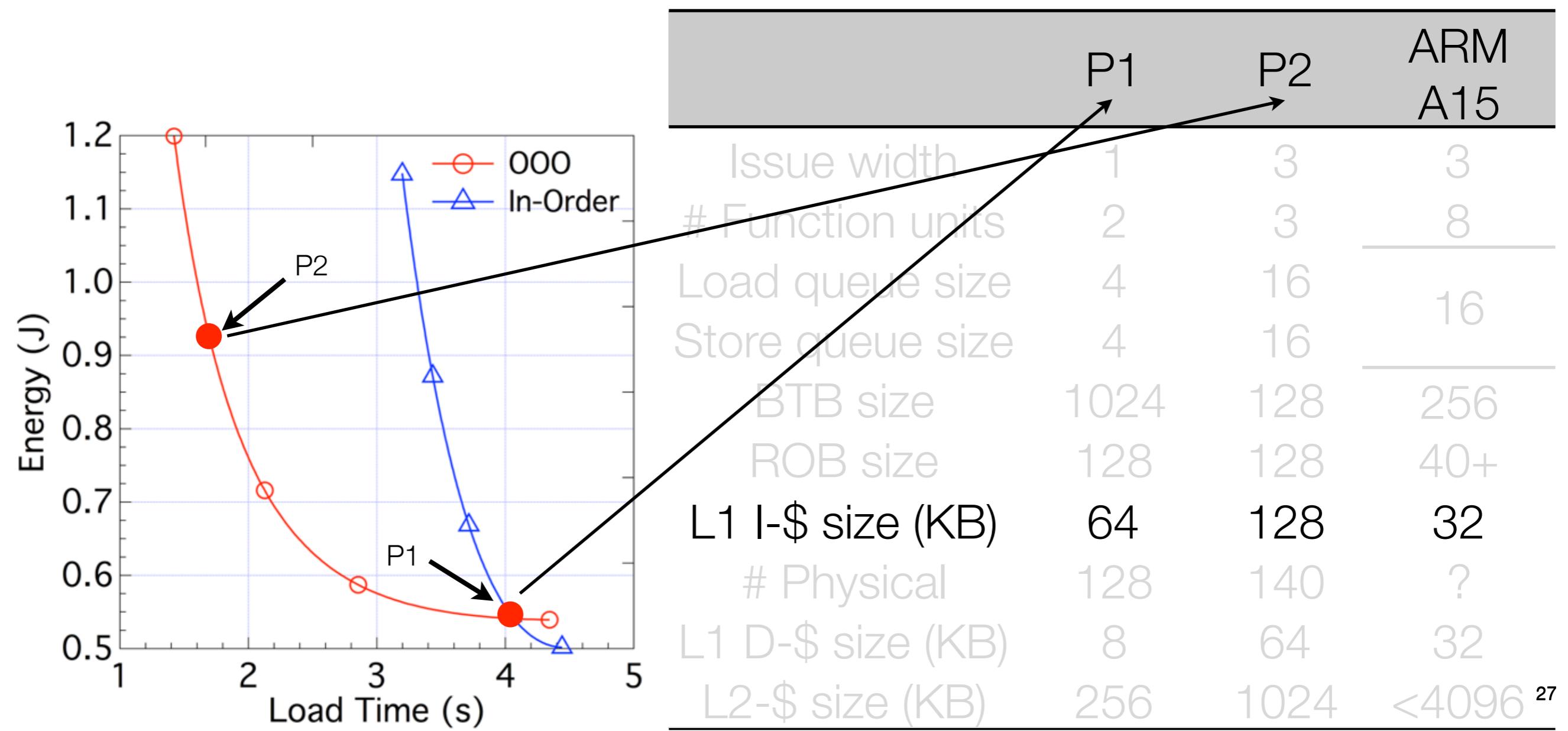


Customization: Identifying Major Sources of Energy Inefficiency



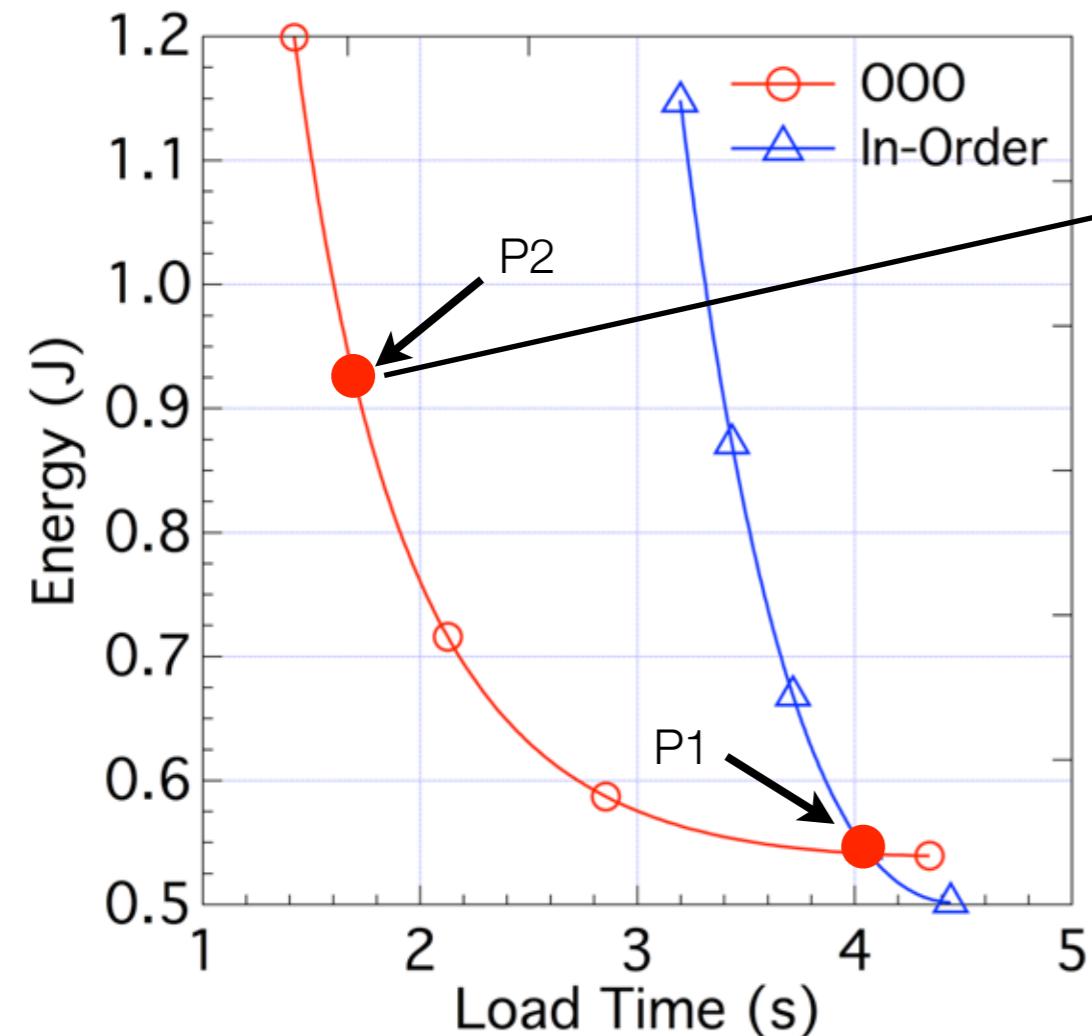
Customization: Identifying Major Sources of Energy Inefficiency

► Instruction supply



Customization: Identifying Major Sources of Energy Inefficiency

- ▶ Instruction supply
- ▶ Data feeding



	P1	P2	ARM A15
Issue width	1	3	3
# Function units	2	3	8
Load queue size	4	16	16
Store queue size	4	16	16
BTB size	1024	128	256
ROB size	128	128	40+
L1 I-\$ size (KB)	64	128	32
# Physical	128	140	?
L1 D-\$ size (KB)	8	64	32
L2-\$ size (KB)	256	1024	<4096

Specialization: Fixing the Pending Inefficiencies

- ▶ Instruction supply
- ▶ Data feeding



Specialization: Fixing the Pending Inefficiencies

- ▶ Instruction supply
 - ▷ Pack more operations in one instruction
- ▶ Data feeding



Specialization: Fixing the Pending Inefficiencies

- ▶ Instruction supply
 - ▷ Pack more operations in one instruction
- ▶ Data feeding
 - ▷ Move operands closer to operations



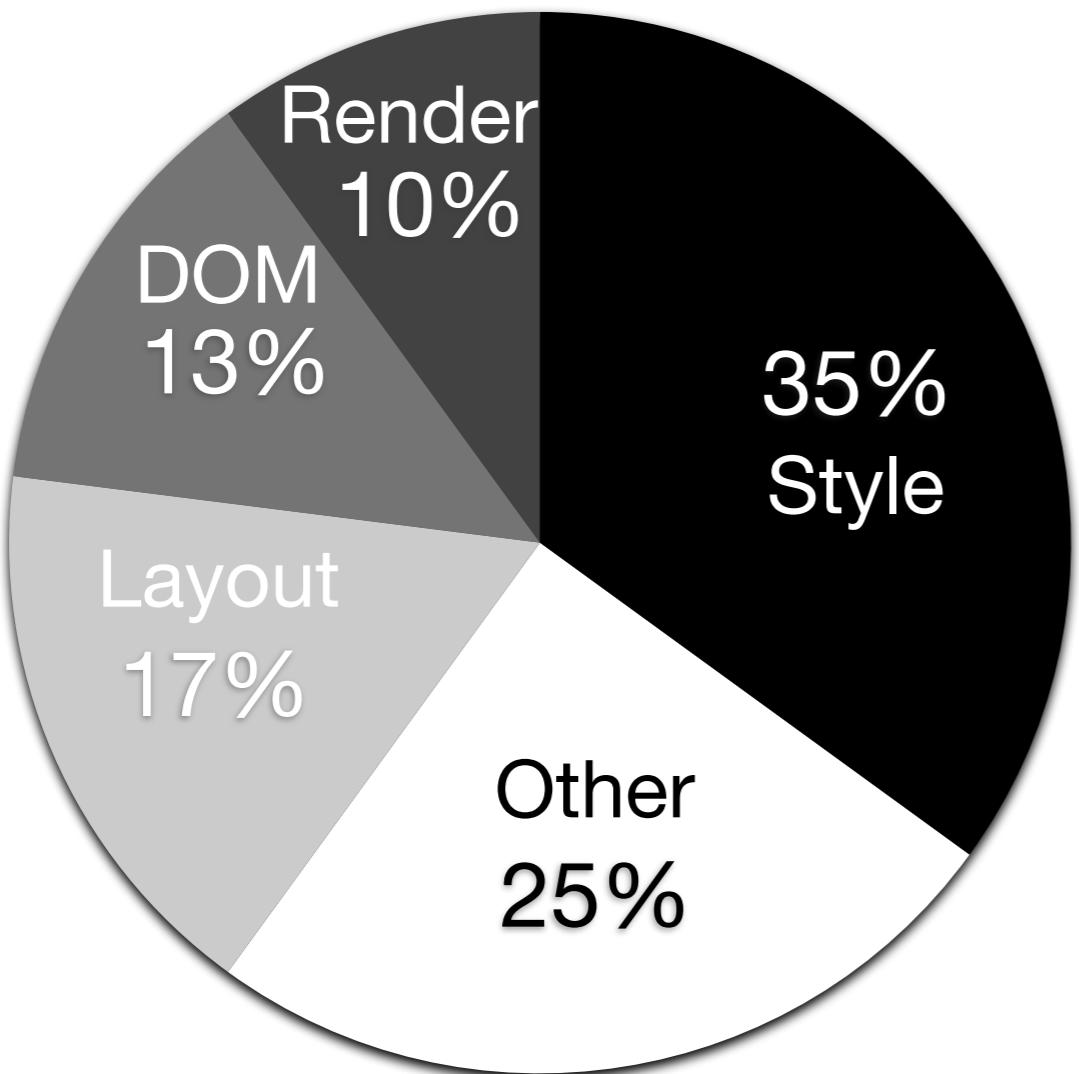
Style Resolution Kernel

- ▶ Choose the **Style** kernel as the specialization target

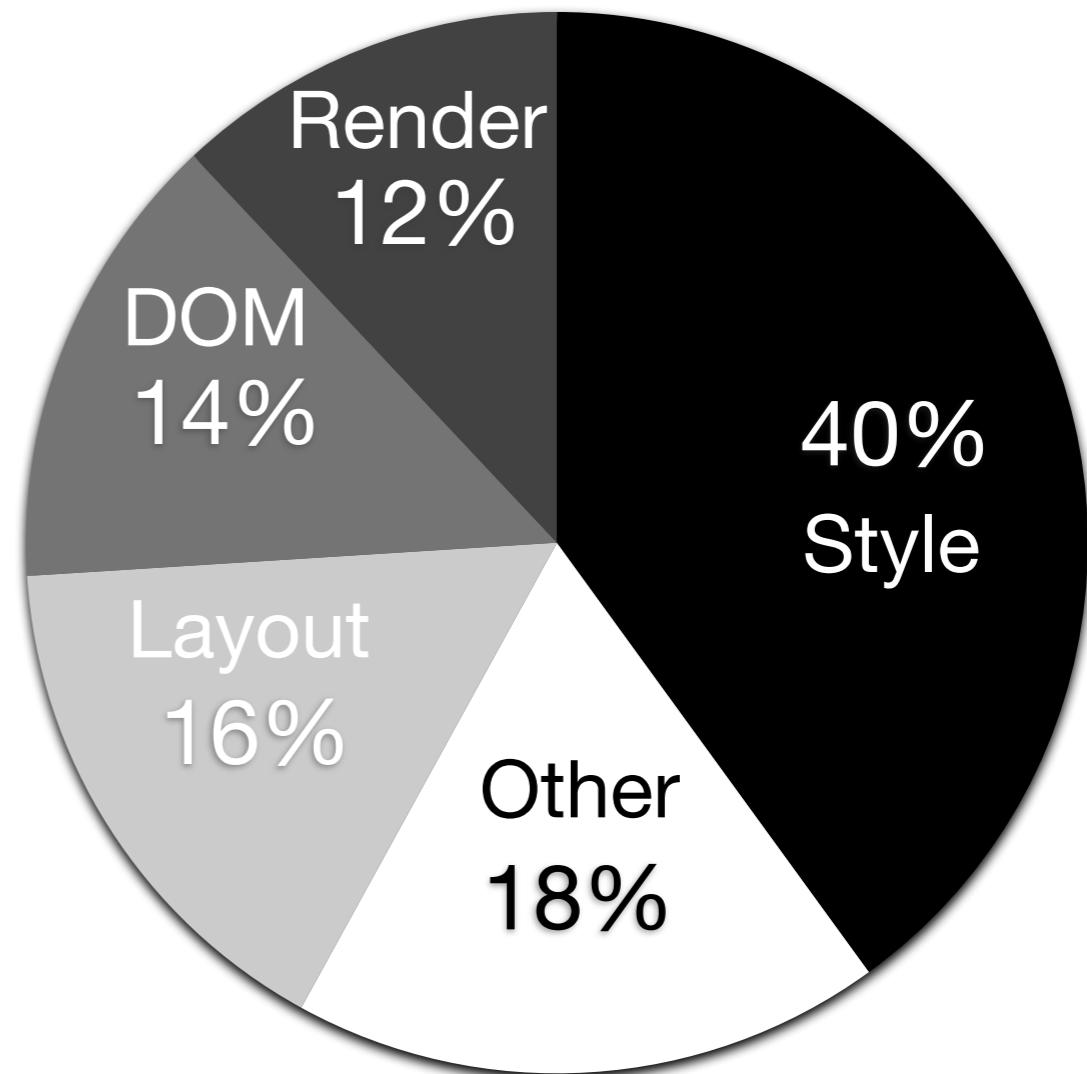


Style Resolution Kernel

- ▶ Choose the **Style** kernel as the specialization target



Execution time
breakdown

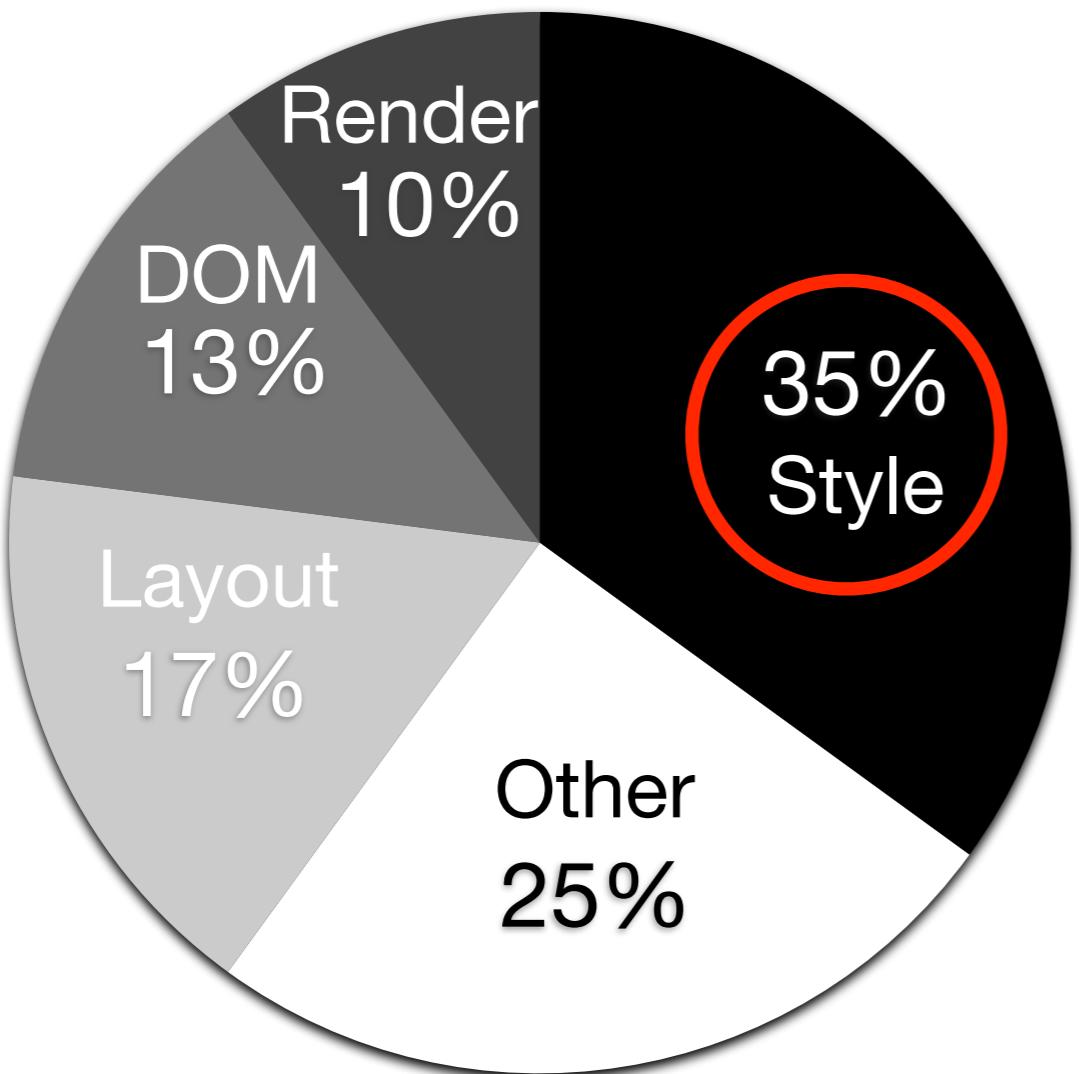


Energy
breakdown

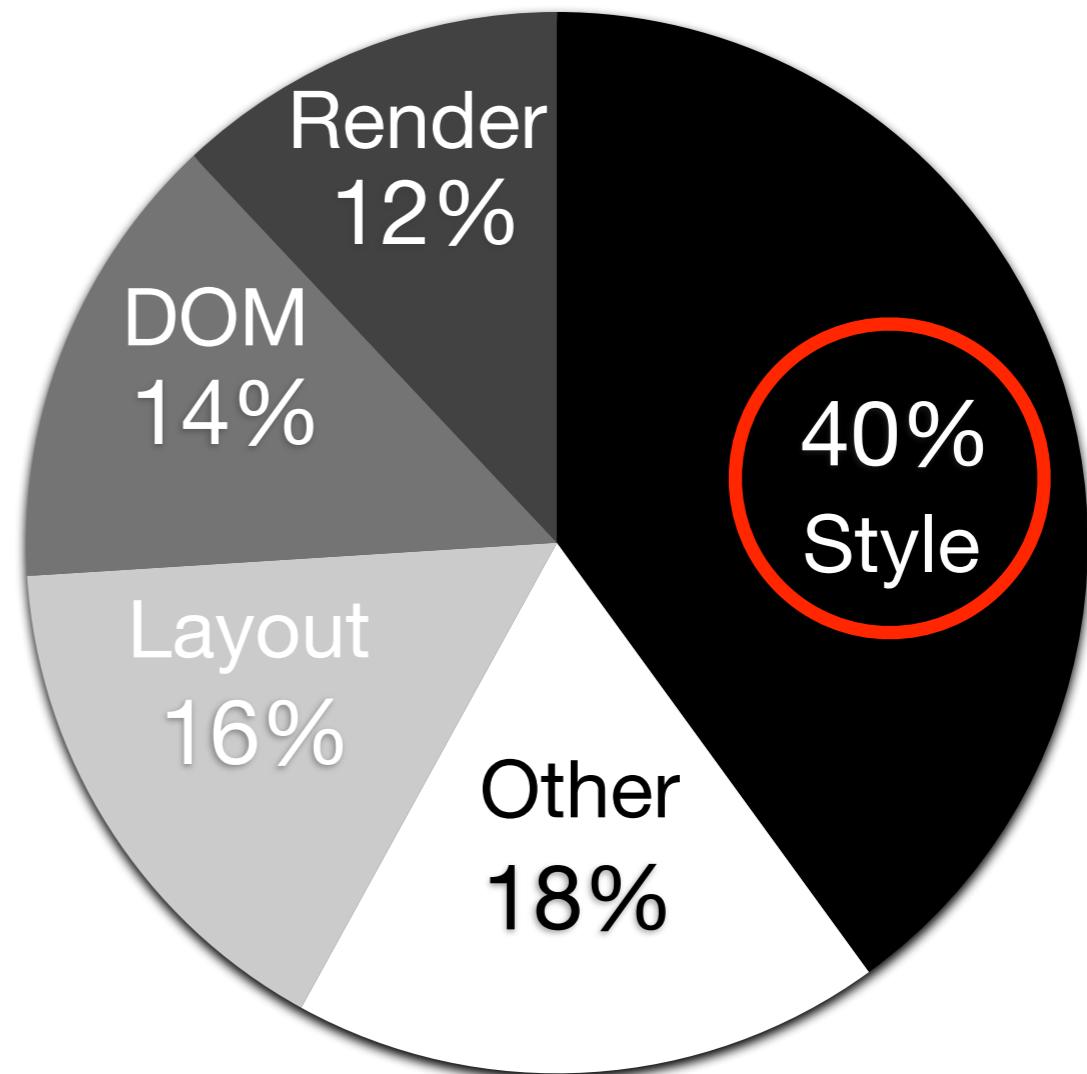


Style Resolution Kernel

- ▶ Choose the **Style** kernel as the specialization target



Execution time
breakdown



Energy
breakdown



Style Resolution Kernel

- ▶ Choose the **Style** kernel as the specialization target

```
for (each rule in matchedRules) {  
  
    for (each property in rule) {  
  
        switch (property.id) {  
            case Font:  
                Style[Font] = Handler(property.value, DOMNode);  
                break;  
            case N: ... } } }
```



Style Resolution Kernel

- ▶ Choose the **Style** kernel as the specialization target

```
for (each rule in matchedRules) {  
  
    for (each property in rule) {  
  
        switch (property.id) {  
            case Font:  
                Style[Font] = Handler(property.value, DOMNode);  
                break;  
            case N: ... } } }
```



Style Resolution Kernel

- ▶ Choose the **Style** kernel as the specialization target

```
for (each rule in matchedRules) {  
    for (each property in rule) {  
        switch (property.id) {  
            case Font:  
                Style[Font] = Handler(property.value, DOMNode);  
                break;  
            case N: ... } } }
```

← **Rule-level
Parallelism (RLP)**



Style Resolution Kernel

- ▶ Choose the **Style** kernel as the specialization target

```
for (each rule in matchedRules) {  
    for (each property in rule) {  
  
        switch (property.id) {  
            case Font:  
                Style[Font] = Handler(property.value, DOMNode);  
                break;  
            case N: ... } } }
```

**Rule-level
Parallelism (RLP)**



Style Resolution Kernel

- ▶ Choose the **Style** kernel as the specialization target

```
for (each rule in matchedRules) {  
    for (each property in rule) {  
        switch (property.id) {  
            case Font:  
                Style[Font] = Handler(property.value, DOMNode);  
                break;  
            case N: ... } } }
```

**Rule-level
Parallelism (RLP)**

**Property-level
Parallelism (PLP)**



Style Resolution Kernel

- ▶ Choose the **Style** kernel as the specialization target

```
for (each rule in matchedRules) {  
    for (each property in rule) {  
        switch (property.id) {  
            case Font:  
                Style[Font] = Handler(property.value, DOMNode);  
                break;  
            case N: ... } } }
```

**Rule-level
Parallelism (RLP)**

**Property-level
Parallelism (PLP)**

- ▶ Exploiting the parallelism to increase the arithmetic intensity



Style Resolution Kernel

- ▶ A running example from www.cnn.com

Style Rules

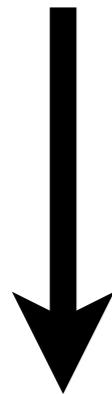
Rule	Property 1		Property 2	
	id	value	id	value
1	padding	0	margin	0
2	padding	6 px	width	36 px



Style Resolution Kernel

- ▶ A running example from www.cnn.com

Style Rules



Rule	Property 1		Property 2	
	id	value	id	value
1	padding	0	margin	0
2	padding	6 px	width	36 px

Final Style Info

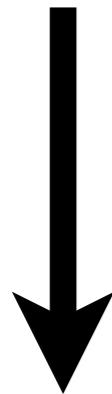
Property 1		Property 2		Property 3	
id	value	id	value	id	value



Style Resolution Kernel

- ▶ A running example from www.cnn.com

Style Rules



High priority

Rule	Property 1		Property 2	
	id	value	id	value
1	padding	0	margin	0
2	padding	6 px	width	36 px

Final Style Info

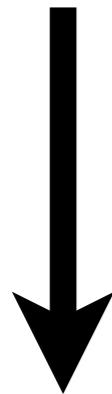
Property 1		Property 2		Property 3	
id	value	id	value	id	value



Style Resolution Kernel

- ▶ A running example from www.cnn.com

Style Rules



Rule	Property 1		Property 2	
	id	value	id	value
1	padding	0	margin	0
2	padding	6 px	width	36 px

Final Style Info

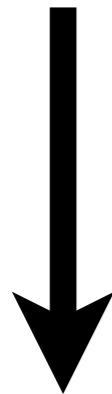
Property 1		Property 2		Property 3	
id	value	id	value	id	value



Style Resolution Kernel

- ▶ A running example from www.cnn.com

Style Rules



High priority

Rule	Property 1		Property 2	
	id	value	id	value
1	padding	0	margin	0
2	padding	6 px	width	36 px

Final Style Info

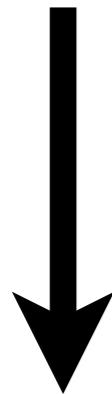
Property 1		Property 2		Property 3	
id	value	id	value	id	value
padding	0				



Style Resolution Kernel

- ▶ A running example from www.cnn.com

Style Rules



High priority

Rule	Property 1		Property 2	
	id	value	id	value
1	padding	0	margin	0
2	padding	6 px	width	36 px

Final Style Info

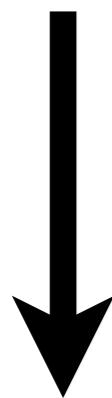
Property 1		Property 2		Property 3	
id	value	id	value	id	value
padding	0	margin	0		



Style Resolution Kernel

- ▶ A running example from www.cnn.com

Style Rules



High priority

Rule	Property 1		Property 2	
	id	value	id	value
1	padding	0	margin	0
2	padding	6 px	width	36 px

Final Style Info

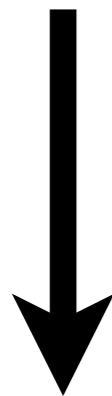
Property 1		Property 2		Property 3	
id	value	id	value	id	value
padding	0	margin	0		



Style Resolution Kernel

- ▶ A running example from www.cnn.com

Style Rules



High priority

Rule	Property 1		Property 2	
	id	value	id	value
1	padding	0	margin	0
2	padding	6 px	width	36 px

Final Style Info

Property 1		Property 2		Property 3	
id	value	id	value	id	value
padding	6 px	margin	0		

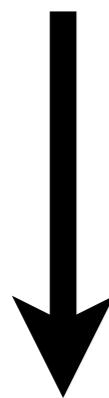


Style Resolution Kernel

- ▶ Order Matters in RLP
- ▶ Order Does **Not** Matter in PLP

- ▶ A running example from www.cnn.com

Style Rules



High priority

Rule	Property 1		Property 2	
	id	value	id	value
1	padding	0	margin	0
2	padding	6 px	width	36 px

Final Style Info

Property 1		Property 2		Property 3	
id	value	id	value	id	value
padding	6 px	margin	0		

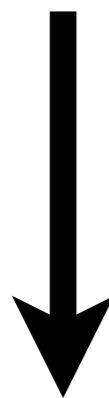


Style Resolution Kernel

- ▶ Order Matters in RLP
- ▶ Order Does **Not** Matter in PLP

- ▶ A running example from www.cnn.com

Style Rules



High priority

Rule	Property 1		Property 2	
	id	value	id	value
1	padding	0	margin	0
2	padding	6 px	width	36 px

Final Style Info

Property 1		Property 2		Property 3	
id	value	id	value	id	value
padding	6 px	margin	0	width	36 px



Style Resolution Unit

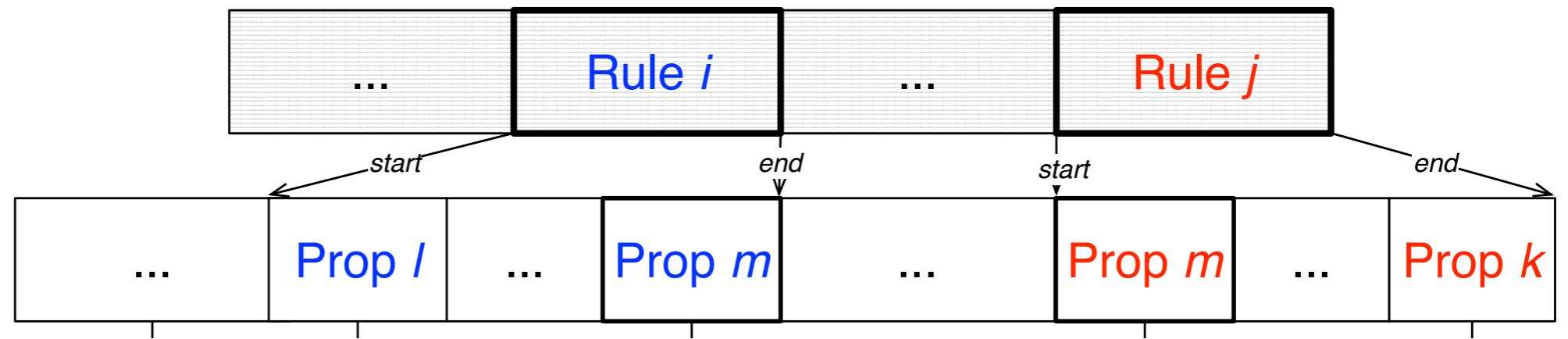
- ▶ Order Matters in RLP
- ▶ Order Does **Not** Matter in PLP



Style Resolution Unit

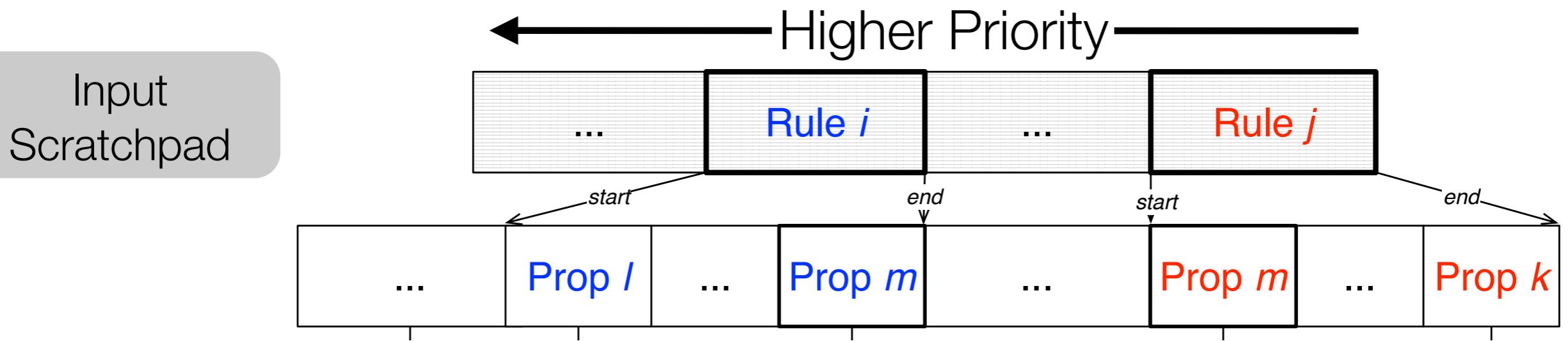
- ▶ Order Matters in RLP
- ▶ Order Does **Not** Matter in PLP

Input
Scratchpad



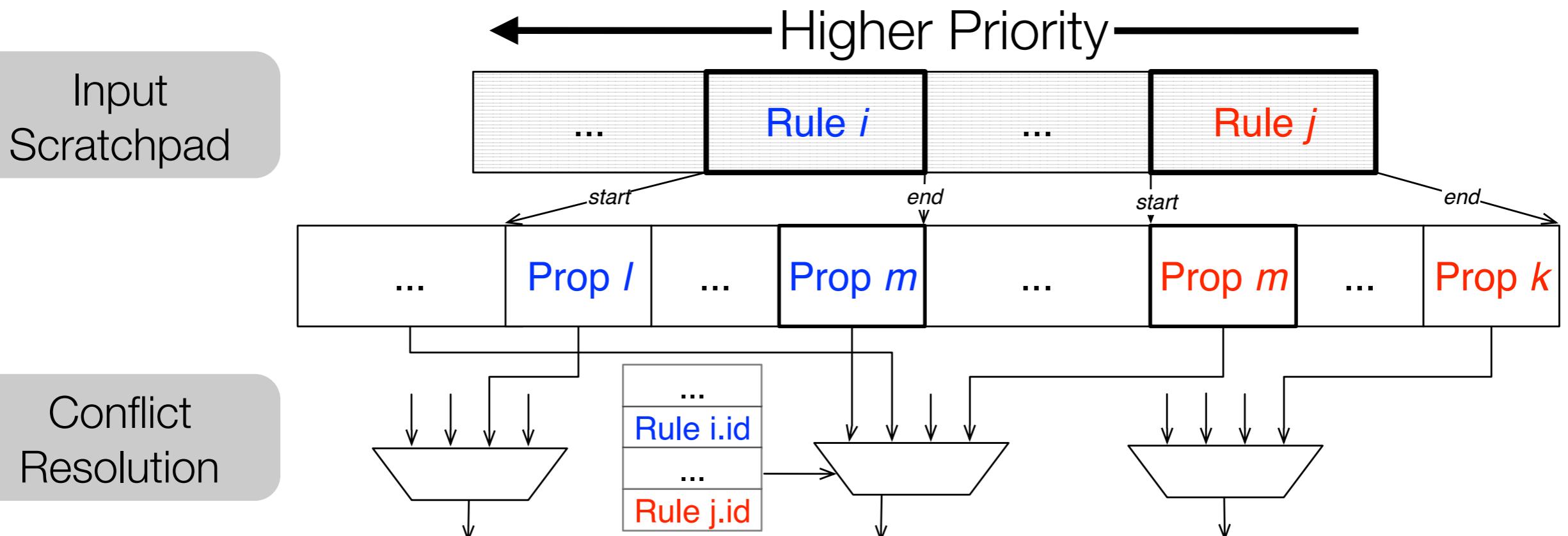
Style Resolution Unit

- ▶ Order Matters in RLP
- ▶ Order Does **Not** Matter in PLP



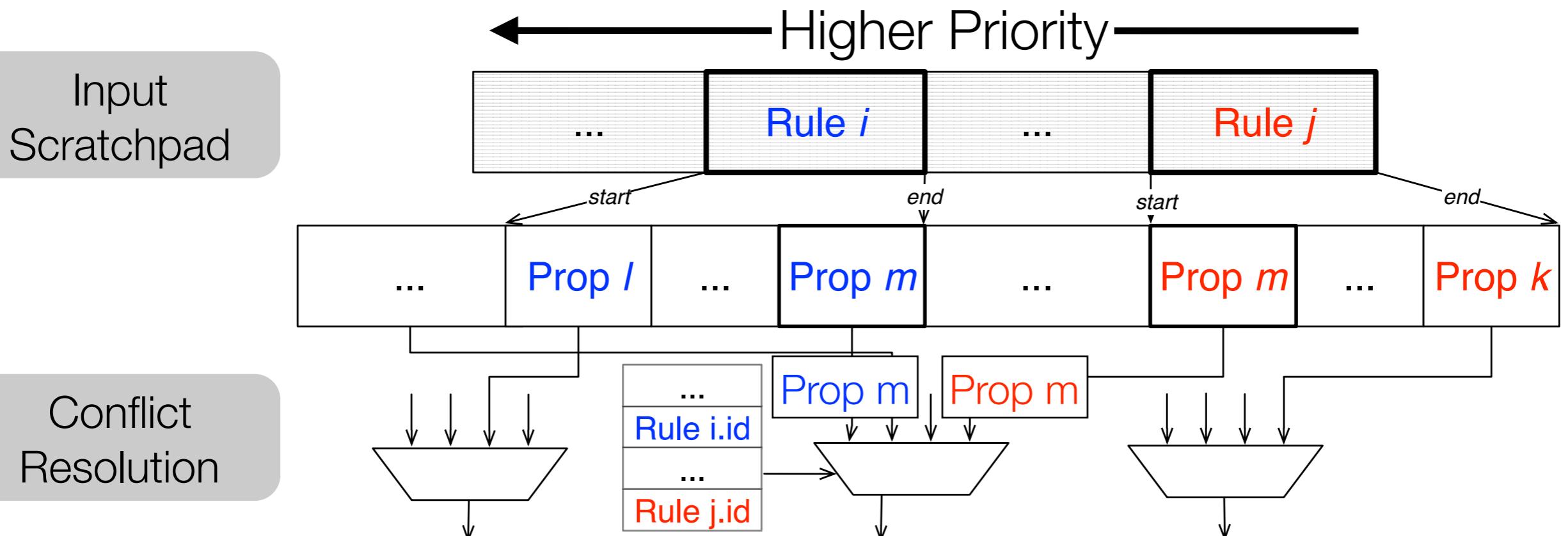
Style Resolution Unit

- ▶ Order Matters in RLP
- ▶ Order Does **Not** Matter in PLP



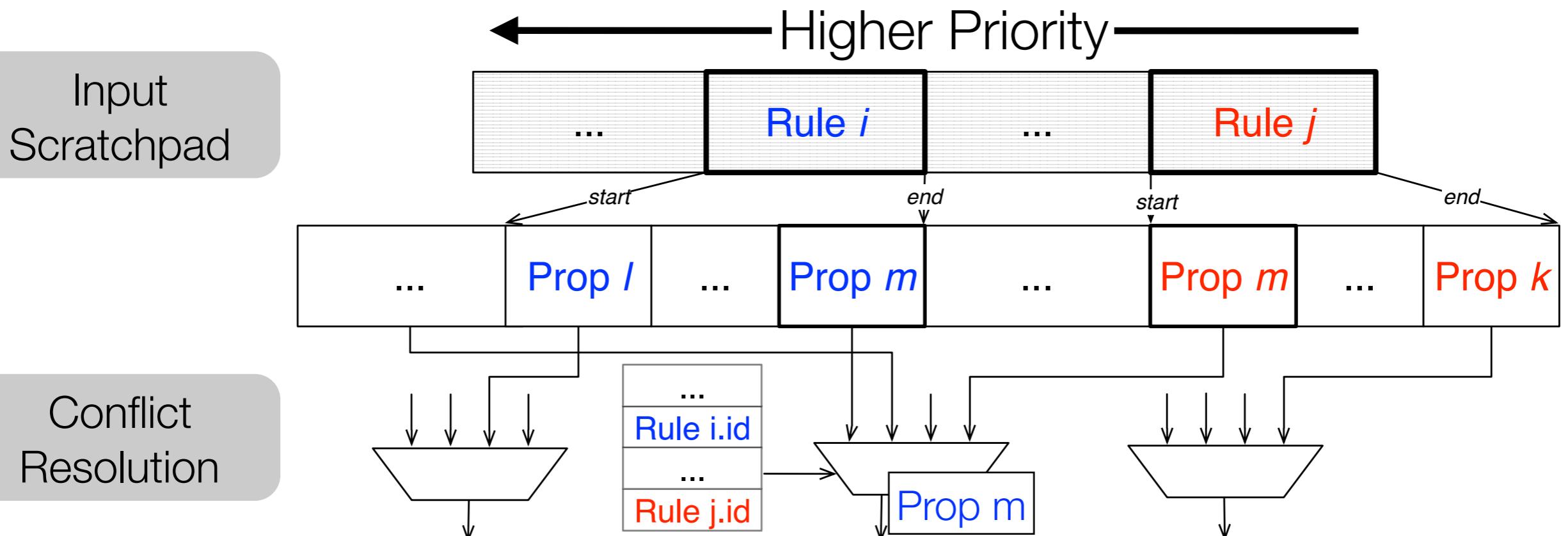
Style Resolution Unit

- ▶ Order Matters in RLP
- ▶ Order Does **Not** Matter in PLP



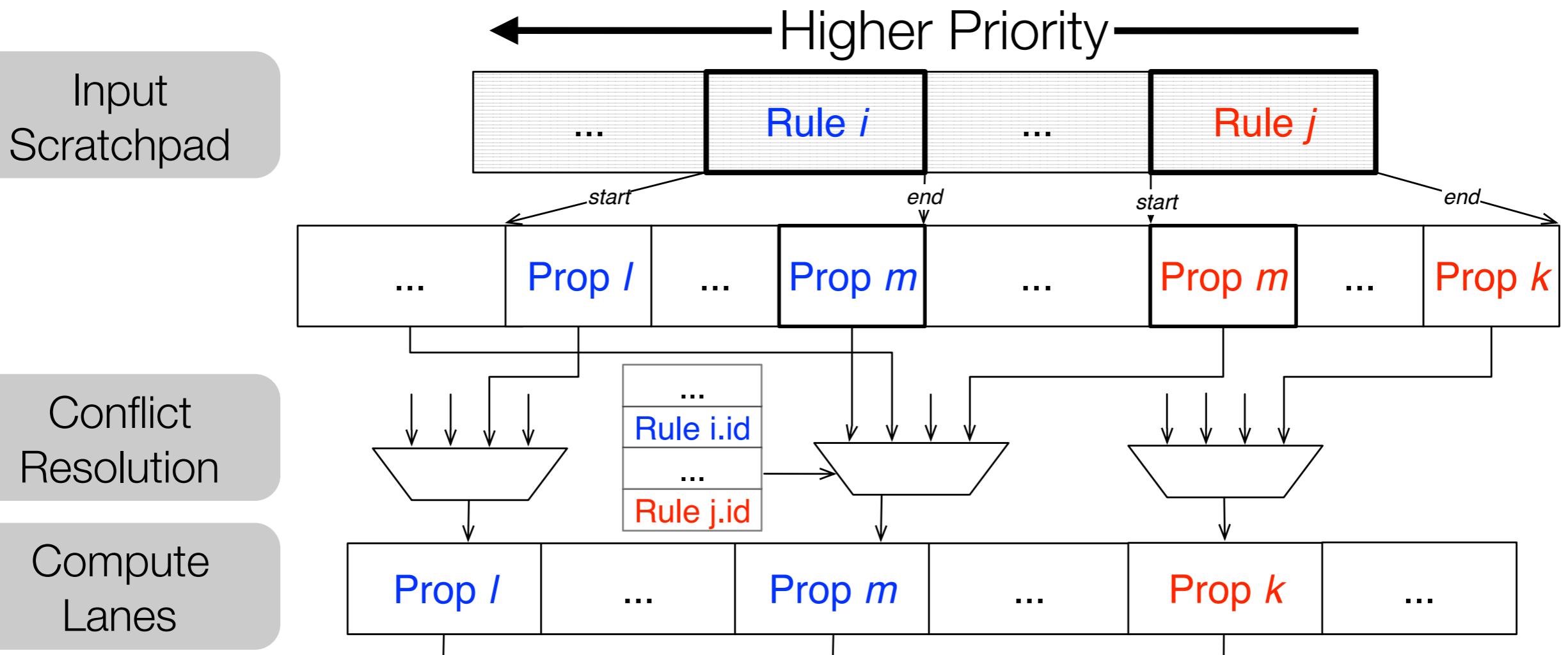
Style Resolution Unit

- ▶ Order Matters in RLP
- ▶ Order Does **Not** Matter in PLP



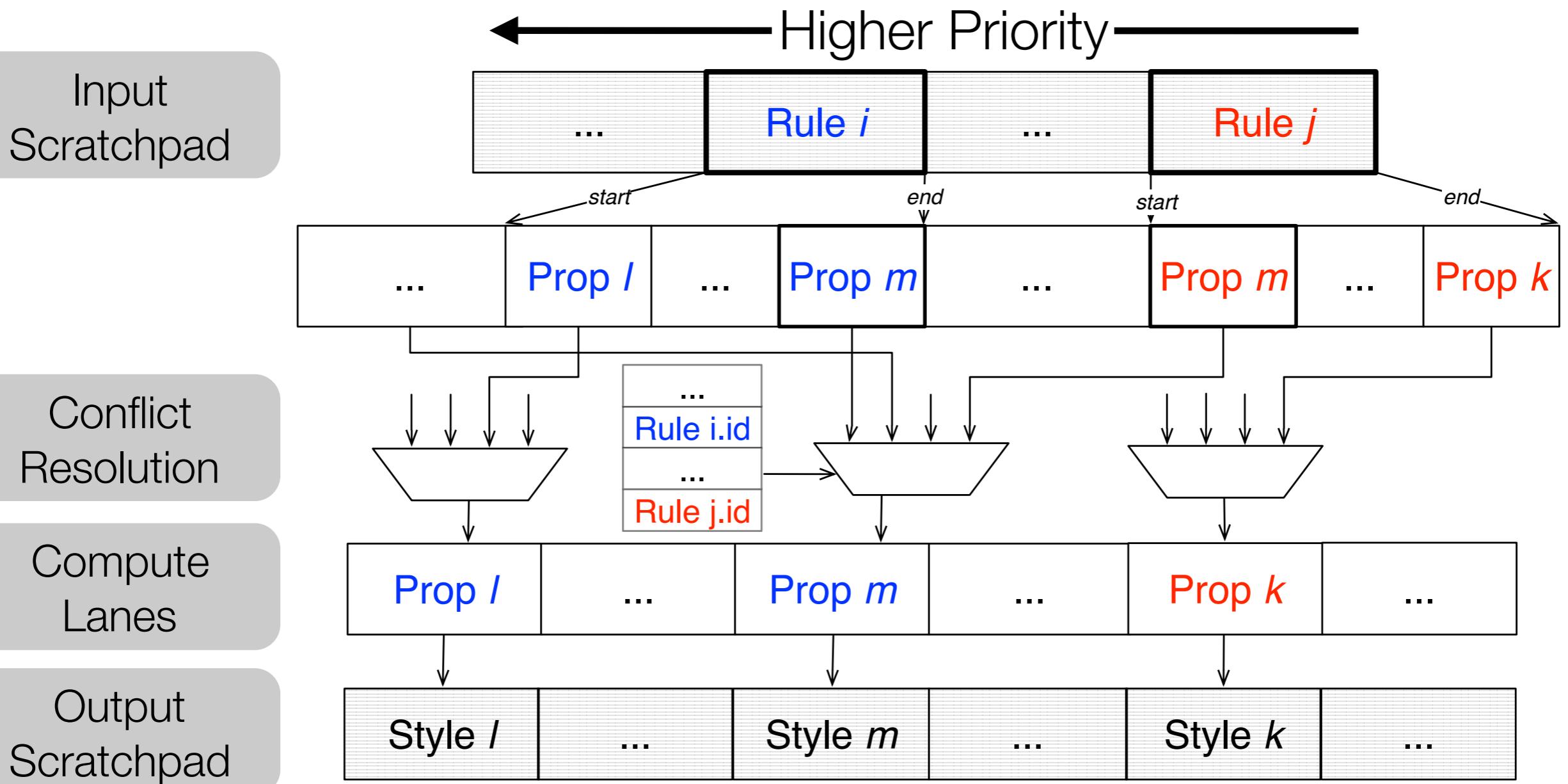
Style Resolution Unit

- ▶ Order Matters in RLP
- ▶ Order Does **Not** Matter in PLP



Style Resolution Unit

- ▶ Order Matters in RLP
- ▶ Order Does **Not** Matter in PLP



Evaluation Results



Evaluation Results

- ▶ Fully synthesized using
Synopsys 28 nm toolchain



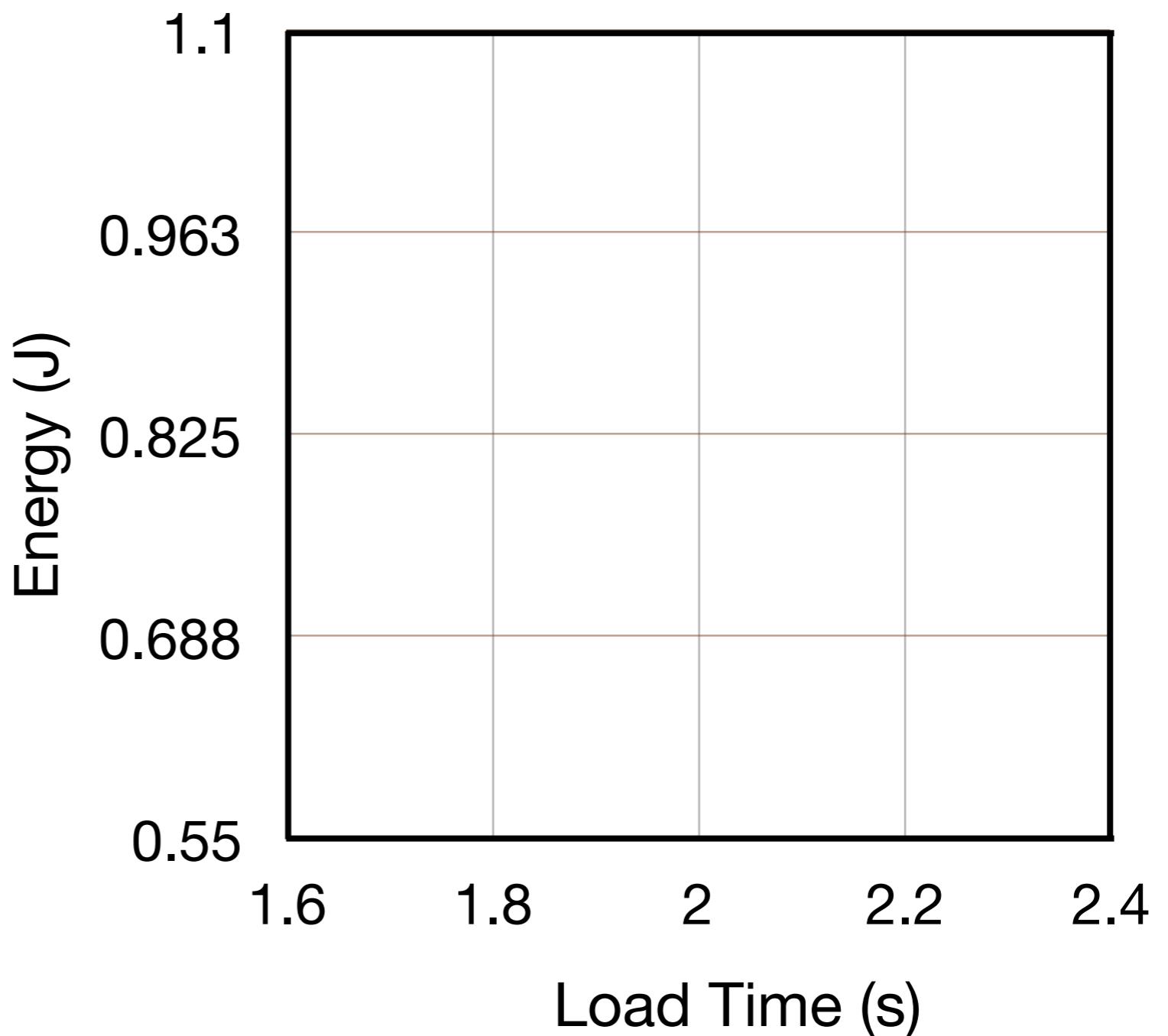
Evaluation Results

- ▶ Fully synthesized using
Synopsys 28 nm toolchain
- ▶ Cost of specialization:
0.59 mm² area overhead
 - ▷ SoC die area is 122 mm² in
Samsung Galaxy S4



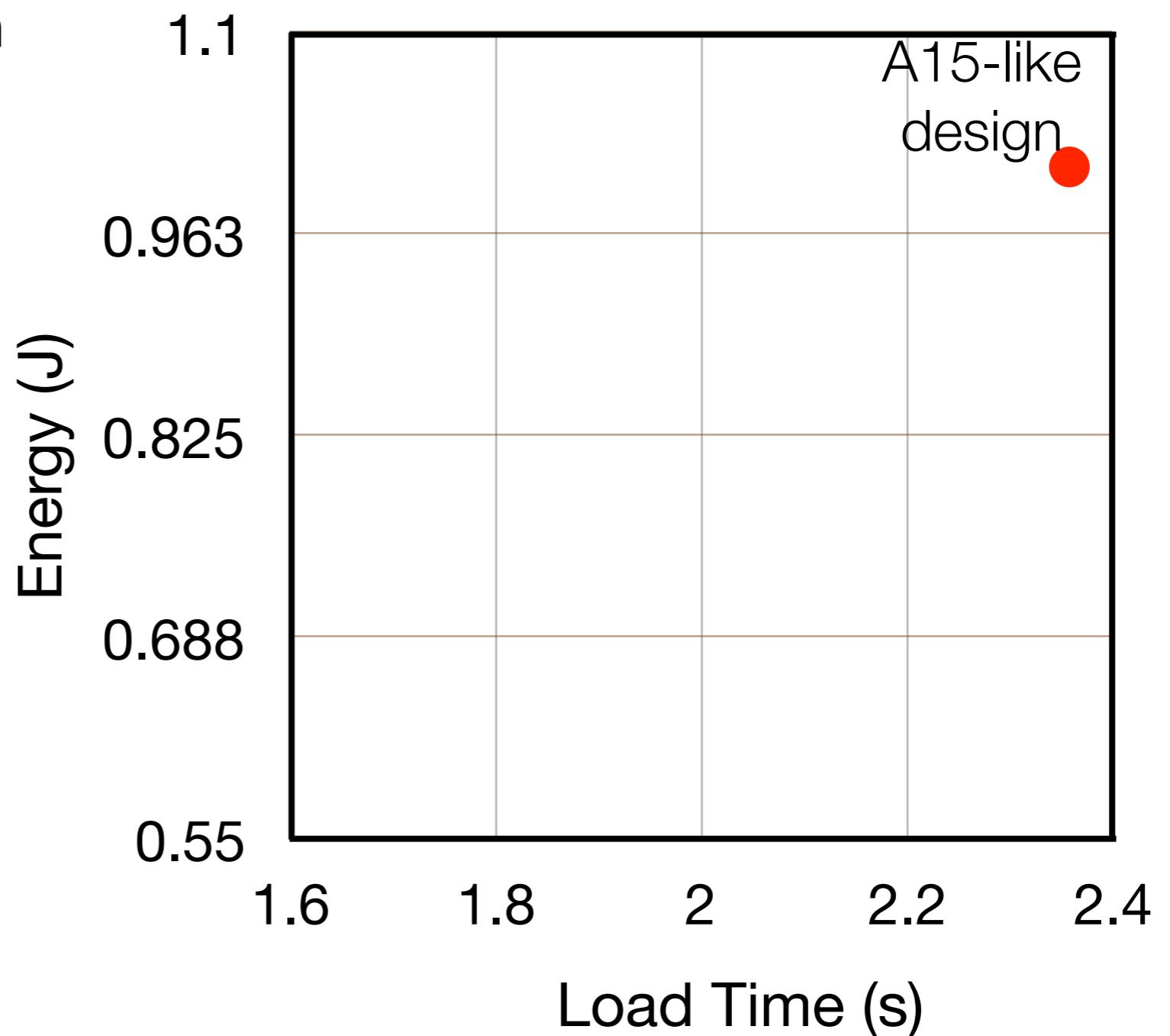
Evaluation Results

- ▶ Fully synthesized using Synopsys 28 nm toolchain
- ▶ Cost of specialization:
0.59 mm² area overhead
 - ▷ SoC die area is 122 mm² in Samsung Galaxy S4



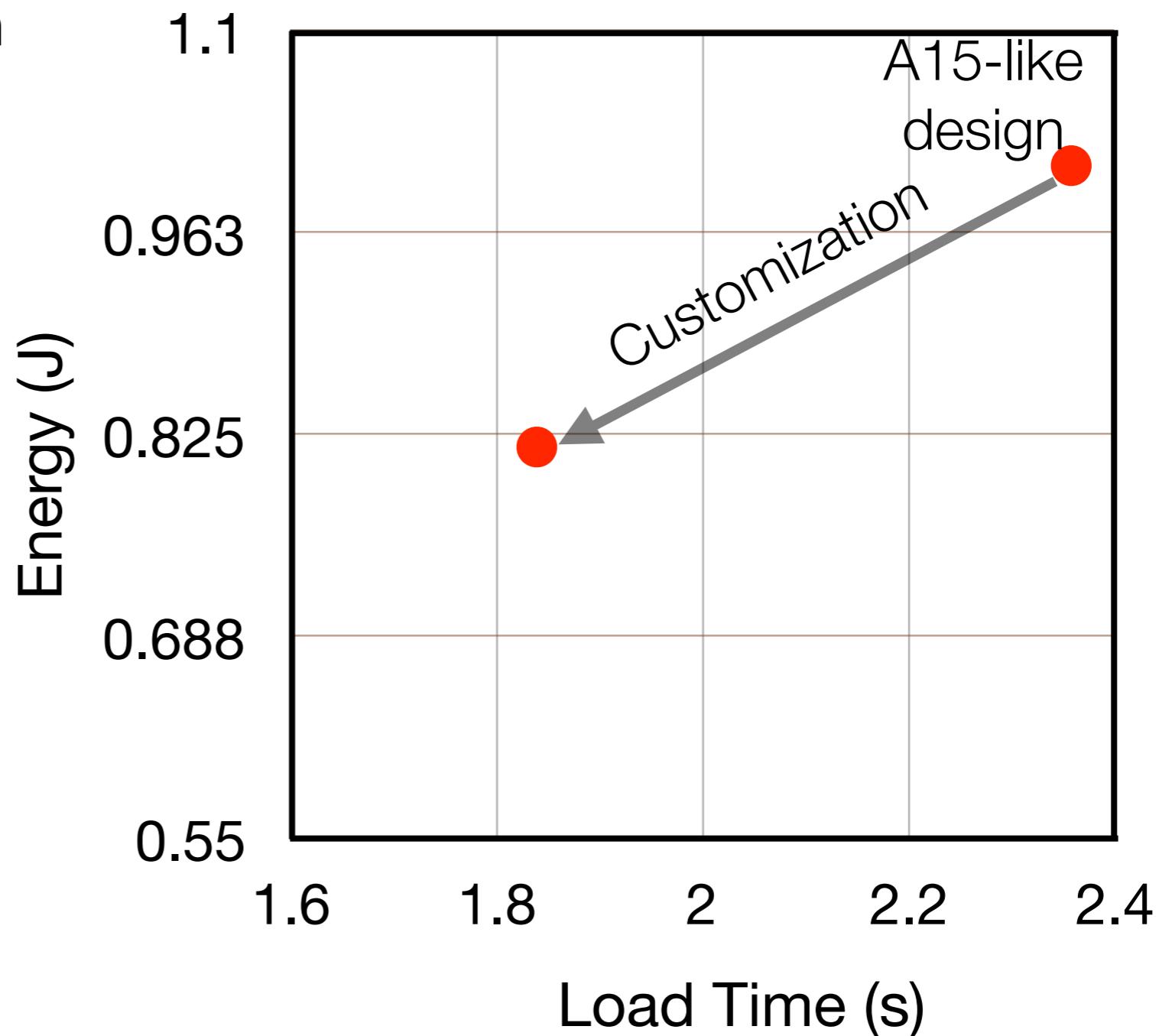
Evaluation Results

- ▶ Fully synthesized using Synopsys 28 nm toolchain
- ▶ Cost of specialization:
0.59 mm² area overhead
 - ▷ SoC die area is 122 mm² in Samsung Galaxy S4



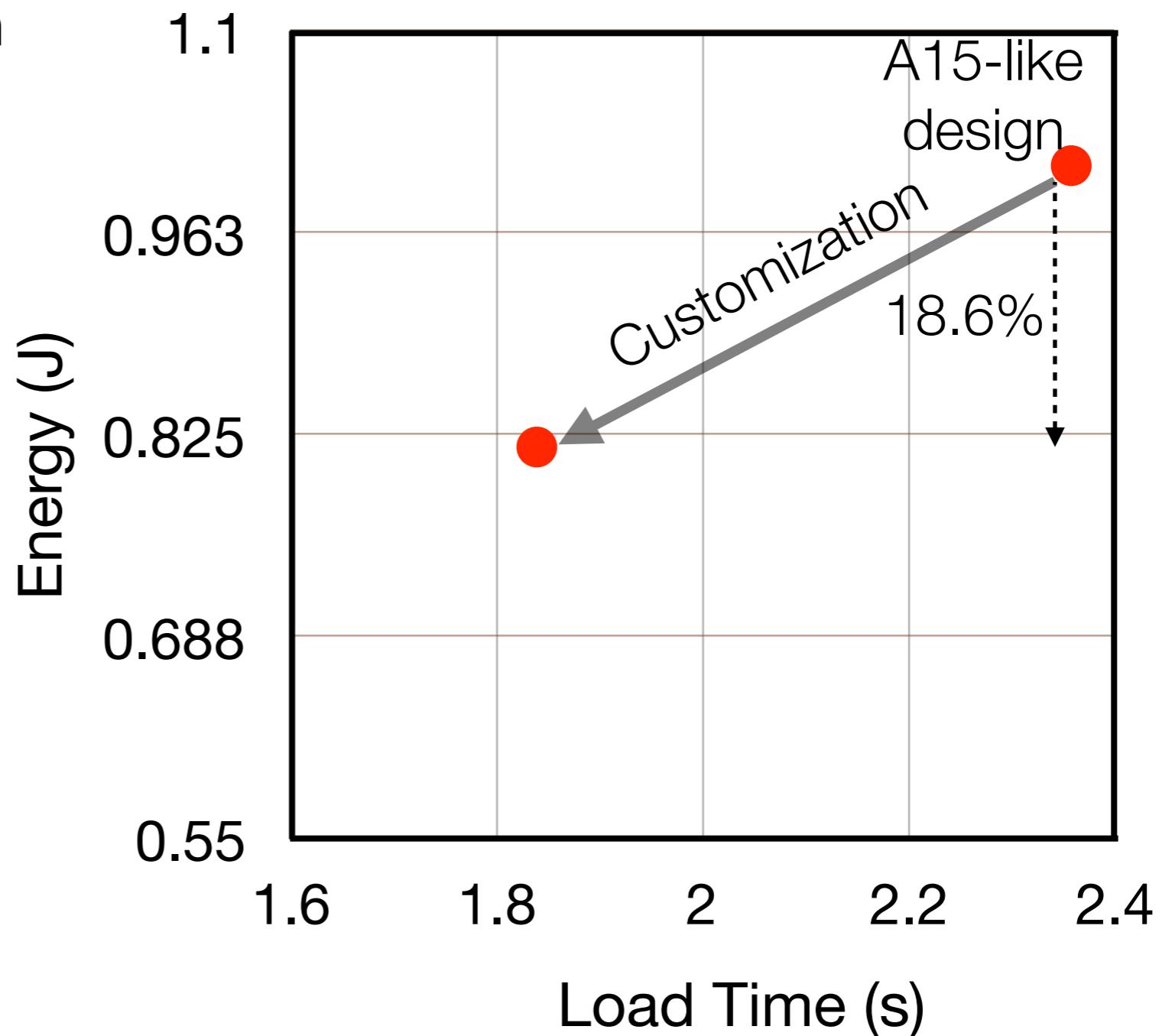
Evaluation Results

- ▶ Fully synthesized using Synopsys 28 nm toolchain
- ▶ Cost of specialization:
0.59 mm² area overhead
 - ▷ SoC die area is 122 mm² in Samsung Galaxy S4



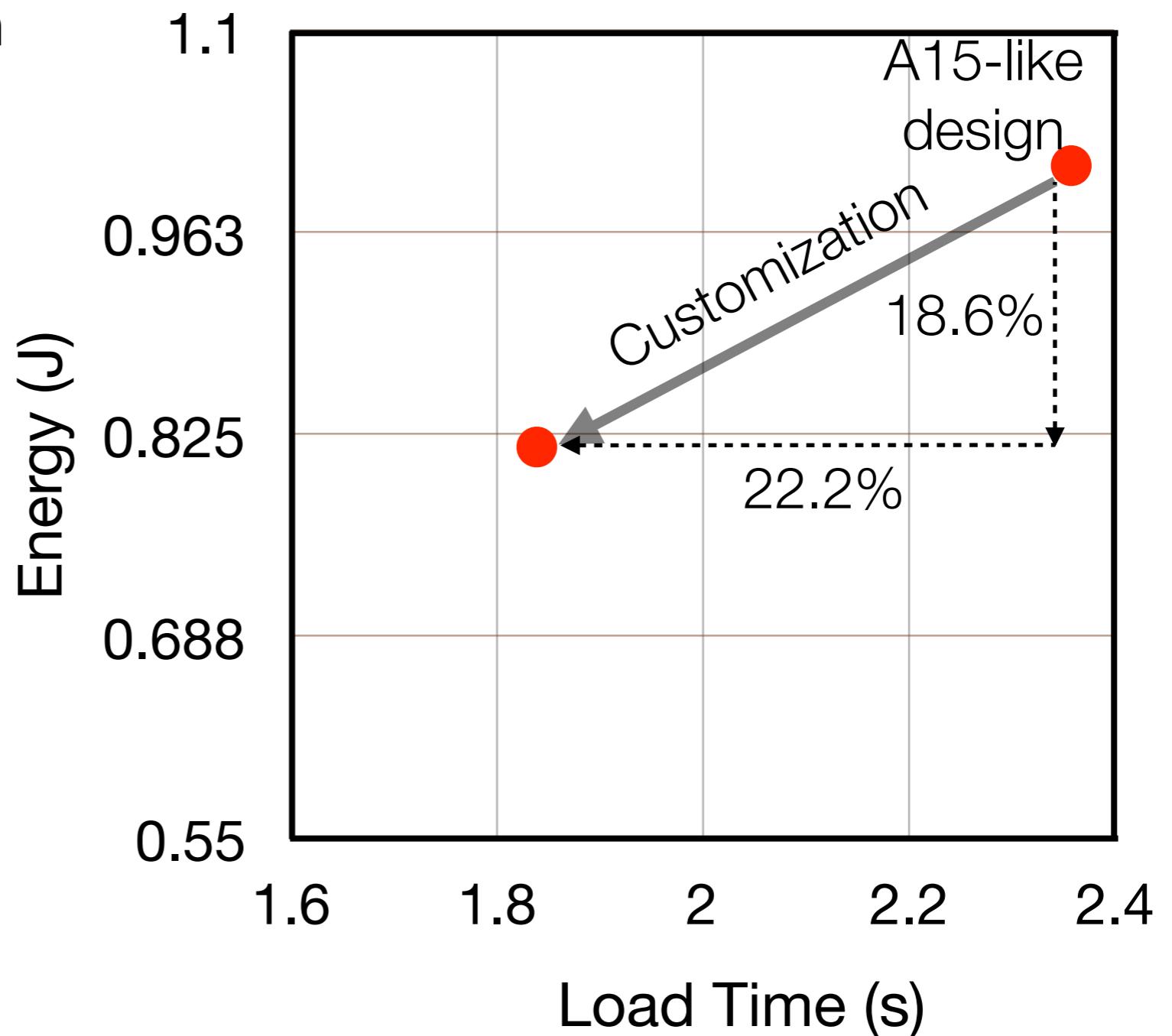
Evaluation Results

- ▶ Fully synthesized using Synopsys 28 nm toolchain
- ▶ Cost of specialization:
0.59 mm² area overhead
 - ▷ SoC die area is 122 mm² in Samsung Galaxy S4



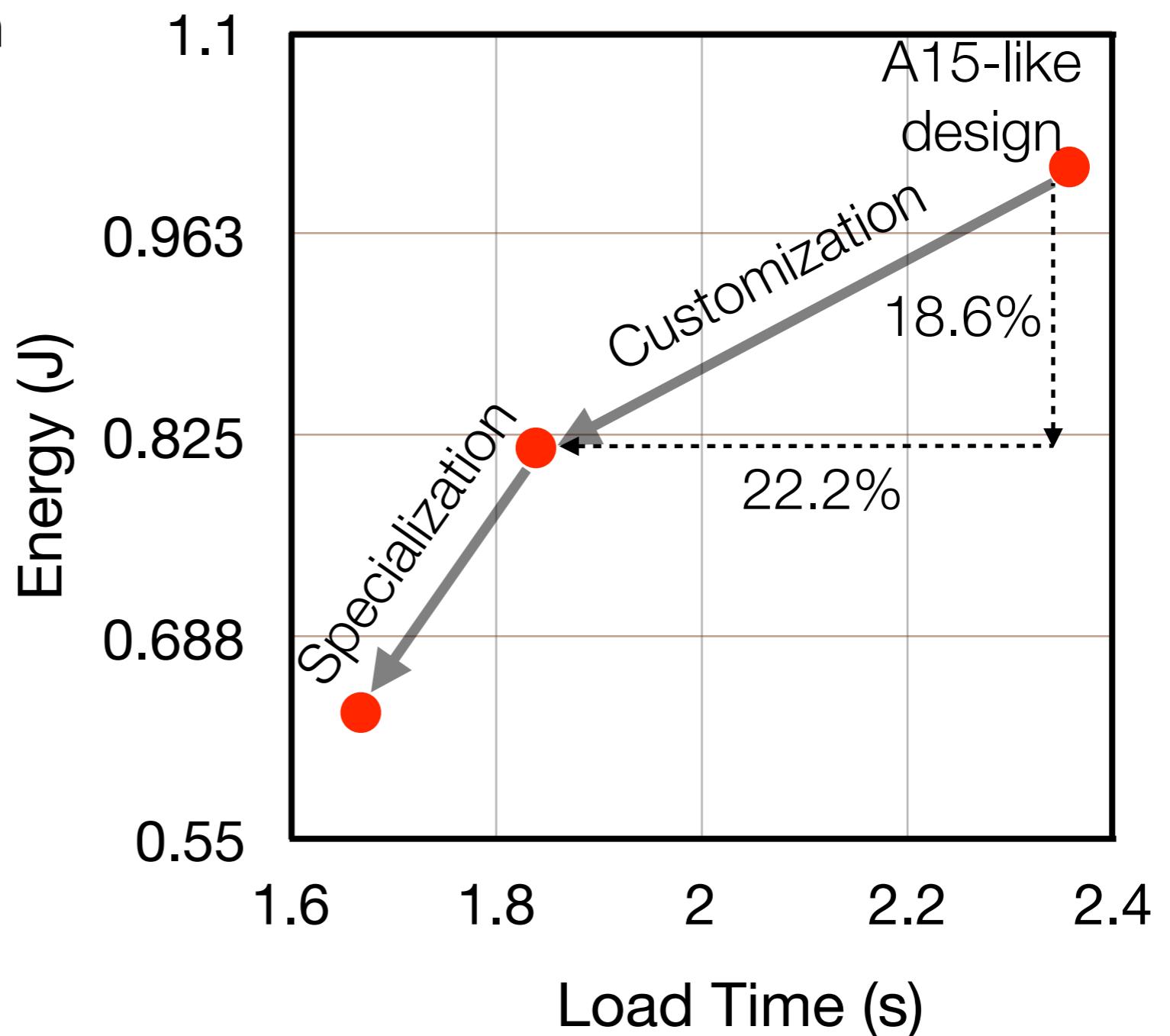
Evaluation Results

- ▶ Fully synthesized using Synopsys 28 nm toolchain
- ▶ Cost of specialization:
0.59 mm² area overhead
 - ▷ SoC die area is 122 mm² in Samsung Galaxy S4



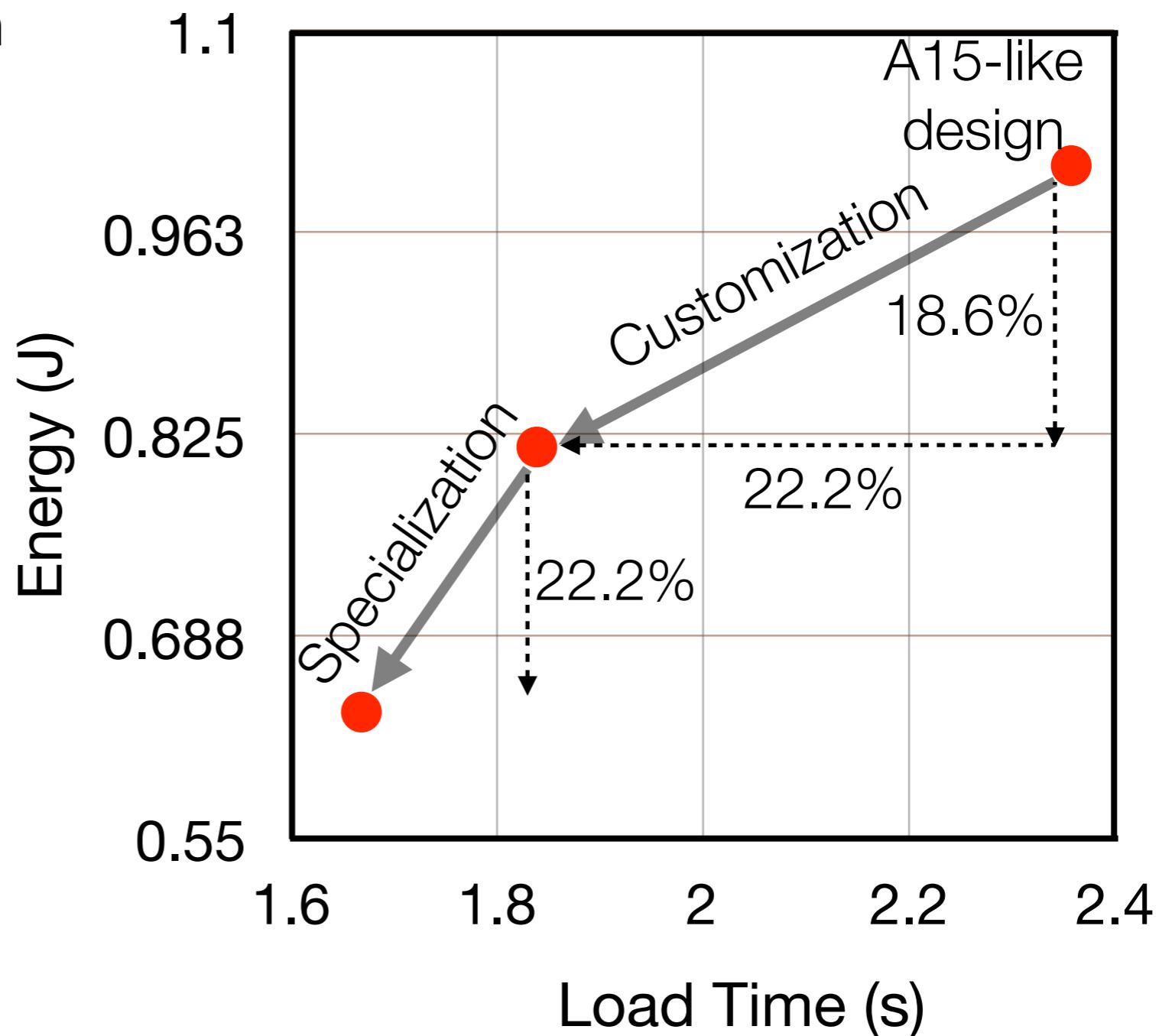
Evaluation Results

- ▶ Fully synthesized using Synopsys 28 nm toolchain
- ▶ Cost of specialization:
0.59 mm² area overhead
 - ▷ SoC die area is 122 mm² in Samsung Galaxy S4



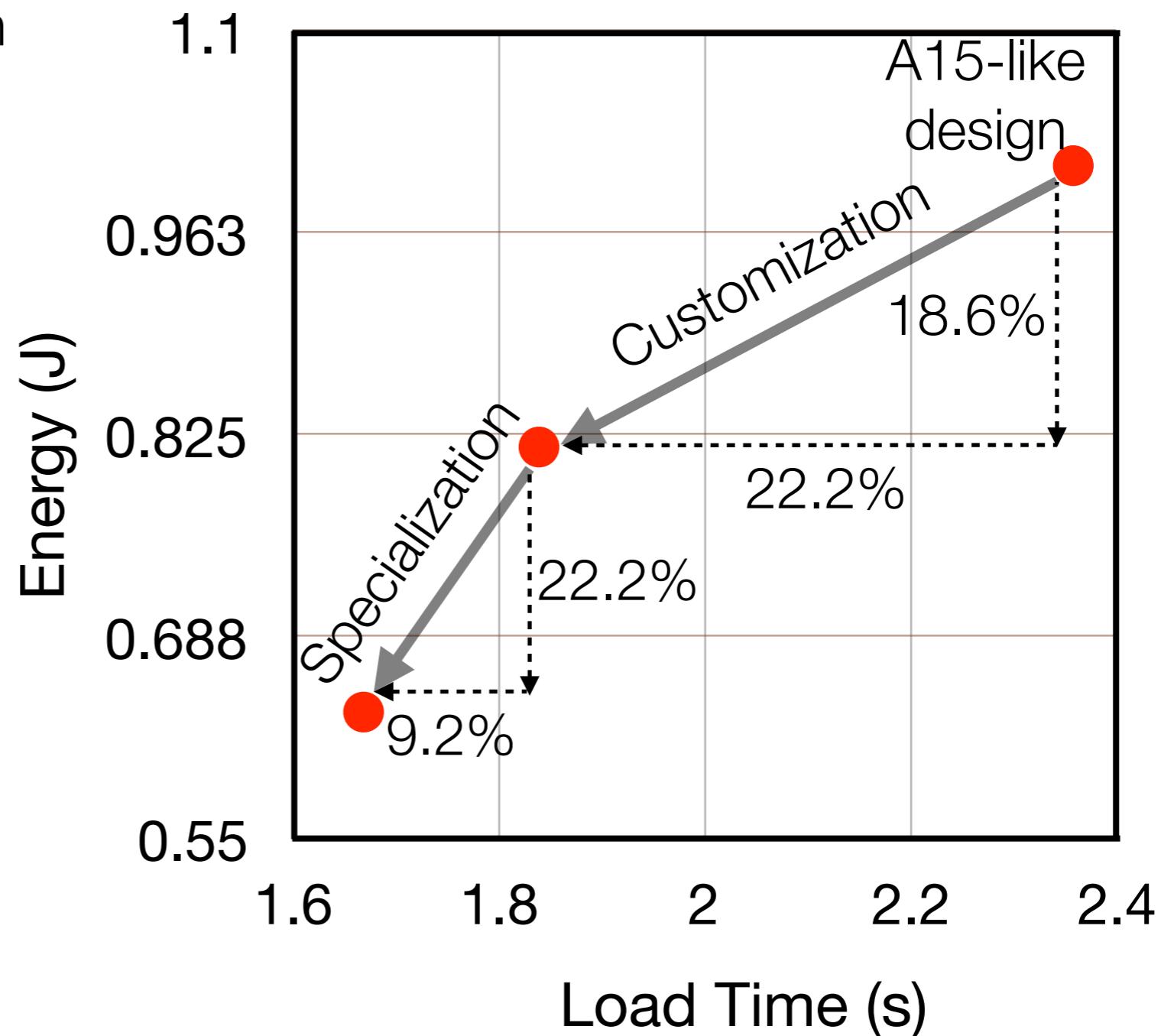
Evaluation Results

- ▶ Fully synthesized using Synopsys 28 nm toolchain
- ▶ Cost of specialization:
0.59 mm² area overhead
 - ▷ SoC die area is 122 mm² in Samsung Galaxy S4



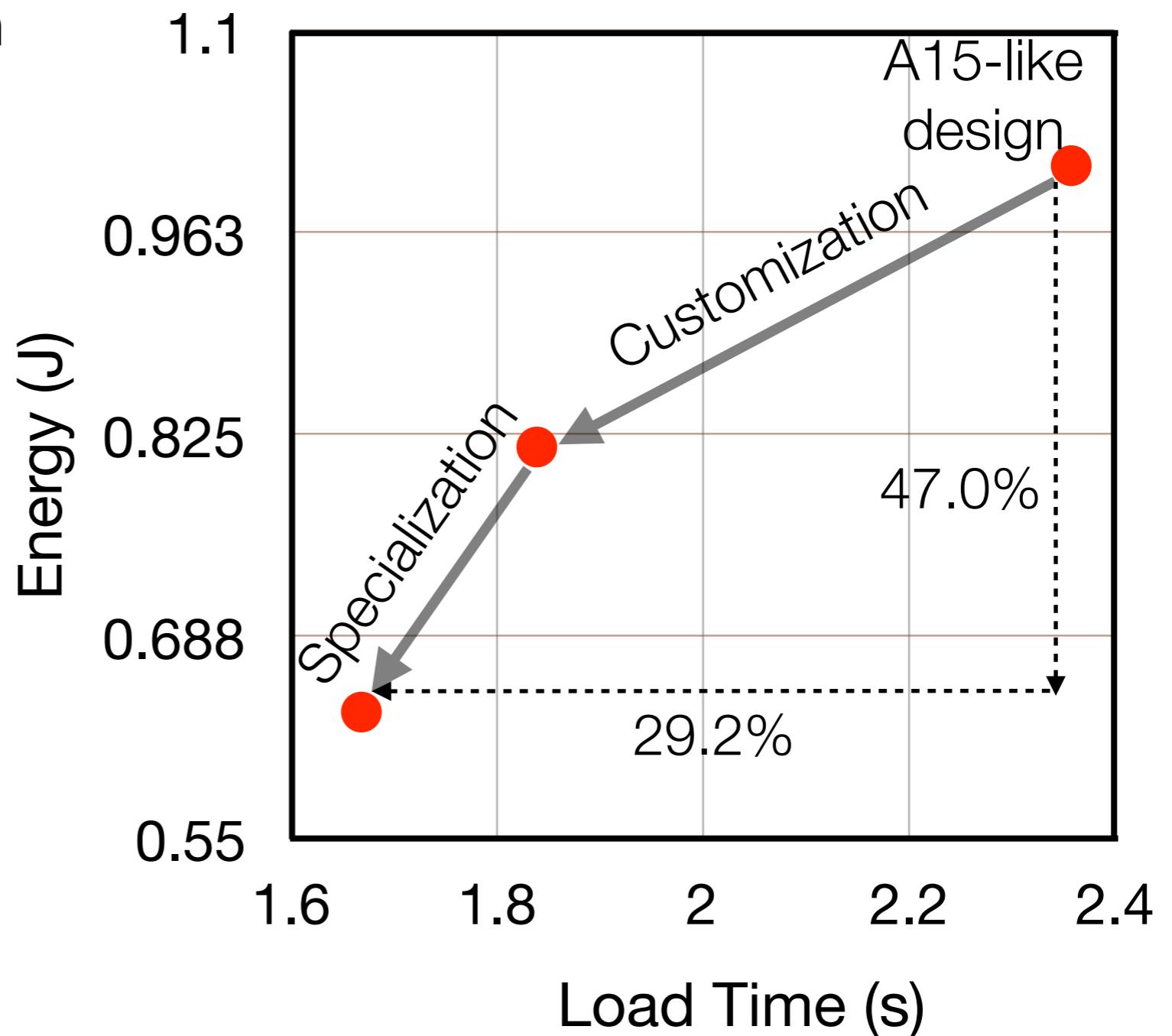
Evaluation Results

- ▶ Fully synthesized using Synopsys 28 nm toolchain
- ▶ Cost of specialization:
0.59 mm² area overhead
 - ▷ SoC die area is 122 mm² in Samsung Galaxy S4

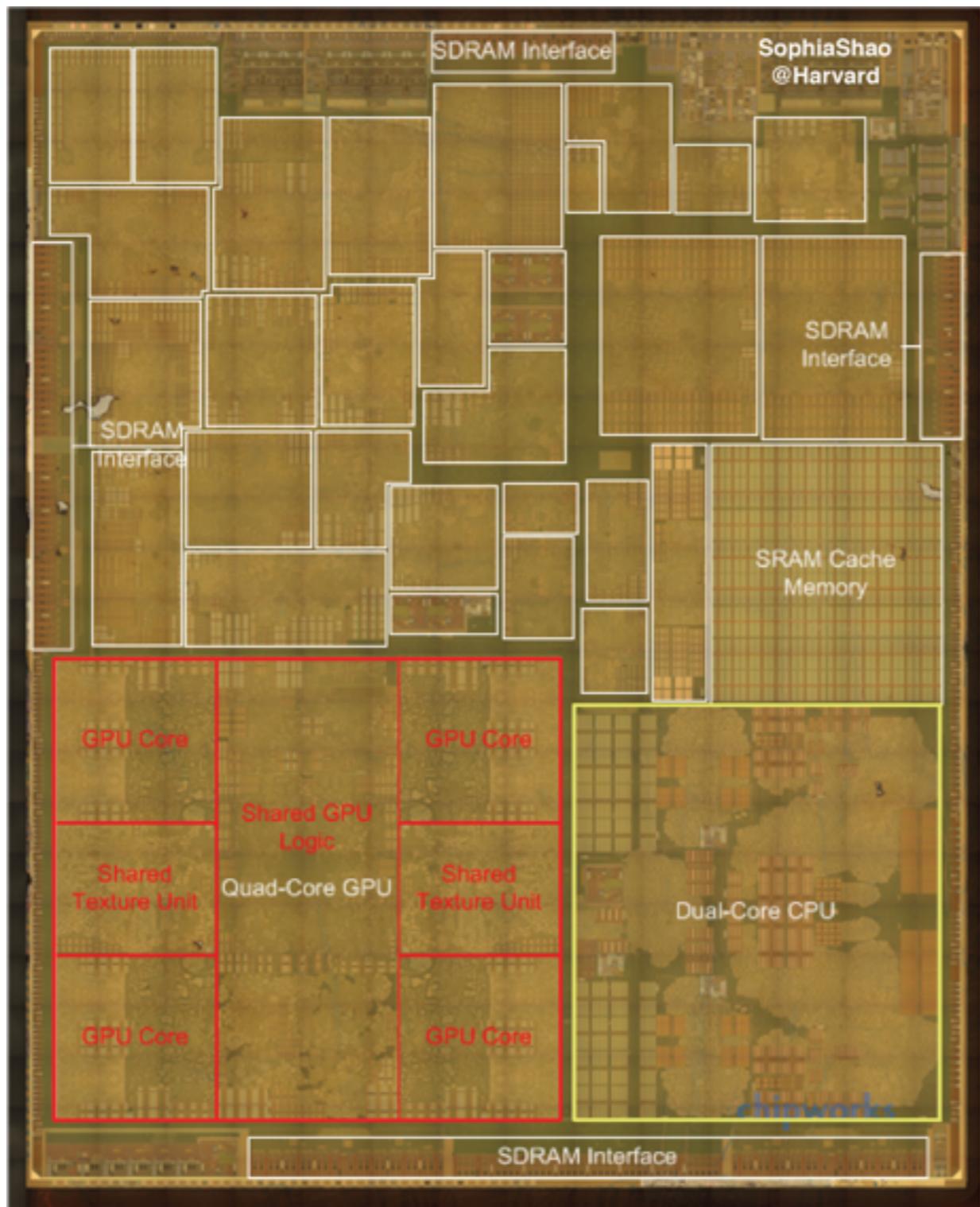


Evaluation Results

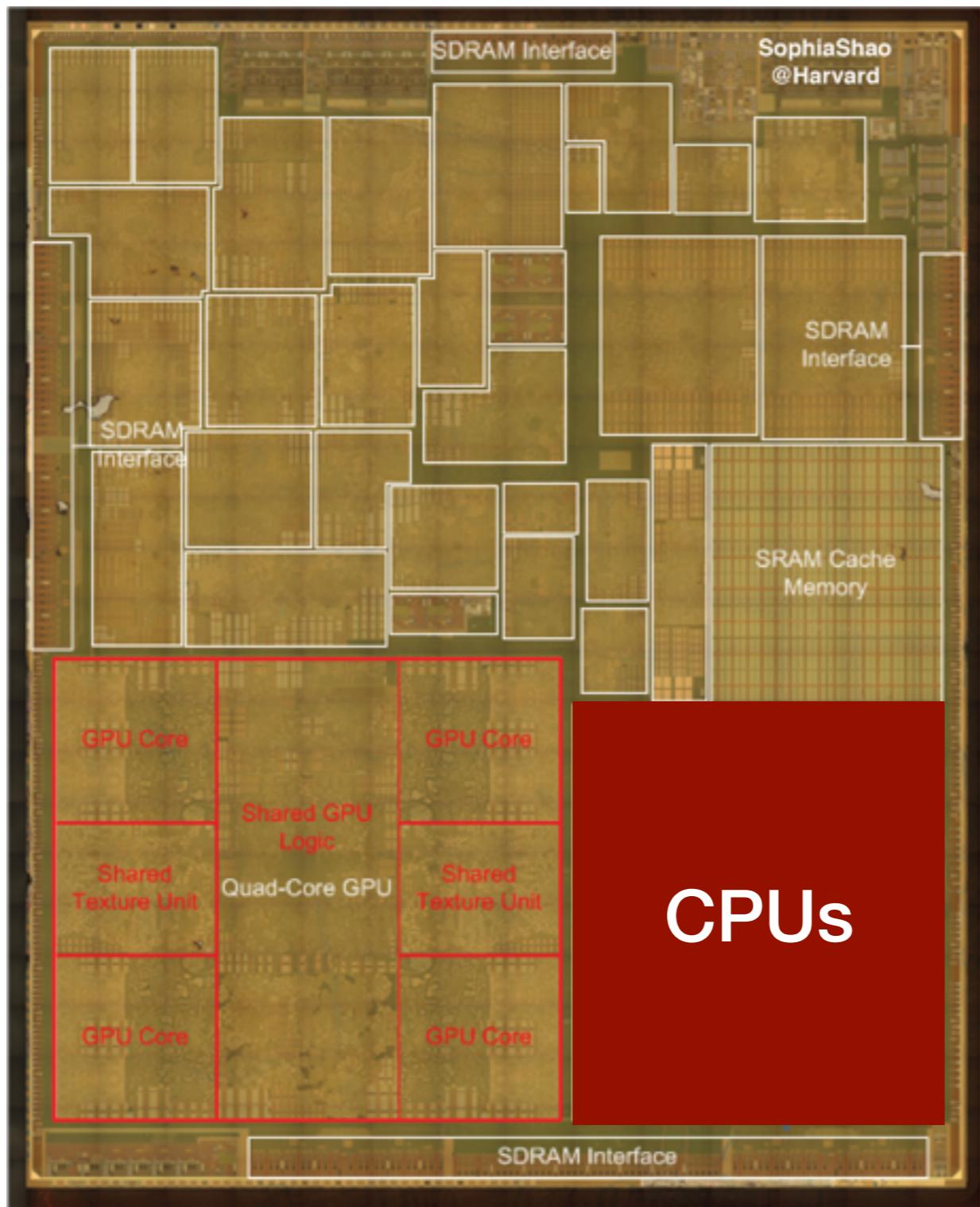
- ▶ Fully synthesized using Synopsys 28 nm toolchain
- ▶ Cost of specialization:
0.59 mm² area overhead
 - ▷ SoC die area is 122 mm² in Samsung Galaxy S4



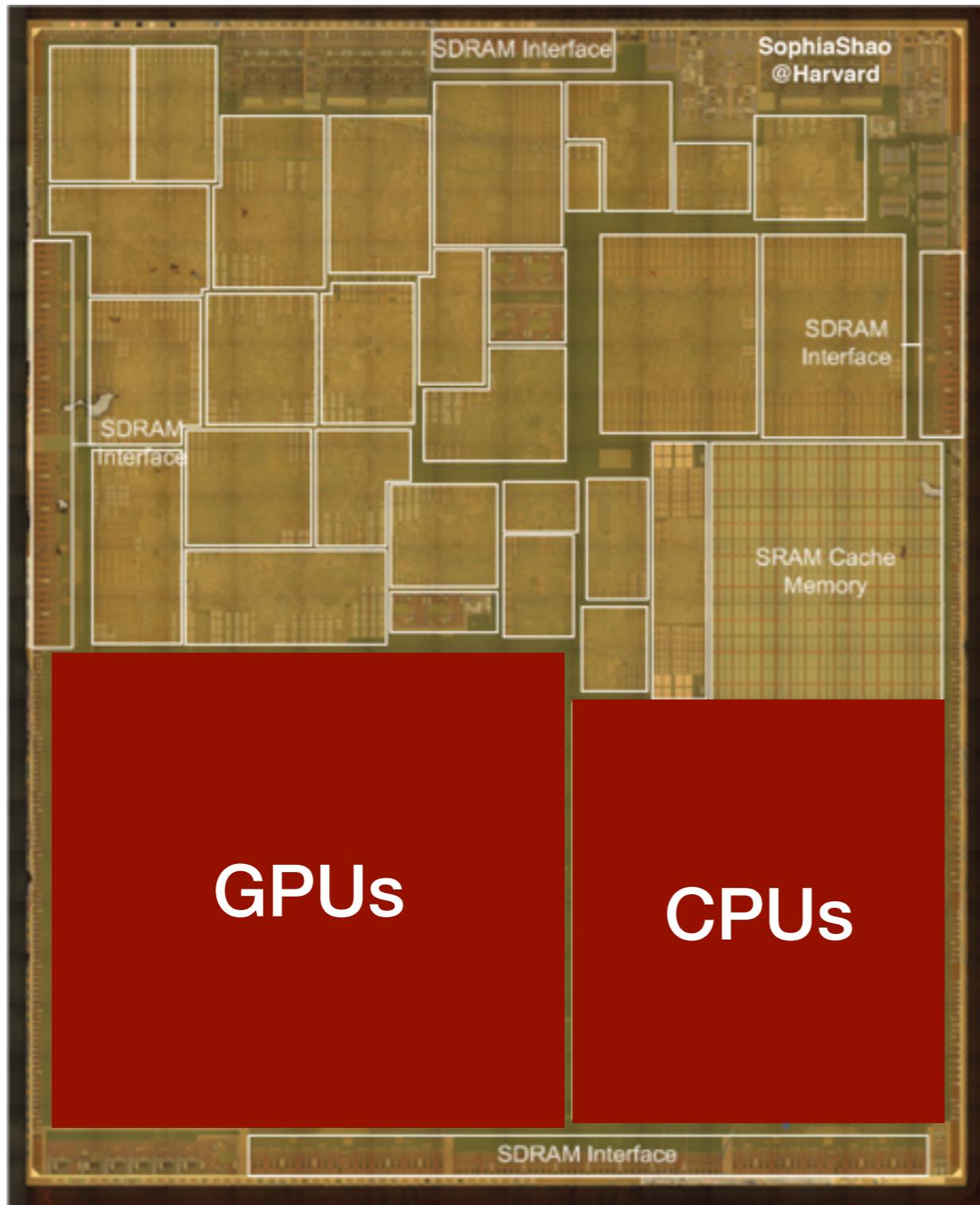
WebCore in SoC



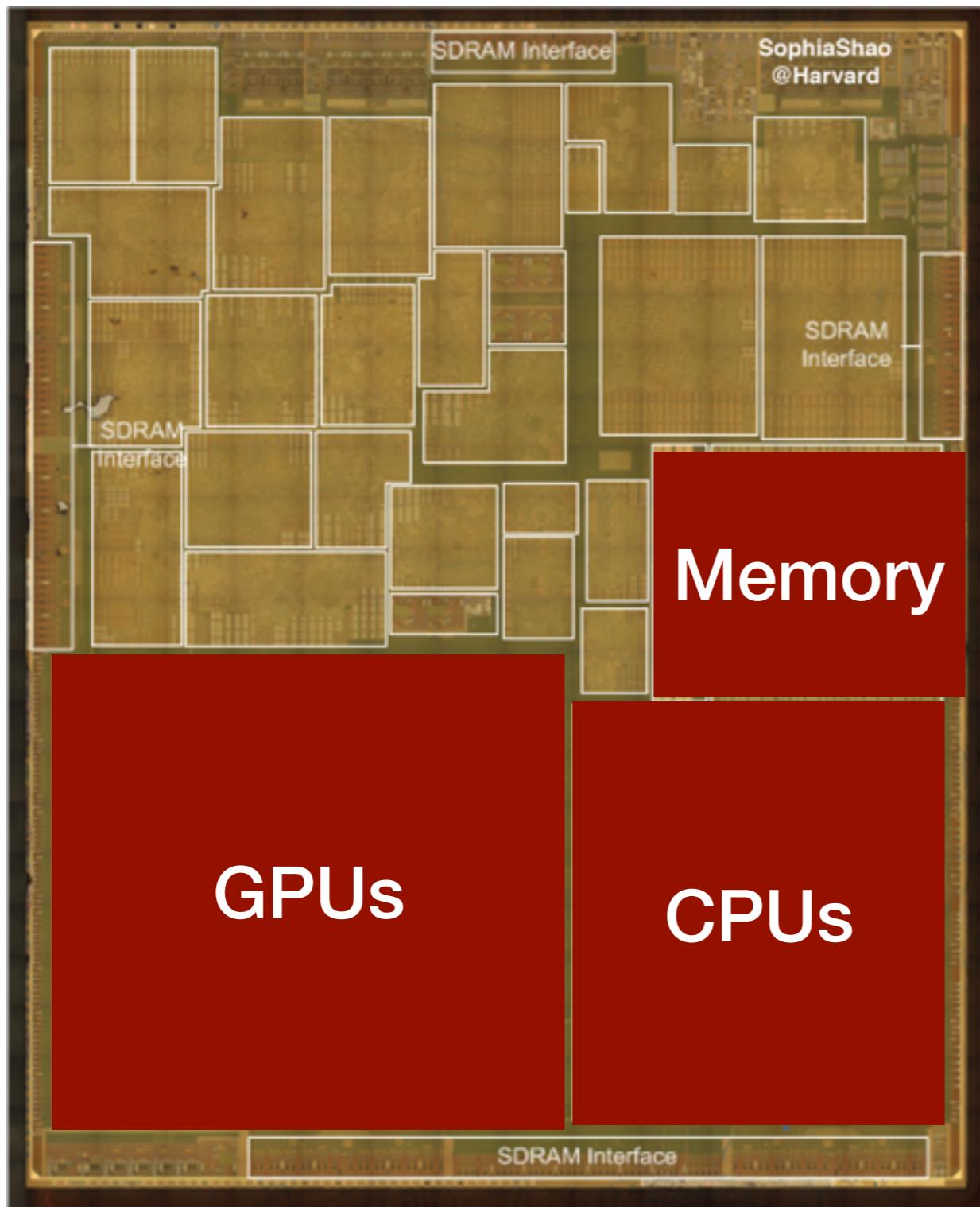
WebCore in SoC



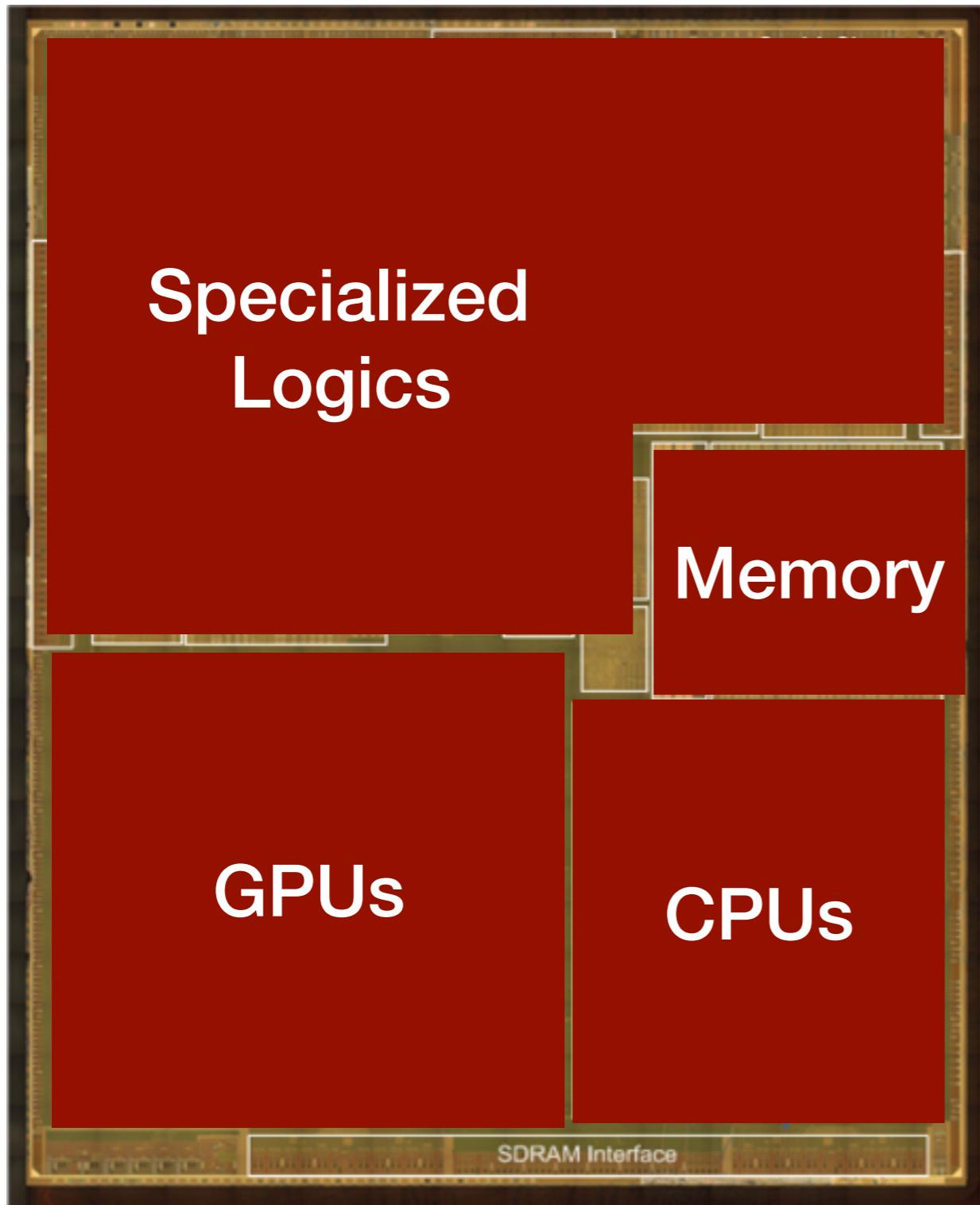
WebCore in SoC



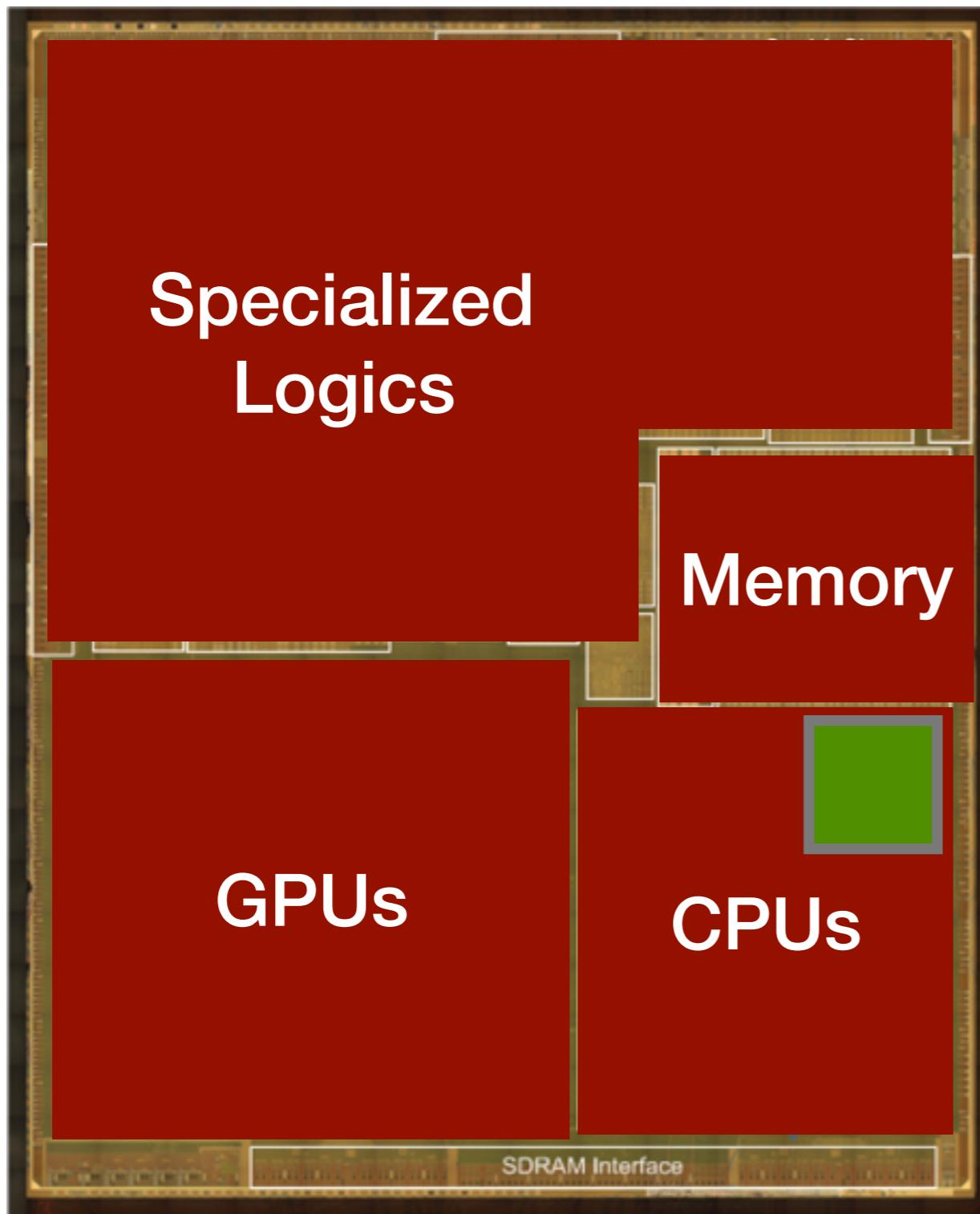
WebCore in SoC



WebCore in SoC

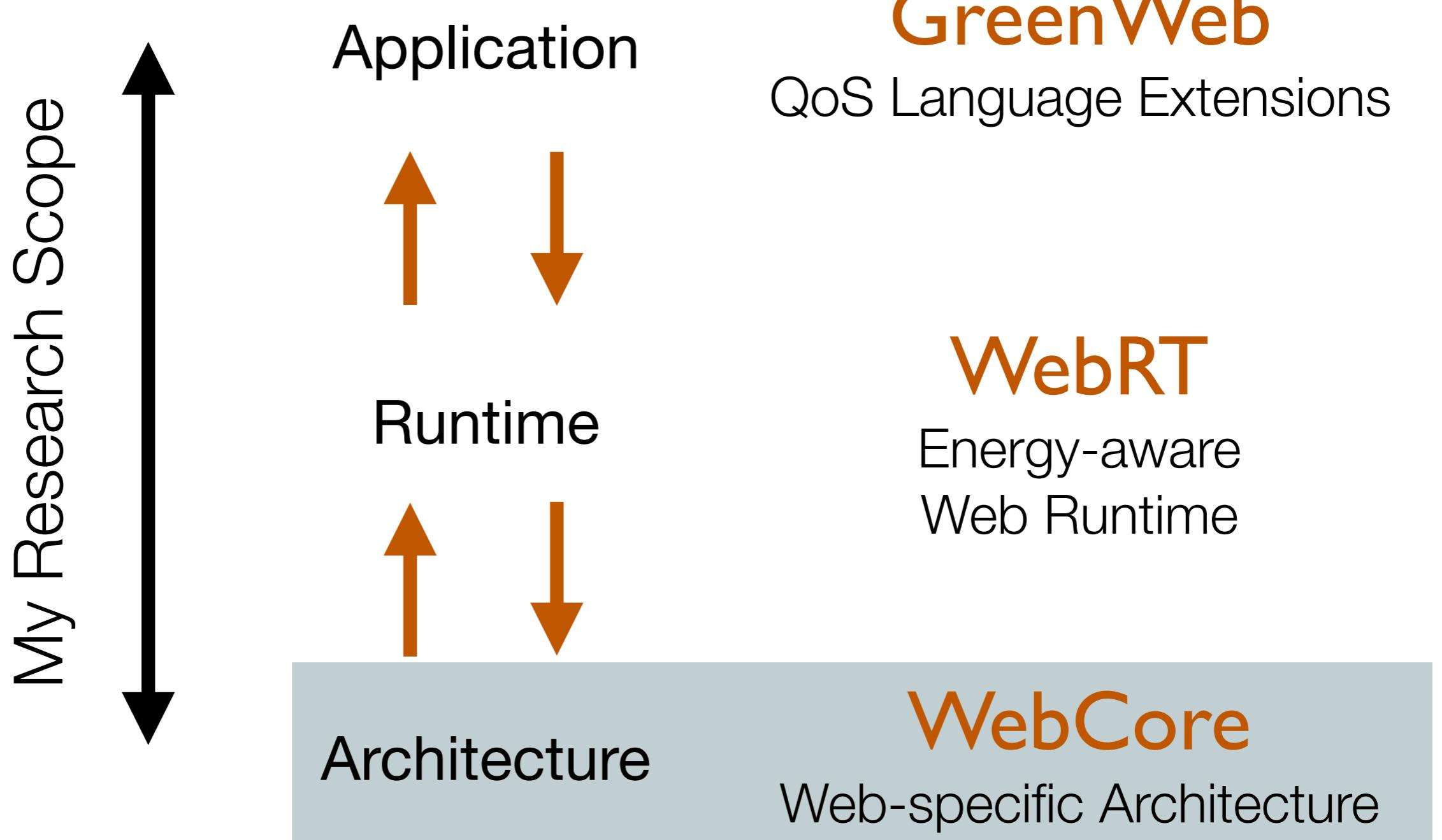


WebCore in SoC

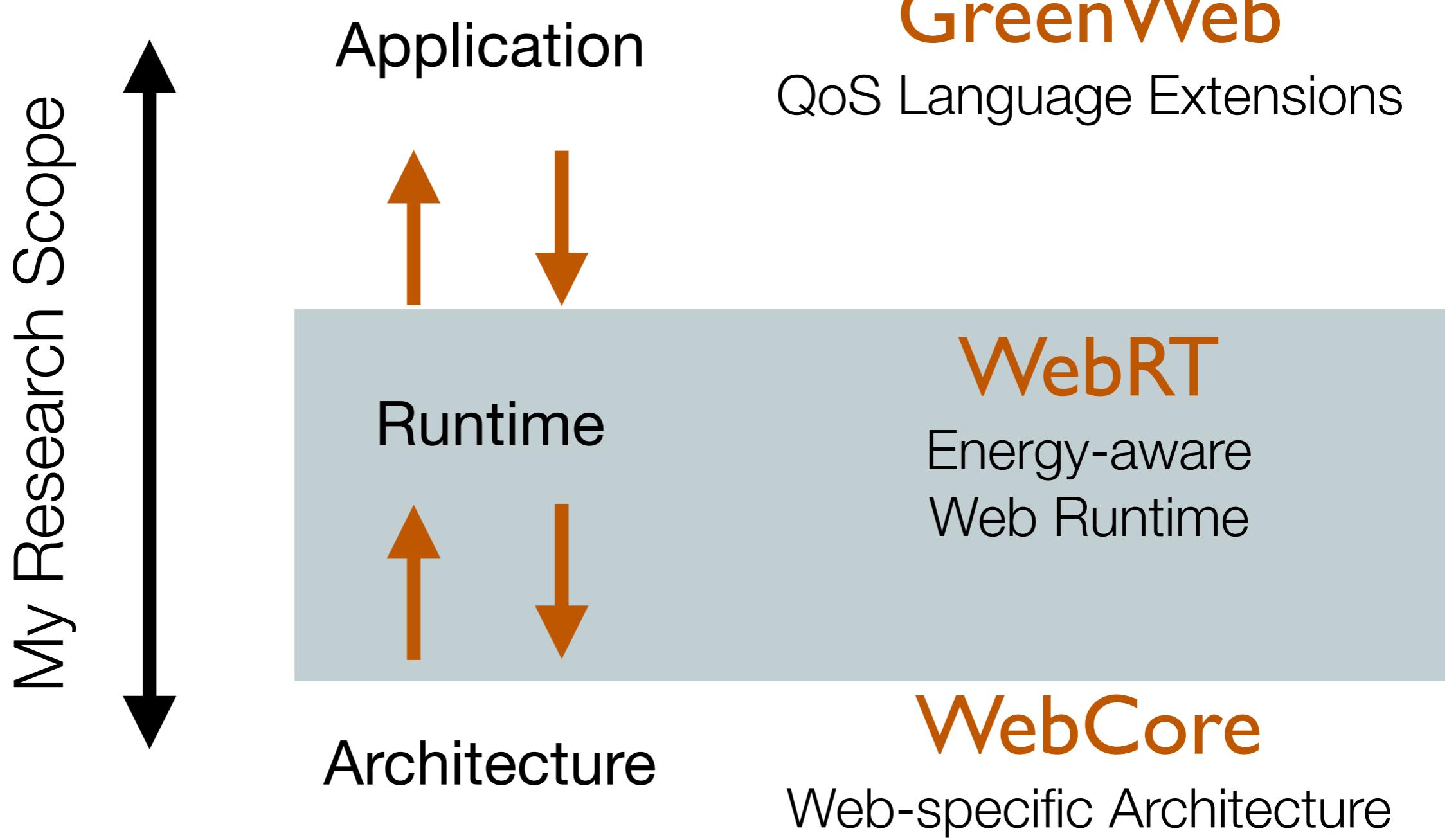


- ▶ One of the cores in the multicore SoC
 - ▶ Becomes “dark” when other applications are executing
- ← WebCore

My Approach



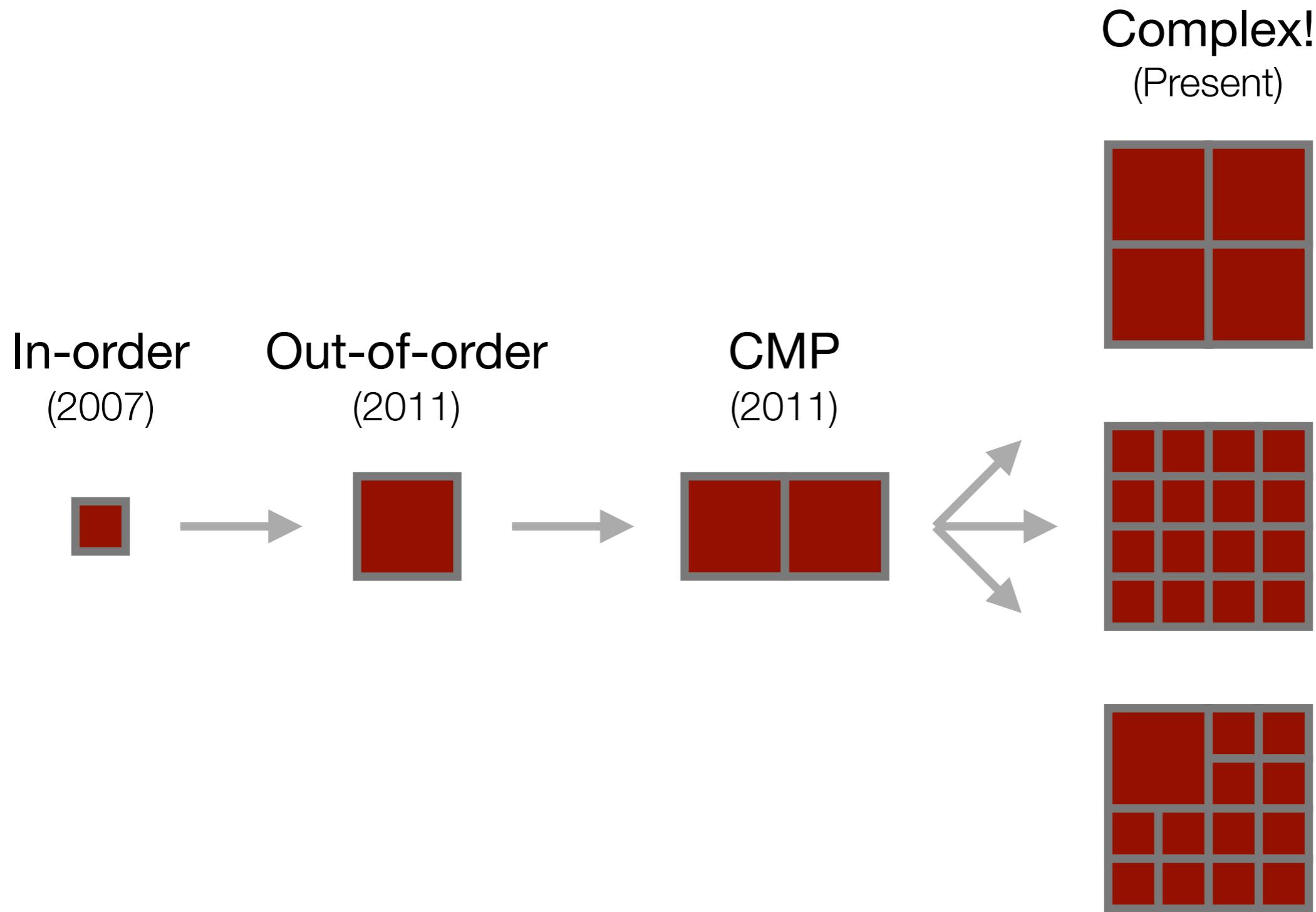
My Approach



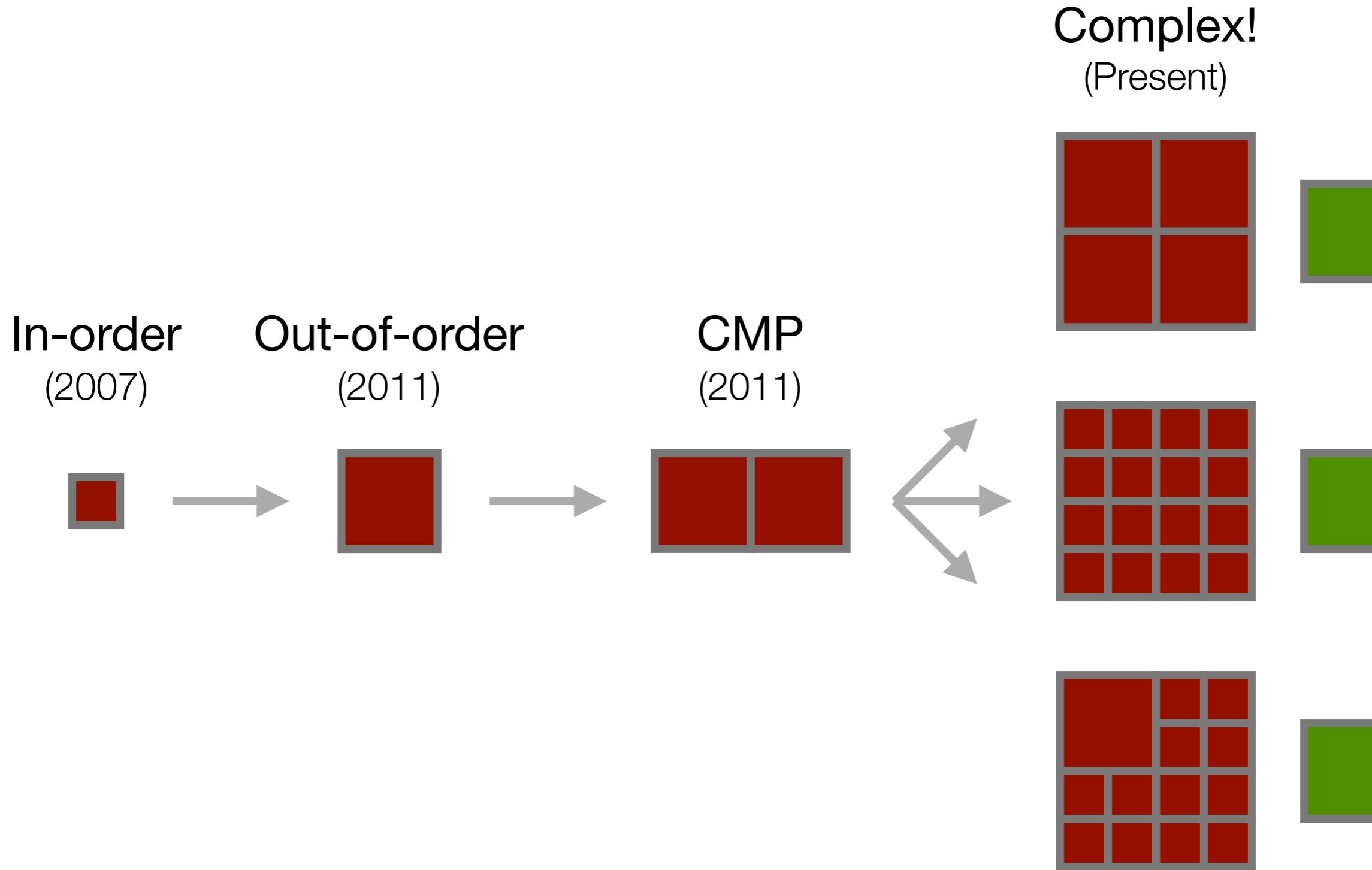
Architecture Evolution



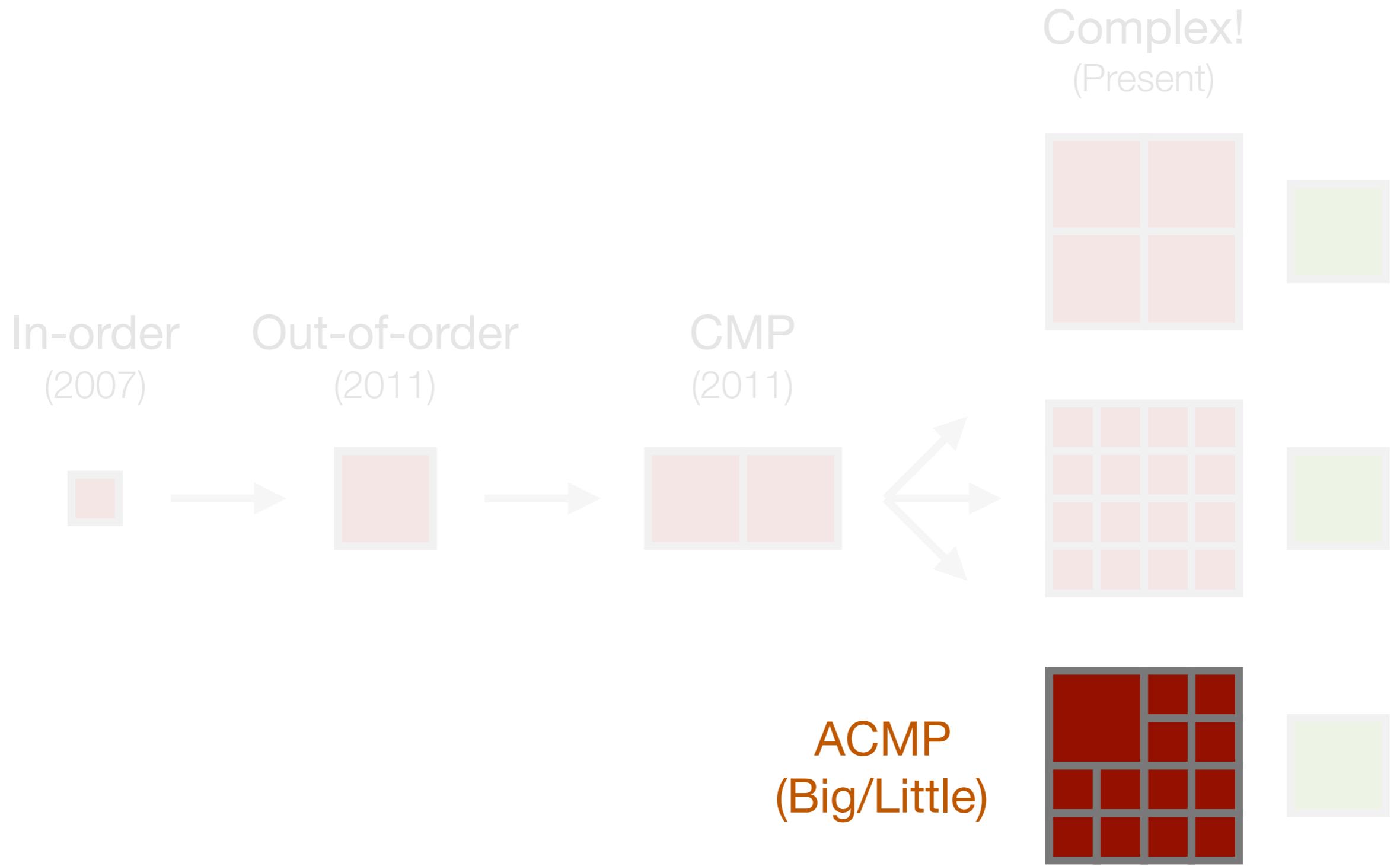
Architecture Evolution



Architecture Evolution



Architecture Evolution



WebRT: Energy-aware Web Runtime



WebRT: Energy-aware Web Runtime

- ▶ **Why ACMP?**: Offer a large performance-energy trade-off space for **energy optimizations**
 - ▷ Different microarchitectures (in-order + out-of-order)
 - ▷ Different frequency settings



WebRT: Energy-aware Web Runtime

- ▶ **Why ACMP?**: Offer a large performance-energy trade-off space for **energy optimizations**
 - ▷ Different microarchitectures (in-order + out-of-order)
 - ▷ Different frequency settings
- ▶ **Idea**: Provide just-enough energy to meet performance target



WebRT: Energy-aware Web Runtime

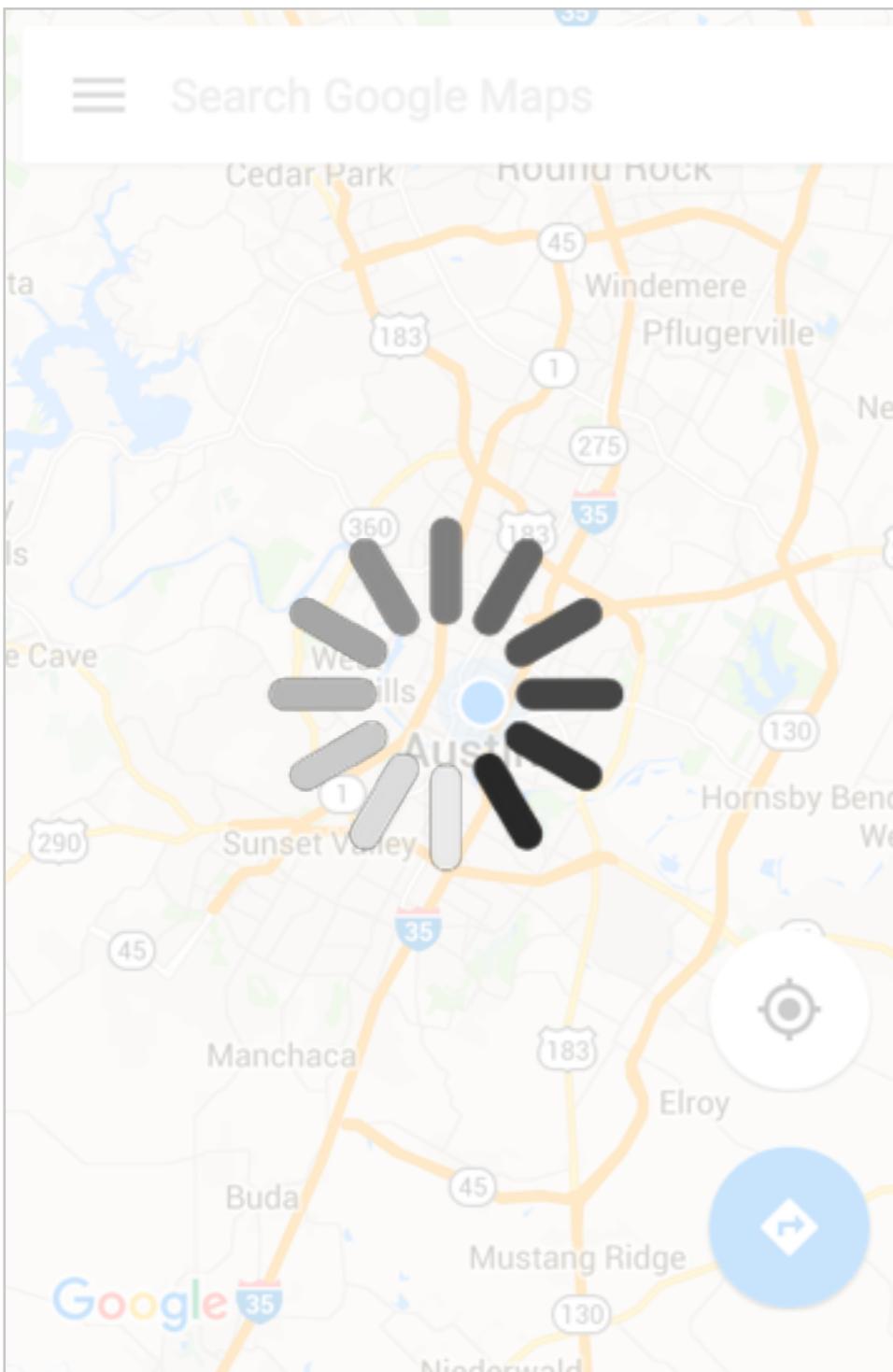
- ▶ **Why ACMP?**: Offer a large performance-energy trade-off space for **energy optimizations**
 - ▷ Different microarchitectures (in-order + out-of-order)
 - ▷ Different frequency settings
- ▶ **Idea**: Provide just-enough energy to meet performance target
- ▶ **Approach**: Systematically understand user interactions and bridge the gap between user behavior and system execution.



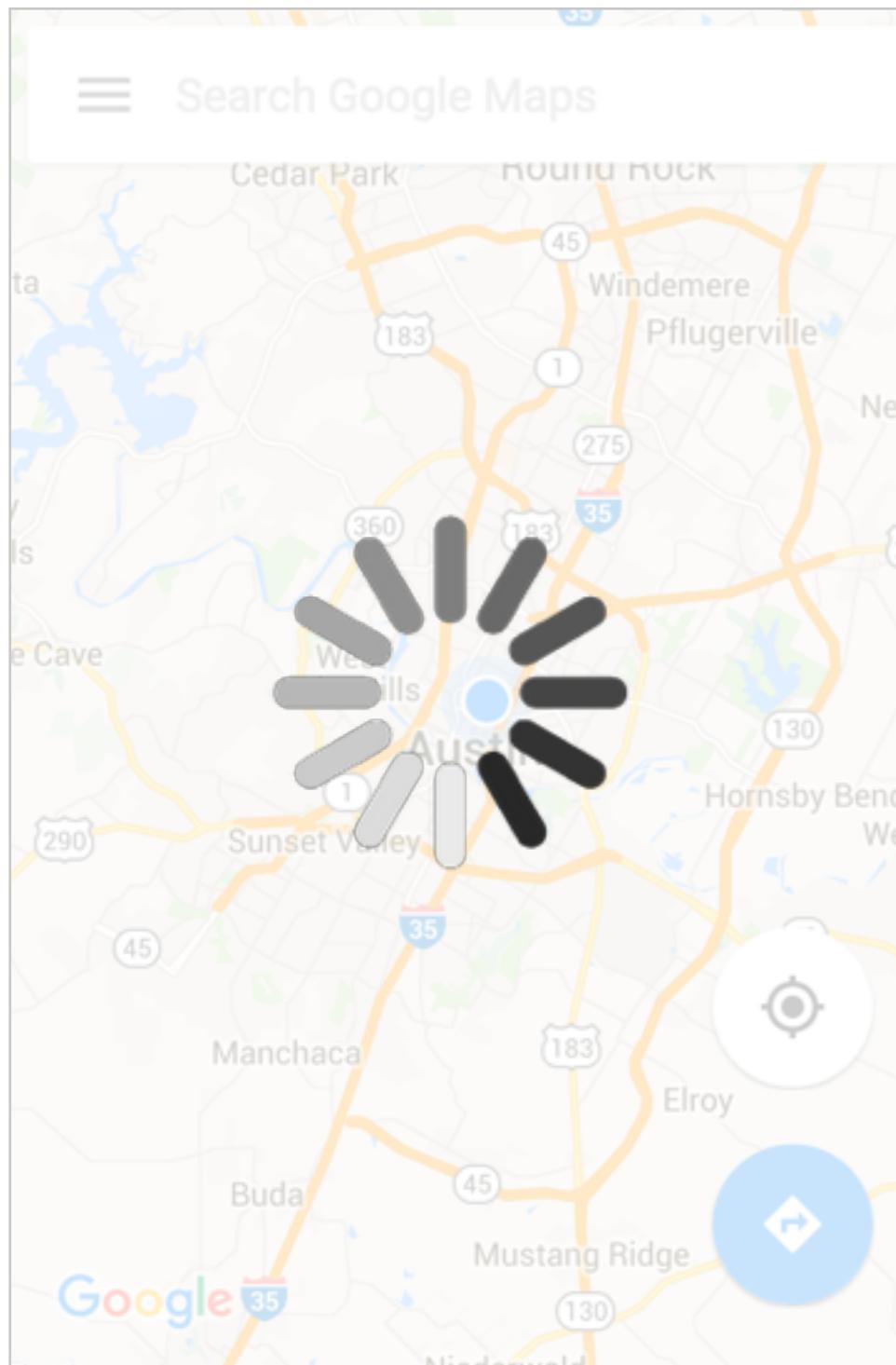
Interacting With a Mobile Web Application



Interacting With a Mobile Web Application

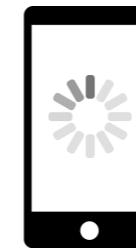


Interacting With a Mobile Web Application

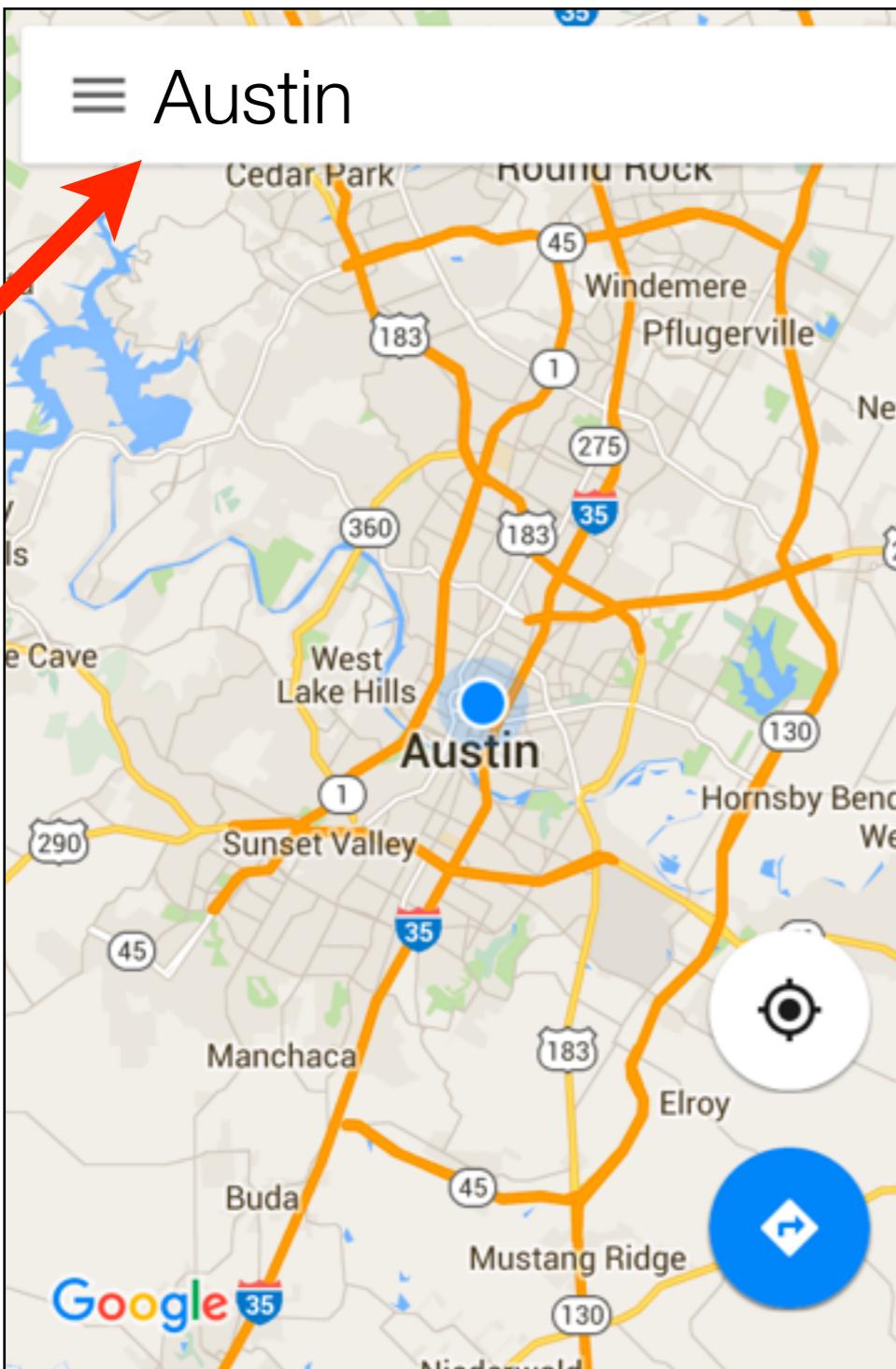


Interactions

Loading

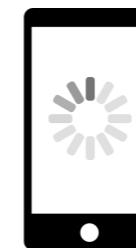


Interacting With a Mobile Web Application

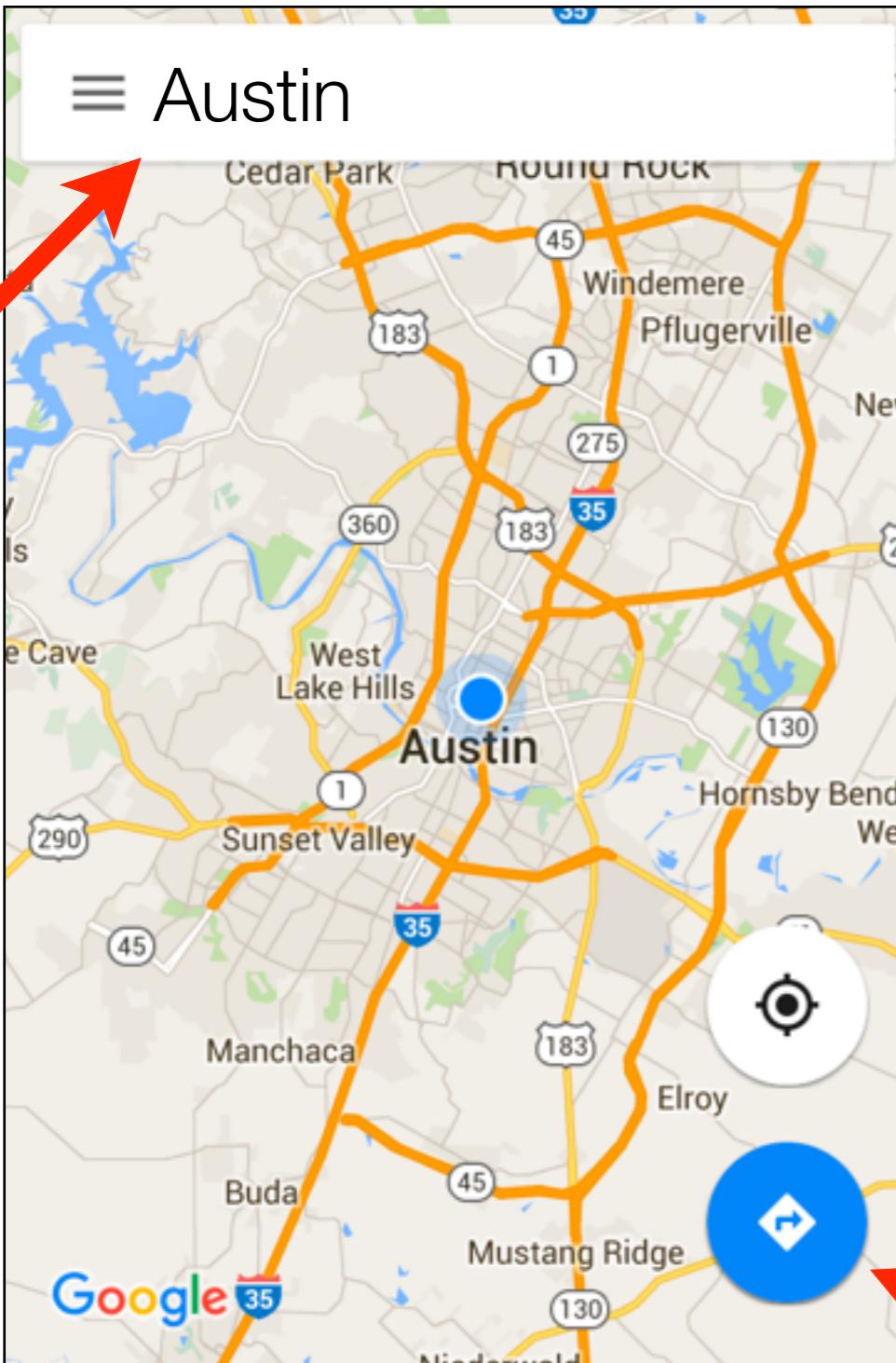


Interactions

Loading

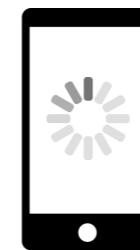


Interacting With a Mobile Web Application

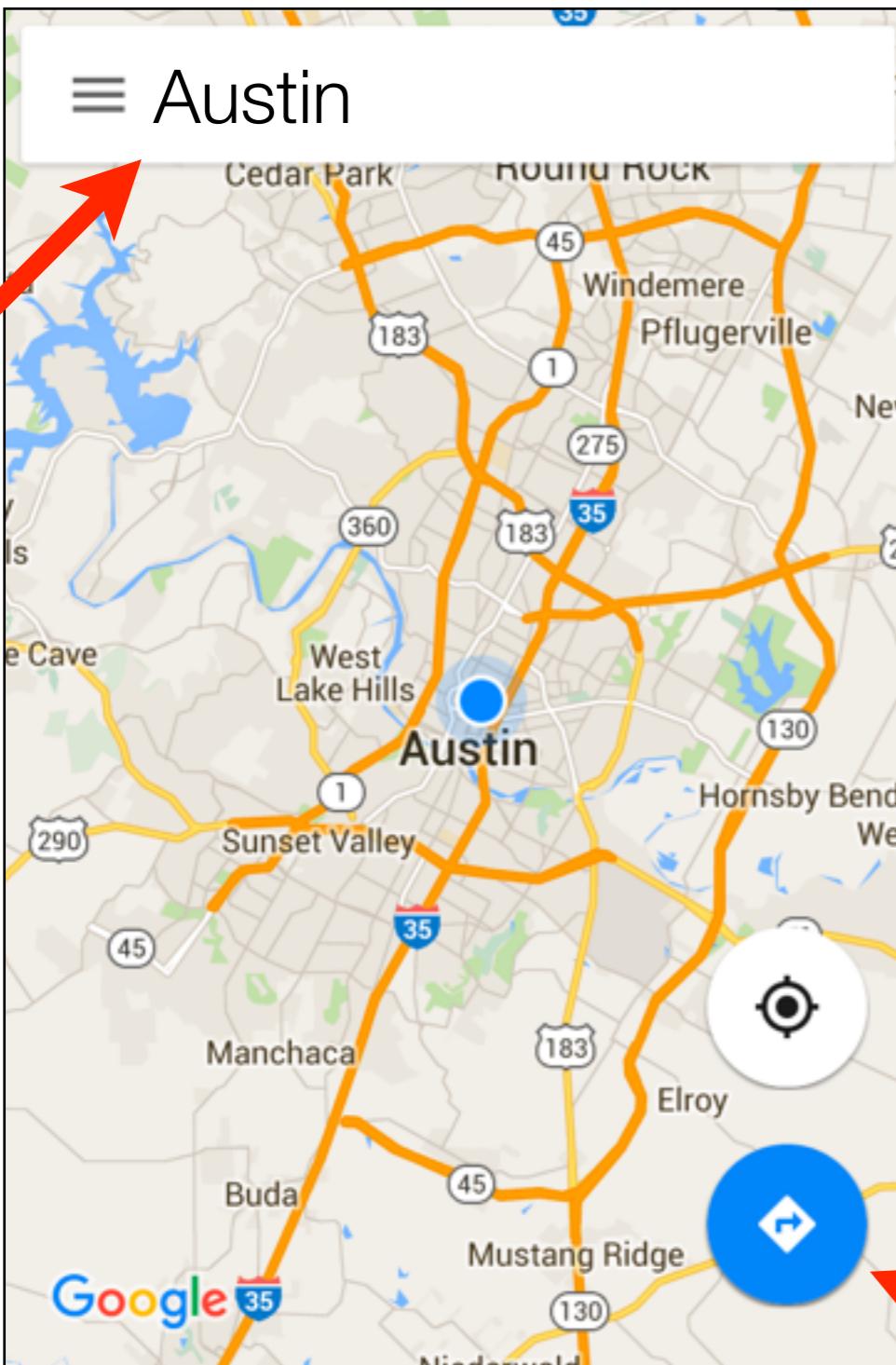


Interactions

Loading

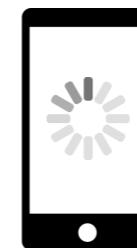


Interacting With a Mobile Web Application



Interactions

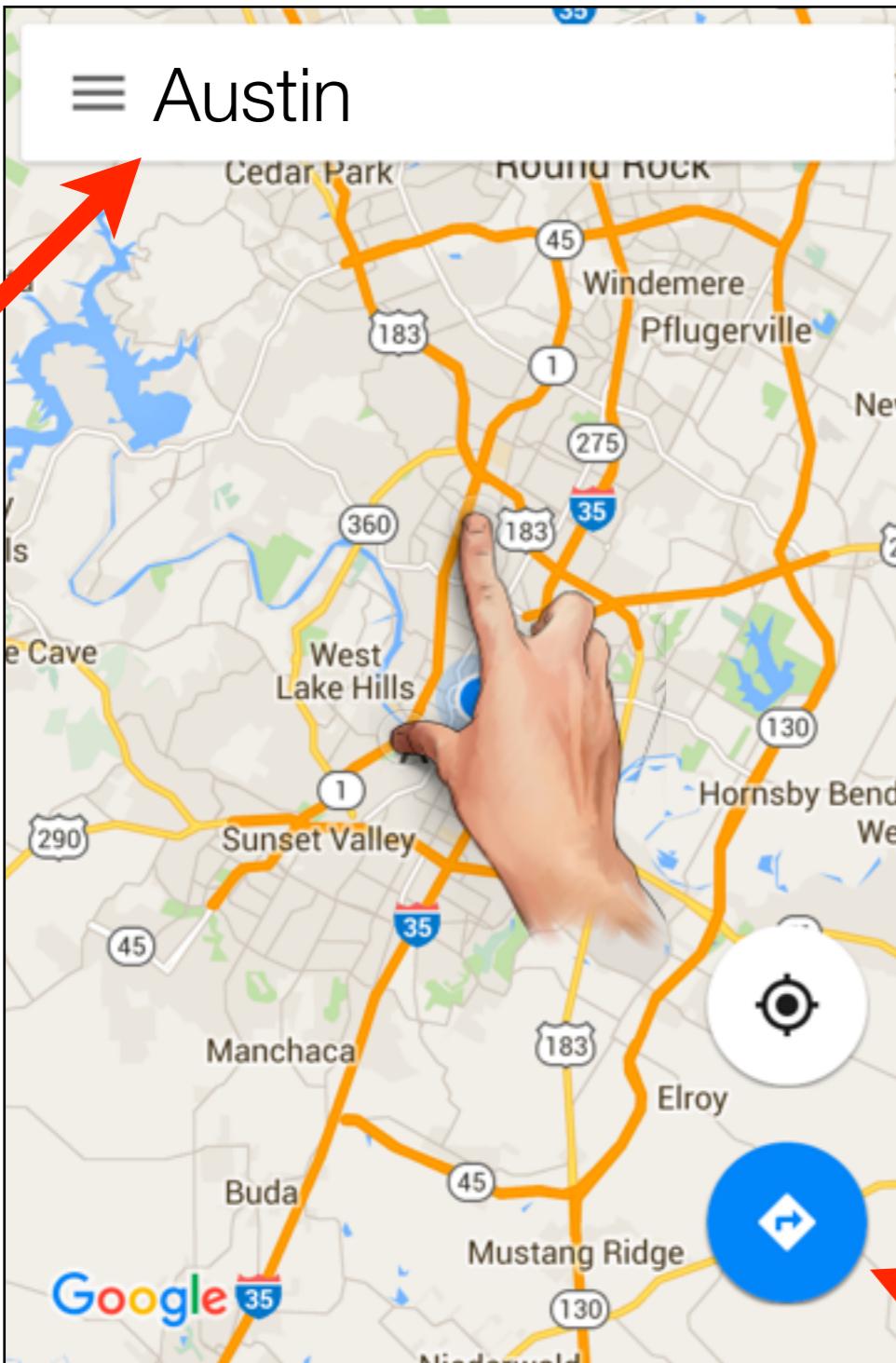
Loading



Touching

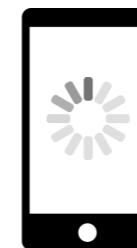


Interacting With a Mobile Web Application



Interactions

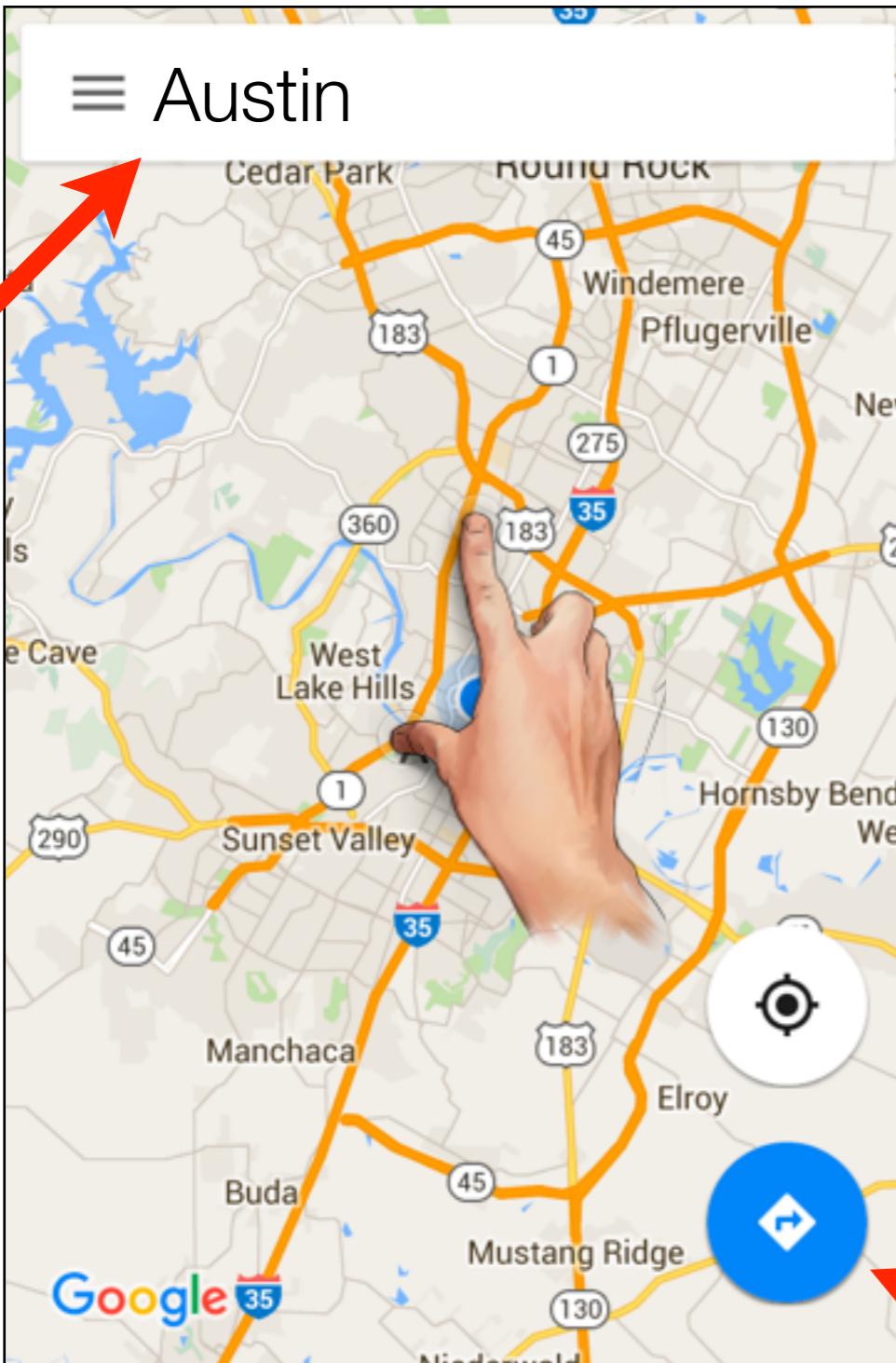
Loading



Touching

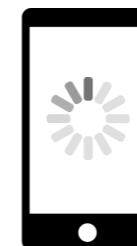


Interacting With a Mobile Web Application



Interactions

Loading



Touching



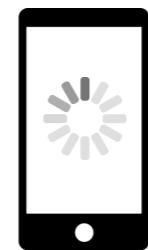
Moving



Interacting With a Mobile Web Application

Interactions

Loading



Touching



Moving



Interacting With a Mobile Web Application

Interactions

Loading



Once per a
usage session

Touching



Moving



Interacting With a Mobile Web Application

Interactions

WebRT
Component

Loading



Proactive
Mechanism

Touching



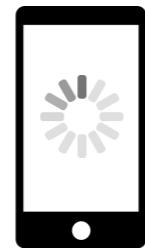
Moving



Interacting With a Mobile Web Application

Interactions

Loading



Touching



Moving



WebRT
Component

Proactive
Mechanism

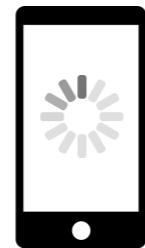
Repetitive in
a usage session



Interacting With a Mobile Web Application

Interactions

Loading



Touching



Moving



WebRT
Component

Proactive
Mechanism

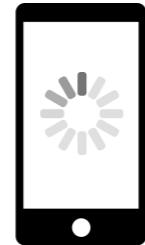
History-
based
Mechanism



WebRT: Energy-aware Web Runtime

Interactions

Loading



Touching



Moving



WebRT
Component

Proactive
Mechanism

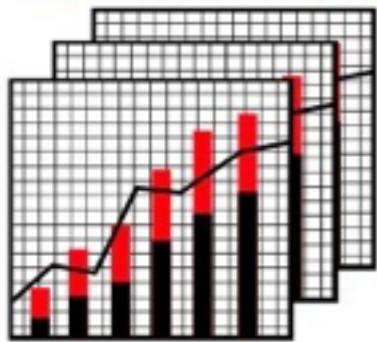
History-
based
Mechanism



Optimizing for Loading



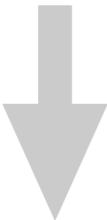
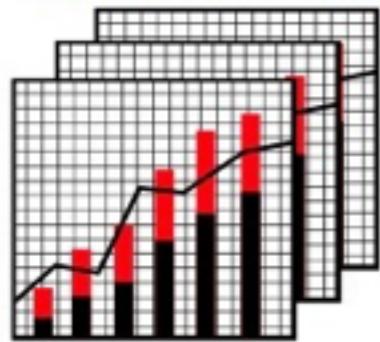
Optimizing for Loading



- ▶ Observation: Web applications have different characteristics that lead to different loading times and energy consumptions

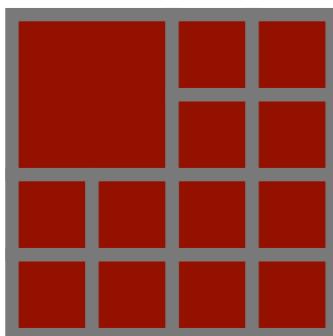
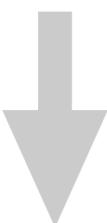
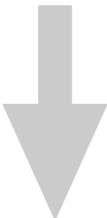
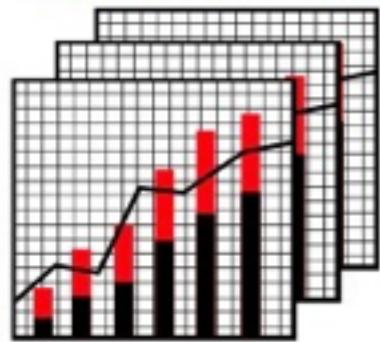


Optimizing for Loading



- ▶ **Observation:** Web applications have different characteristics that lead to different loading times and energy consumptions
- ▶ **Mechanism:** Predict the ideal ACMP configuration (<core, frequency>) and schedule application loading accordingly

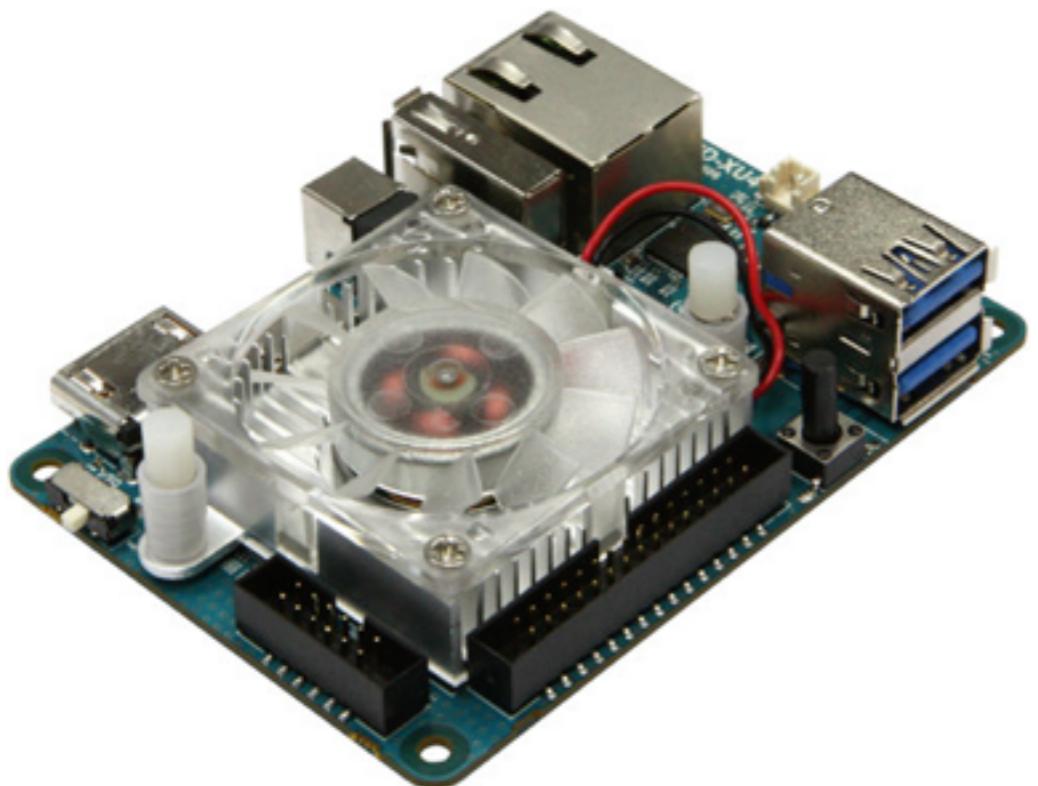
Optimizing for Loading



- ▶ **Observation:** Web applications have different characteristics that lead to different loading times and energy consumptions
- ▶ **Mechanism:** Predict the ideal ACMP configuration (<core, frequency>) and schedule application loading accordingly
- ▶ **Effect:** Properly provision the hardware resources based on application characteristics

Big/Little Setup

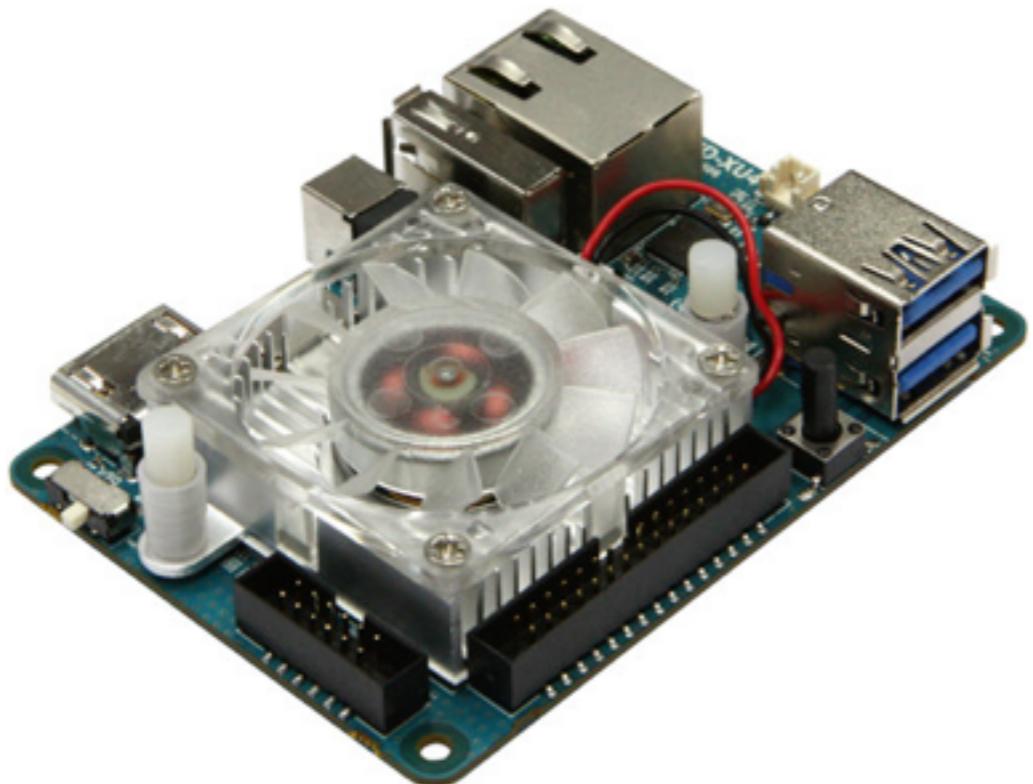
ODroid XU+E development board,
which contains an Exynos 5410 SoC
used in Samsung Galaxy S4.



Big/Little Setup

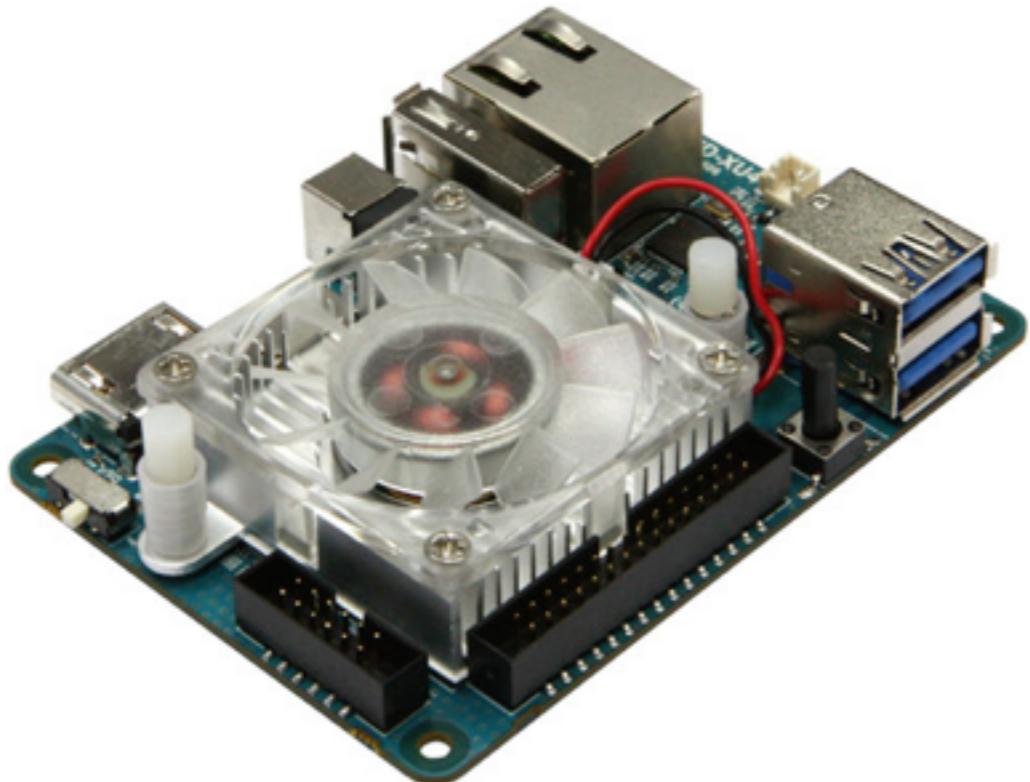
ODroid XU+E development board,
which contains an Exynos 5410 SoC
used in Samsung Galaxy S4.

Big core cluster: ARM Cortex A15, OoO with 3 issue
DVFS: 800 MHz ~ 1.8 GHz at a 100 MHz granularity



Big/Little Setup

ODroid XU+E development board,
which contains an Exynos 5410 SoC
used in Samsung Galaxy S4.

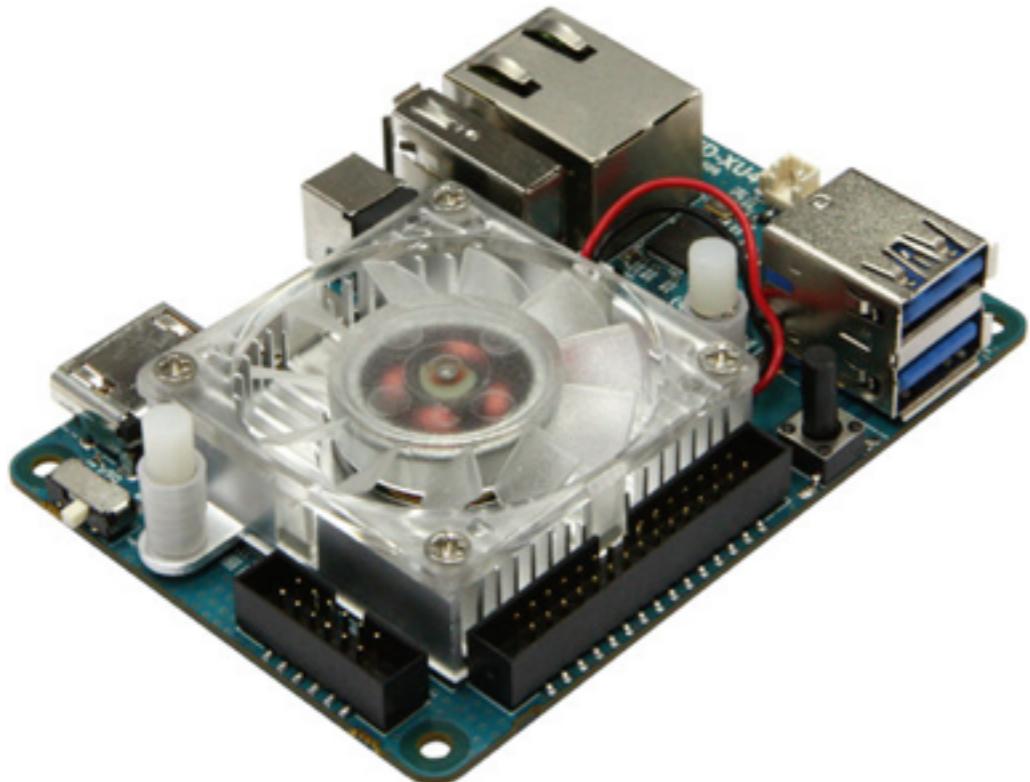


Big core cluster: ARM Cortex A15, OoO with 3 issue
DVFS: 800 MHz ~ 1.8 GHz at a 100 MHz granularity

Little core cluster: ARM Cortex A7, In-order with 2 issue
DVFS: 350 MHz ~ 600 MHz at a 50 MHz granularity

Big/Little Setup

ODroid XU+E development board,
which contains an Exynos 5410 SoC
used in Samsung Galaxy S4.



Big core cluster: ARM Cortex A15, OoO with 3 issue

DVFS: 800 MHz ~ 1.8 GHz at a 100 MHz granularity

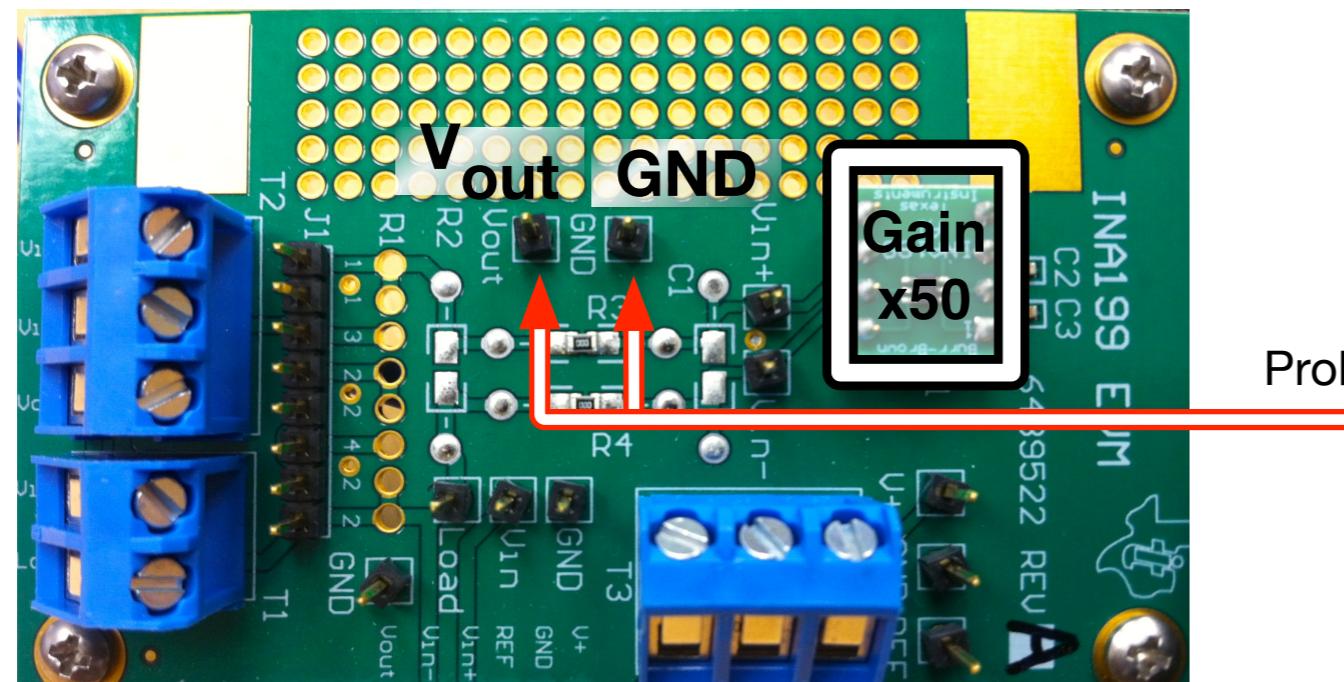
Little core cluster: ARM Cortex A7, In-order with 2 issue

DVFS: 350 MHz ~ 600 MHz at a 50 MHz granularity

Overhead:

- ▶ Frequency switch: 100 us
- ▶ Core migration: 20 us

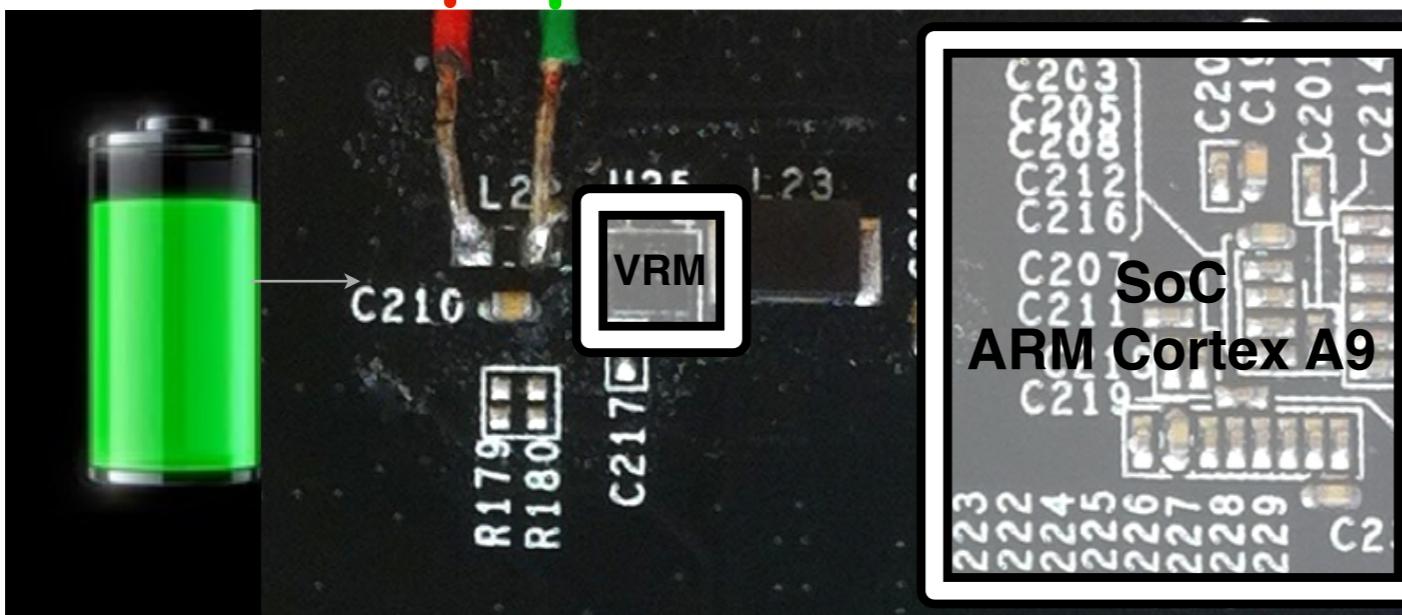
Power and Energy Measurements



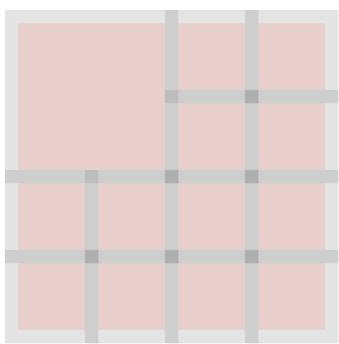
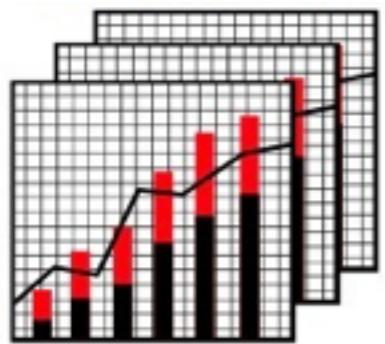
Probe



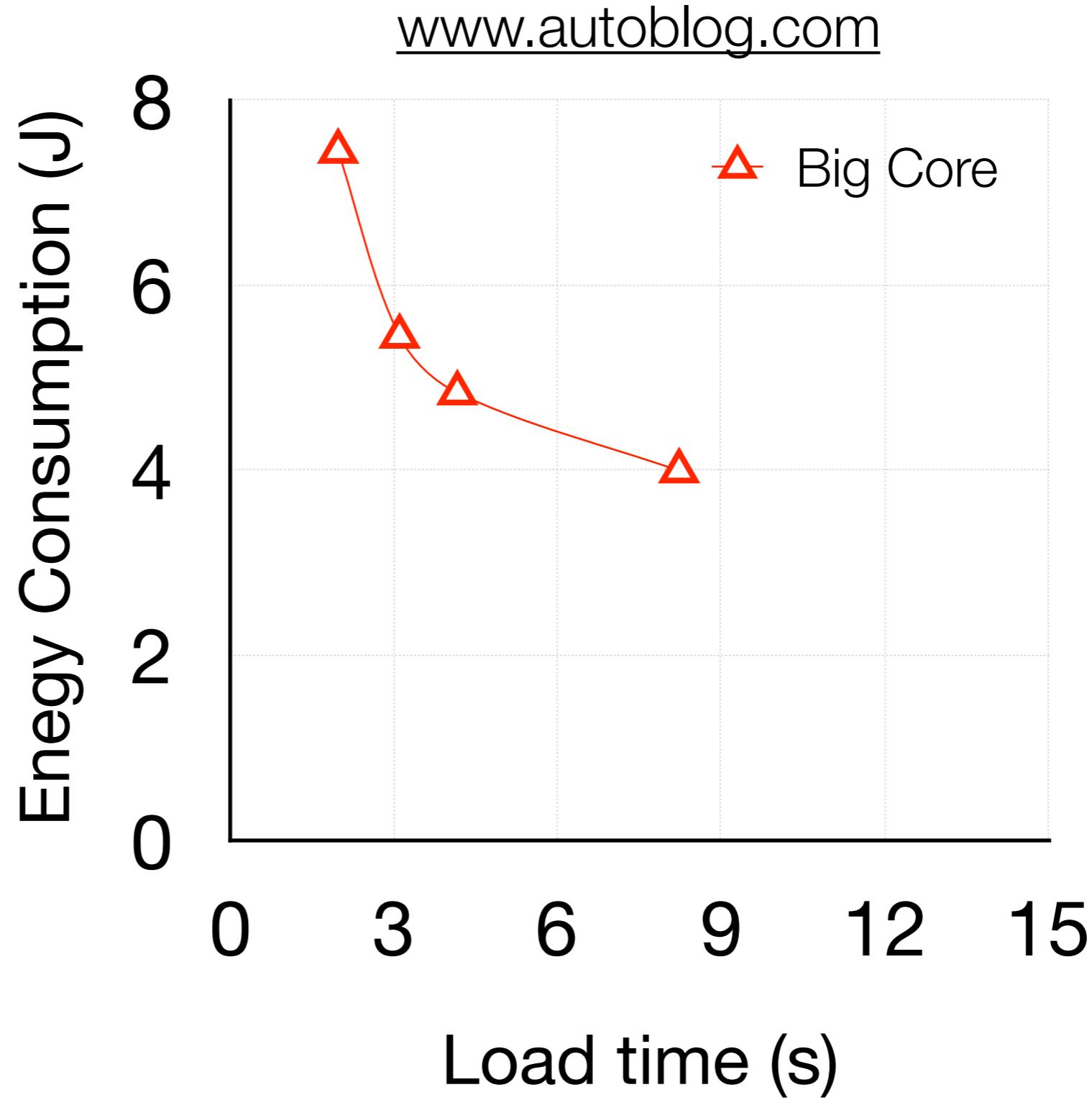
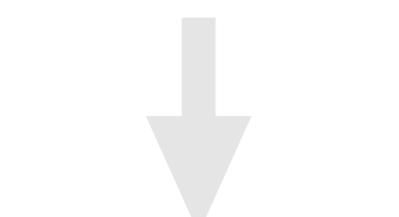
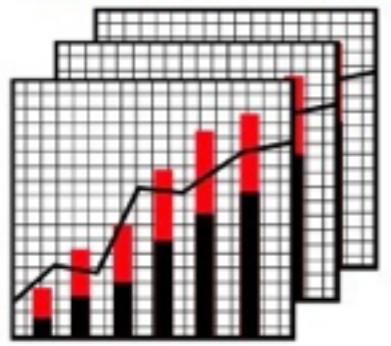
Data Acquisition (DAQ)



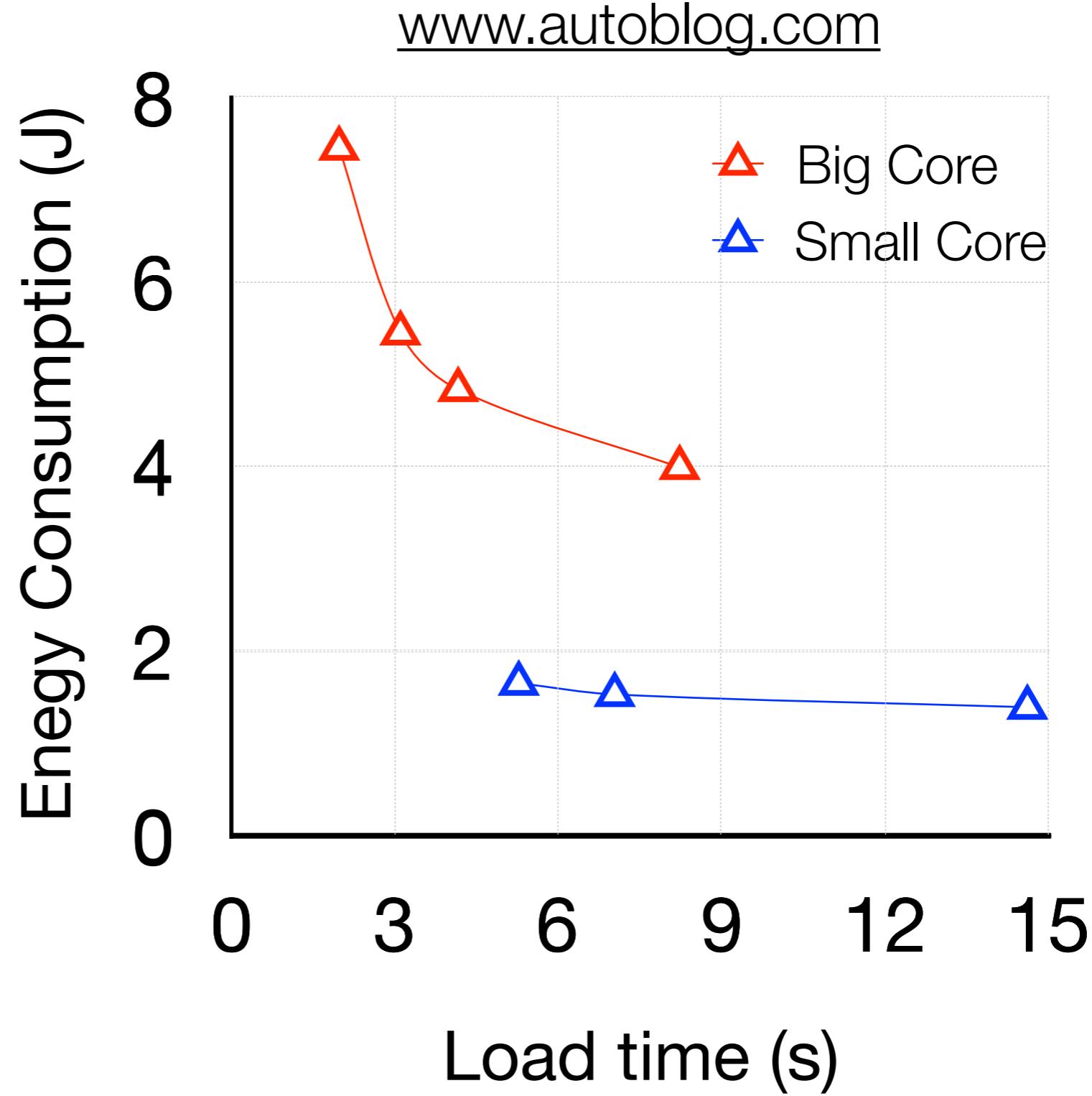
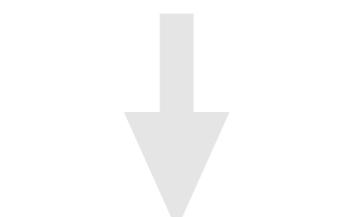
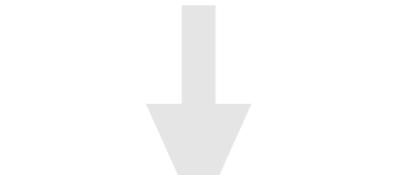
Performance-Energy Trade-off



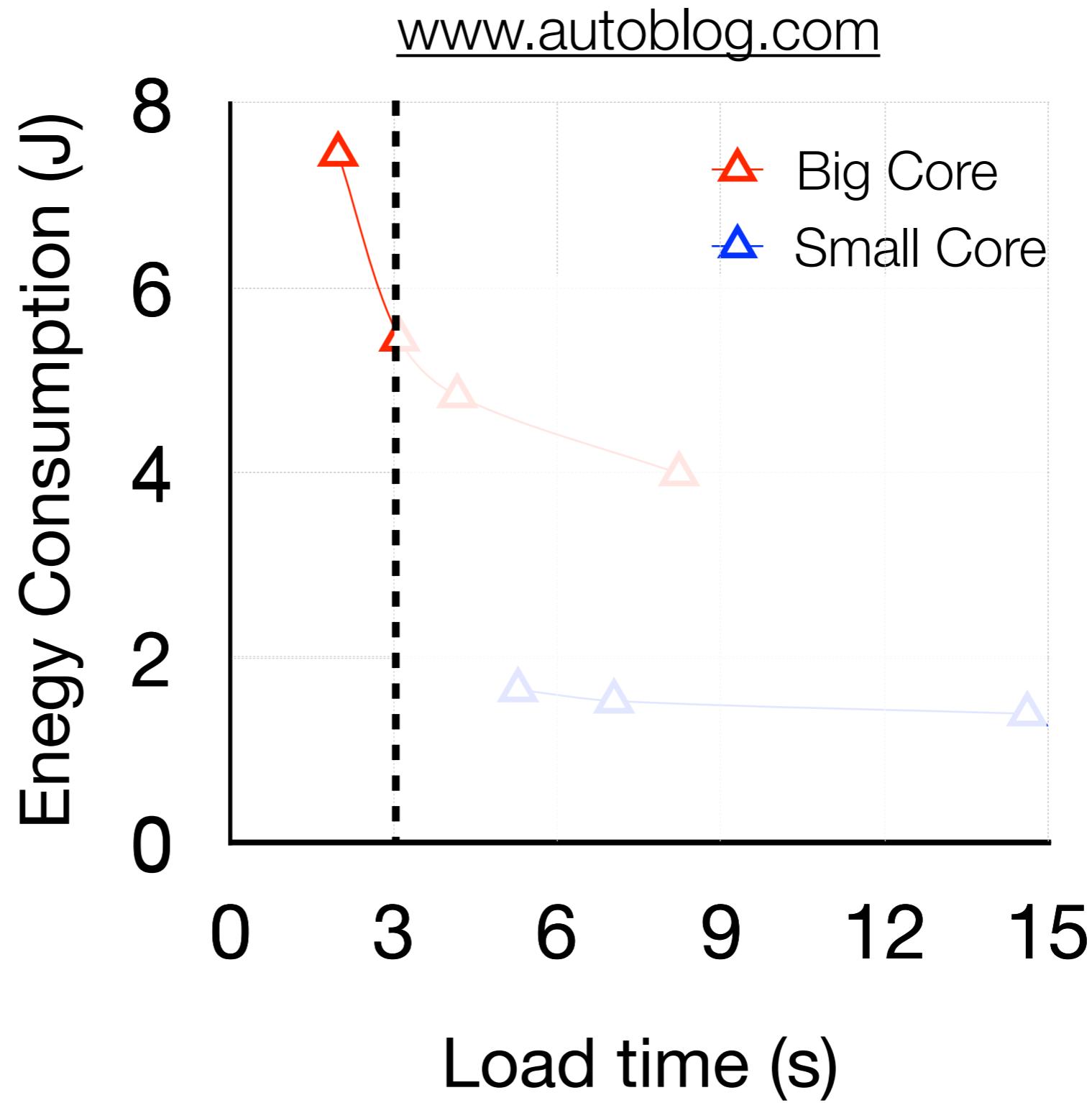
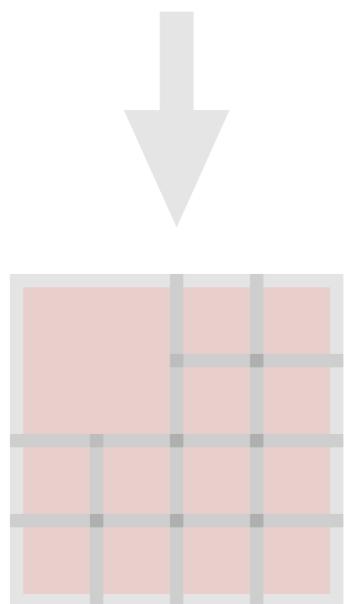
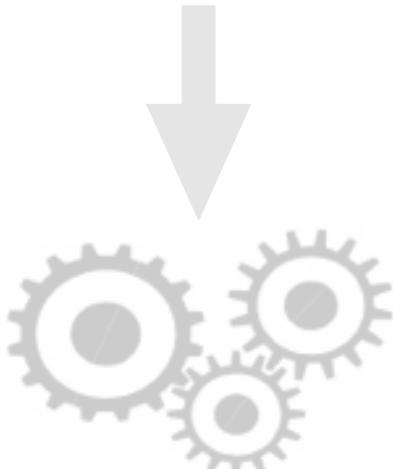
Performance-Energy Trade-off



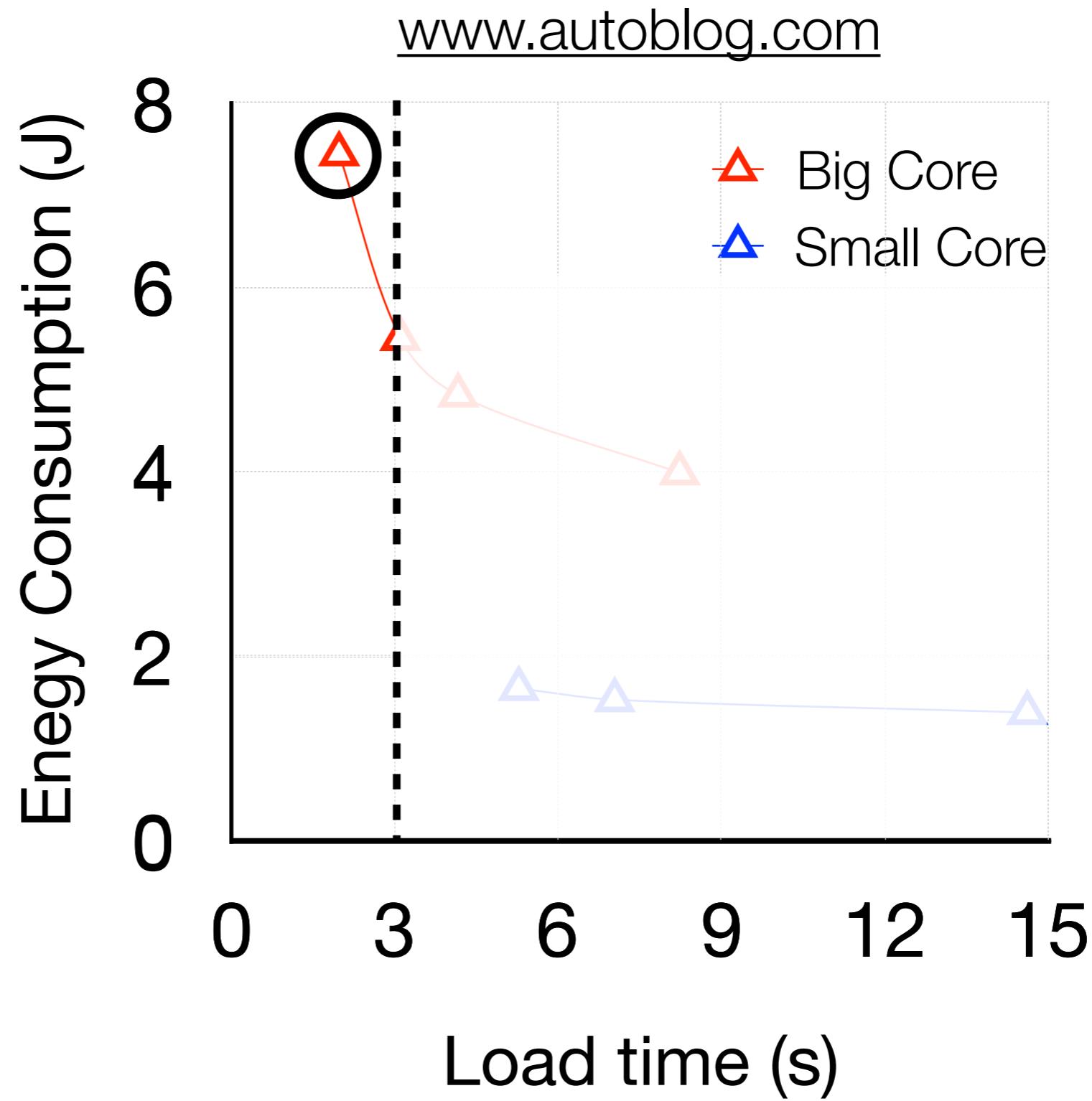
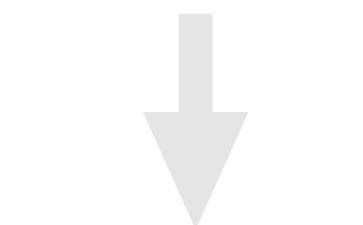
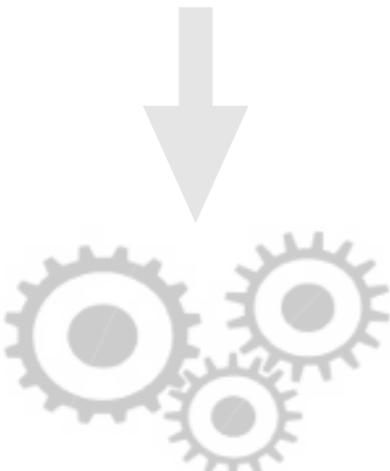
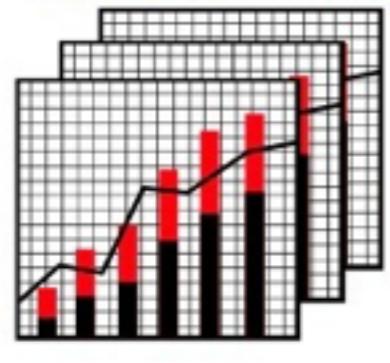
Performance-Energy Trade-off



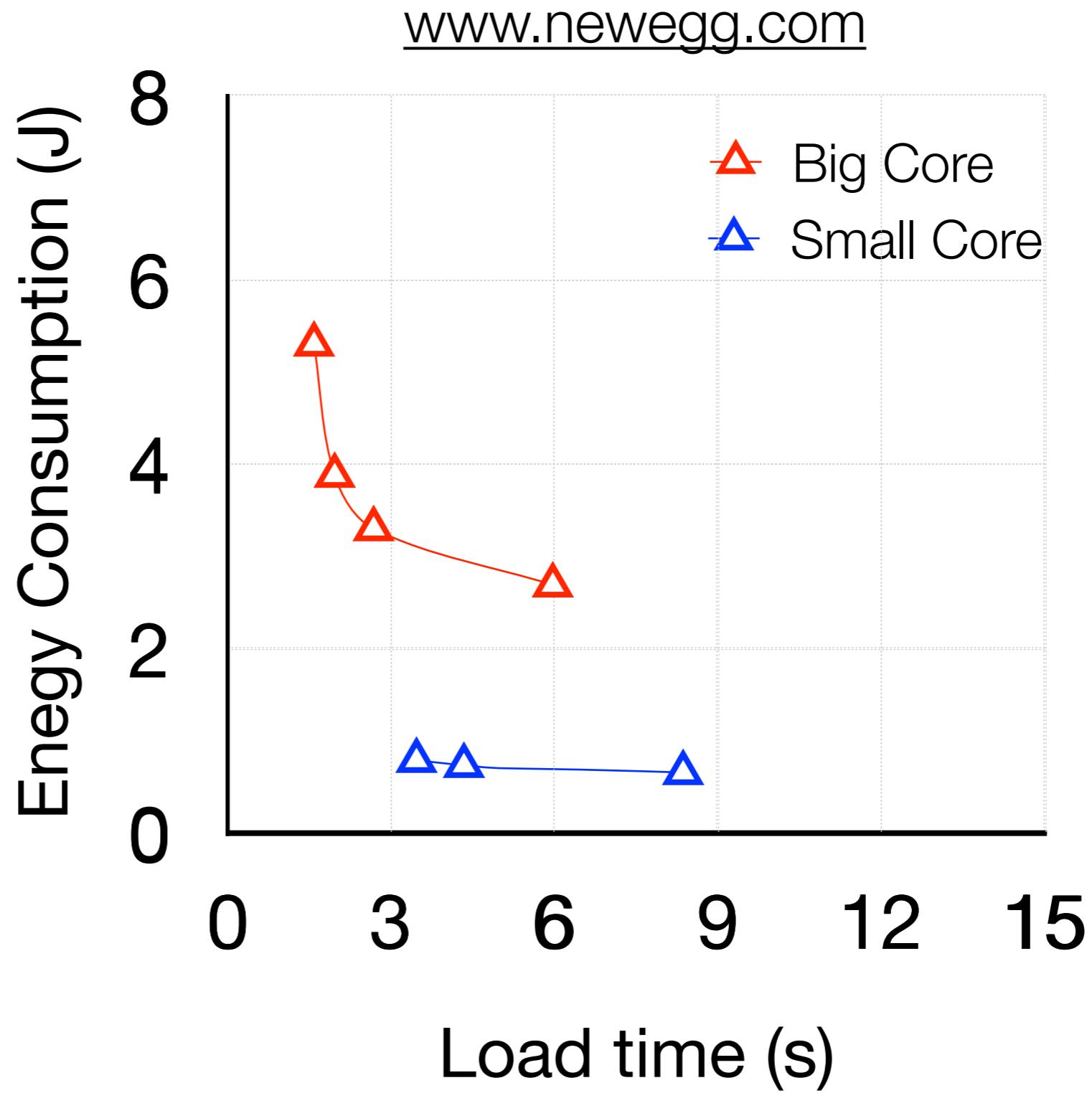
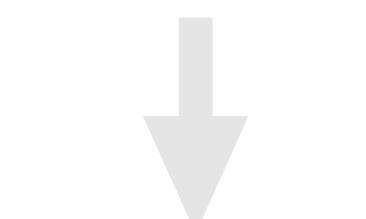
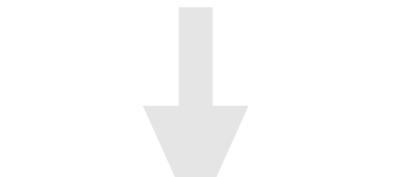
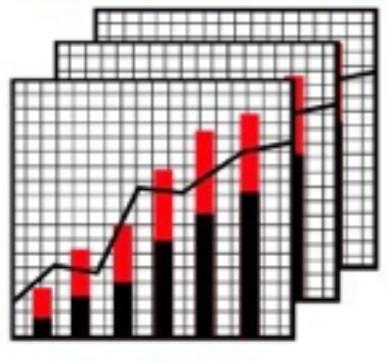
Performance-Energy Trade-off



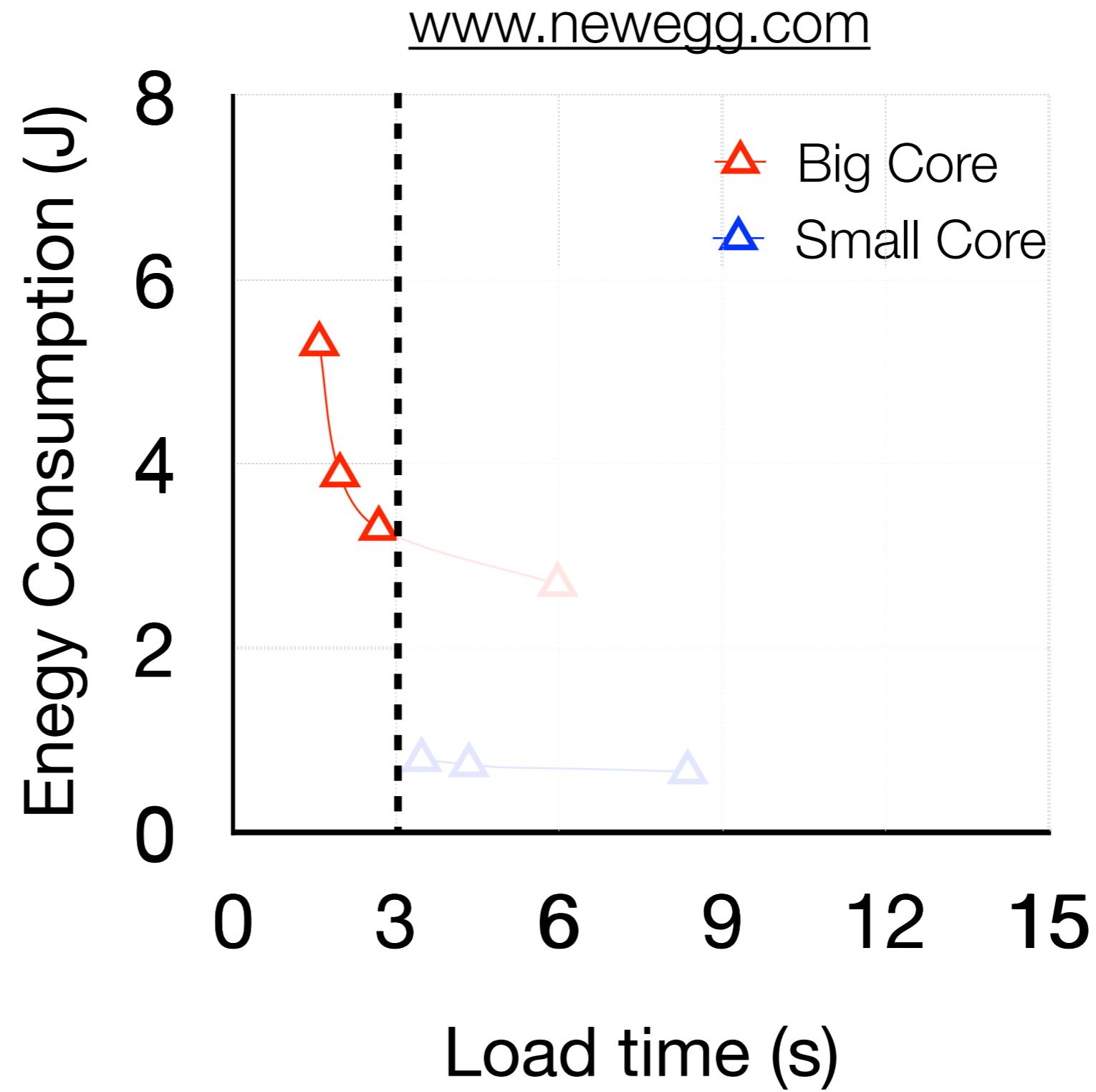
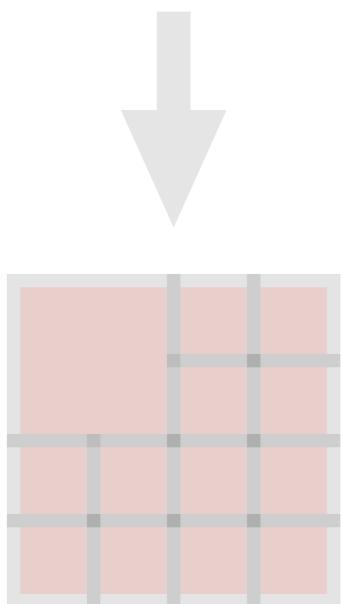
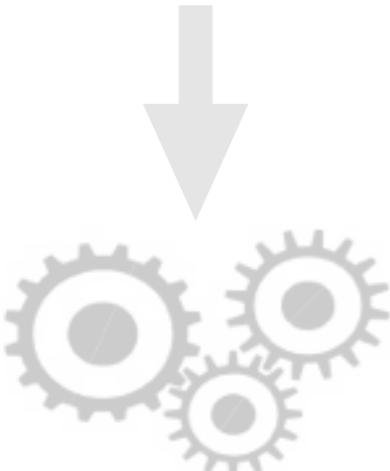
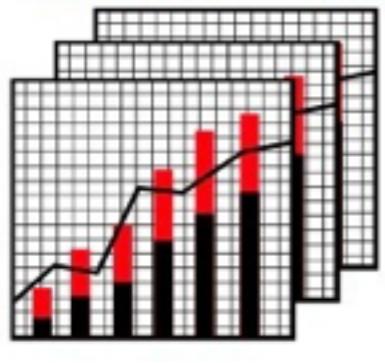
Performance-Energy Trade-off



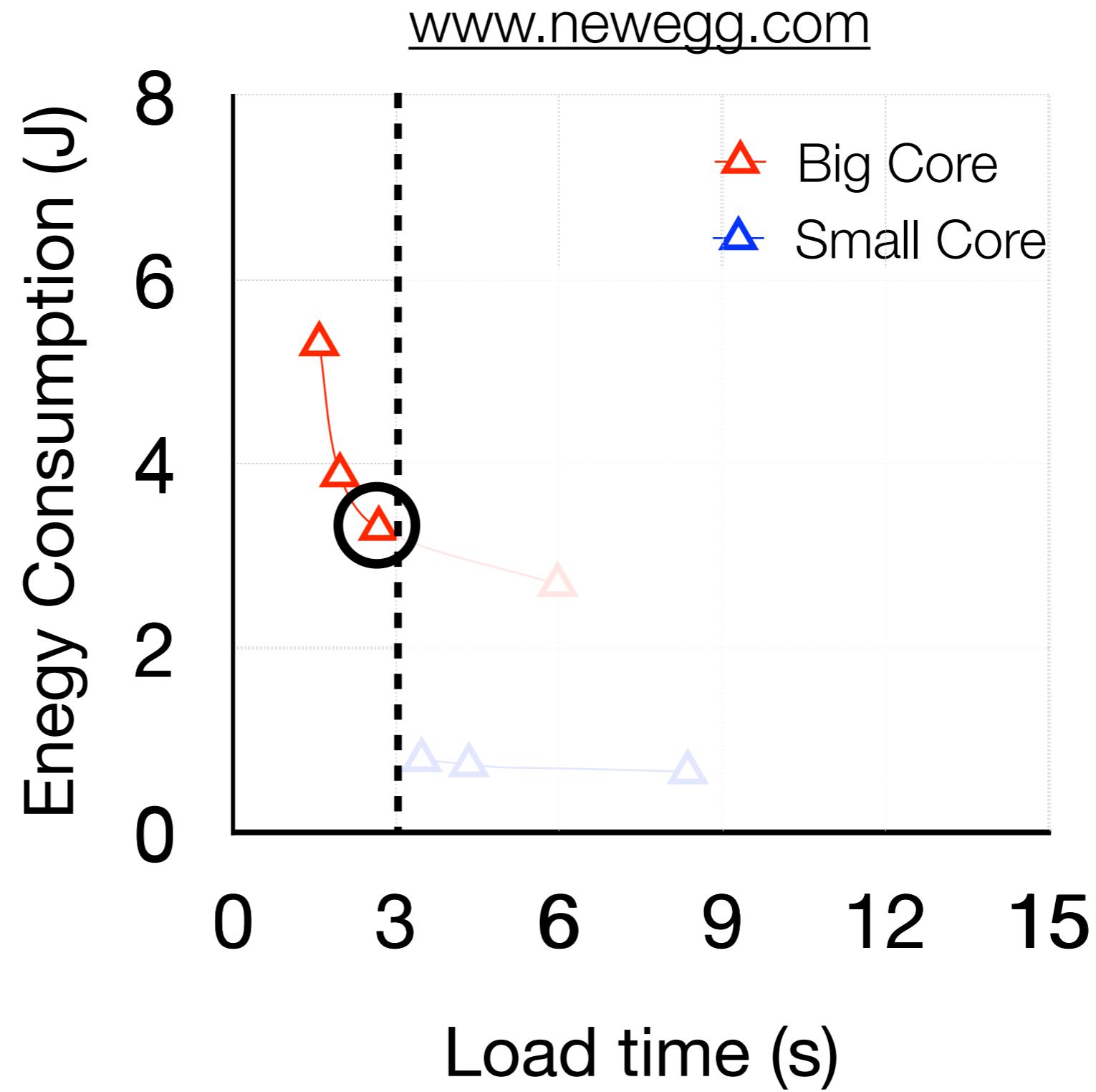
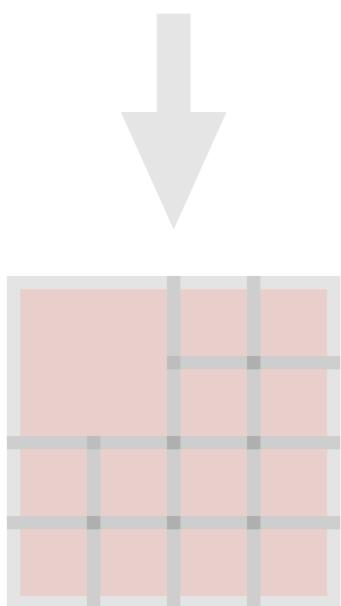
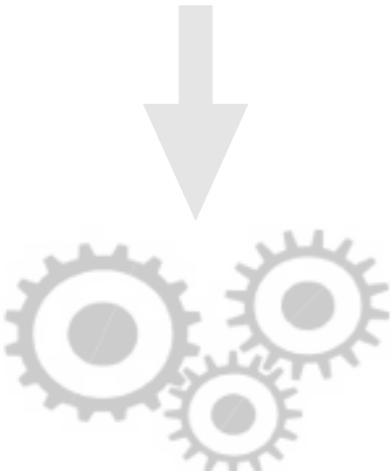
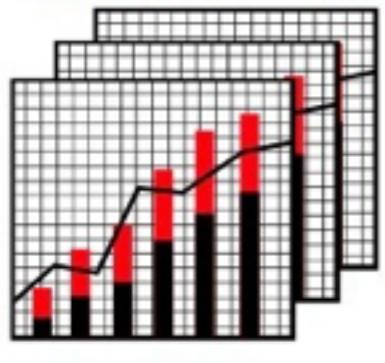
Performance-Energy Trade-off



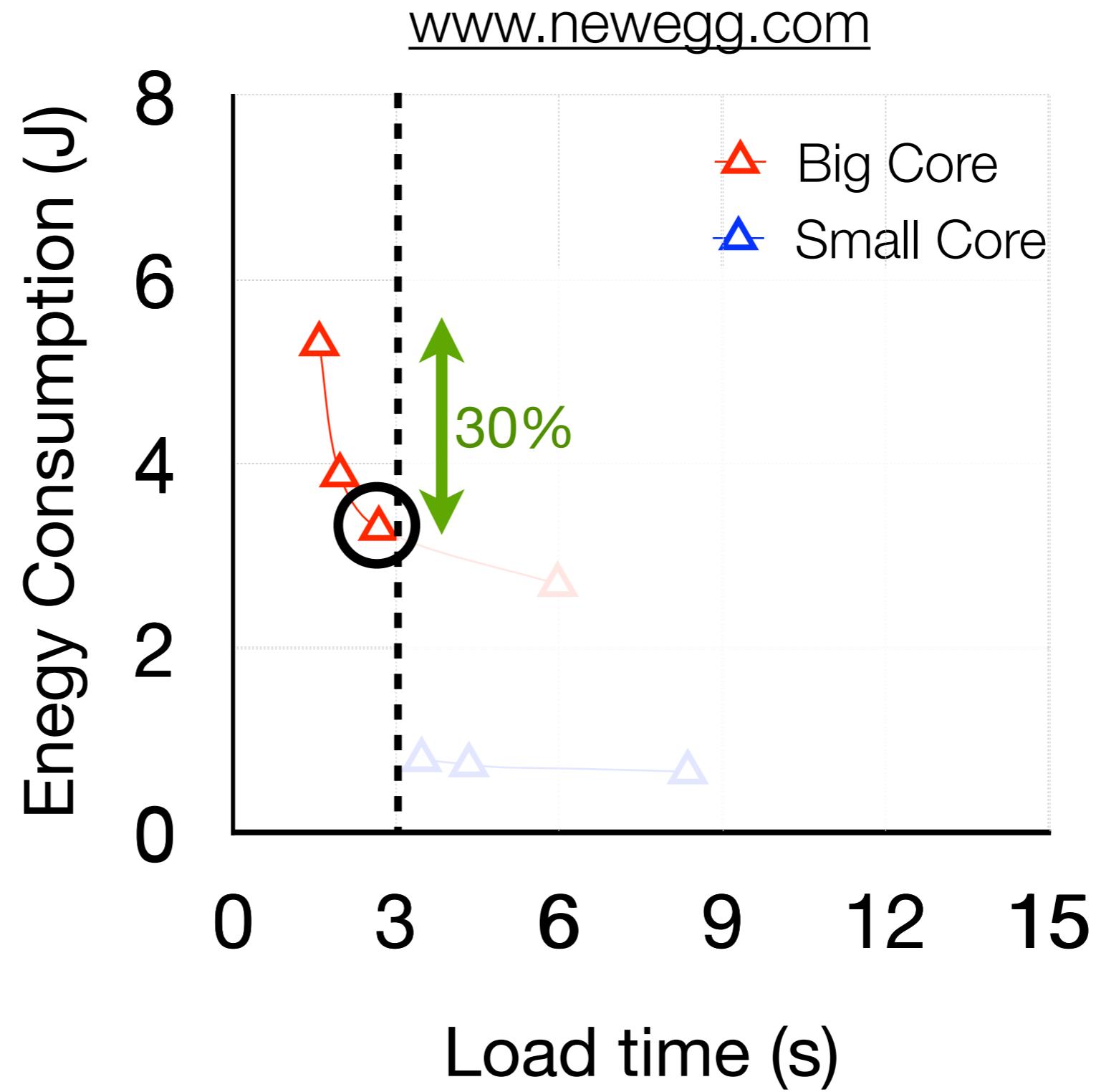
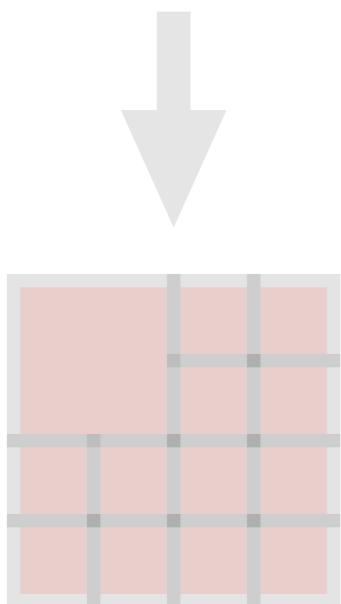
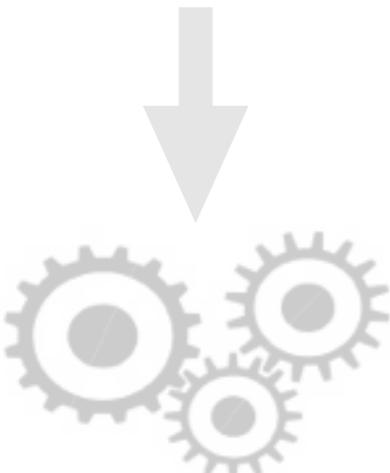
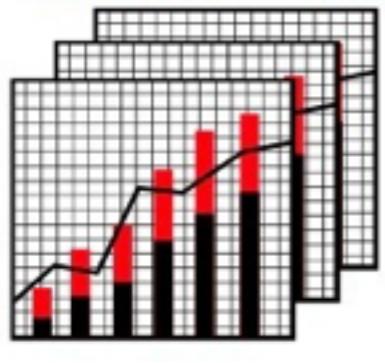
Performance-Energy Trade-off



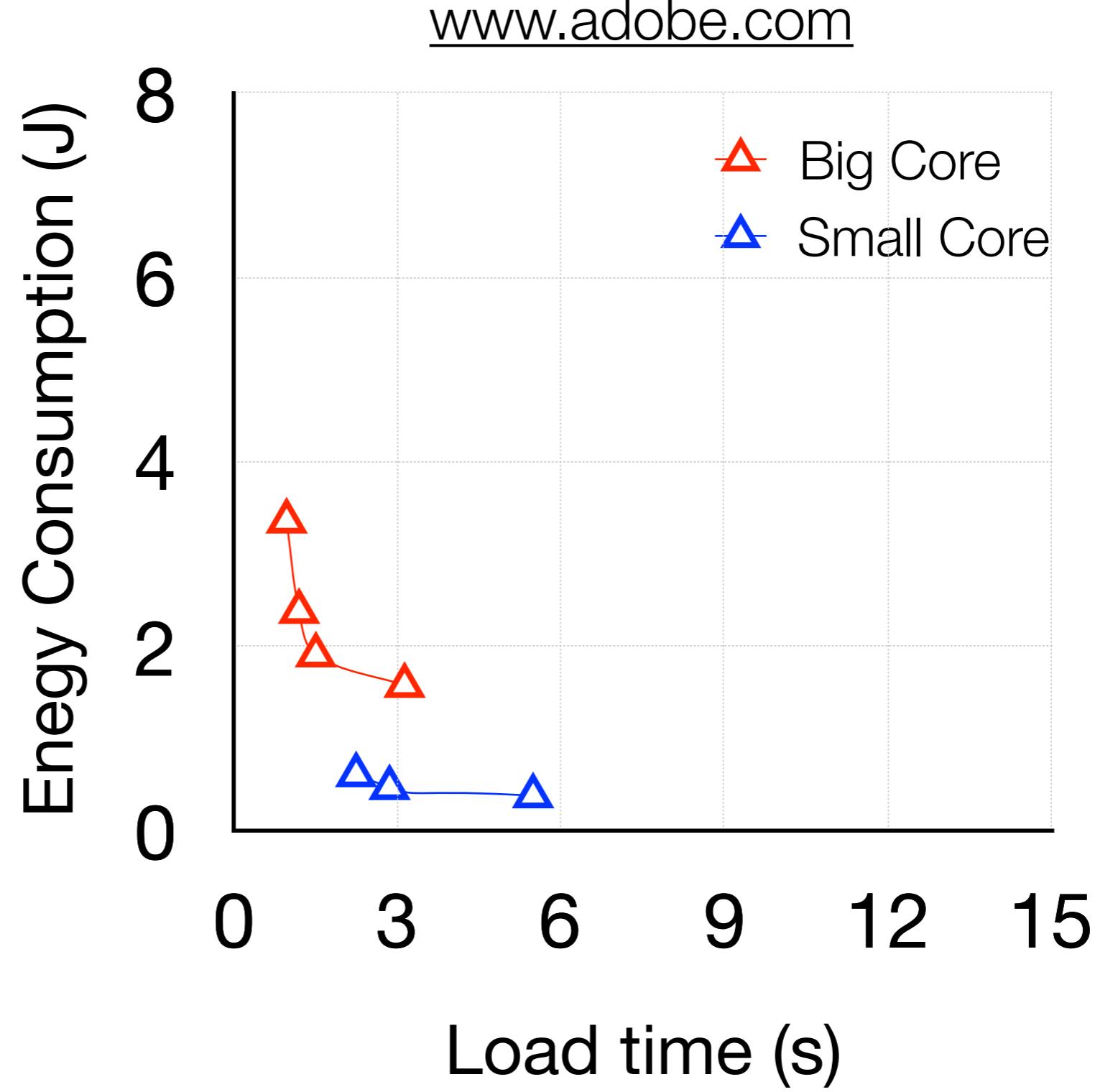
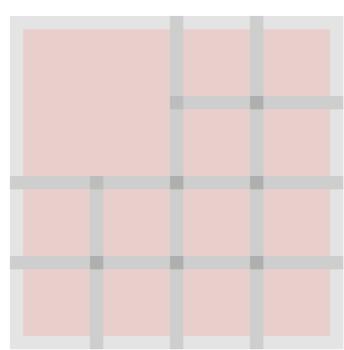
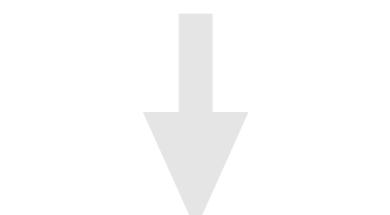
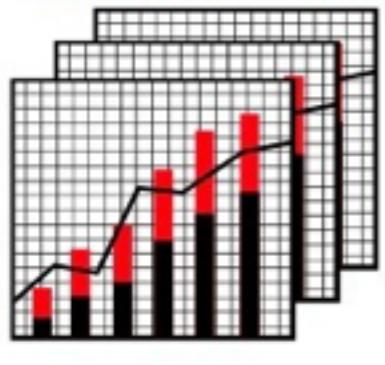
Performance-Energy Trade-off



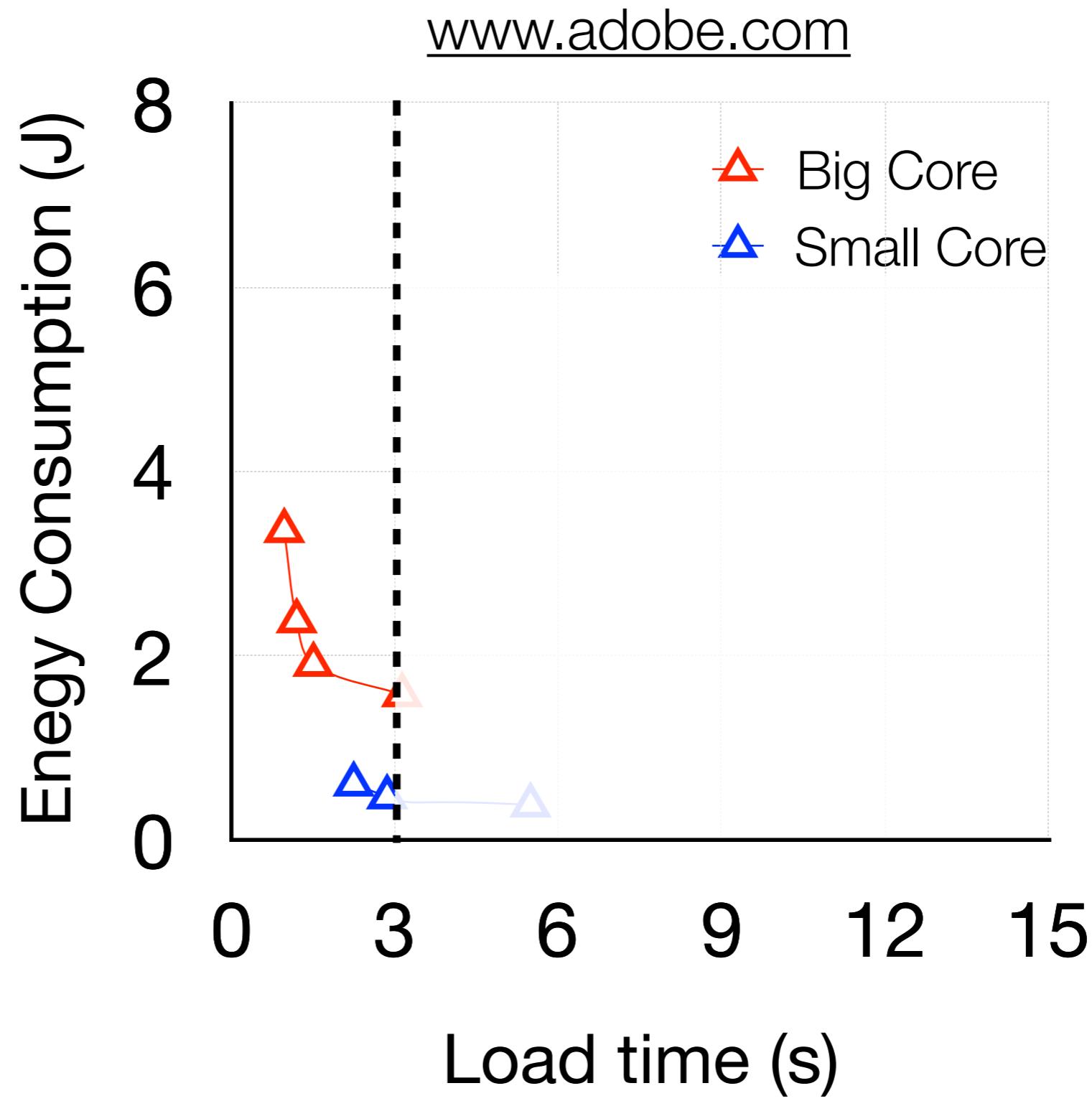
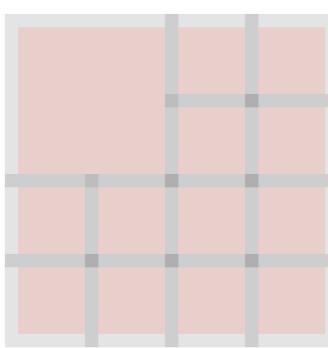
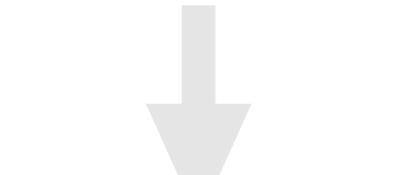
Performance-Energy Trade-off



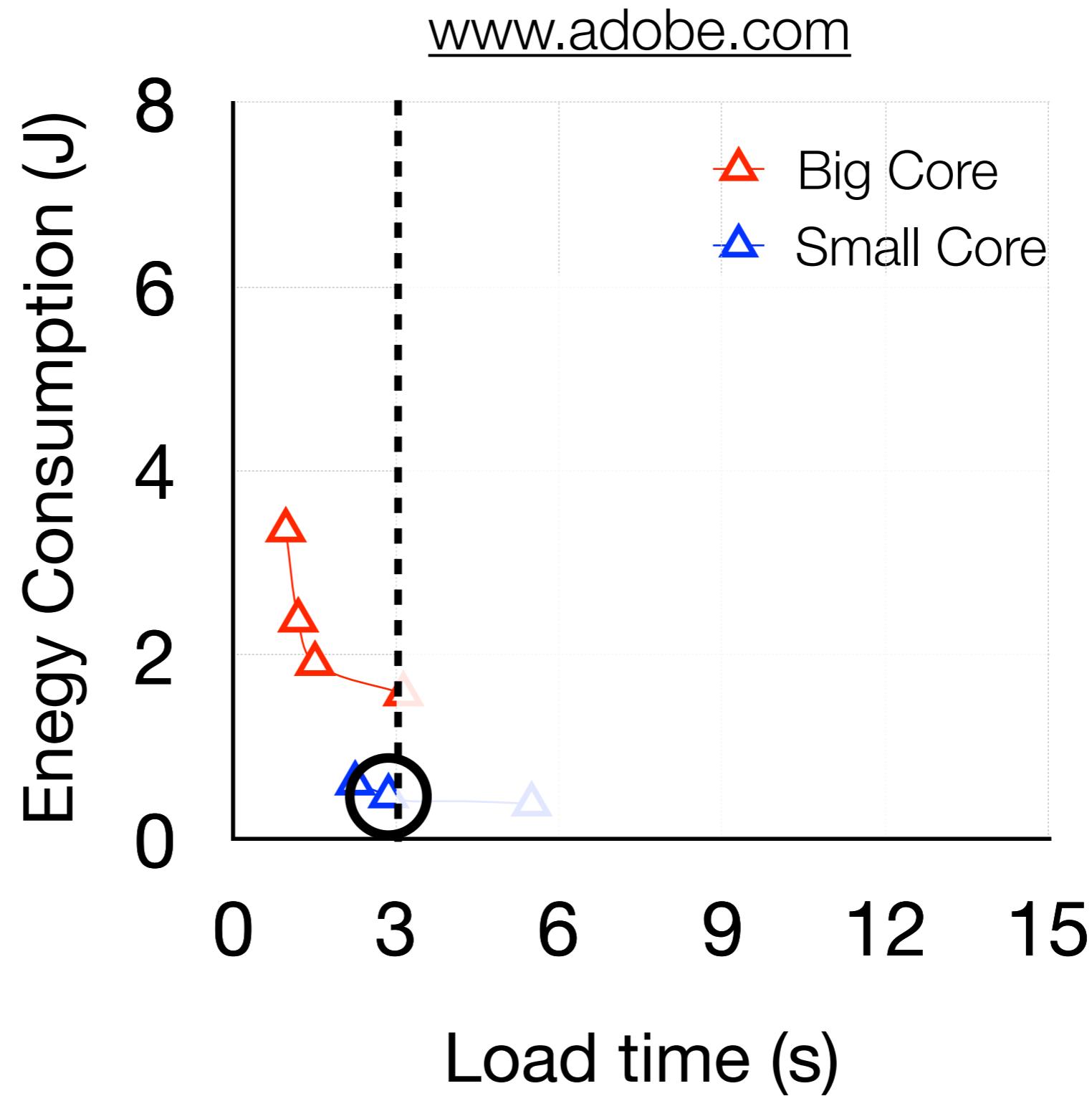
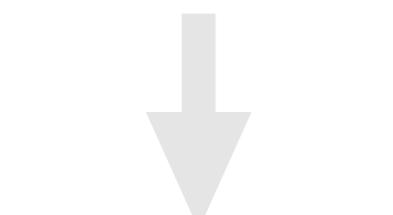
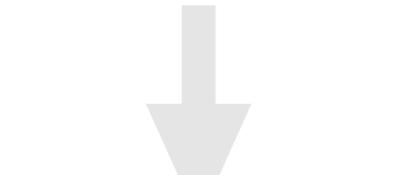
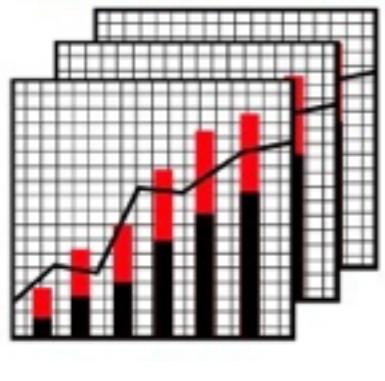
Performance-Energy Trade-off



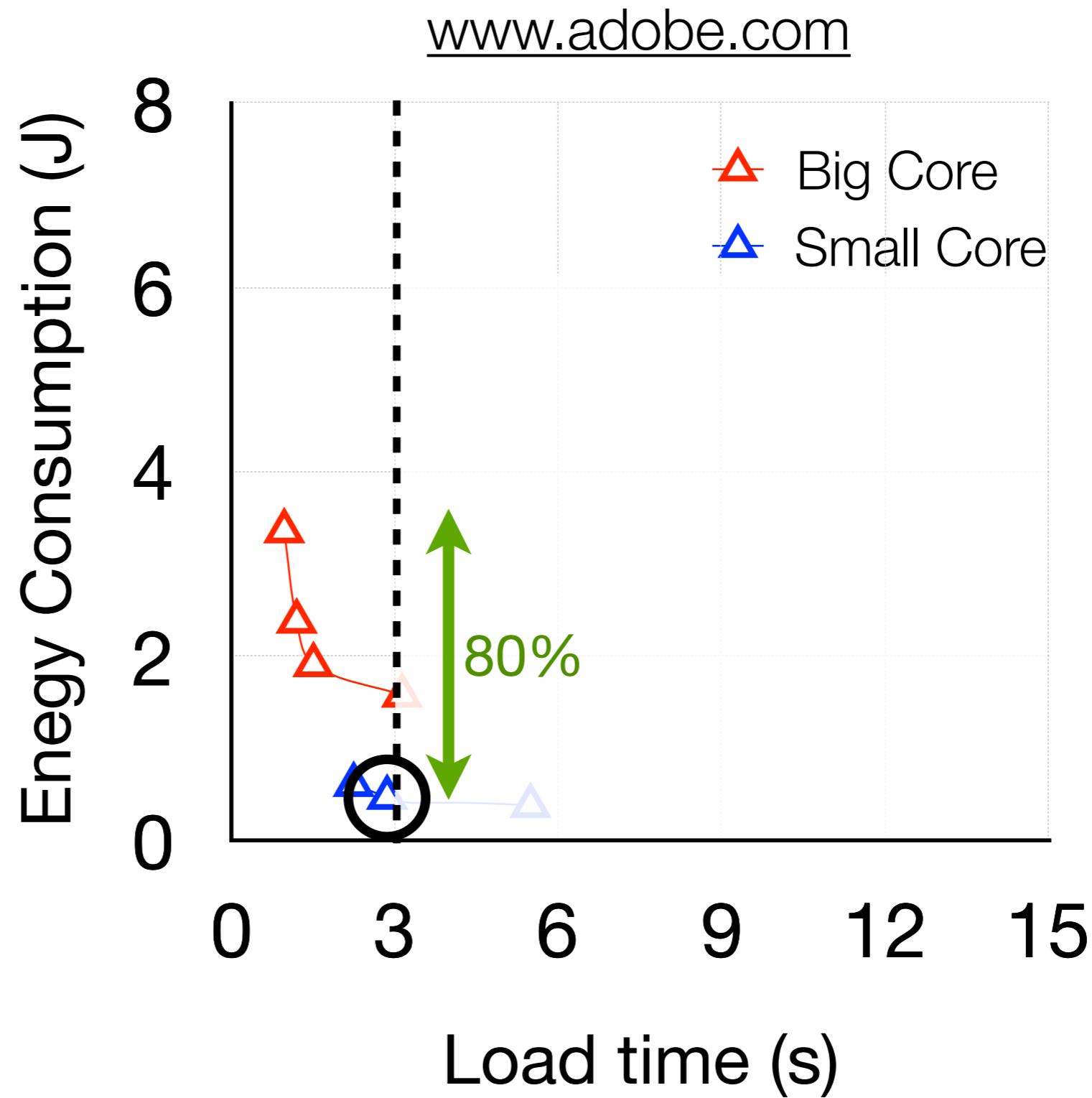
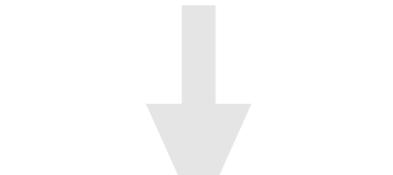
Performance-Energy Trade-off



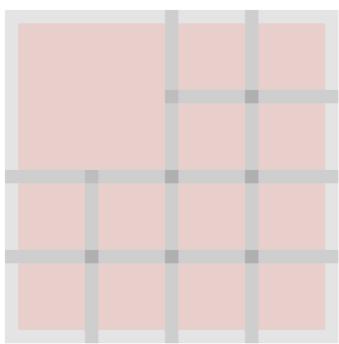
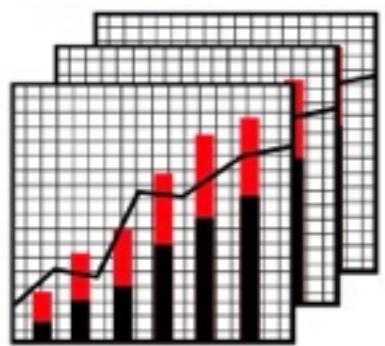
Performance-Energy Trade-off



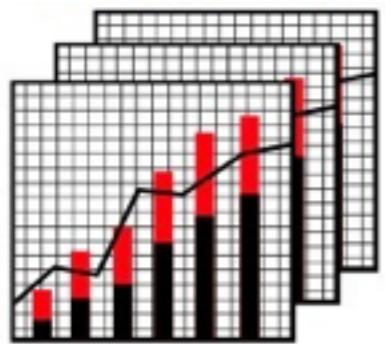
Performance-Energy Trade-off



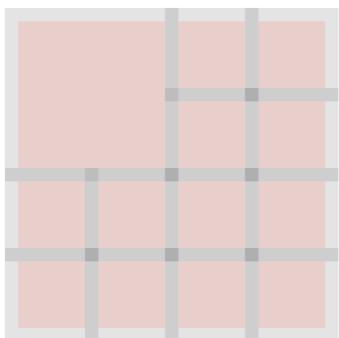
Breaking Down the Computations



Breaking Down the Computations



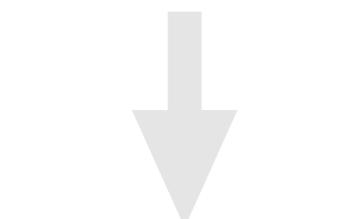
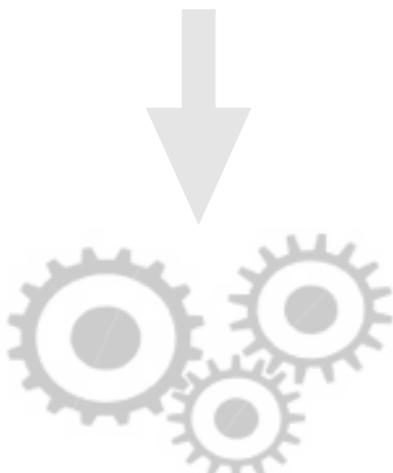
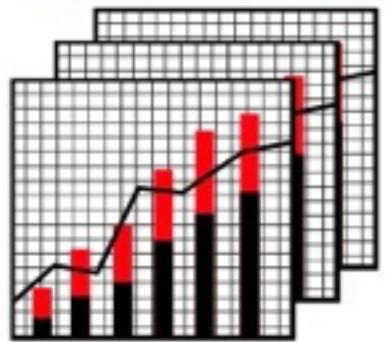
HTML (Structure)



CSS (Style)



Breaking Down the Computations

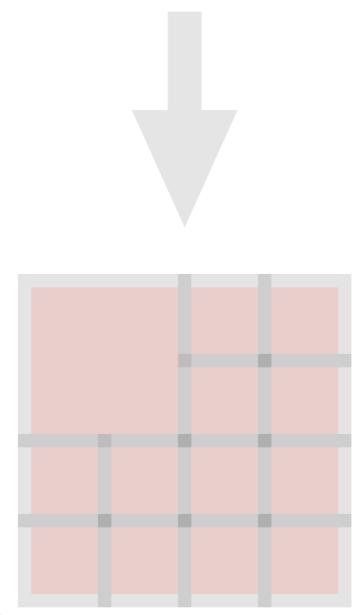
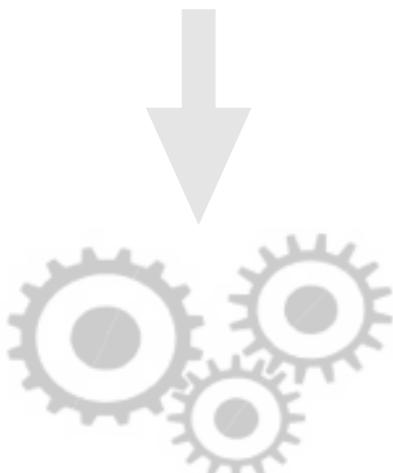
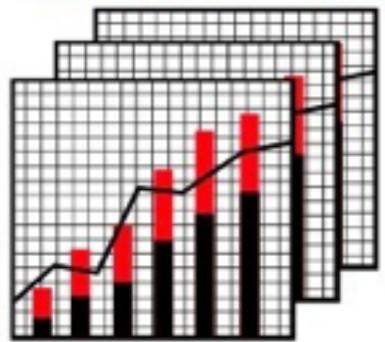


HTML (Structure)

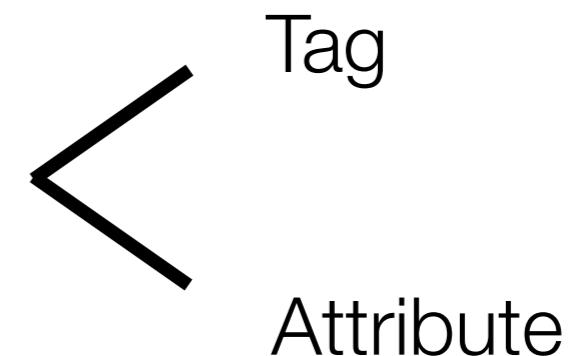
Tag
Attribute

CSS (Style)

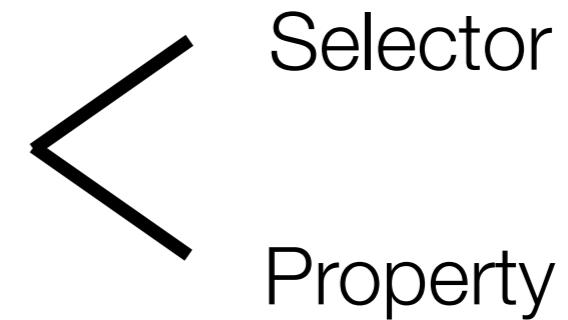
Breaking Down the Computations



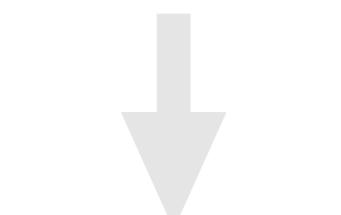
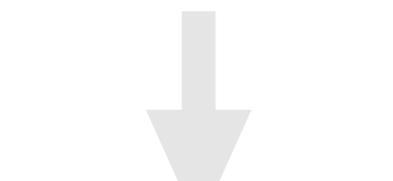
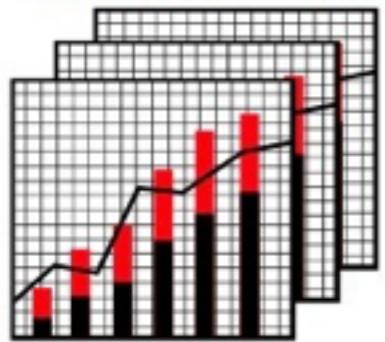
HTML (Structure)



CSS (Style)



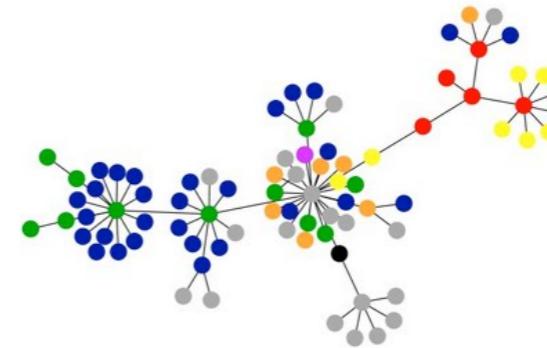
Breaking Down the Computations



HTML (Structure)

DOM Tree

CSS (Style)



Tag

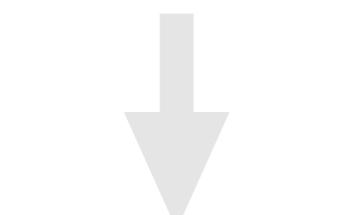
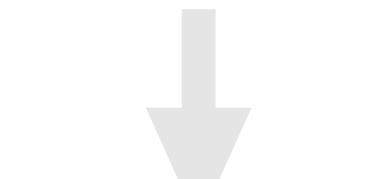
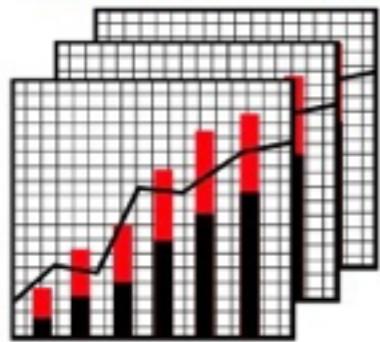
Attribute

Selector

Property



Breaking Down the Computations

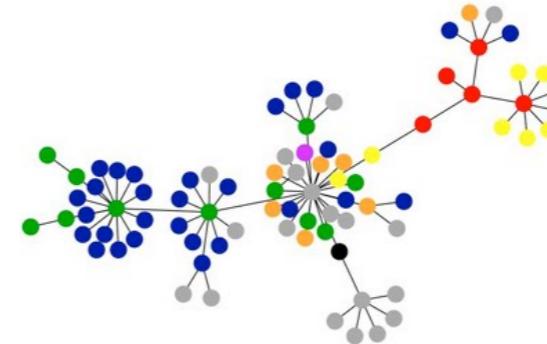


Web Primitives

HTML (Structure)

DOM Tree

CSS (Style)



Tag

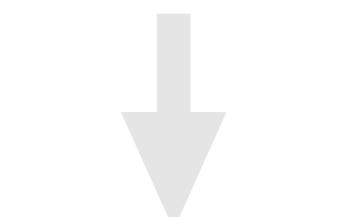
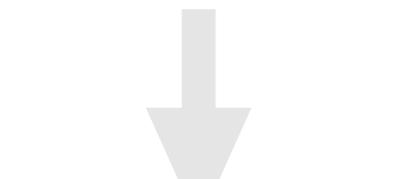
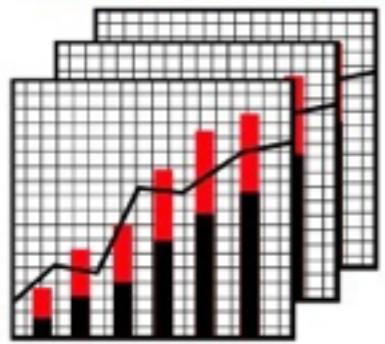
Attribute

Selector

Property

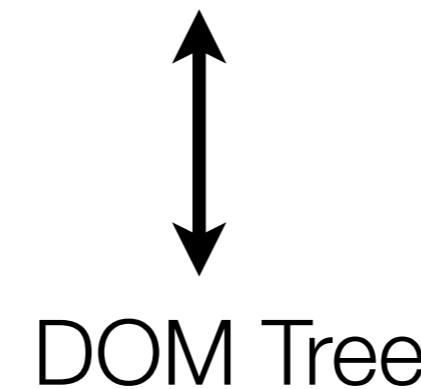


Breaking Down the Computations

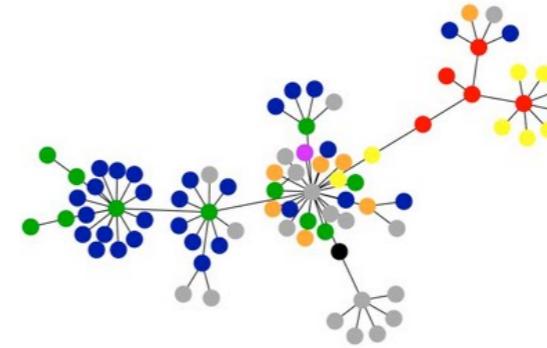


Web Primitives

HTML (Structure)



CSS (Style)



Tag

Attribute

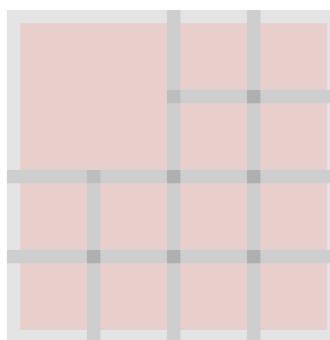
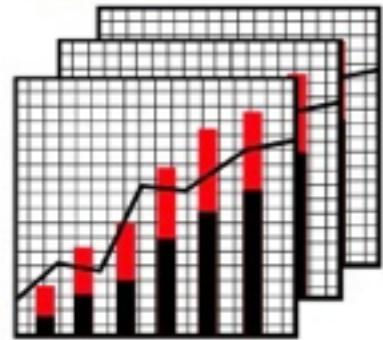
Selector

Property

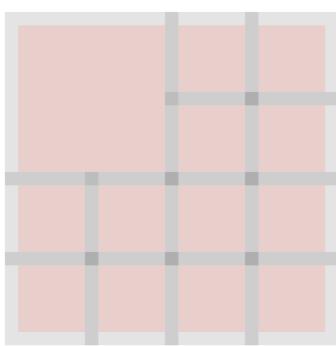
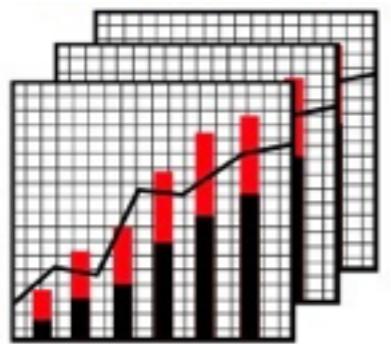


HTML Tag Analysis

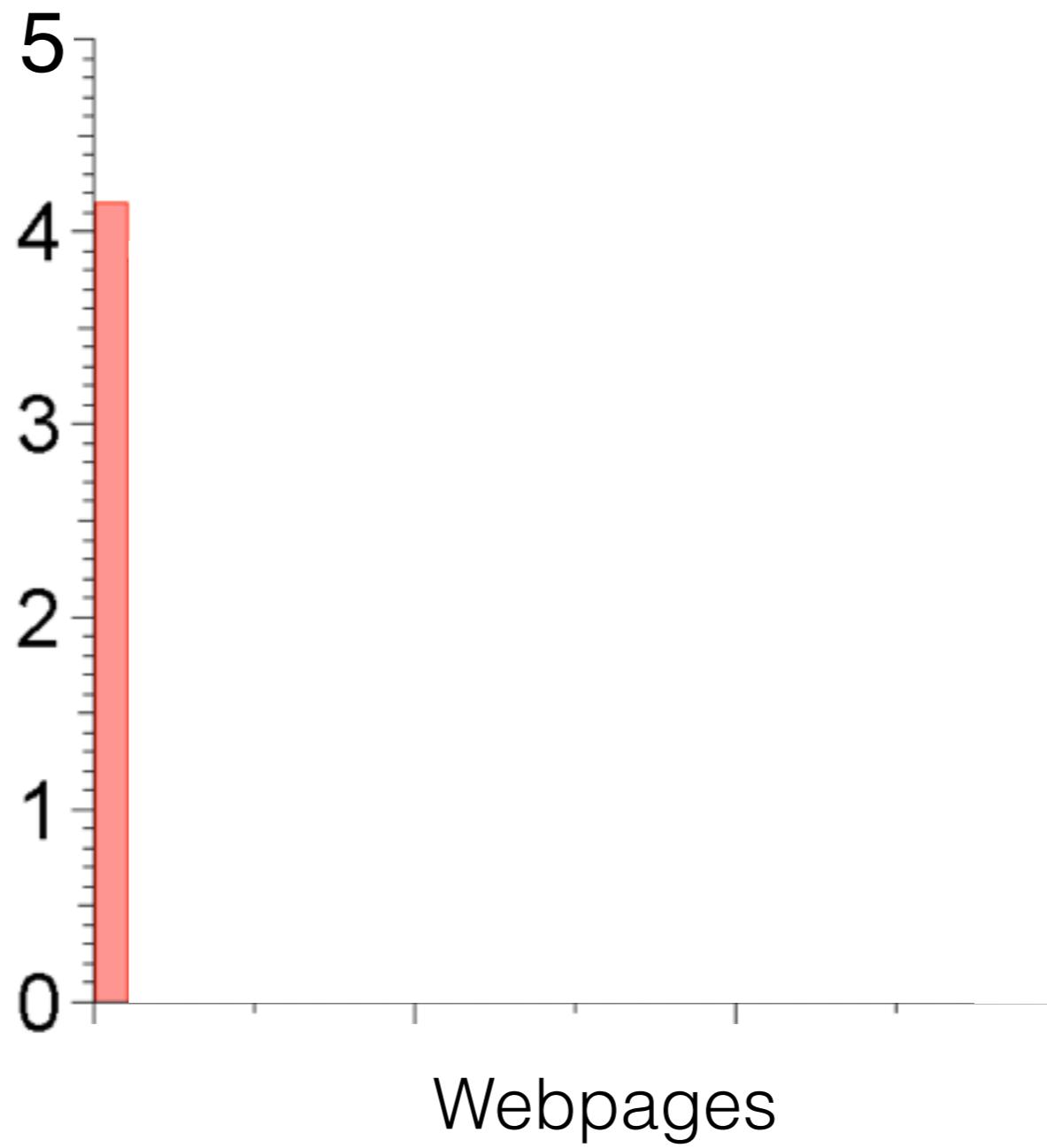
www.163.com



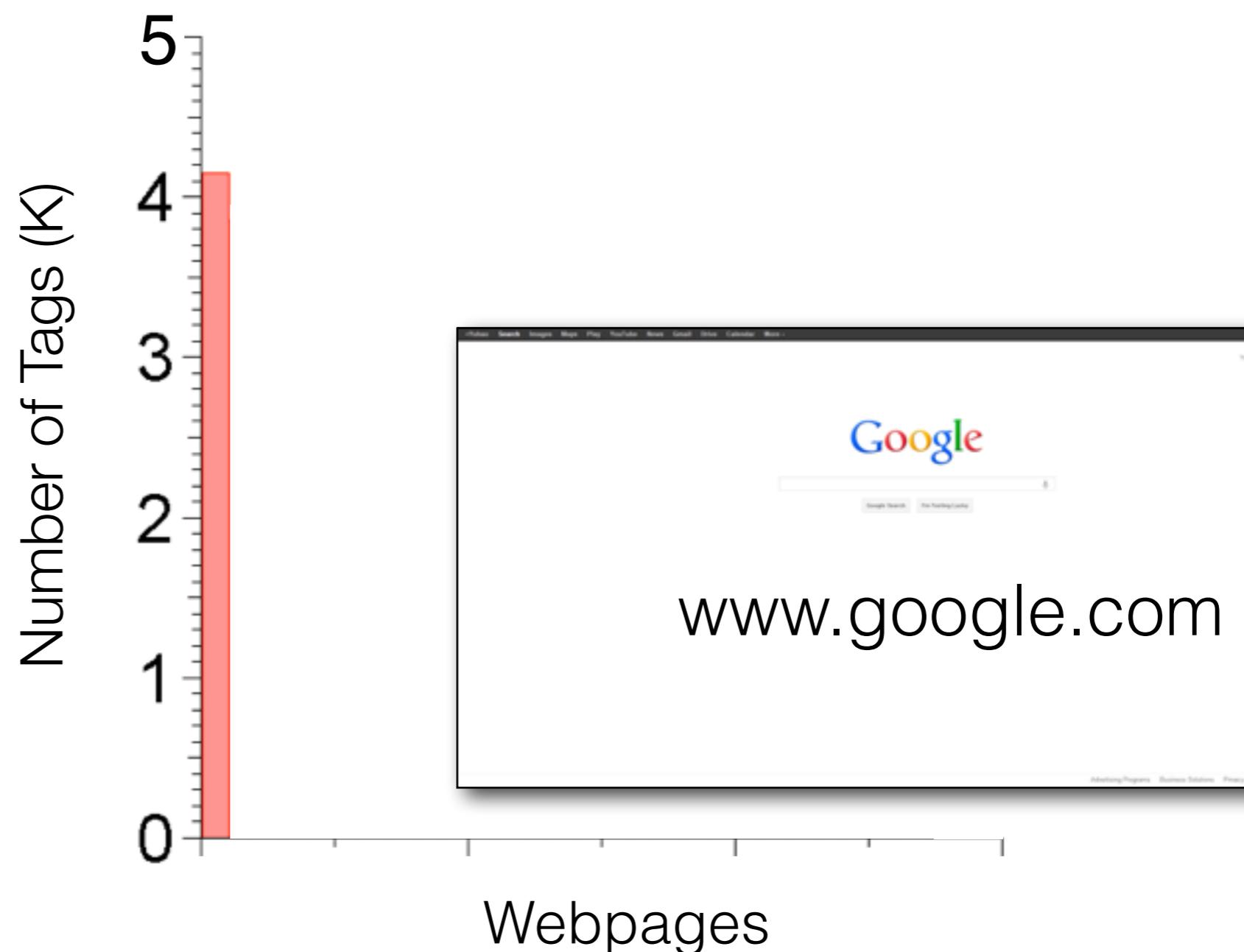
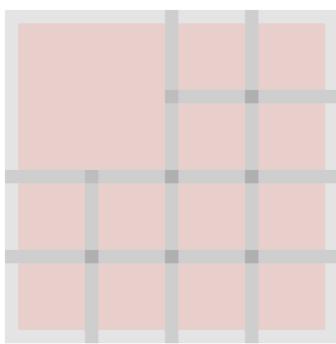
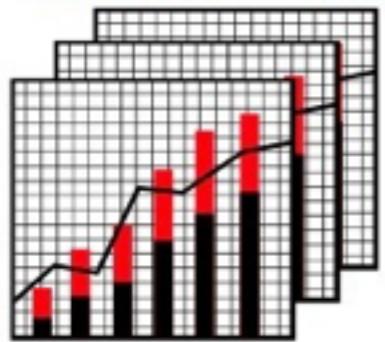
HTML Tag Analysis



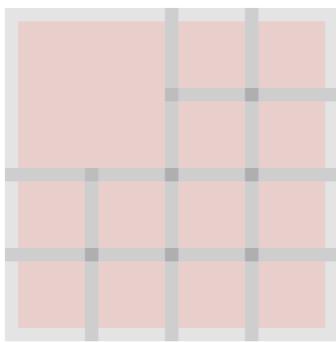
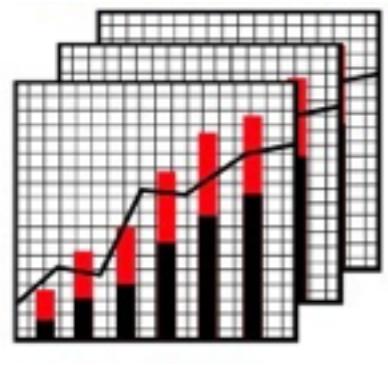
Number of Tags (K)



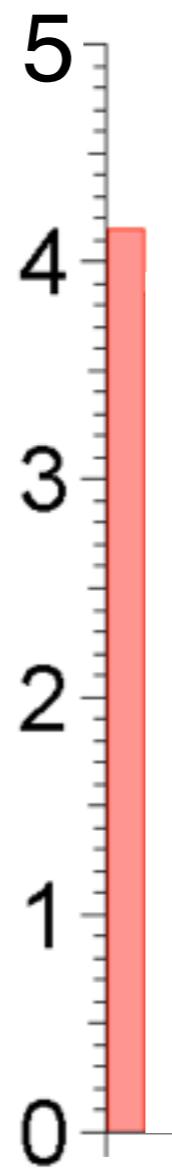
HTML Tag Analysis



HTML Tag Analysis

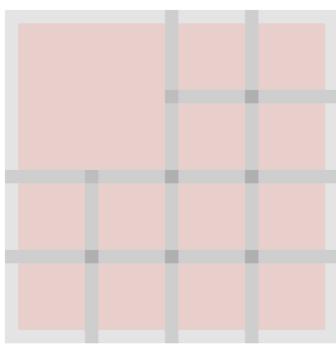
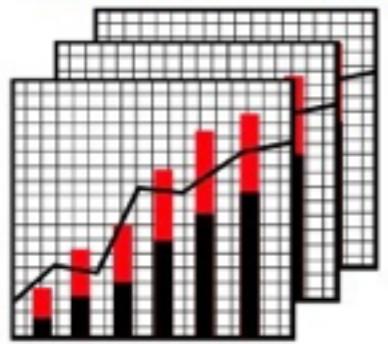


Number of Tags (K)

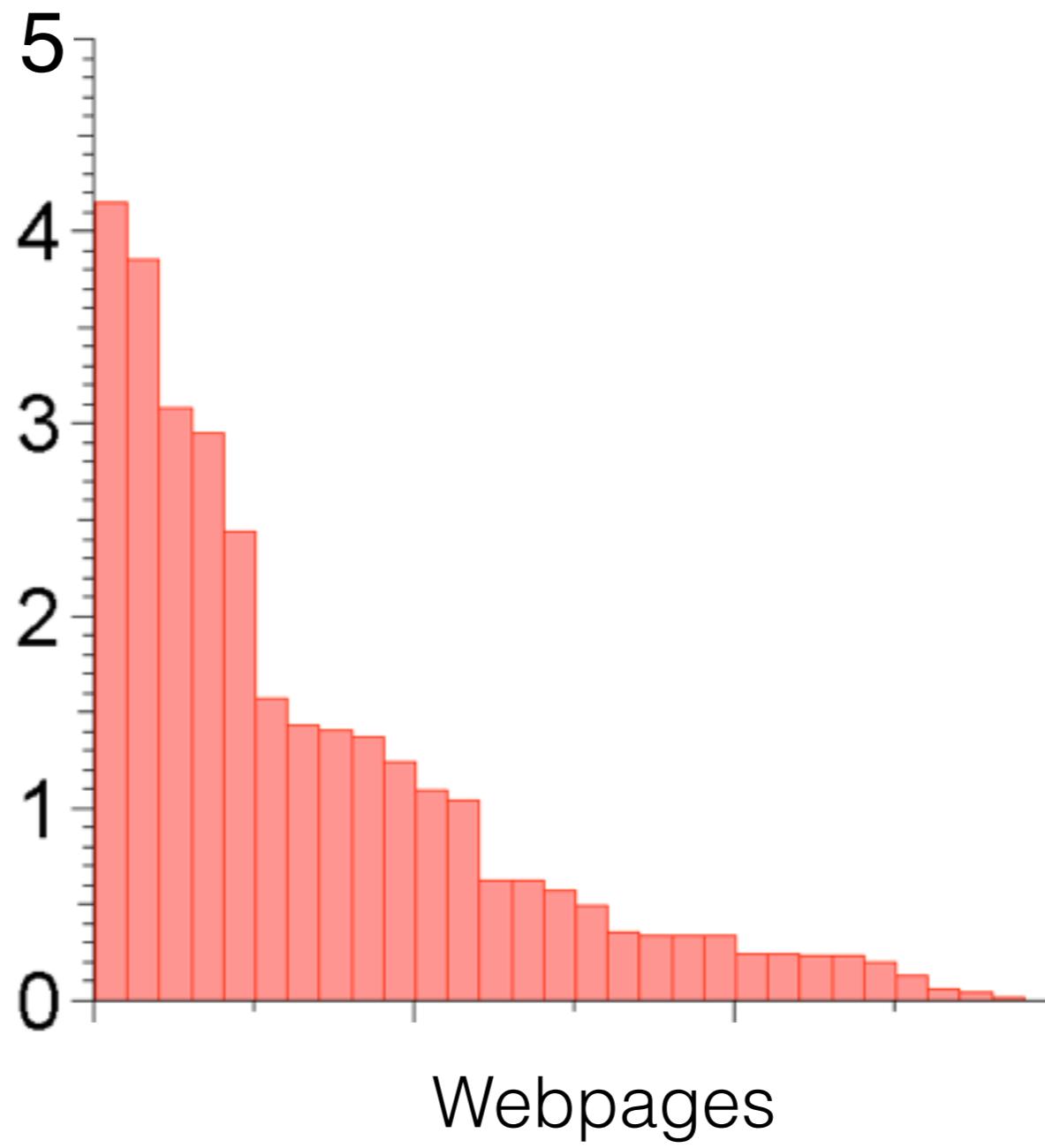


Webpages

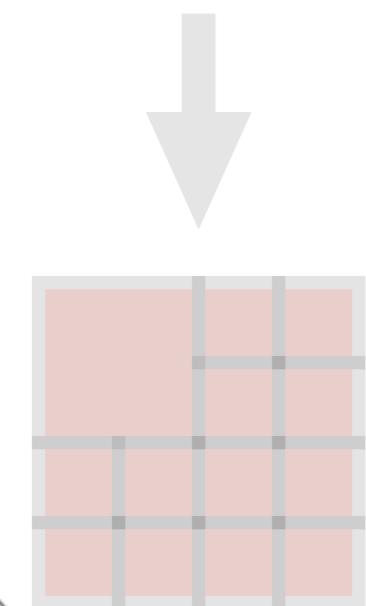
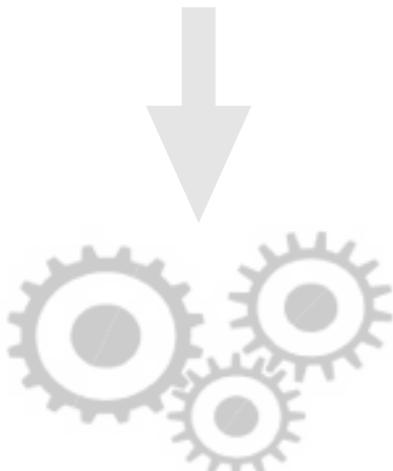
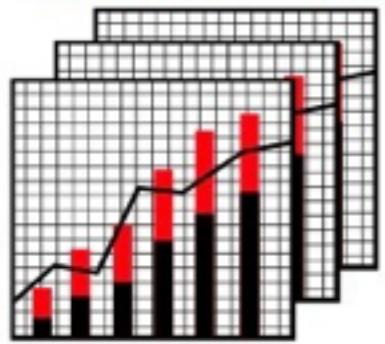
HTML Tag Analysis



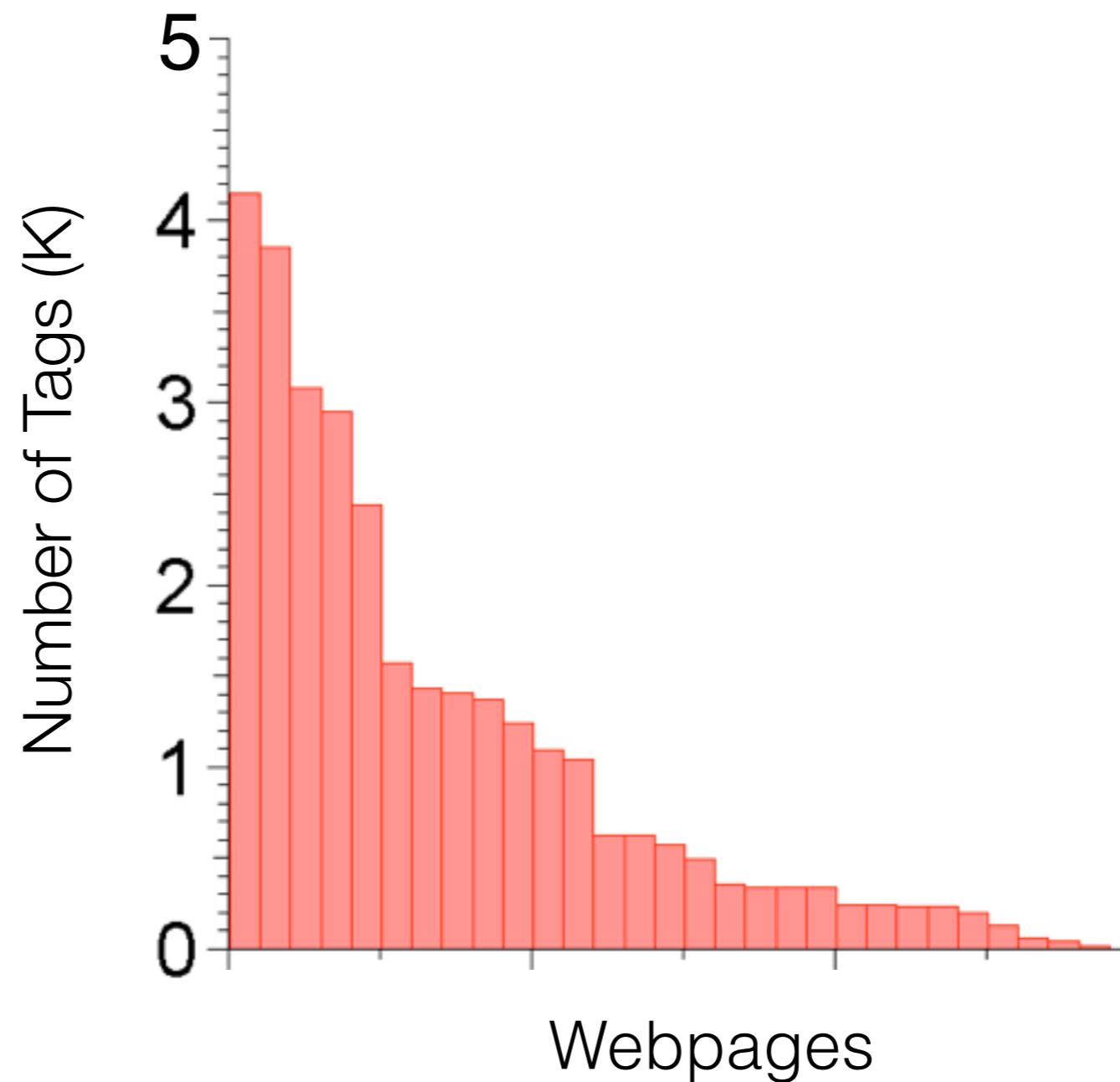
Number of Tags (K)



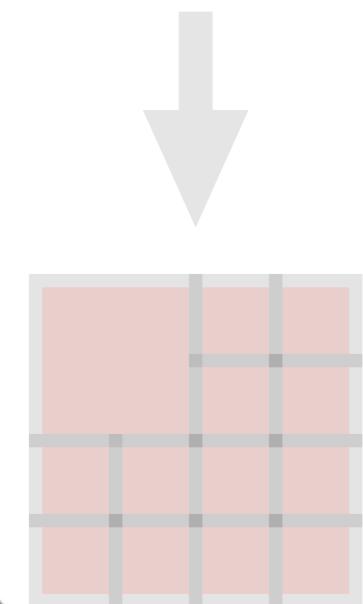
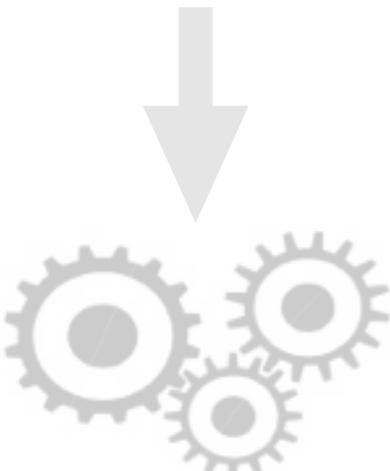
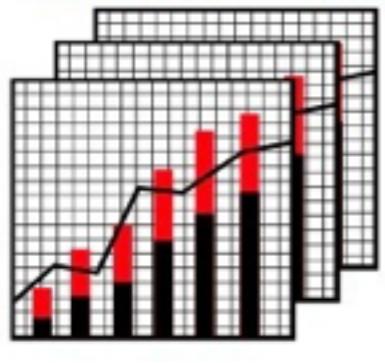
HTML Tag Analysis



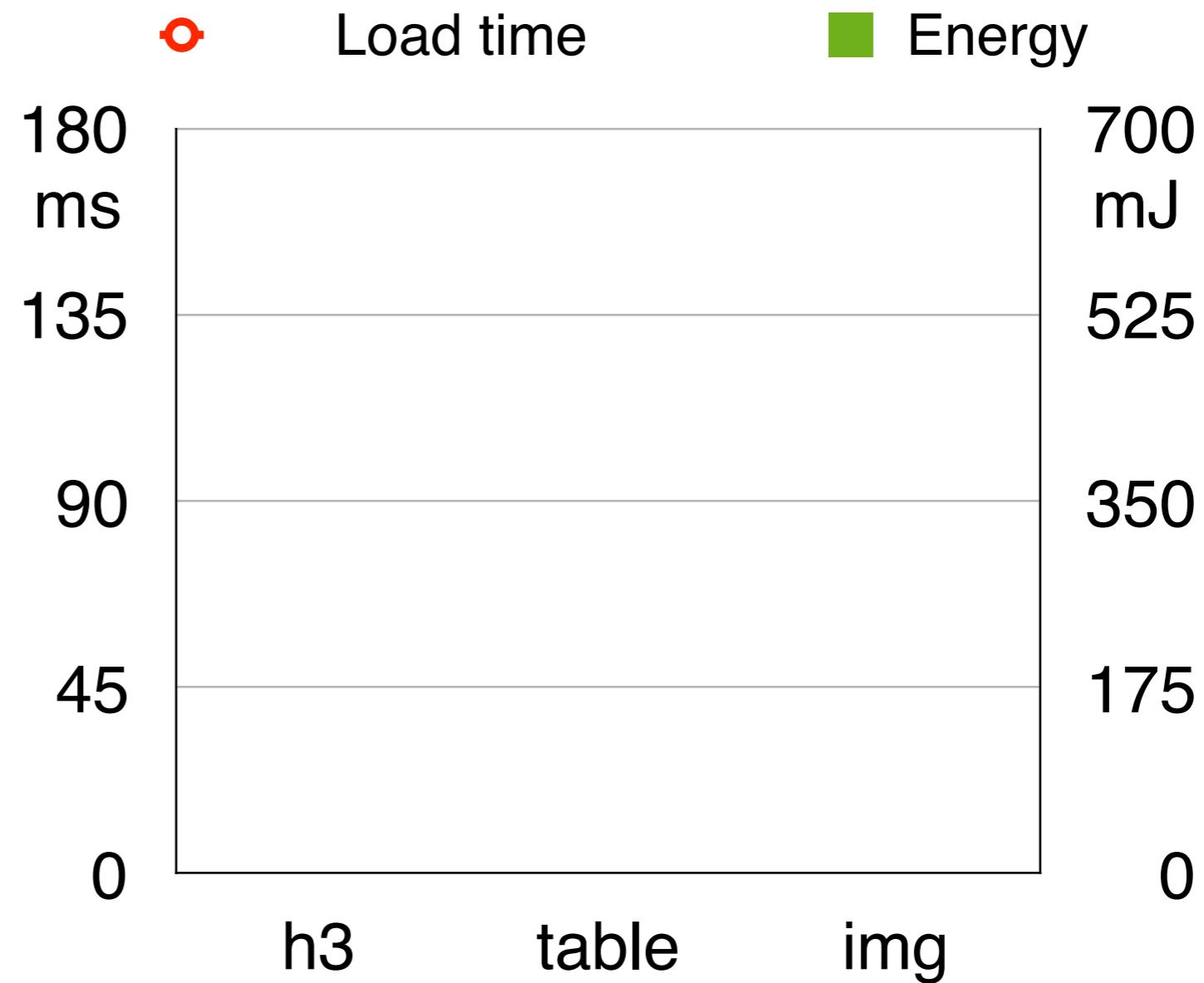
- ▶ Web applications have different tag counts



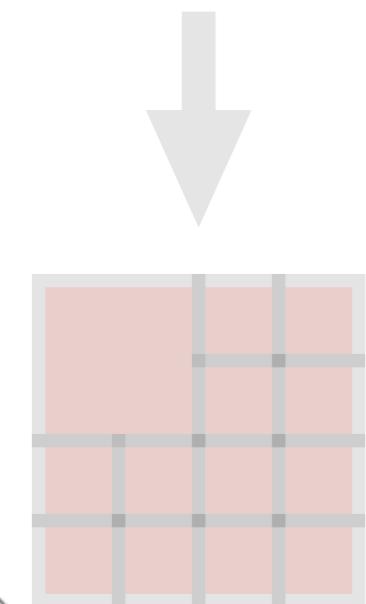
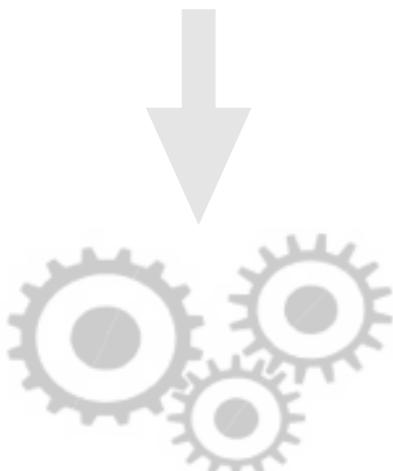
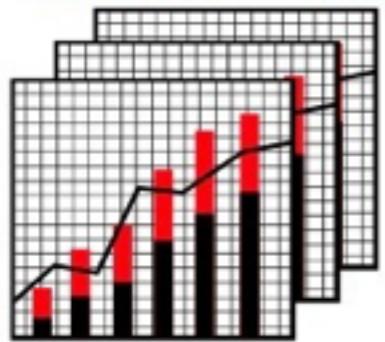
Tag Processing Overhead



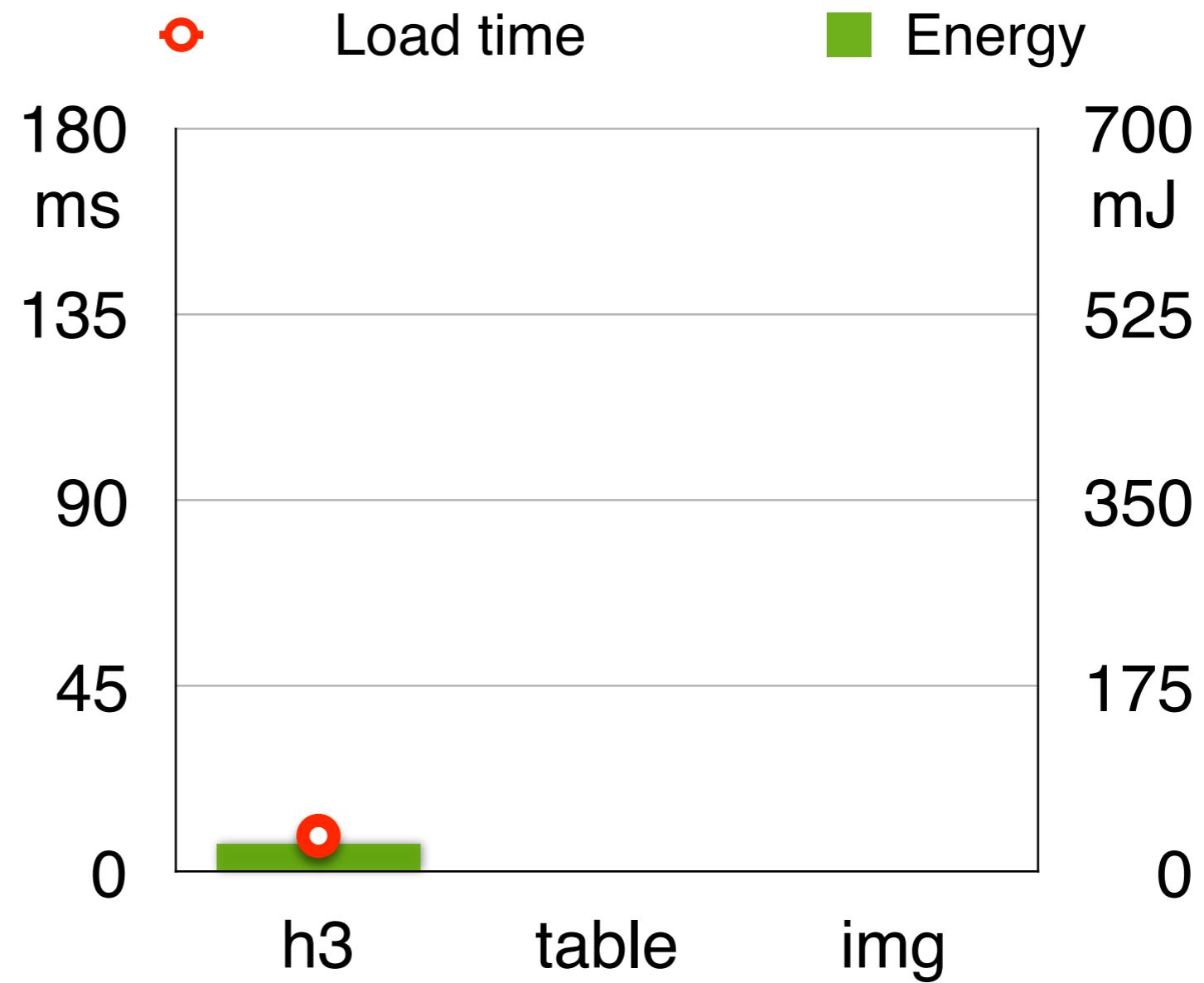
- ▶ Web applications have different tag counts



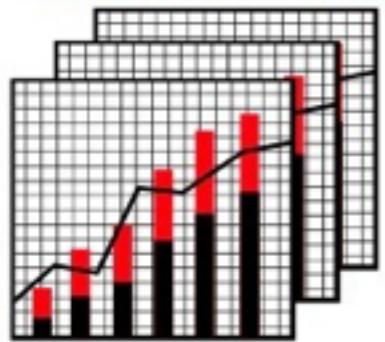
Tag Processing Overhead



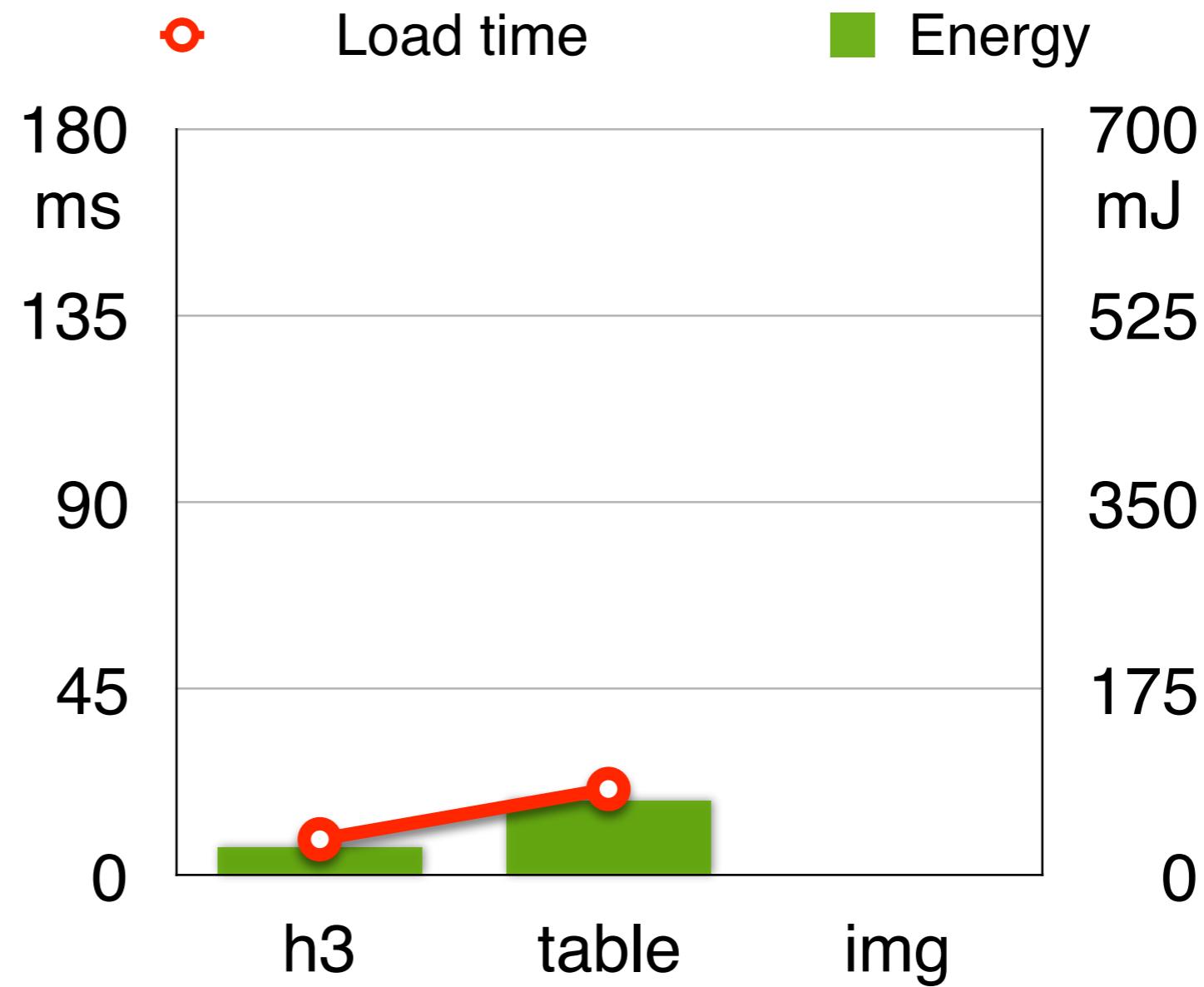
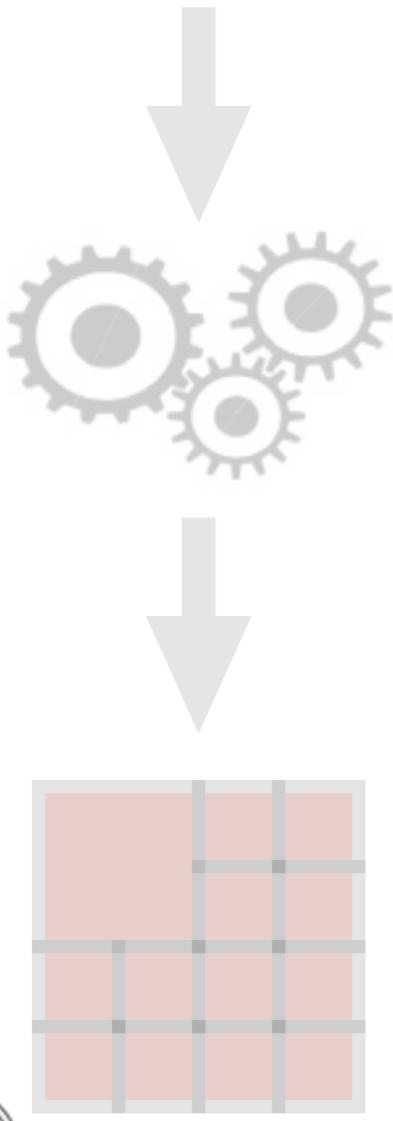
- ▶ Web applications have different tag counts



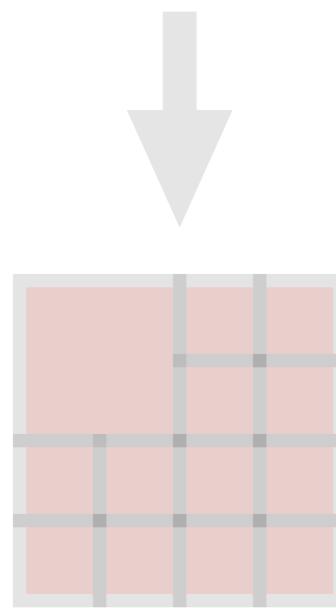
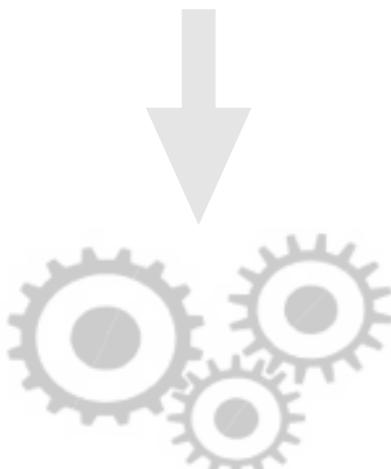
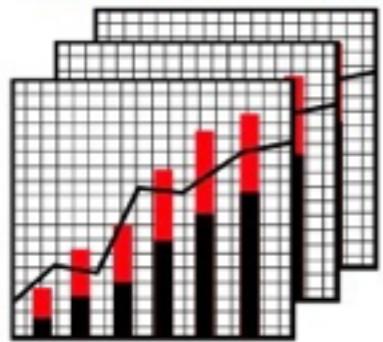
Tag Processing Overhead



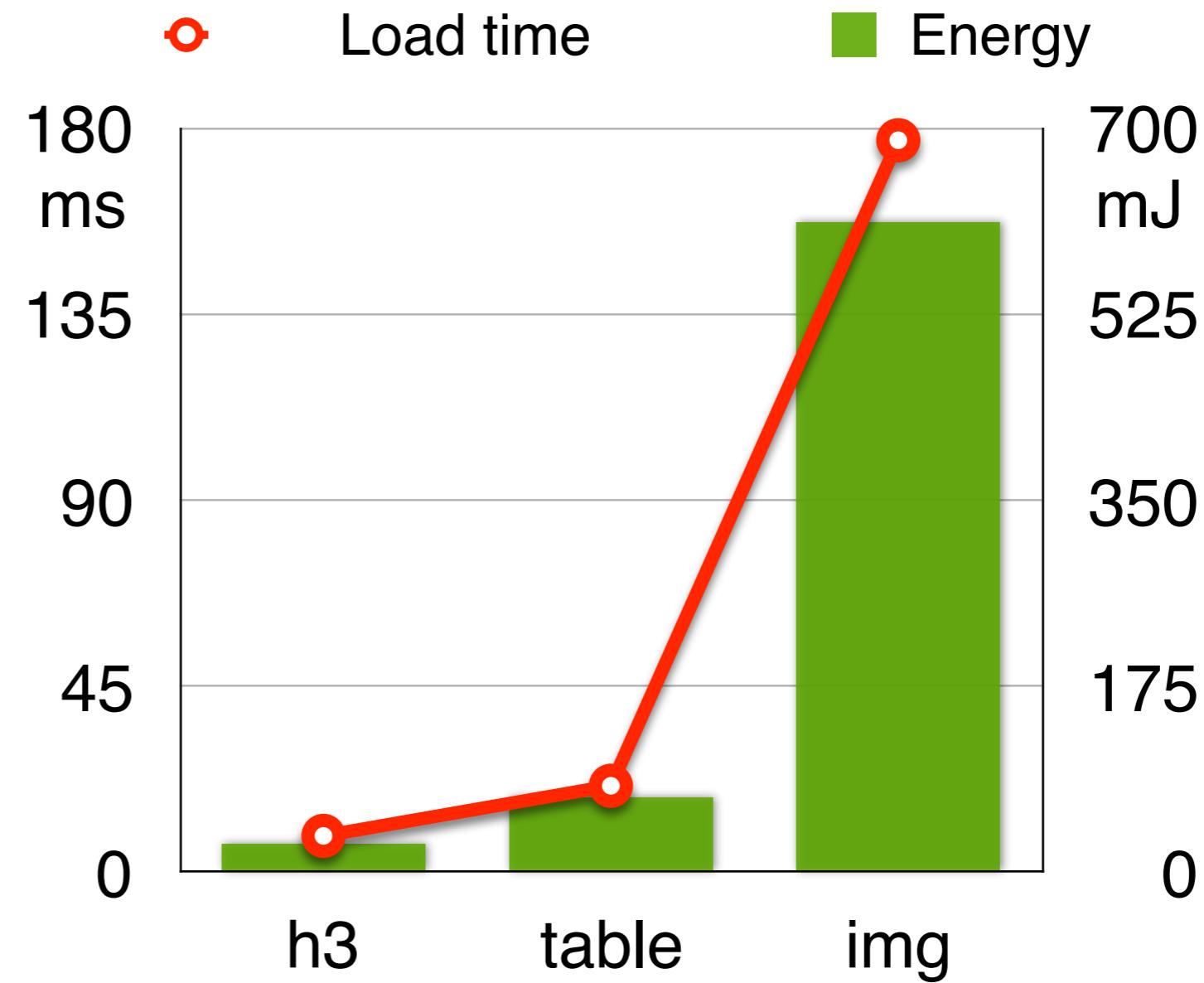
- ▶ Web applications have different tag counts



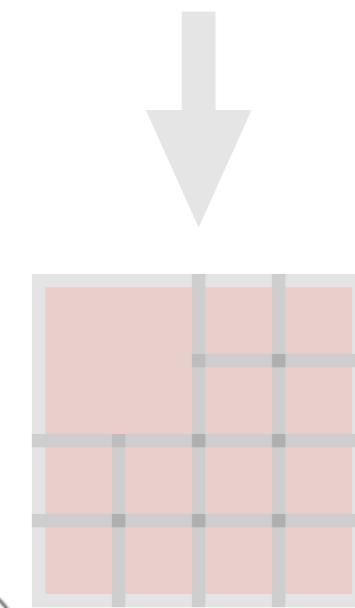
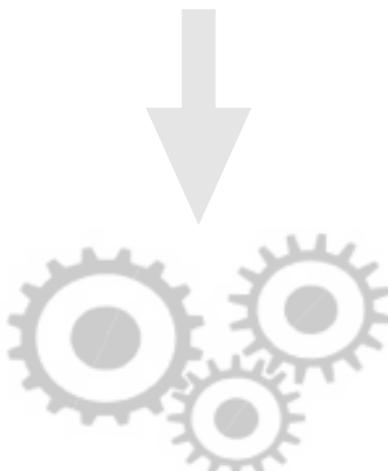
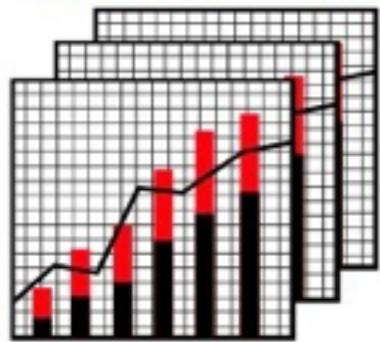
Tag Processing Overhead



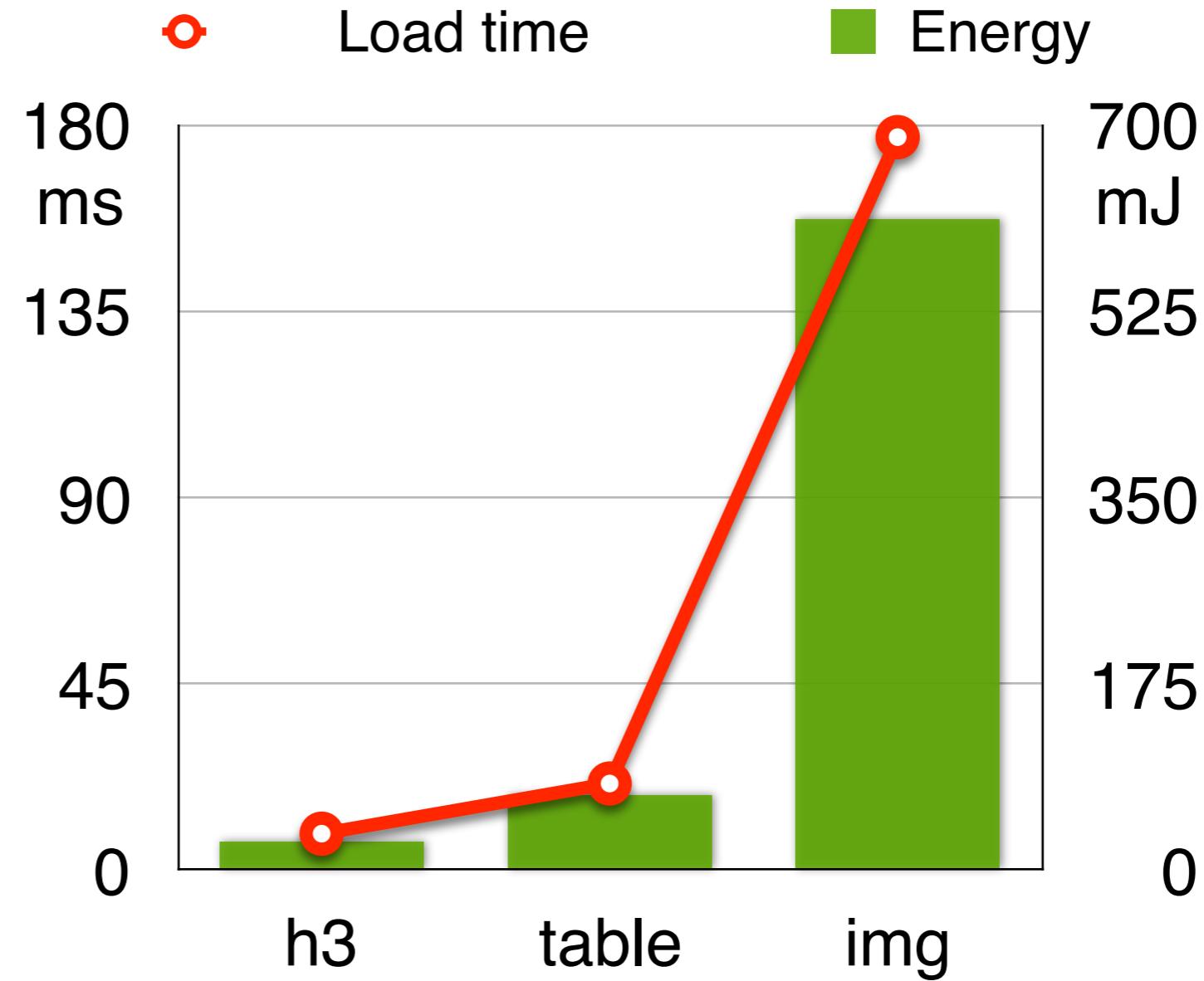
- ▶ Web applications have different tag counts



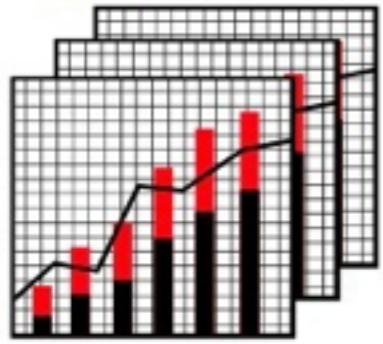
Tag Processing Overhead



- ▶ Web applications have different tag counts
- ▶ Tags have different processing overheads

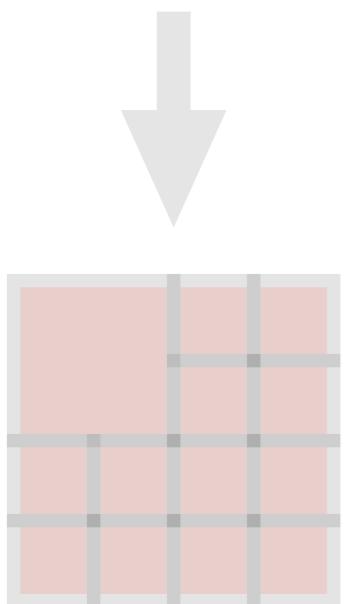
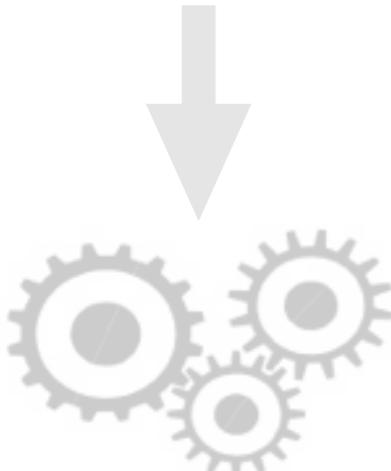


Tag Processing Overhead

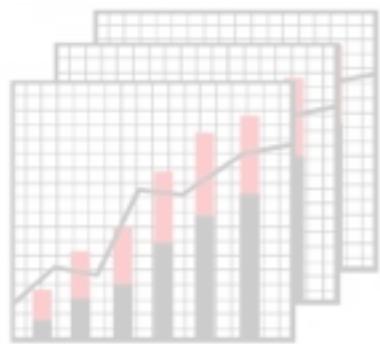


- ▶ Web applications have different tag counts
- ▶ Tags have different processing overheads

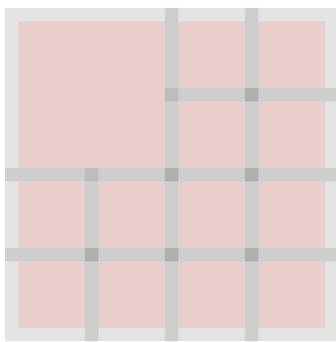
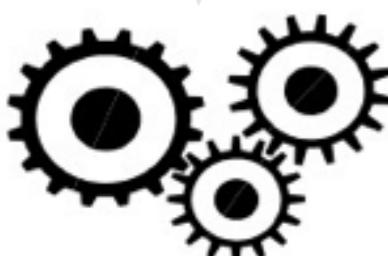
Root-cause of Web Application Variance



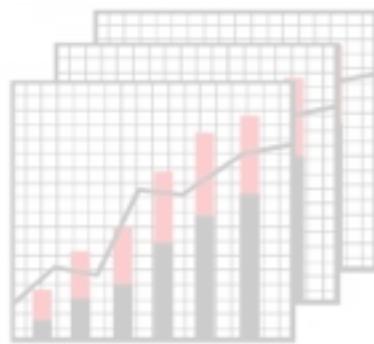
Predicting Loading Performance & Energy



Idea: predict load time & energy (responses)
based on Web primitives (predictors)



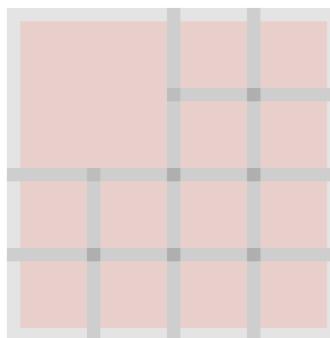
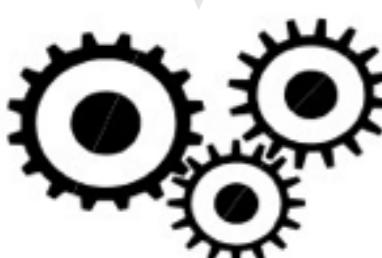
Predicting Loading Performance & Energy



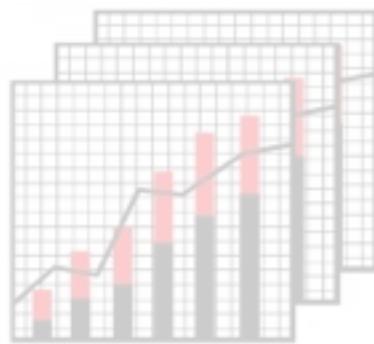
Identify Predictors
Training using hottest
2,500 webpages

Predictors
(HTML, CSS)

Responses
(Time, Energy)



Predicting Loading Performance & Energy

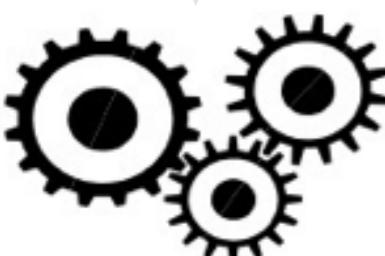


Identify Predictors

Training using hottest
2,500 webpages

Predictors
(HTML, CSS)

Responses
(Time, Energy)



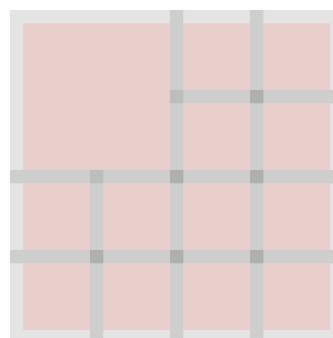
Model Construction & Refinement

Refine the linear model

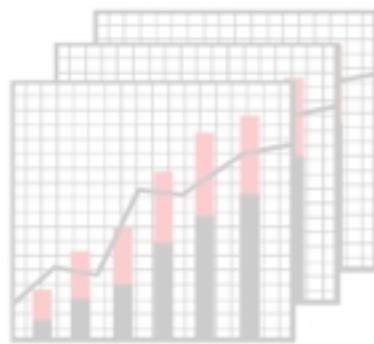
Linear Regression

Mitigate Over-fitting

Model Non-Linearity



Predicting Loading Performance & Energy



Identify Predictors

Training using hottest
2,500 webpages

Predictors
(HTML, CSS)

Responses
(Time, Energy)



Model Construction & Refinement

Refine the linear model



Linear Regression



Mitigate Over-fitting

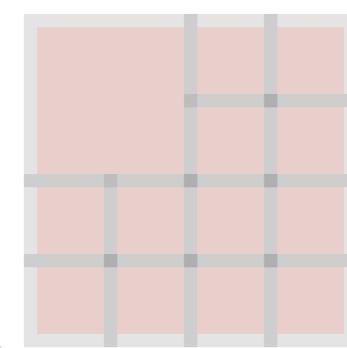


Model Non-Linearity



Loading Time
Model

Energy Model

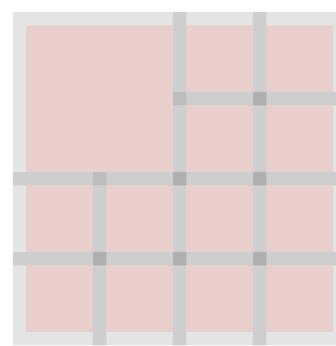
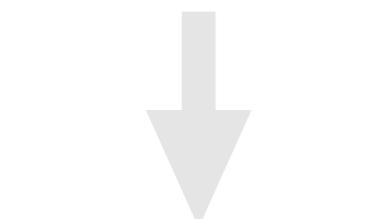
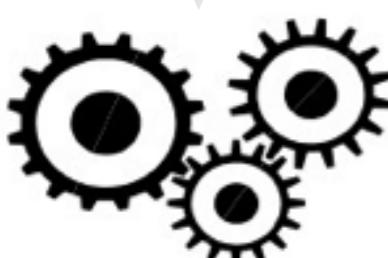
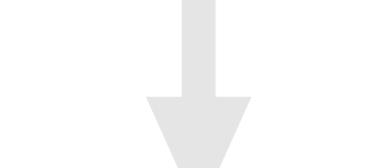
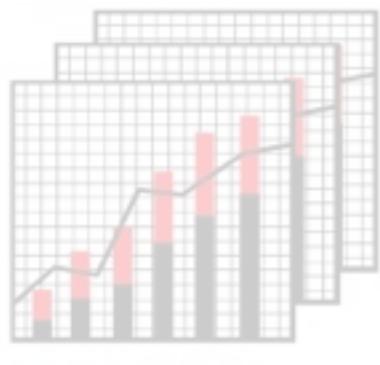


Model Validation

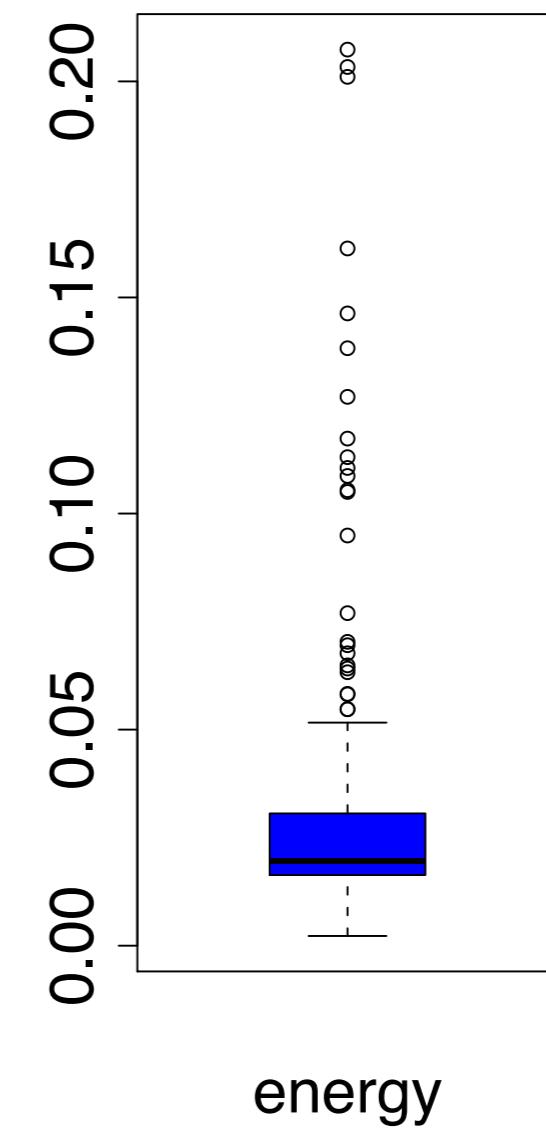
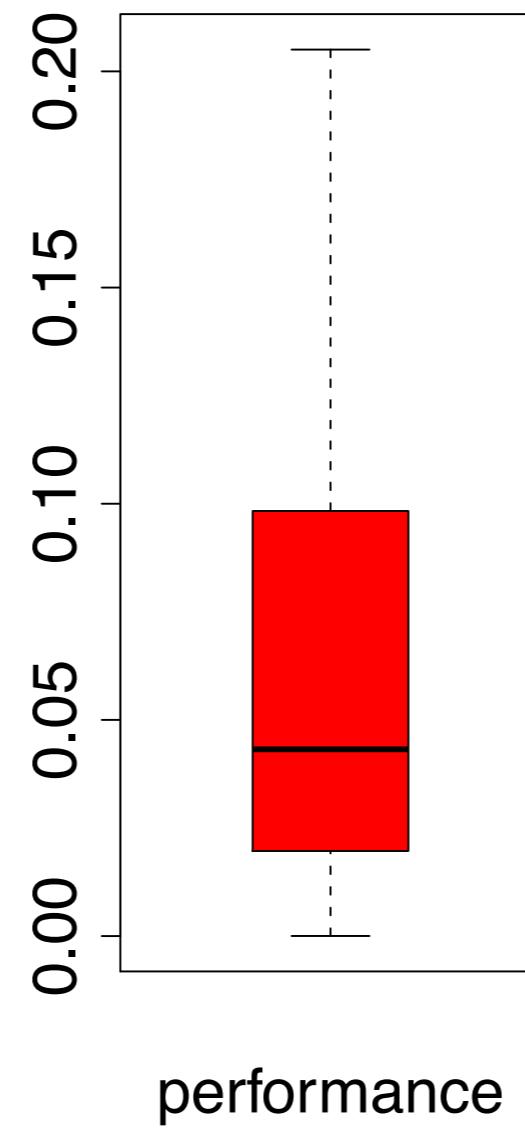
Validating on another
2,500 webpages



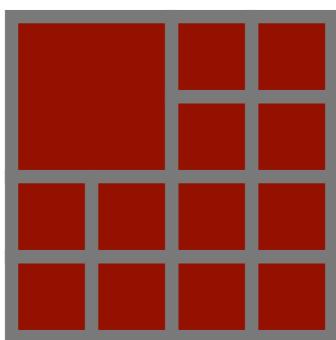
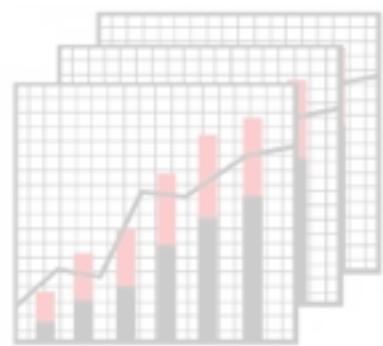
Predicting Loading Performance & Energy



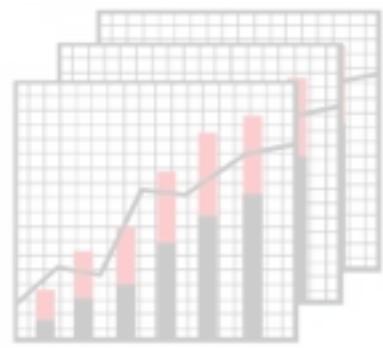
Median prediction error is less than 5%



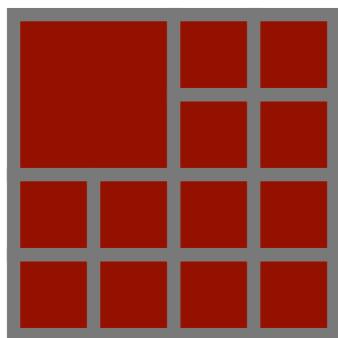
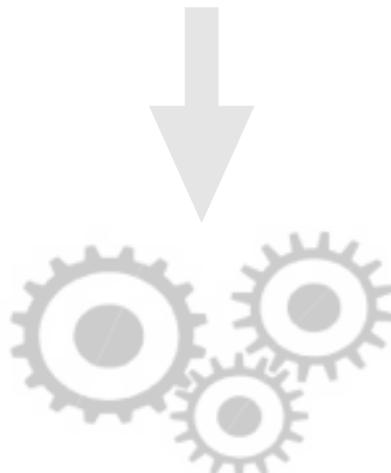
Webpage-aware Scheduler



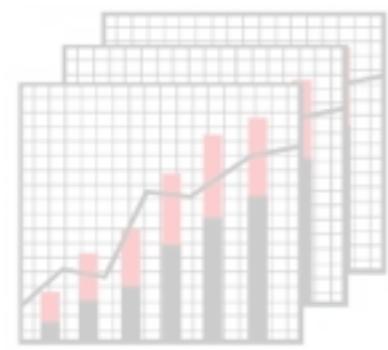
Webpage-aware Scheduler



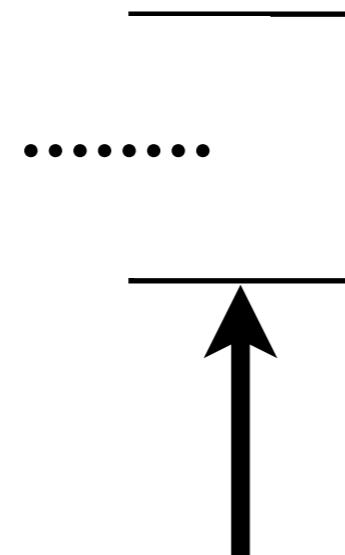
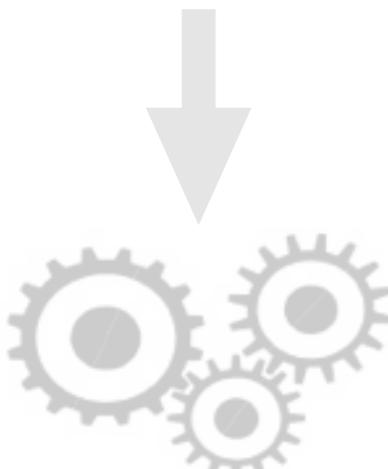
Normal Web application loading
Scheduler operations



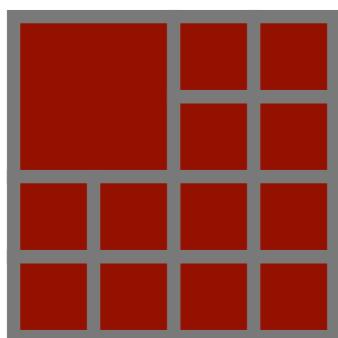
Webpage-aware Scheduler



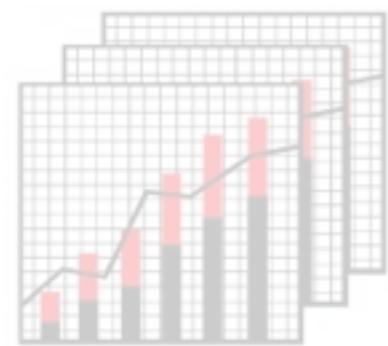
Normal Web application loading
Scheduler operations



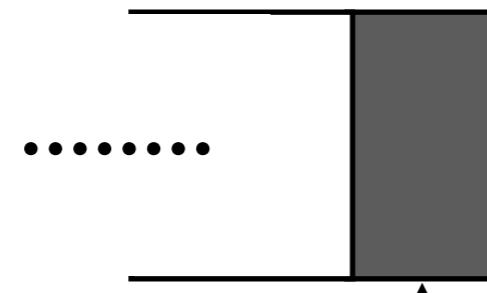
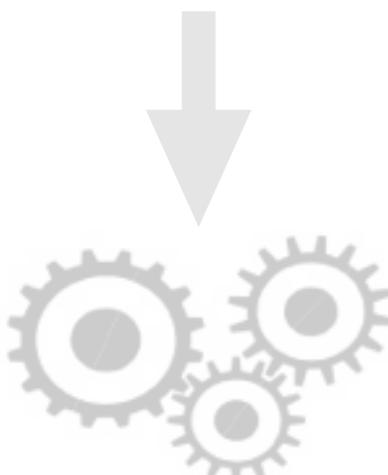
Network



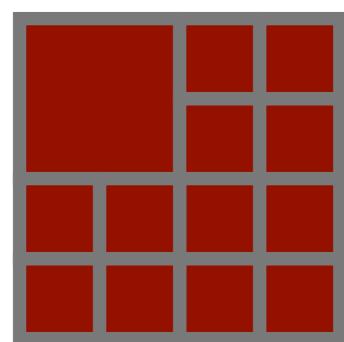
Webpage-aware Scheduler



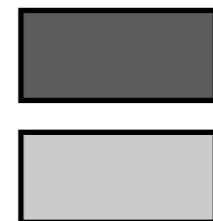
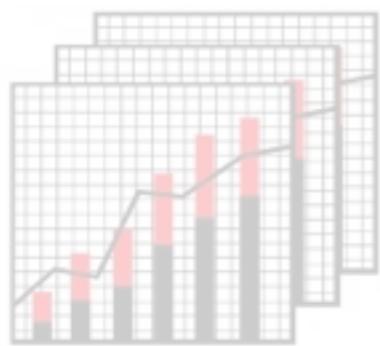
Normal Web application loading
Scheduler operations



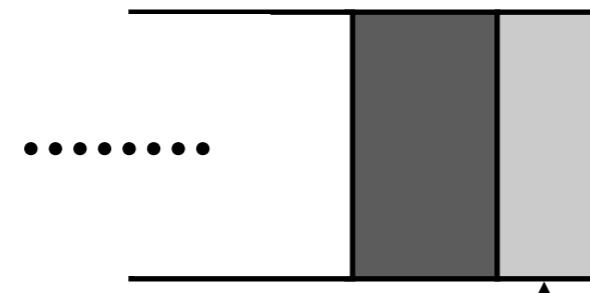
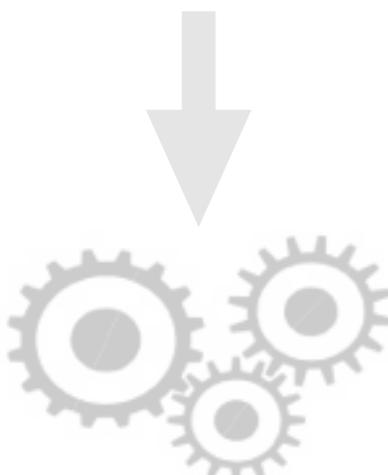
Parsing
(1~%)



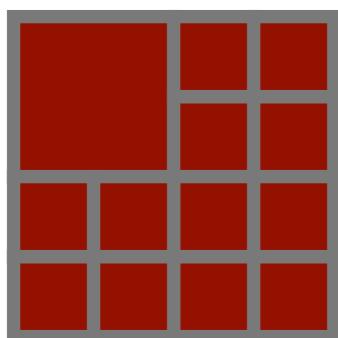
Webpage-aware Scheduler



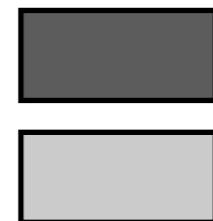
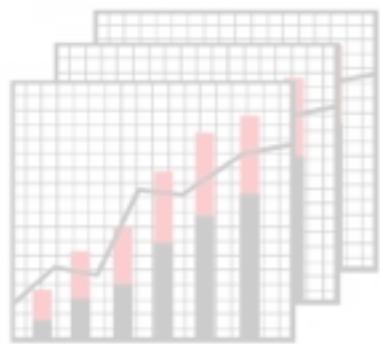
Normal Web application loading
Scheduler operations



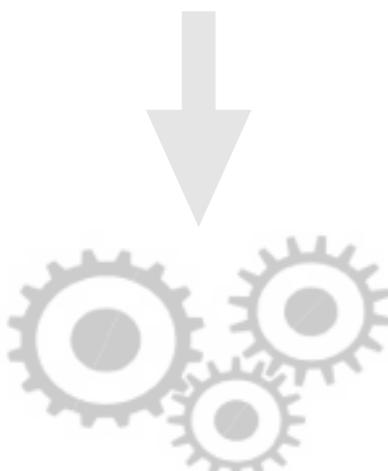
Prediction
(minimal overhead)



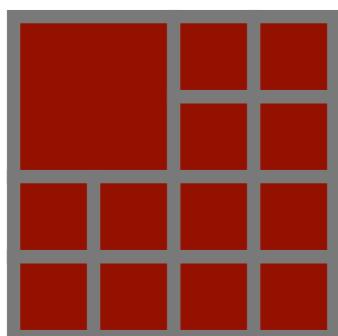
Webpage-aware Scheduler



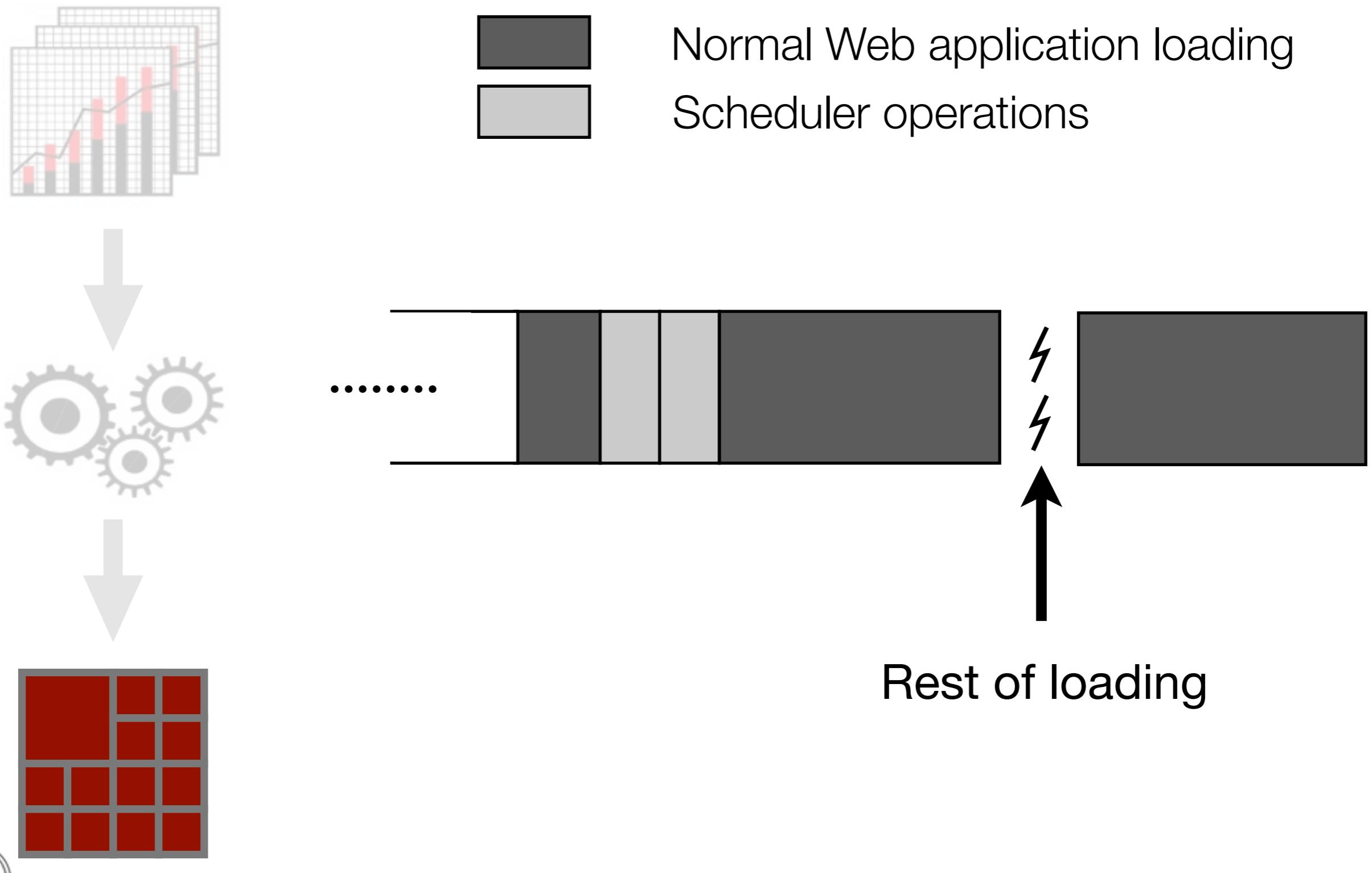
Normal Web application loading
Scheduler operations



Scheduling Overhead
(~120 us)



Webpage-aware Scheduler



Evaluation



Evaluation

- ▶ Highest performance (Perf)
 - ▷ Highest frequency on big core
 - ▷ Standard to guarantee responsiveness



Evaluation

- ▶ Highest performance (Perf)
 - ▷ Highest frequency on big core
 - ▷ Standard to guarantee responsiveness
- ▶ OS DVFS strategies (OS)
 - ▷ Ondemand governor (across big and little cores)



Evaluation

- ▶ Highest performance (Perf)
 - ▷ Highest frequency on big core
 - ▷ Standard to guarantee responsiveness
- ▶ OS DVFS strategies (OS)
 - ▷ Ondemand governor (across big and little cores)
- ▶ Metrics:
 - ▷ Energy Savings
 - ▷ QoS Violations



Evaluation

- ▶ Highest performance (Perf)
 - ▷ Highest frequency on big core
 - ▷ Standard to guarantee responsiveness

83.0% energy savings over Perf, 4.1% more QoS violations

- ▷ Ondemand governor (across big and little cores)

- ▶ Metrics:
 - ▷ Energy Savings
 - ▷ QoS Violations



Evaluation

- ▶ Highest performance (Perf)
 - ▷ Highest frequency on big core
 - ▷ Standard to guarantee responsiveness

83.0% energy savings over Perf, **4.1%** more QoS violations

8.6% energy savings over OS, **0.1%** more QoS violations

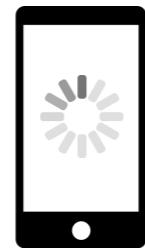
- ▷ Energy Savings
- ▷ QoS Violations



WebRT: Energy-aware Web Runtime

Interactions

Loading



Touching



Moving



WebRT
Component

Proactive
Mechanism

History-
based
Mechanism



WebRT: Energy-aware Web Runtime

Interactions

Loading



Touching



Moving



WebRT Component

Proactive
Mechanism

History-
based
Mechanism



Optimizing Post-loading Interactions



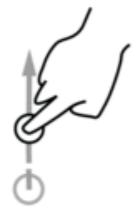
Optimizing Post-loading Interactions

Interactions

Touching



Moving



Optimizing Post-loading Interactions

Interactions → Events

Touching



Moving



Optimizing Post-loading Interactions

Interactions → Events

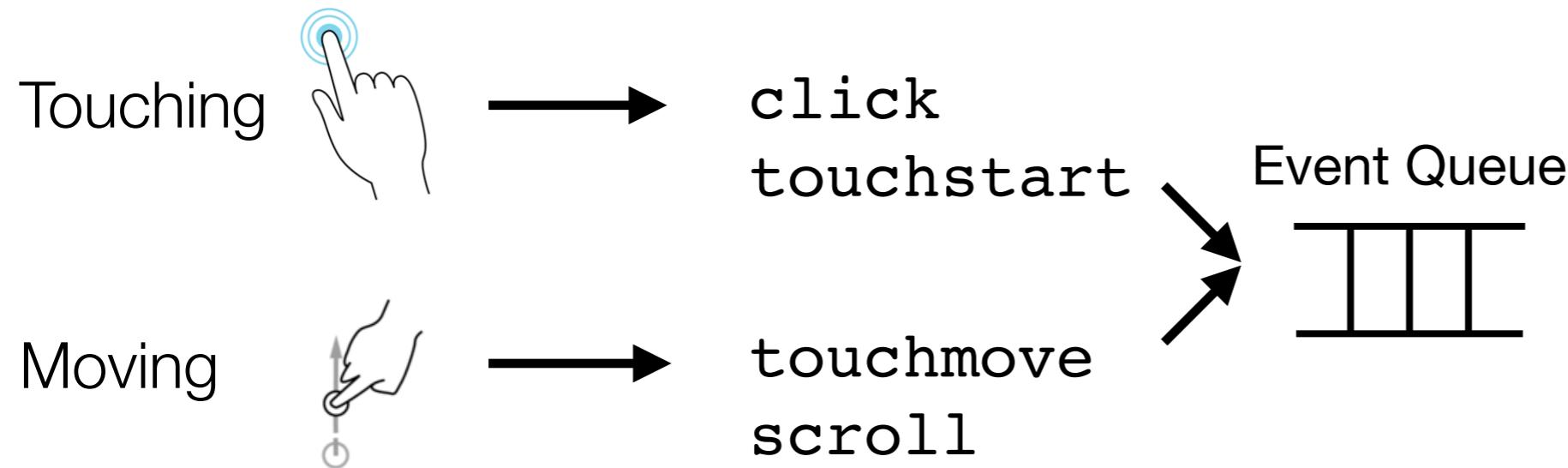
Touching  → click
touchstart

Moving  → touchmove
scroll



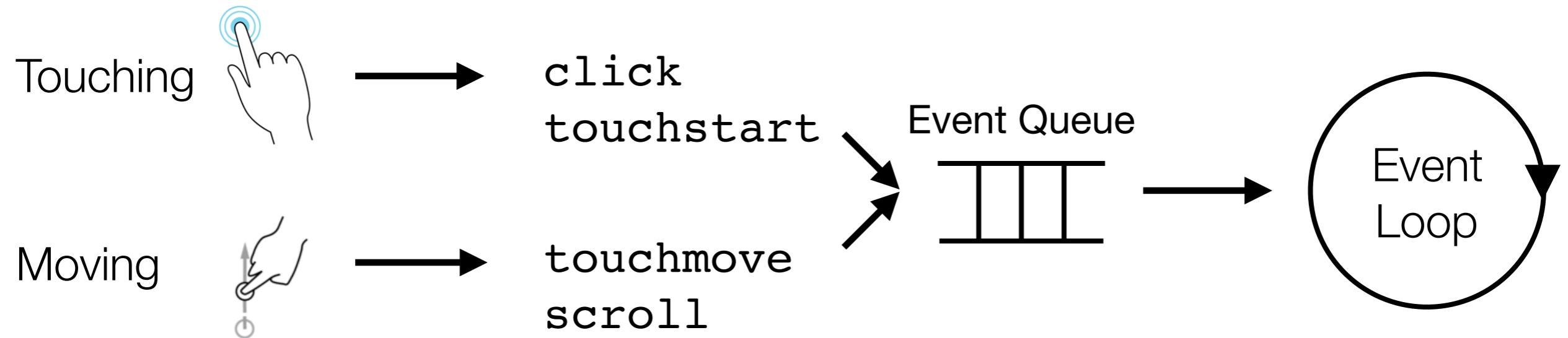
Optimizing Post-loading Interactions

Interactions → Events



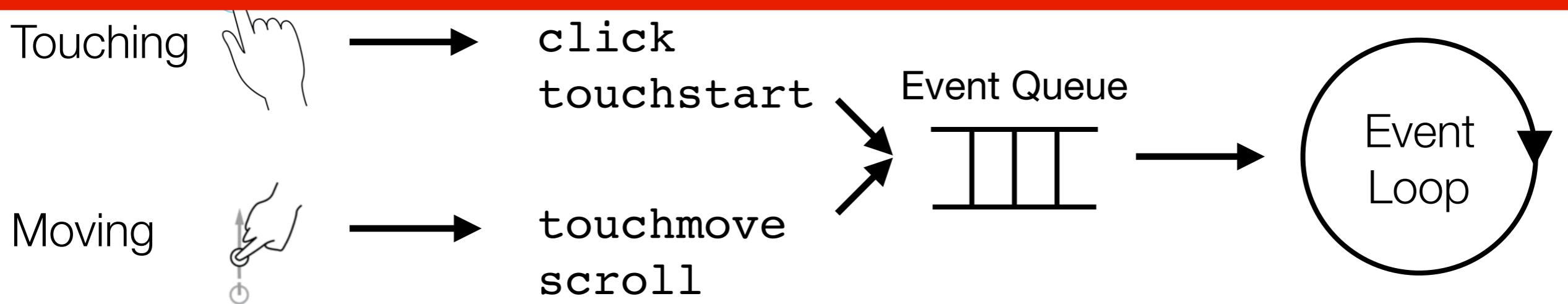
Optimizing Post-loading Interactions

Interactions → Events

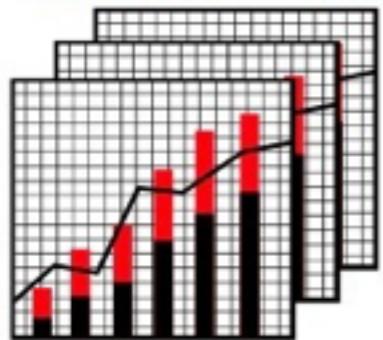


Optimizing Post-loading Interactions

Optimize post-loading at an event-granularity

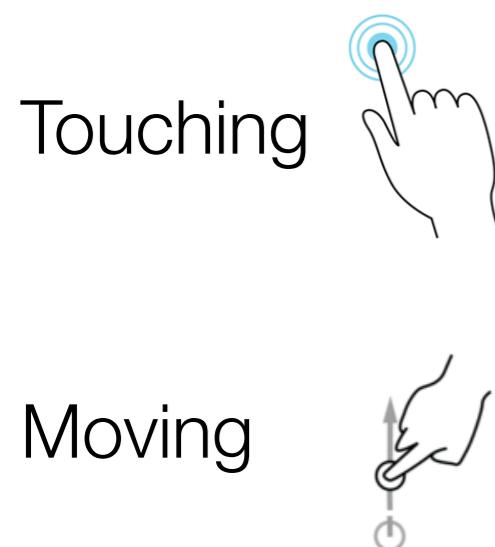


Optimizing Post-loading Interactions



- ▶ Observation: Events have different execution latencies that enable energy optimizations

Interactions → Events



Touching

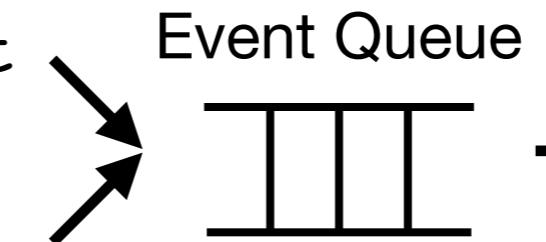
click

touchstart

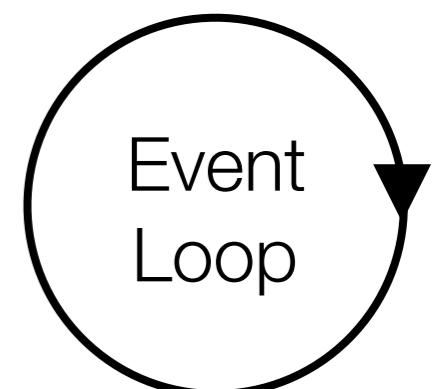


Moving

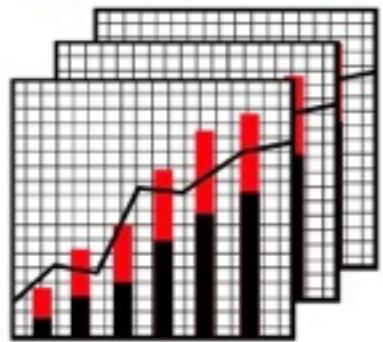
touchmove
scroll



Event Queue



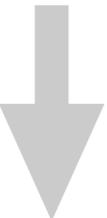
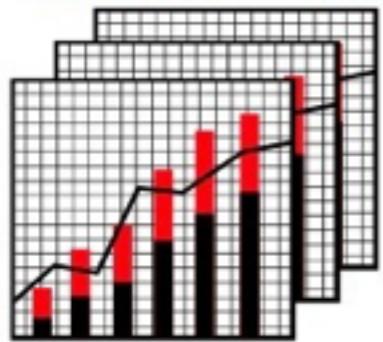
Optimizing Post-loading Interactions



- ▶ Observation: Events have different execution latencies that enable energy optimizations

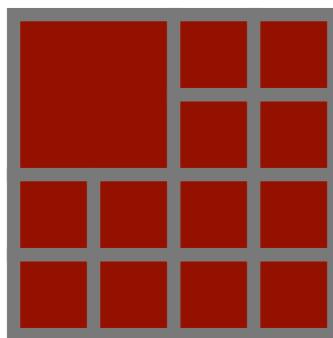
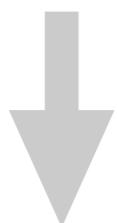
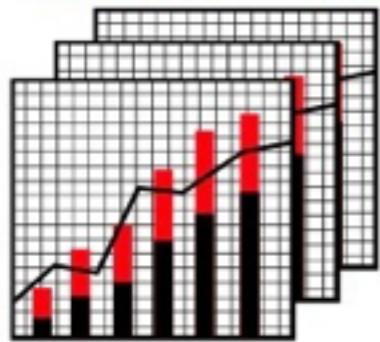


Optimizing Post-loading Interactions



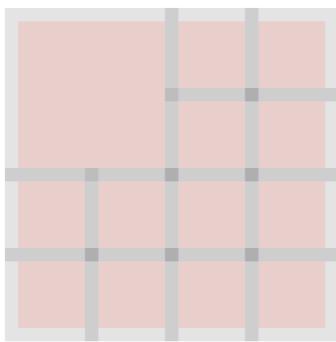
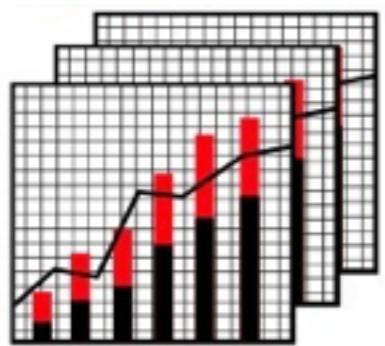
- ▶ **Observation:** Events have different execution latencies that enable energy optimizations
- ▶ **Mechanism:** Event-based scheduling to predict the ACMP configuration that exploits event slacks and saves energy

Optimizing Post-loading Interactions

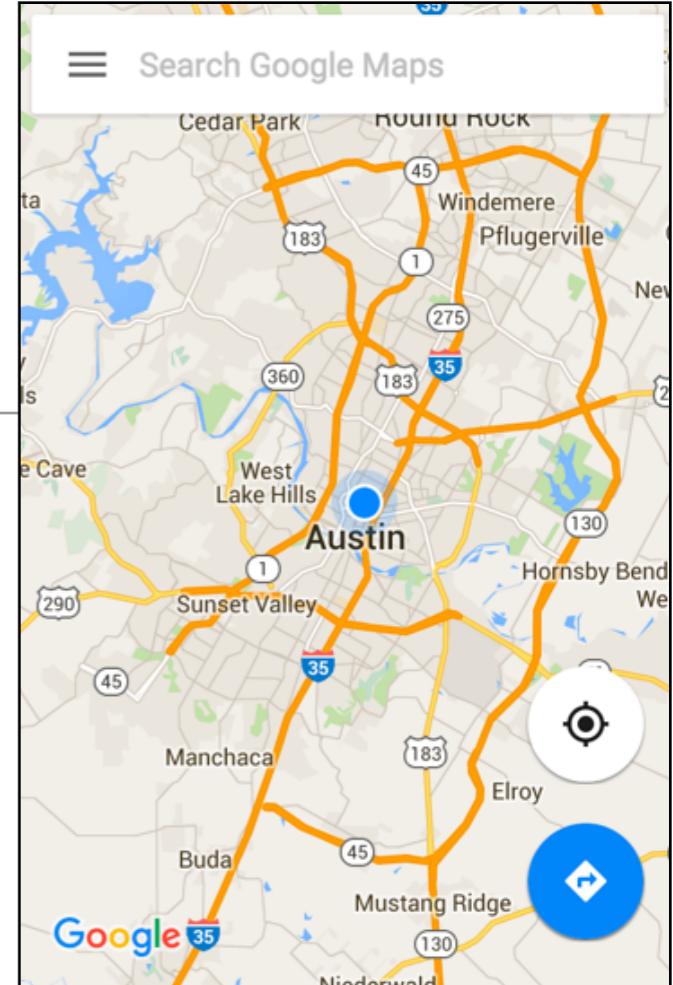
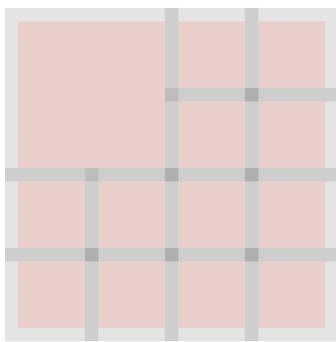
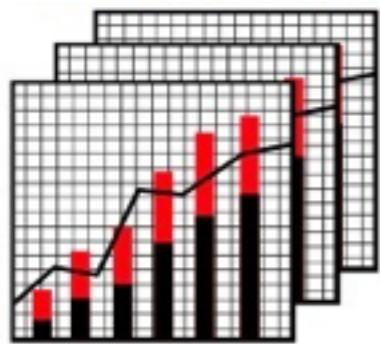


- ▶ **Observation:** Events have different execution latencies that enable energy optimizations
- ▶ **Mechanism:** Event-based scheduling to predict the ACMP configuration that exploits event slacks and saves energy
- ▶ **Effect:** Properly provision the hardware resources based on event characteristics

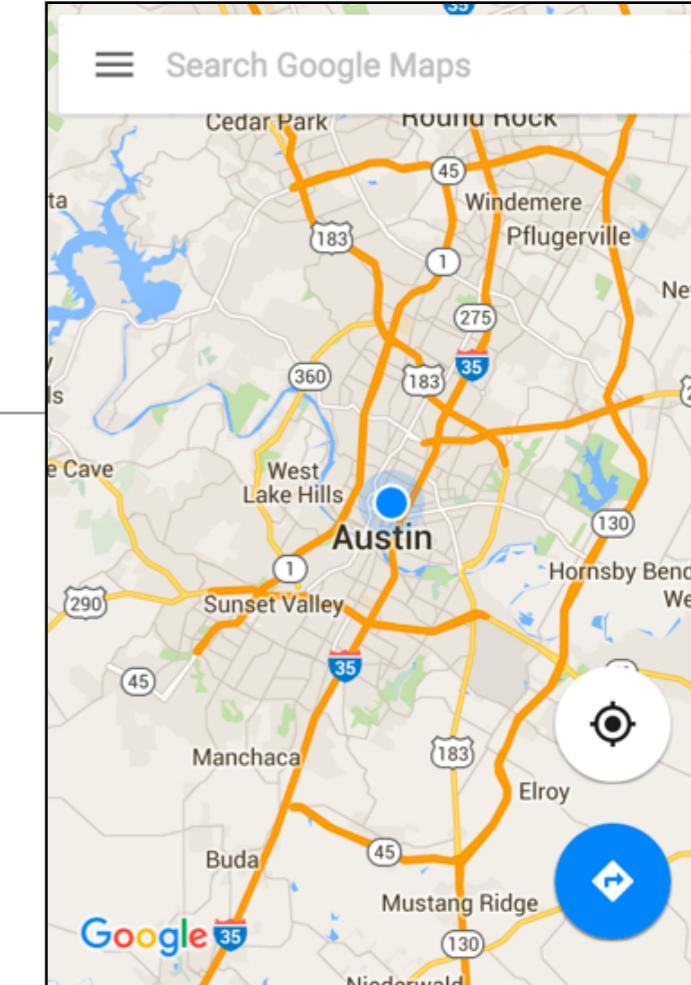
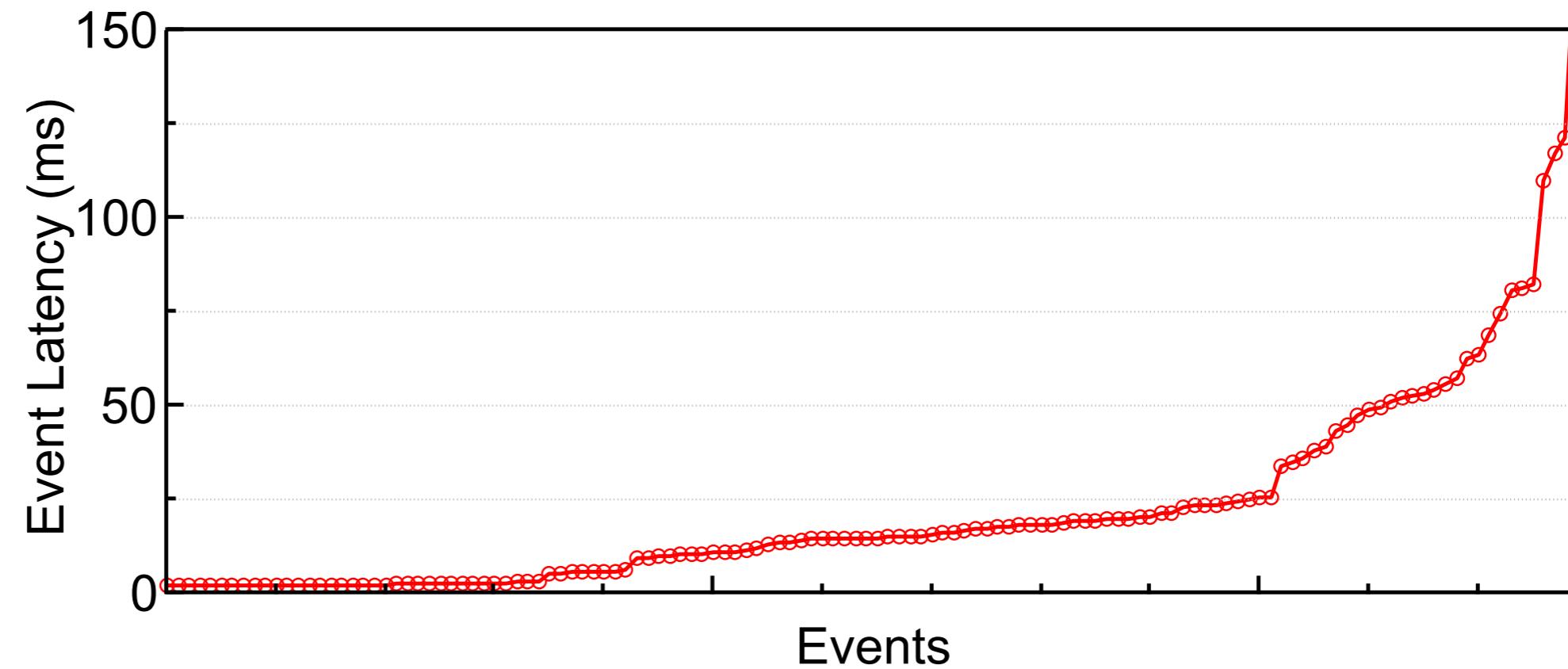
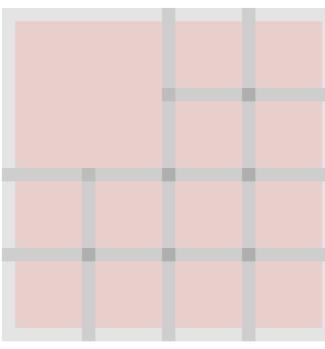
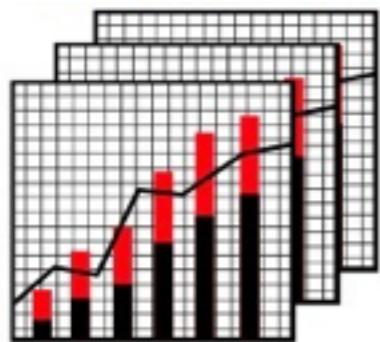
Event-Level Characterization



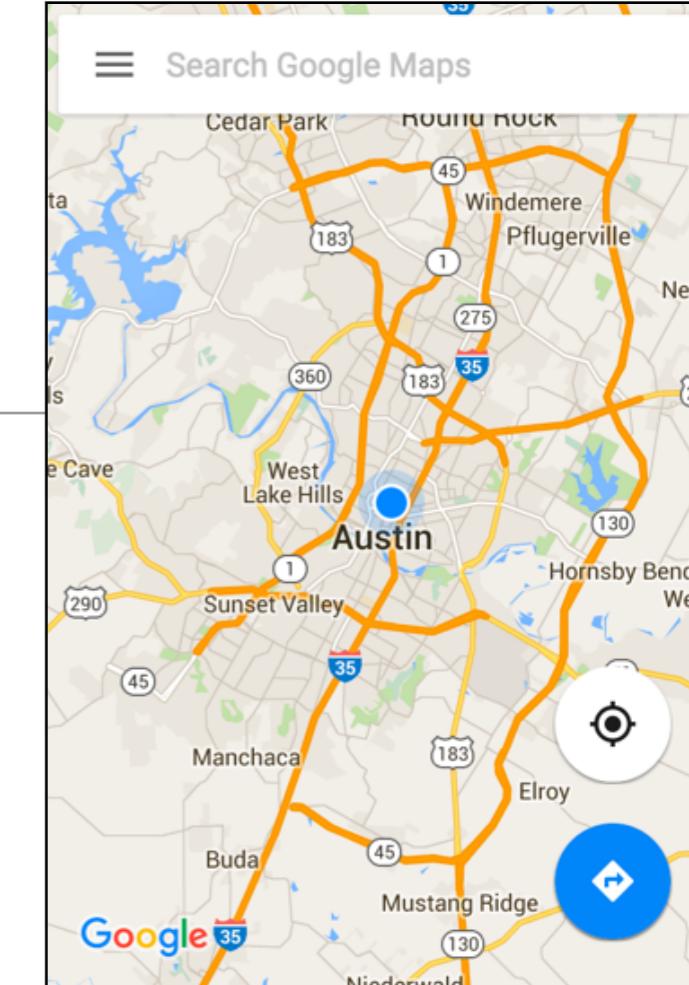
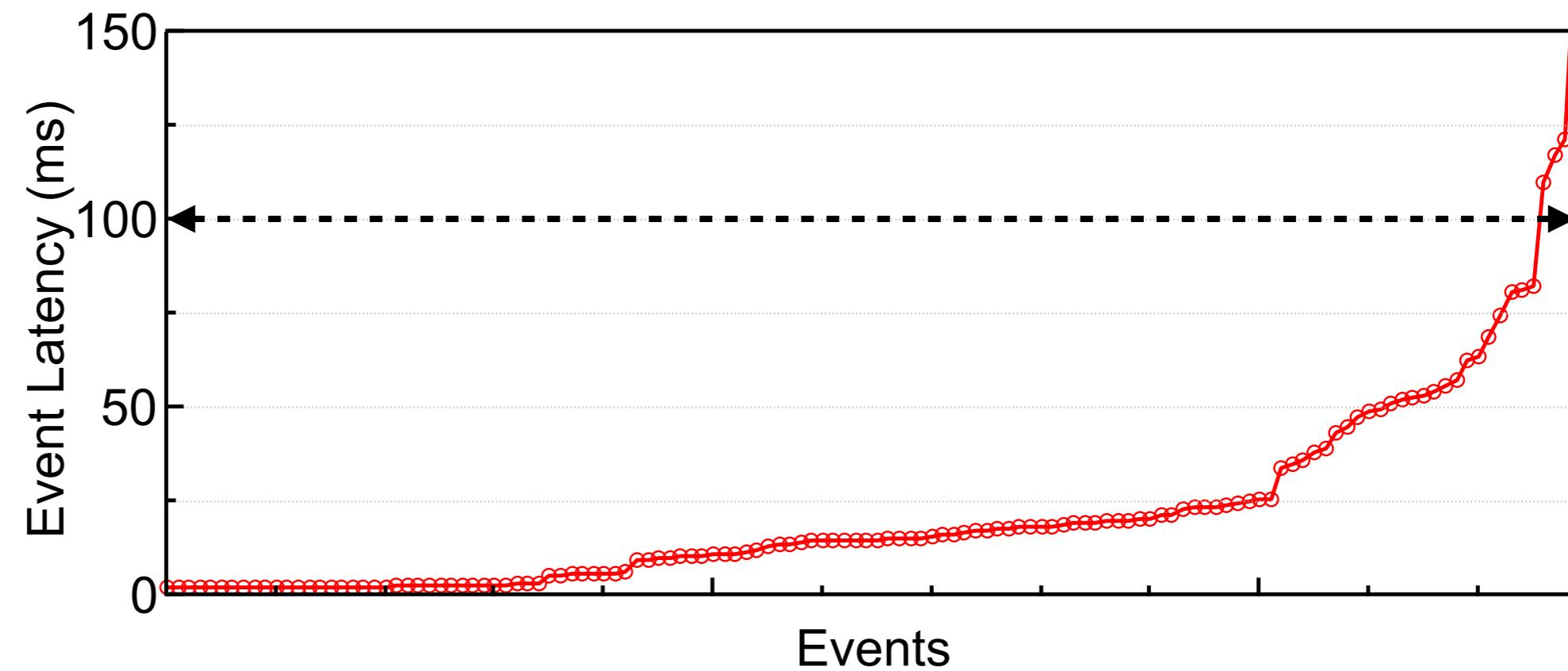
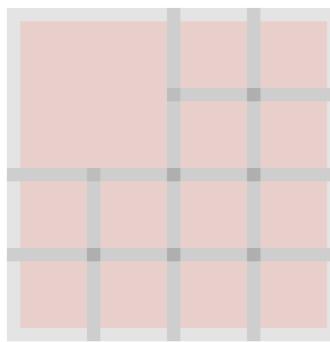
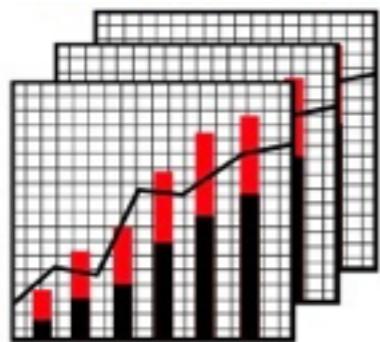
Event-Level Characterization



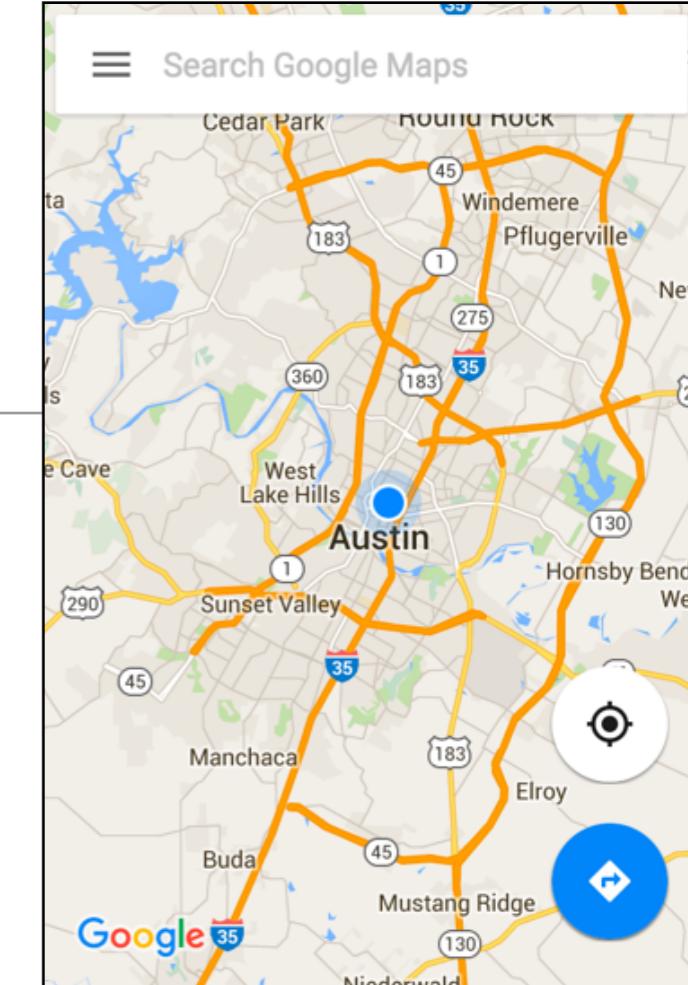
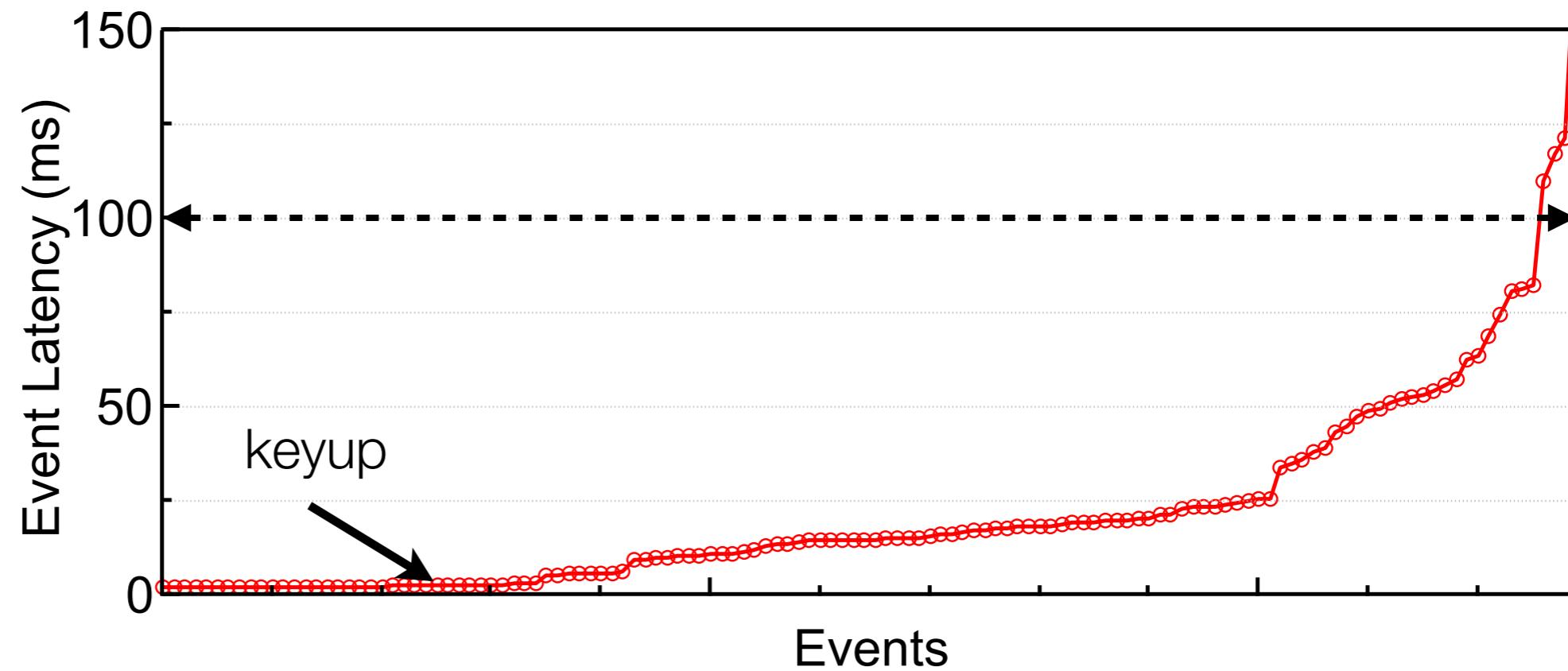
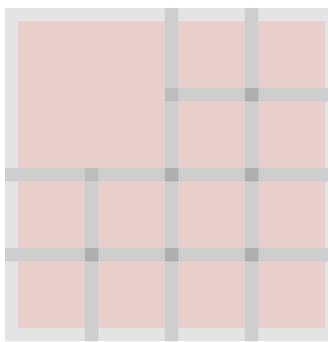
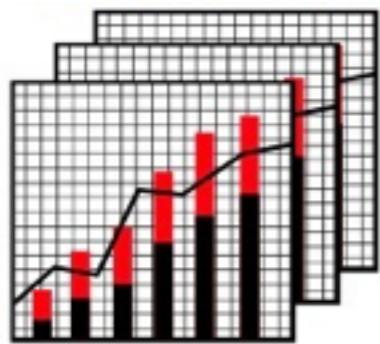
Event-Level Characterization



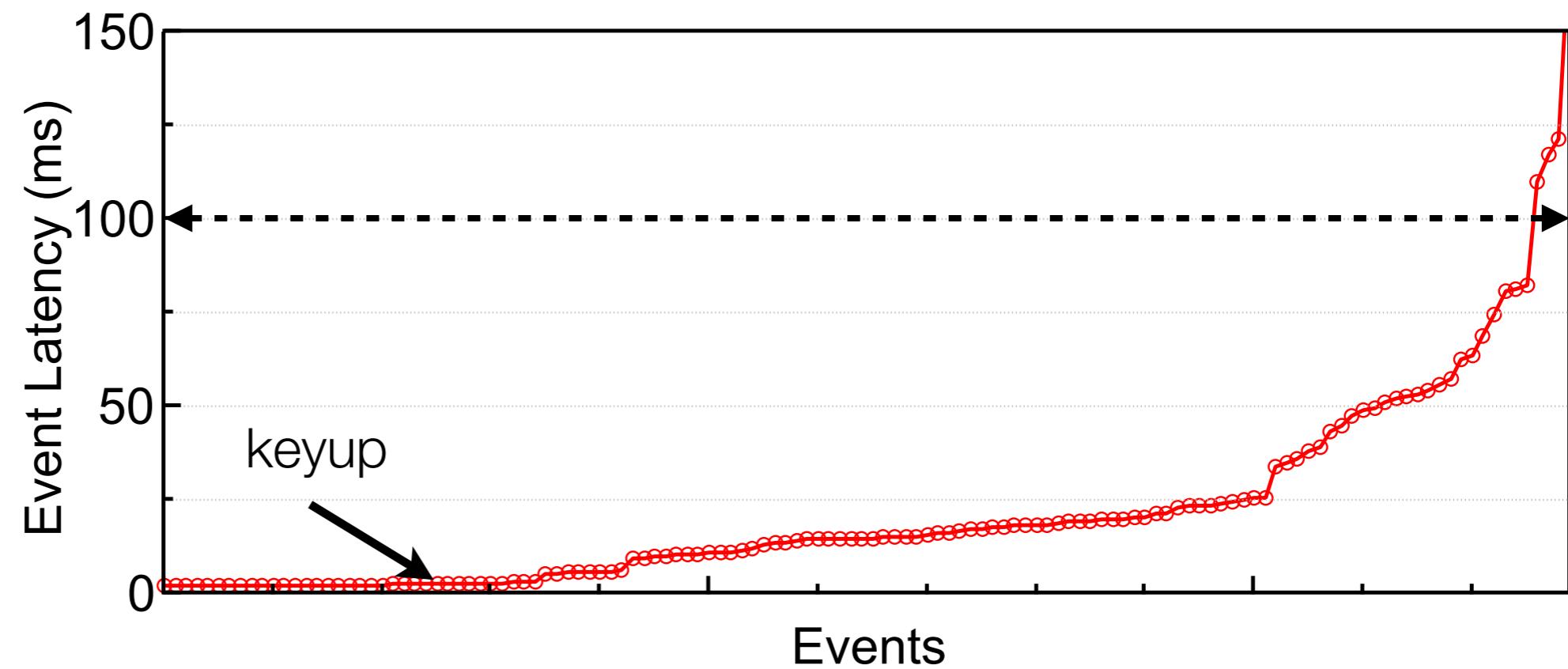
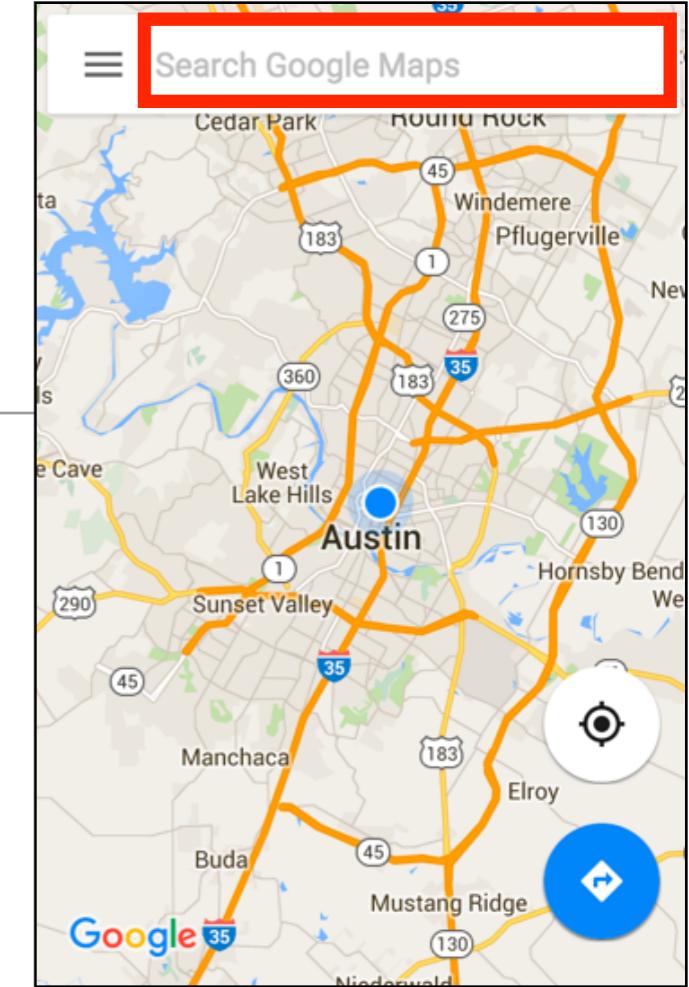
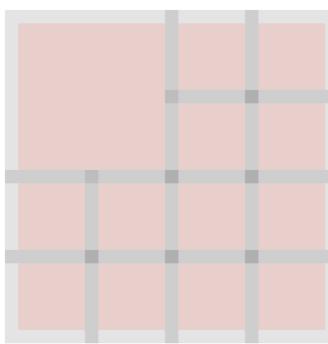
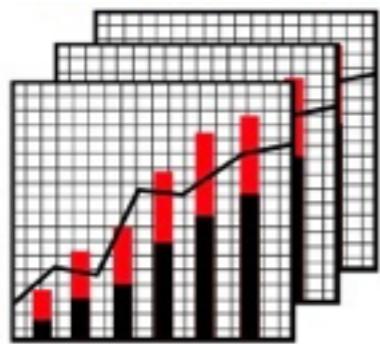
Event-Level Characterization



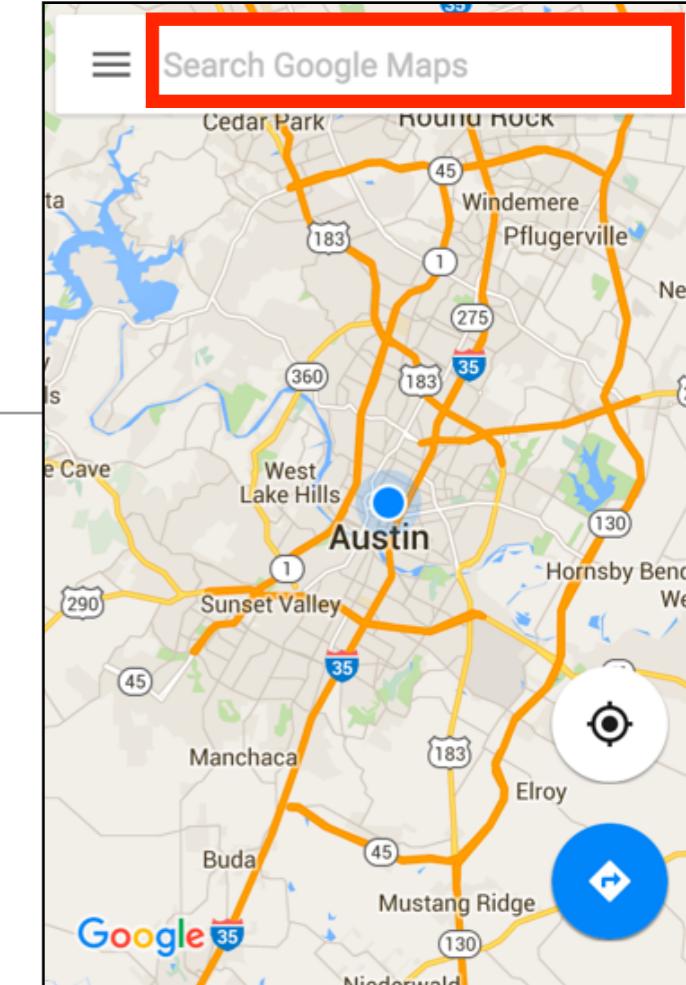
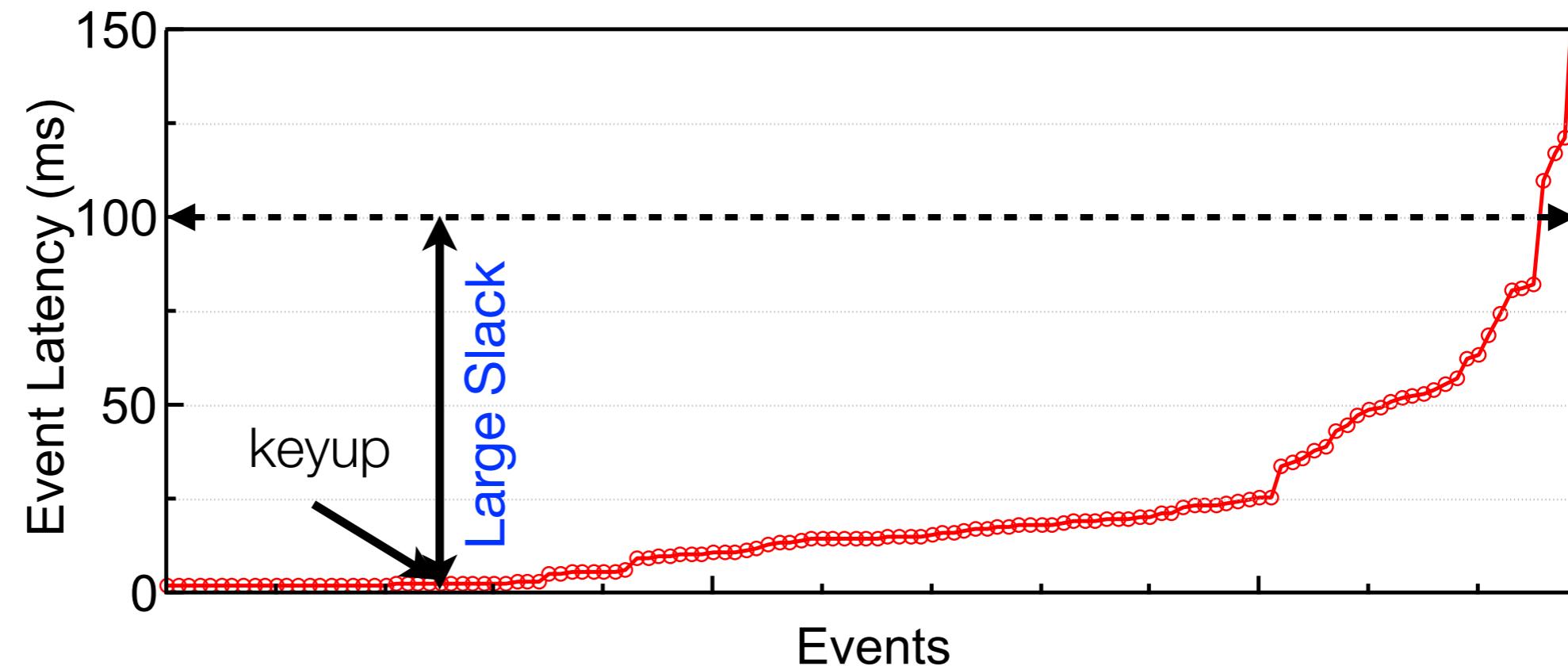
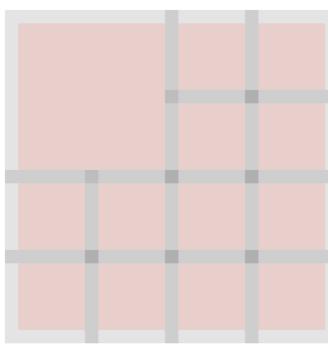
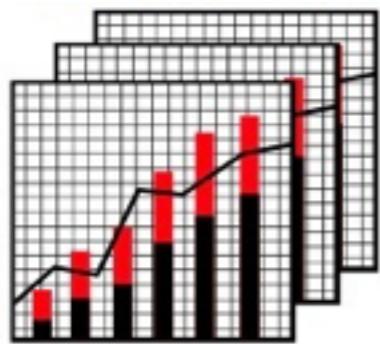
Event-Level Characterization



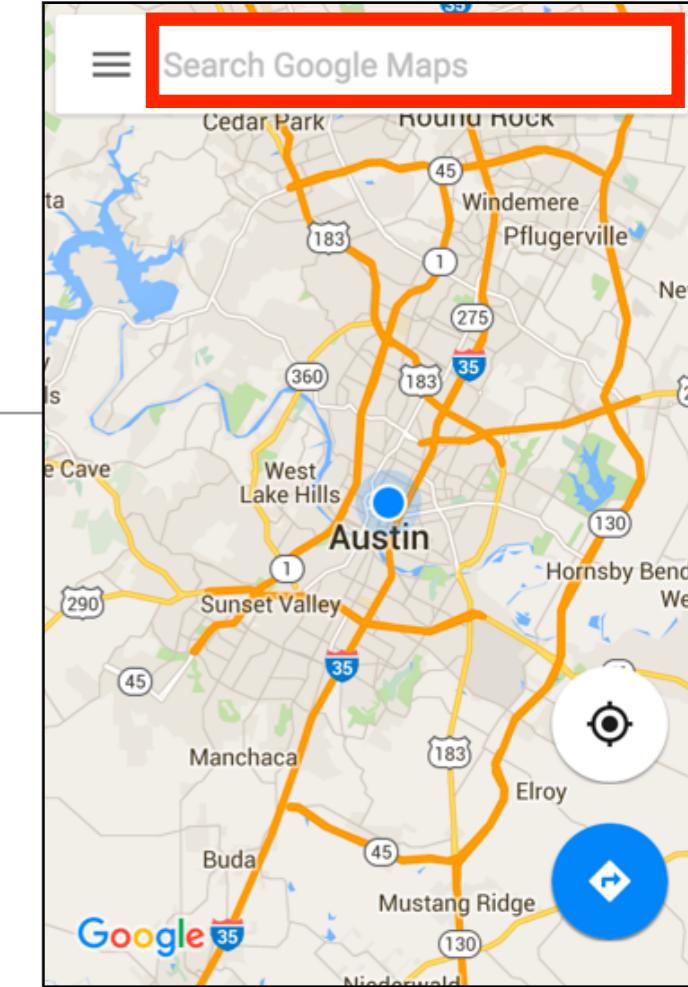
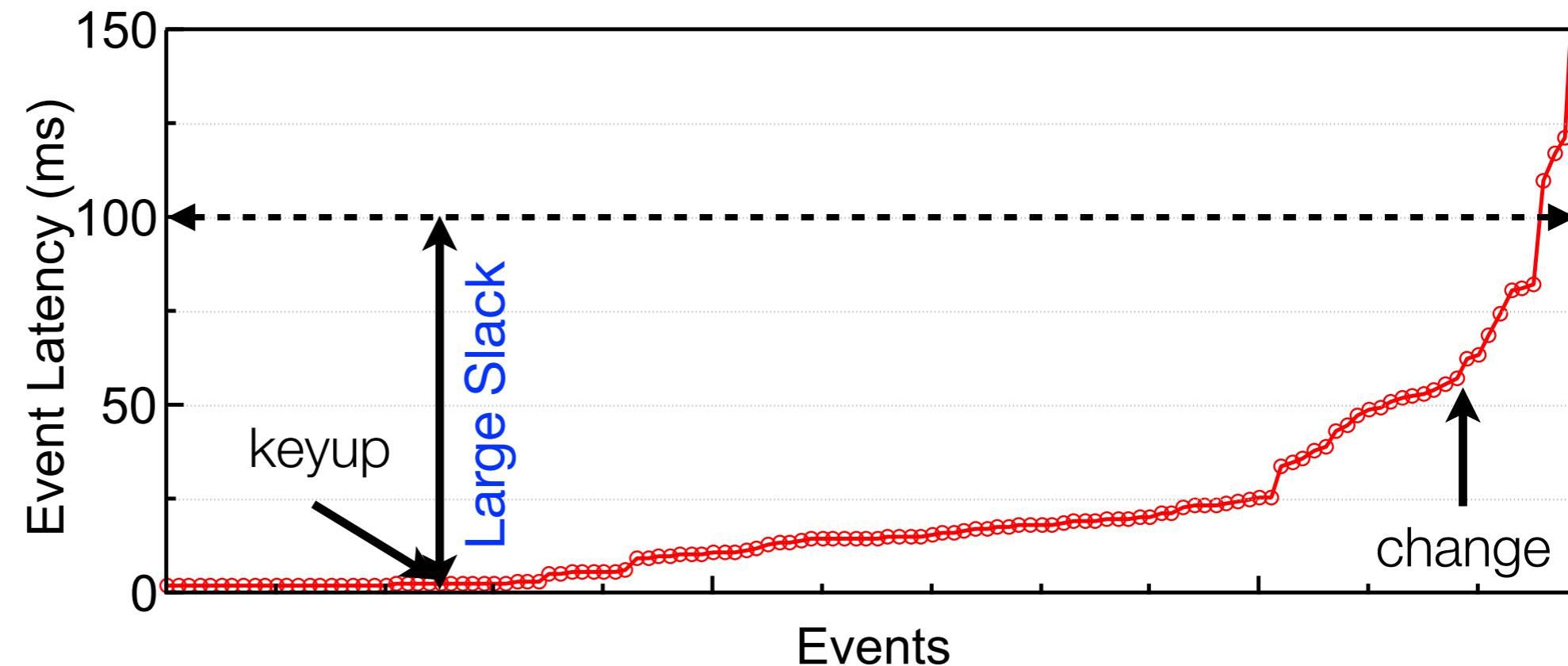
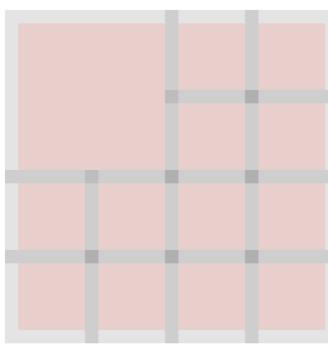
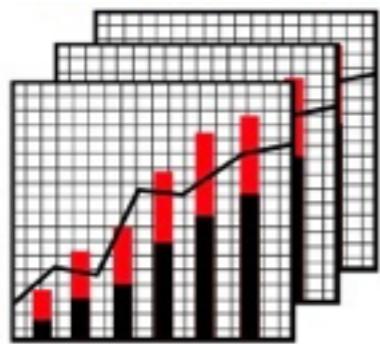
Event-Level Characterization



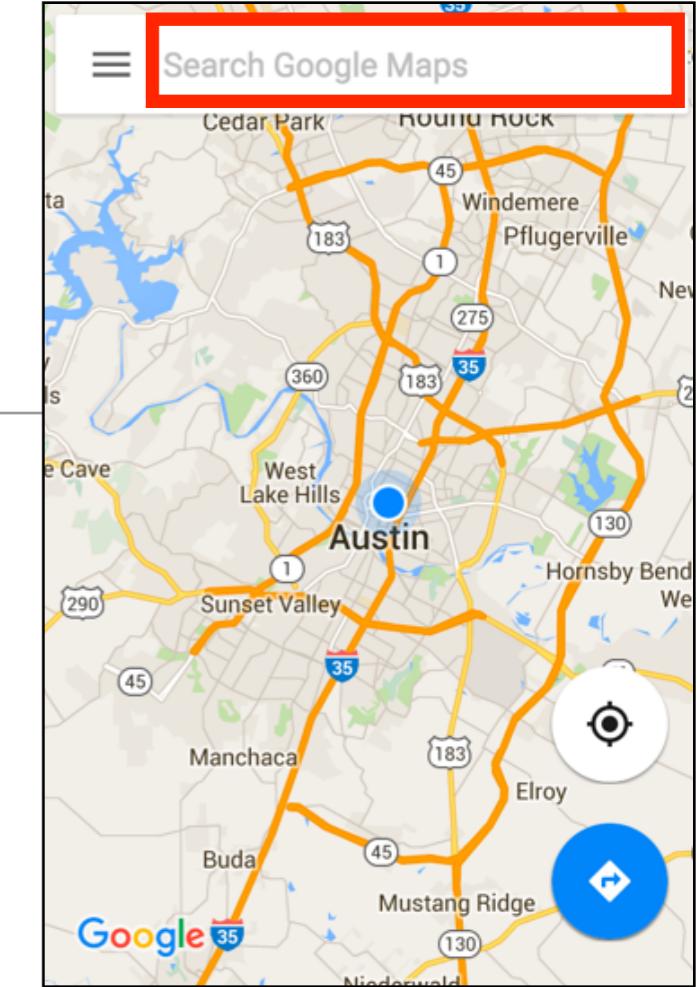
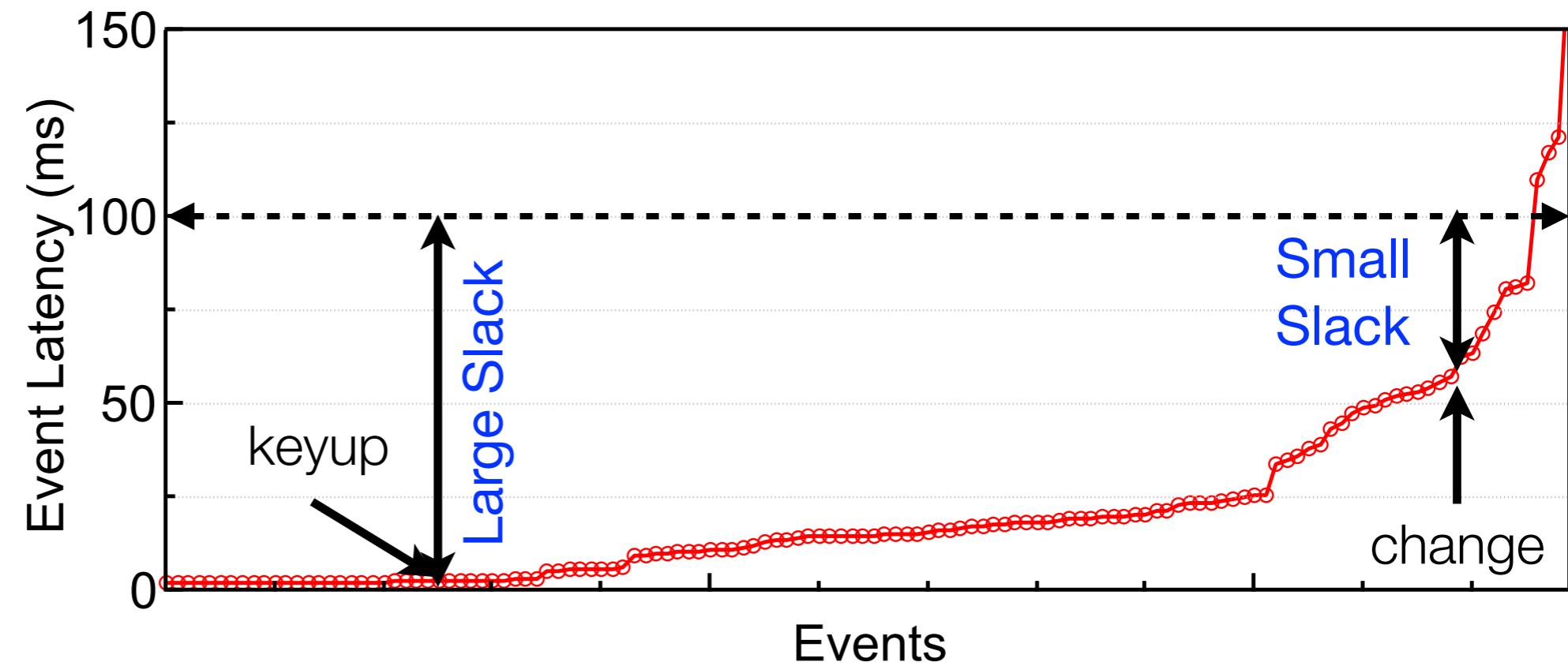
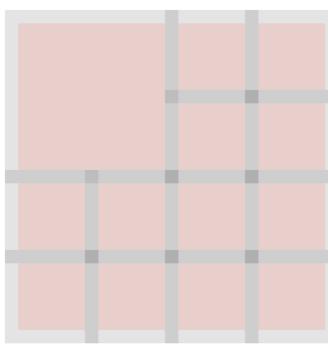
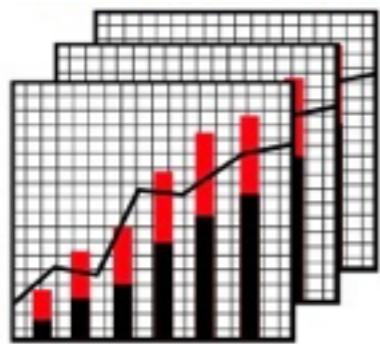
Event-Level Characterization



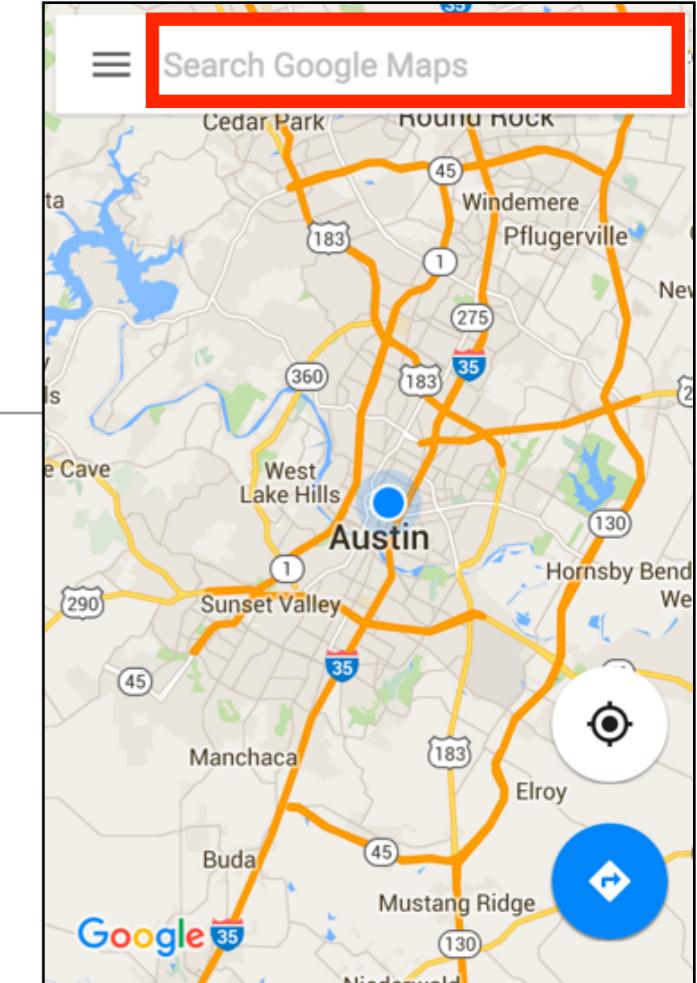
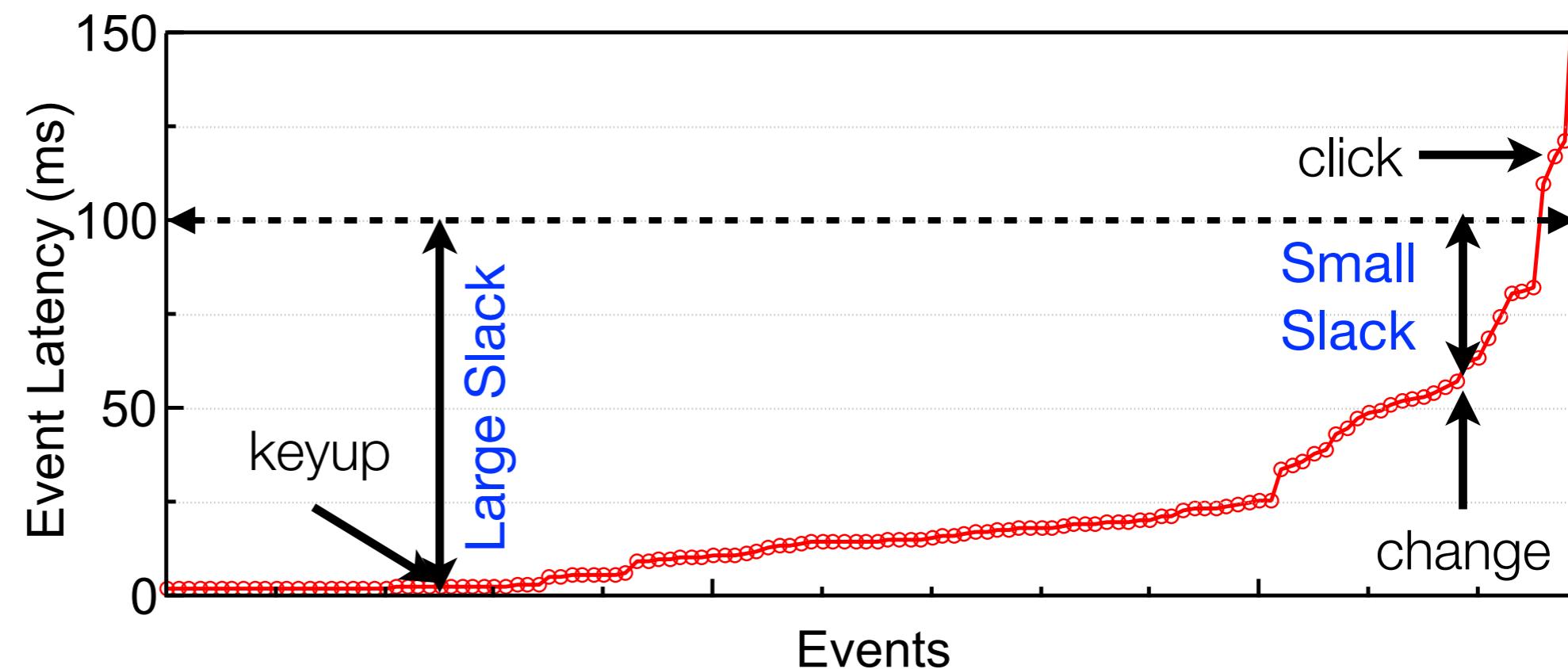
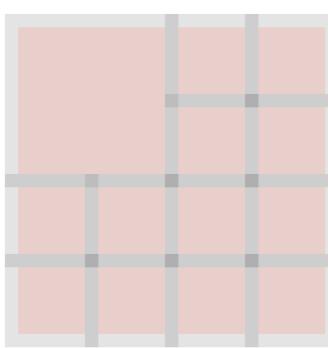
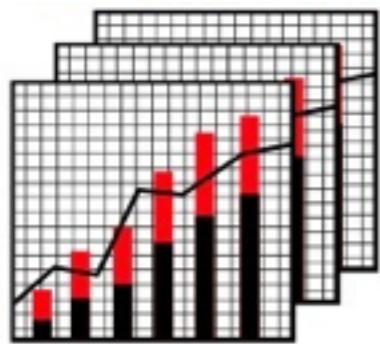
Event-Level Characterization



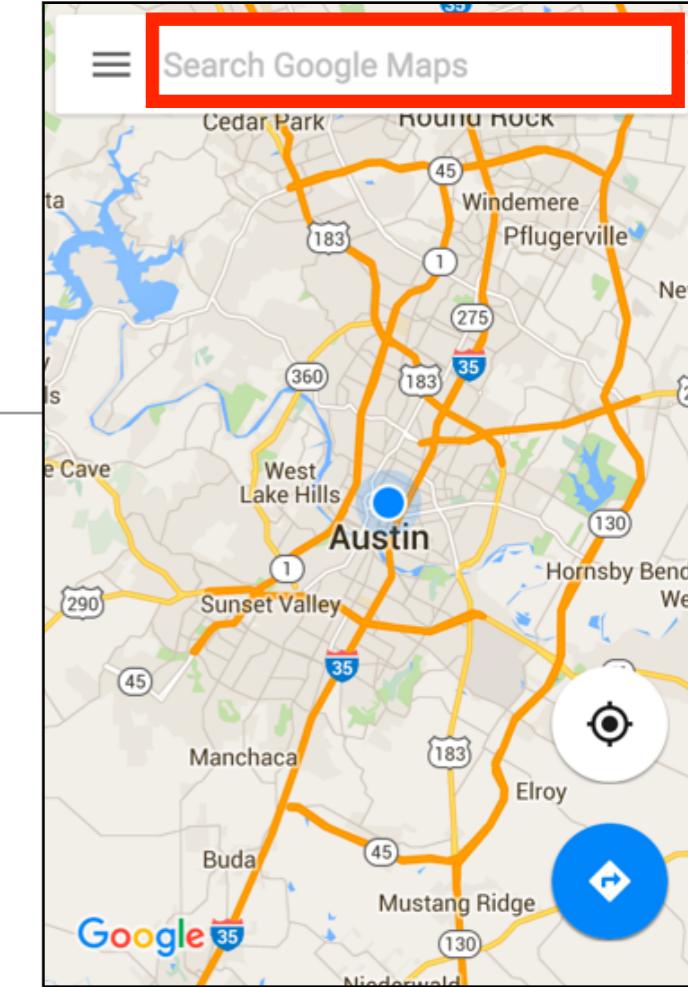
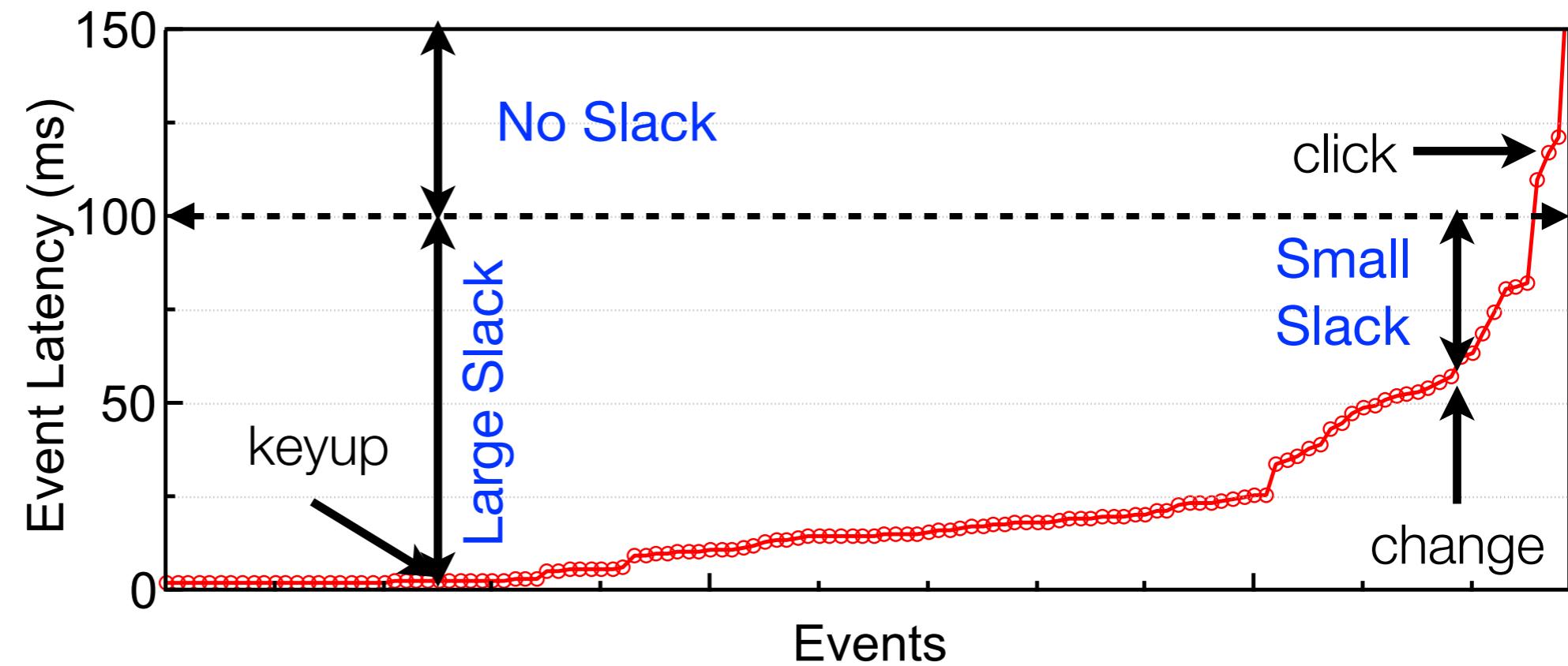
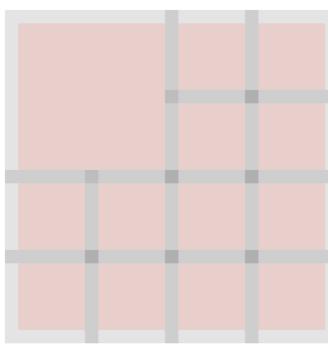
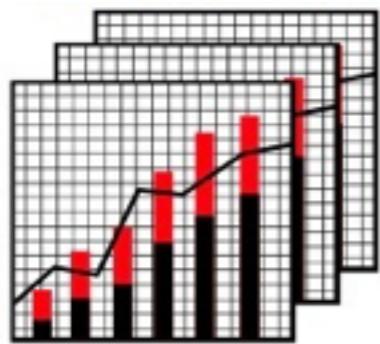
Event-Level Characterization



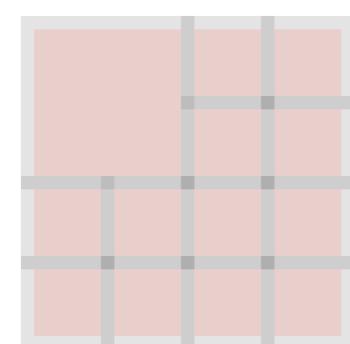
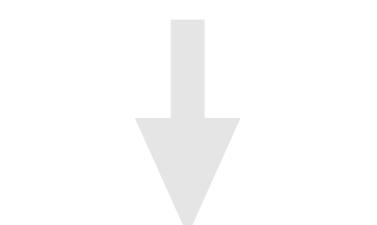
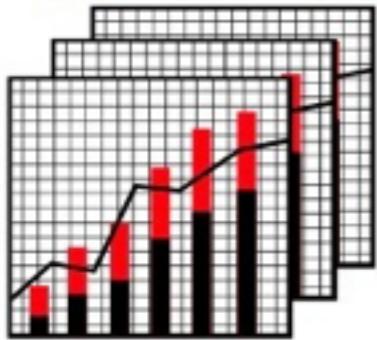
Event-Level Characterization



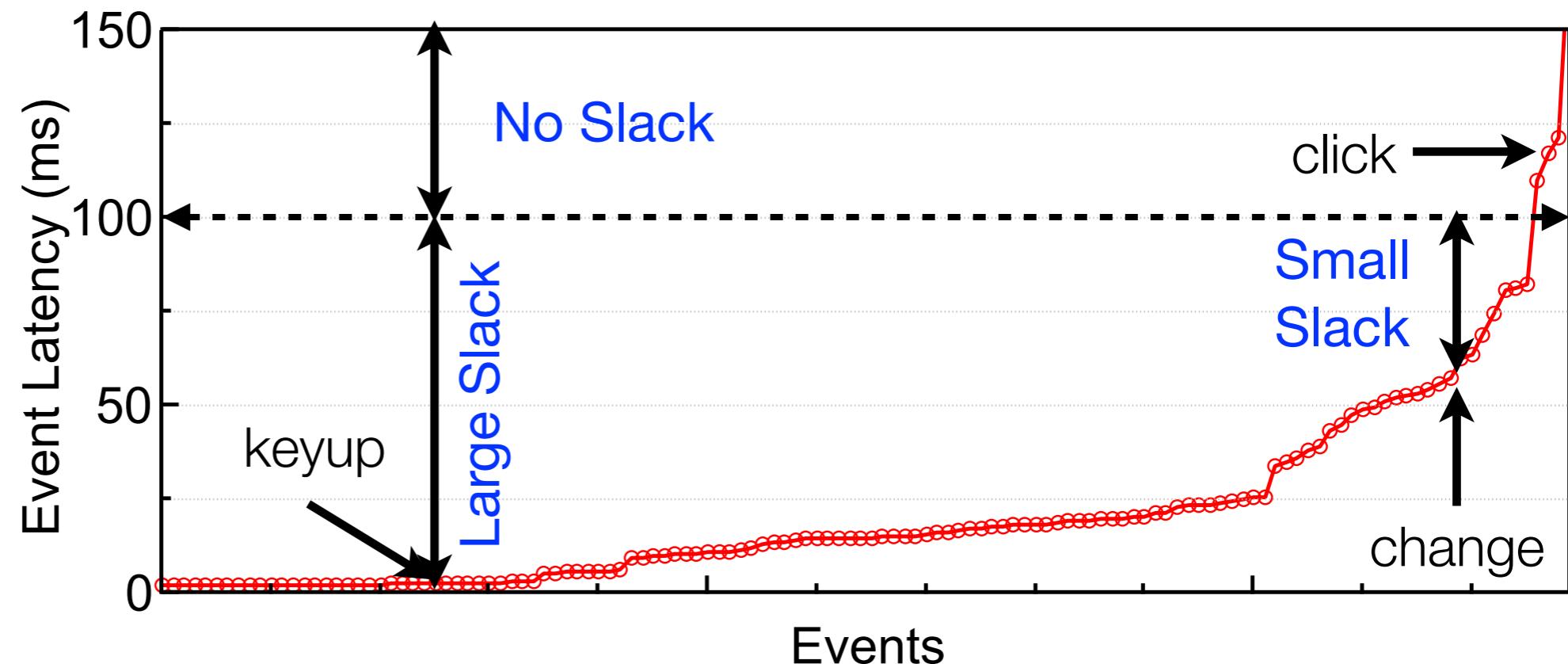
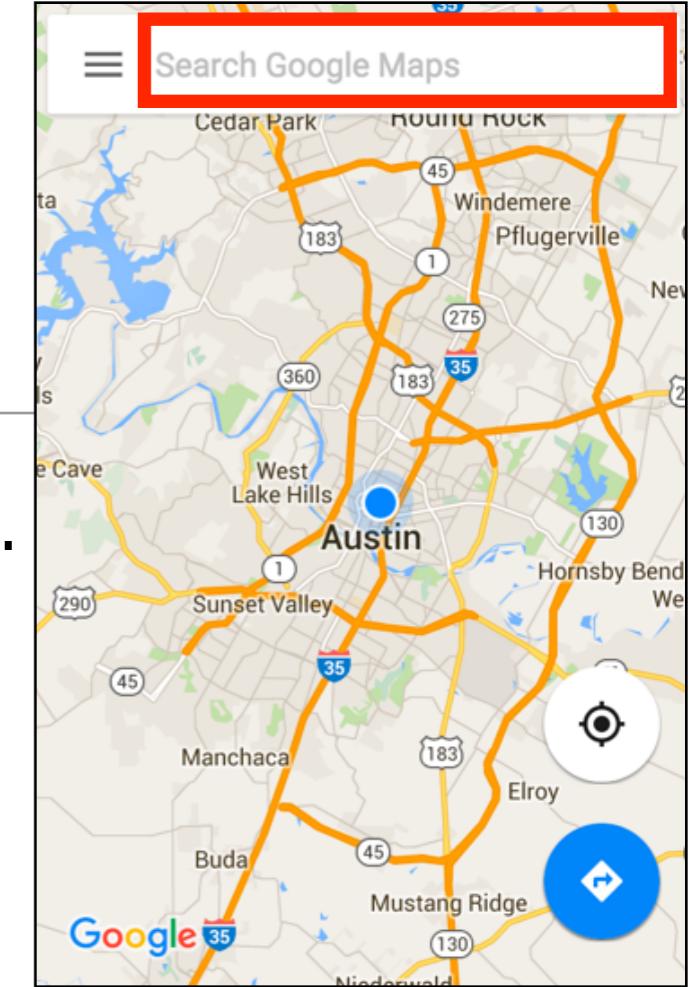
Event-Level Characterization



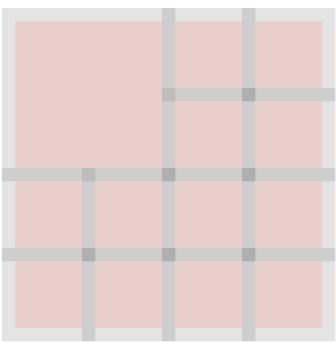
Event-Level Characterization



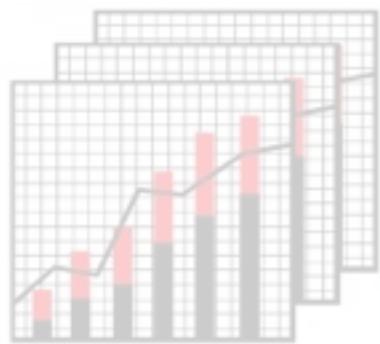
- Wide distribution of event latencies.
Events exhibit different slacks.
- ▷ How to exploit event slacks?



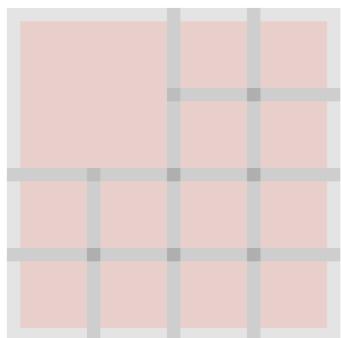
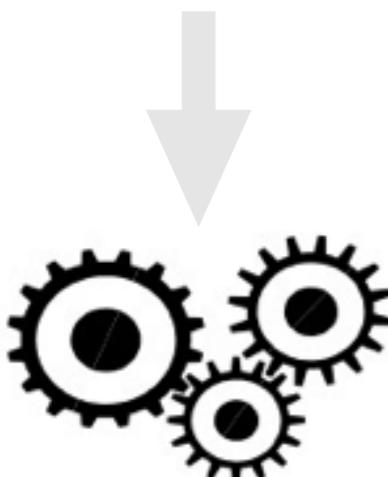
Event-based Scheduler (EBS)



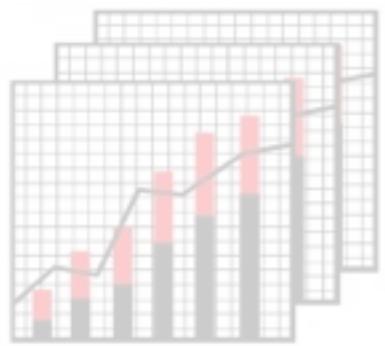
Event-based Scheduler (EBS)



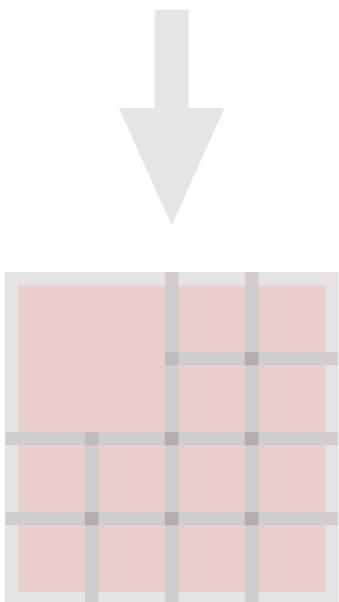
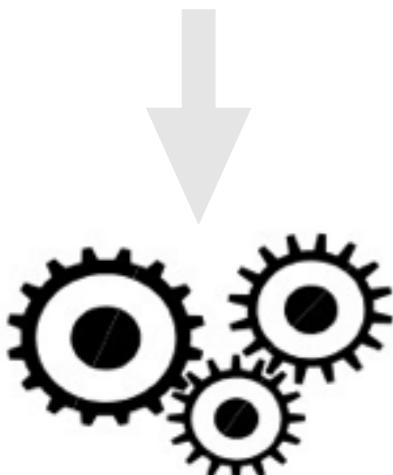
- ▶ **Goal:** For each event, find the most energy-efficient ACMP configuration that meets the latency target



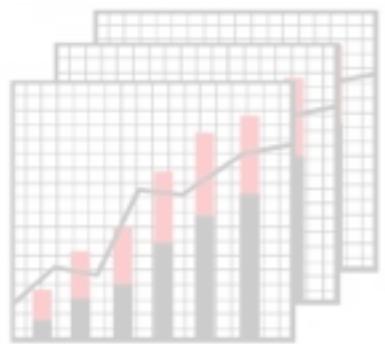
Event-based Scheduler (EBS)



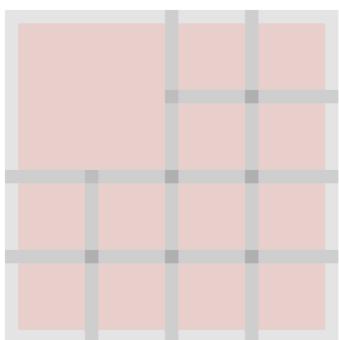
Thread Scheduling



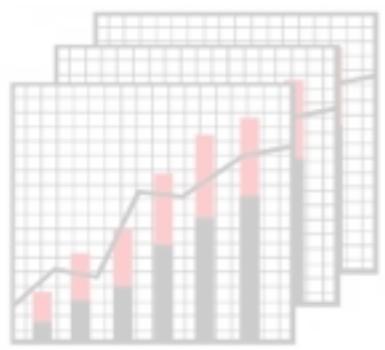
Event-based Scheduler (EBS)



Thread Scheduling



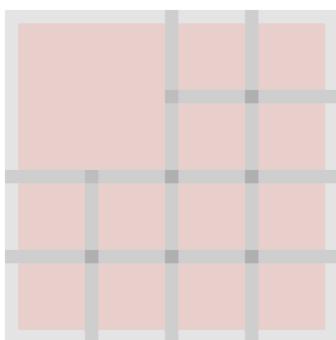
Event-based Scheduler (EBS)



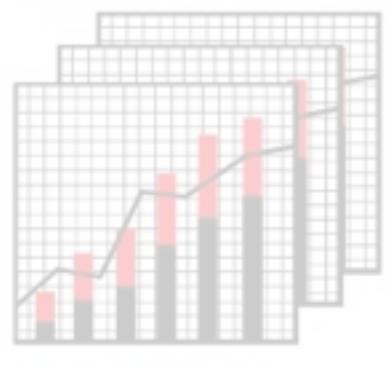
Thread Scheduling



Thread-based Scheduler



Event-based Scheduler (EBS)

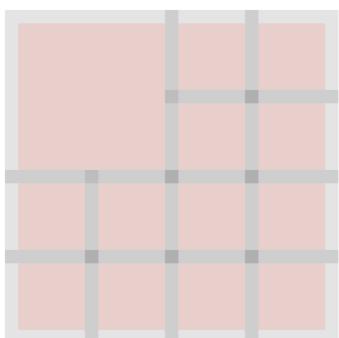


Thread Scheduling

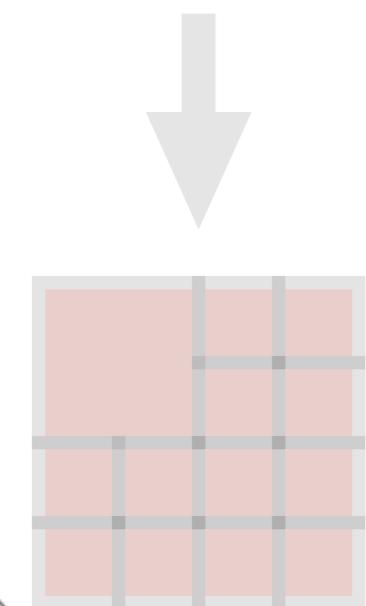
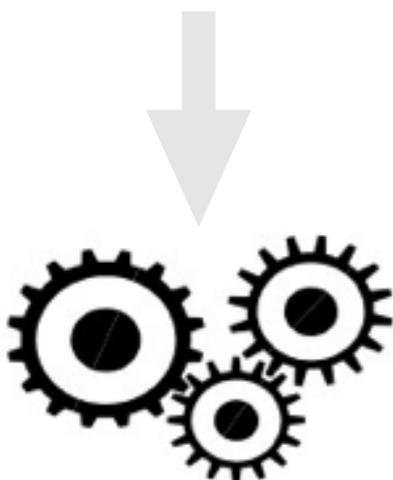
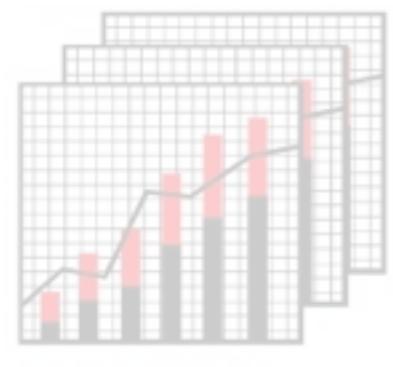


Thread-based Scheduler

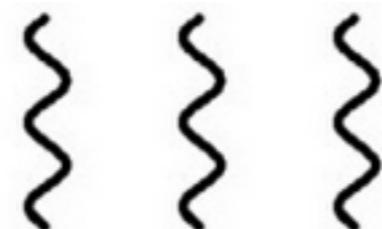
Throughput Fairness



Event-based Scheduler (EBS)



Thread Scheduling

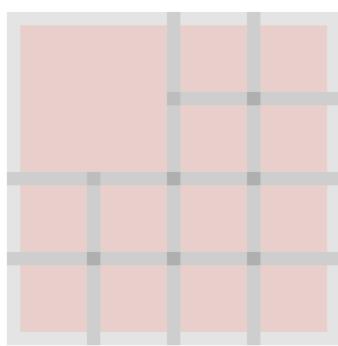
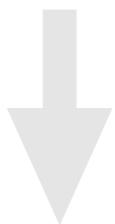
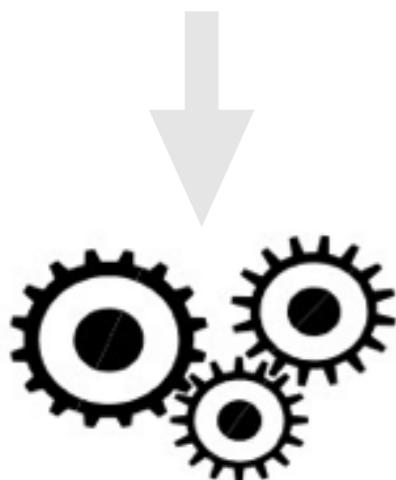
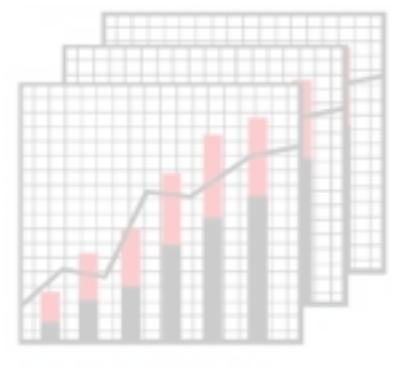


Thread-based Scheduler

Throughput Fairness

Events-based Scheduling

Event-based Scheduler (EBS)



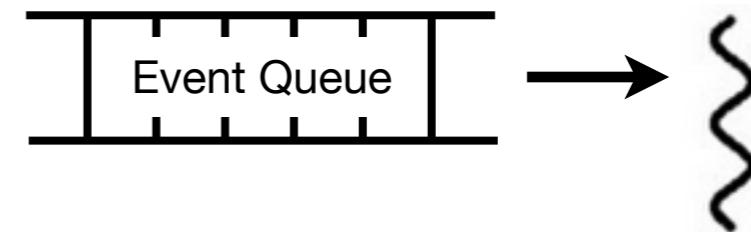
Thread Scheduling



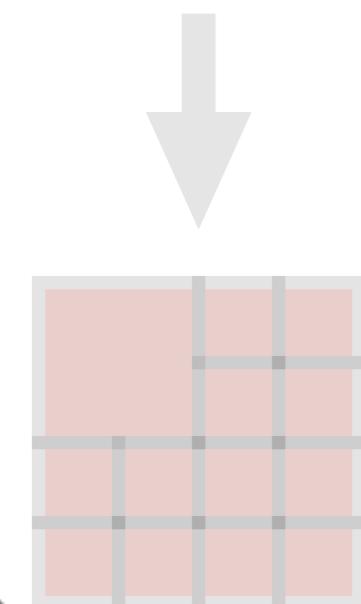
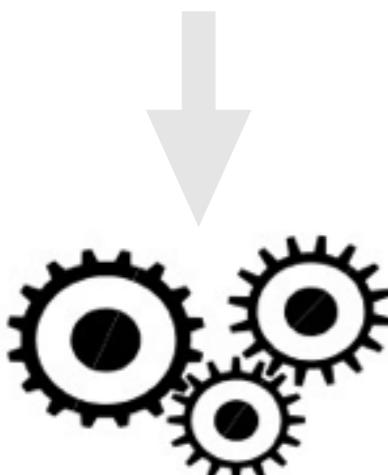
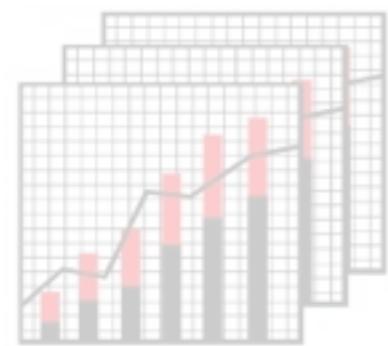
Thread-based Scheduler

Throughput Fairness

Events-based Scheduling



Event-based Scheduler (EBS)



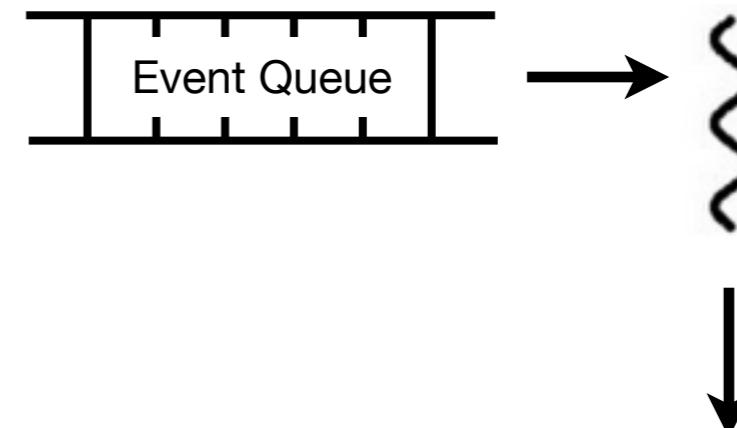
Thread Scheduling



Thread-based Scheduler

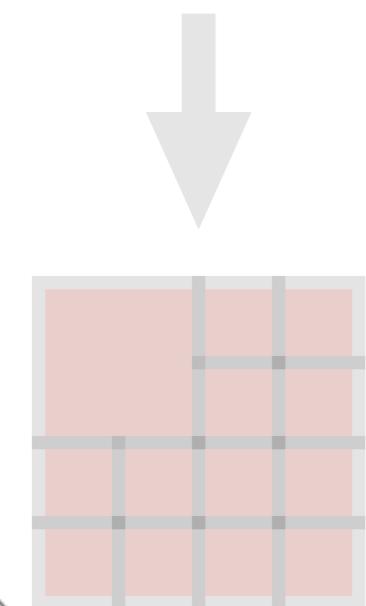
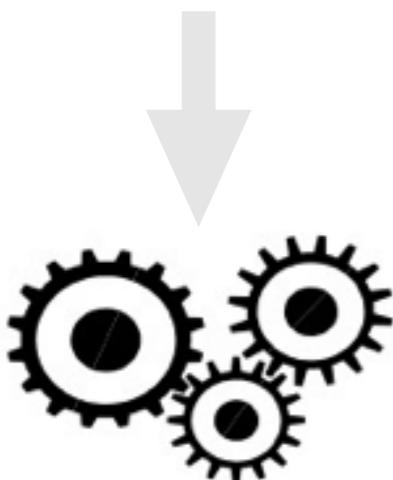
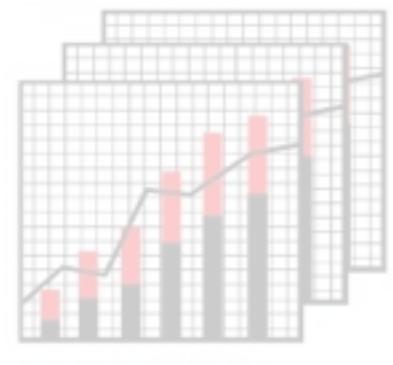
Throughput Fairness

Events-based Scheduling



Event-based Scheduler

Event-based Scheduler (EBS)



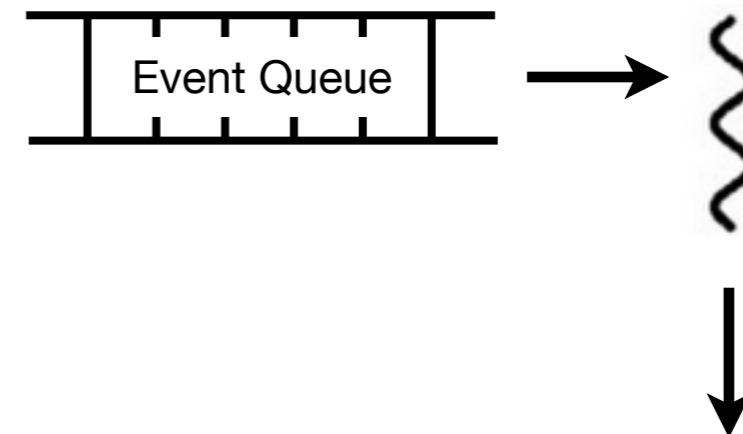
Thread Scheduling



Thread-based Scheduler

Throughput Fairness

Events-based Scheduling

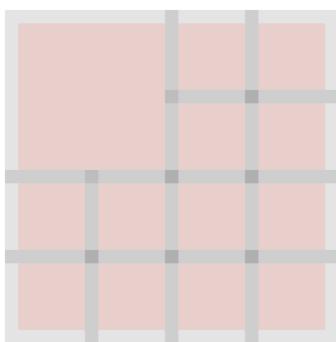
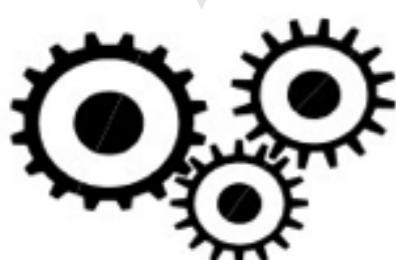
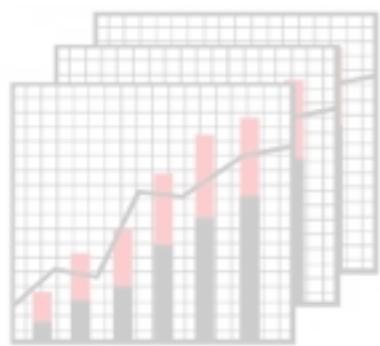


Event-based Scheduler

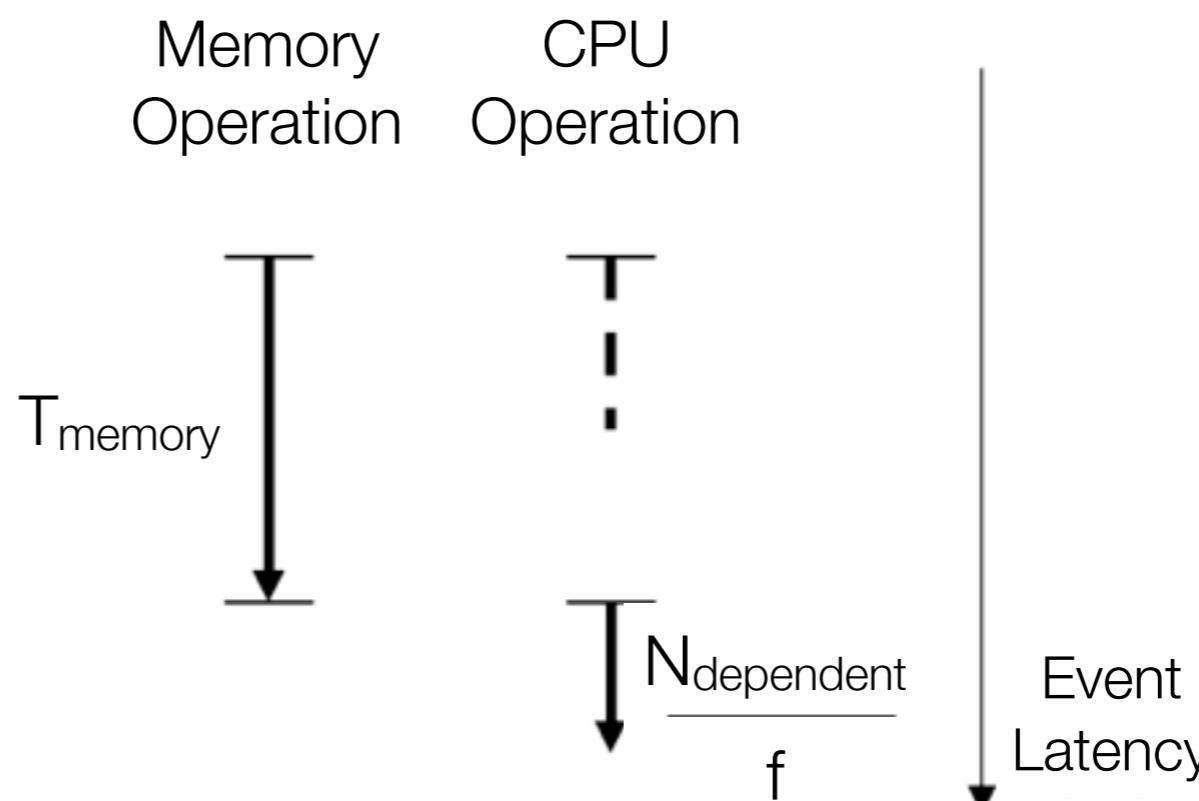
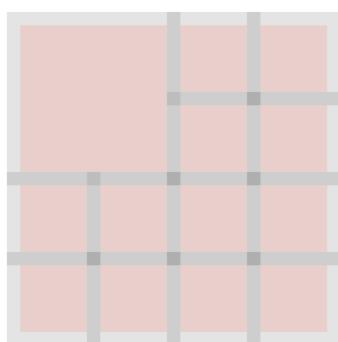
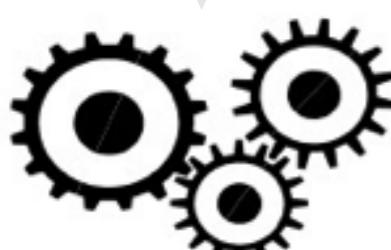
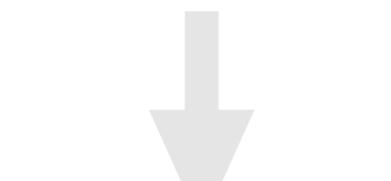
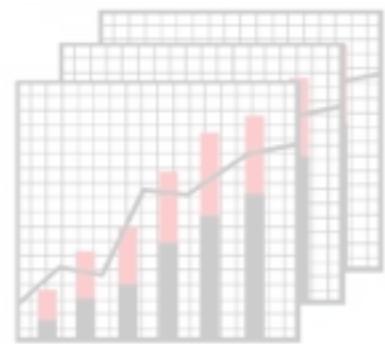
Event
Latency Event
Energy



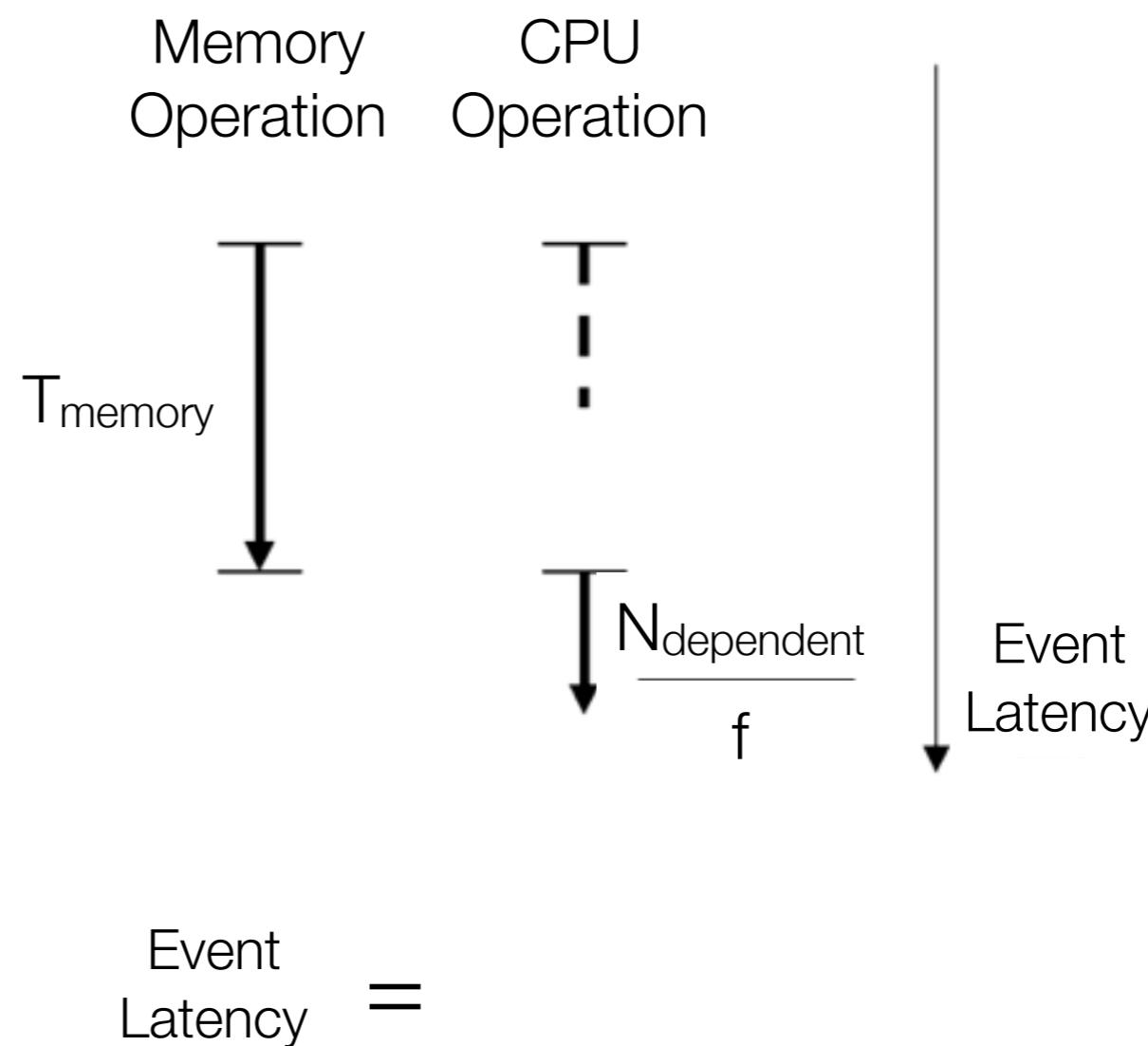
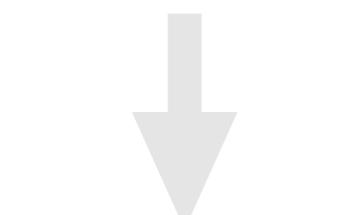
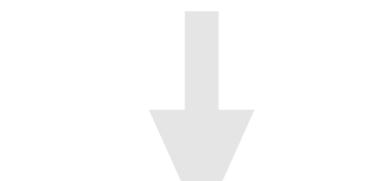
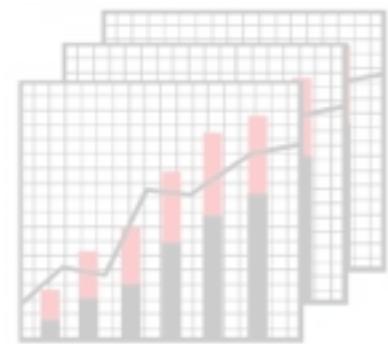
Predicting Event Latency



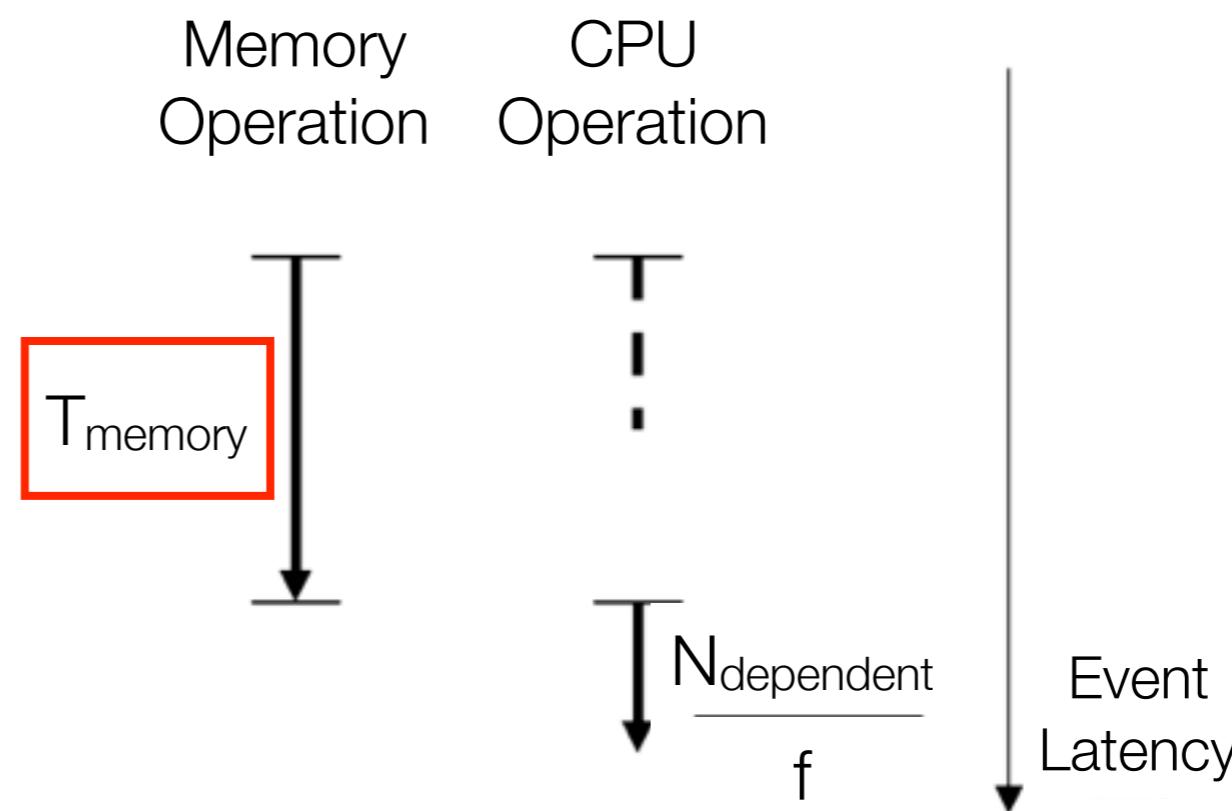
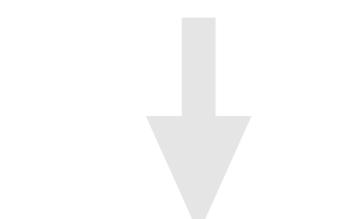
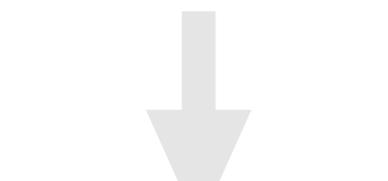
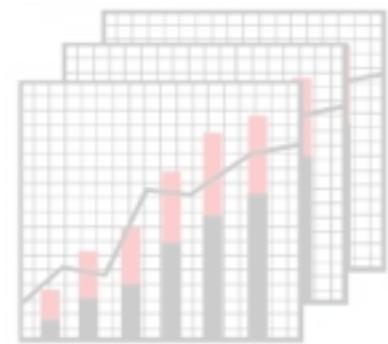
Predicting Event Latency



Predicting Event Latency



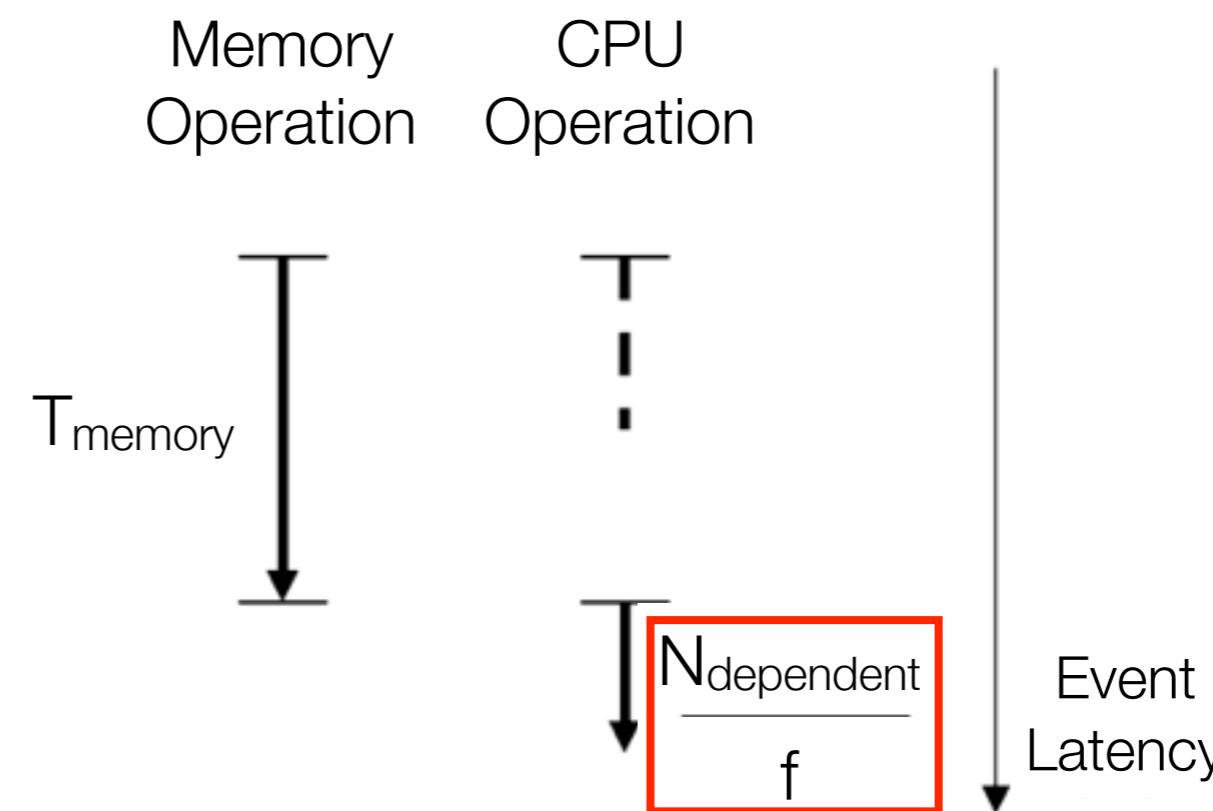
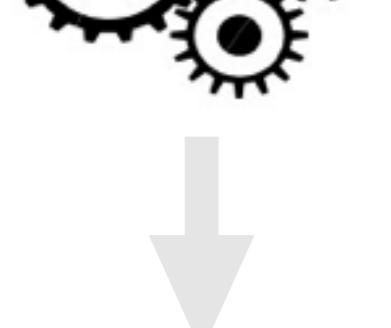
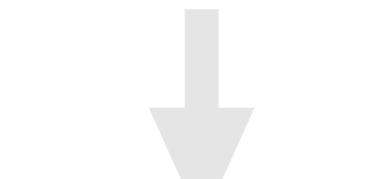
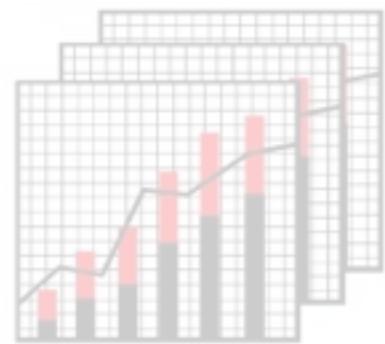
Predicting Event Latency



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



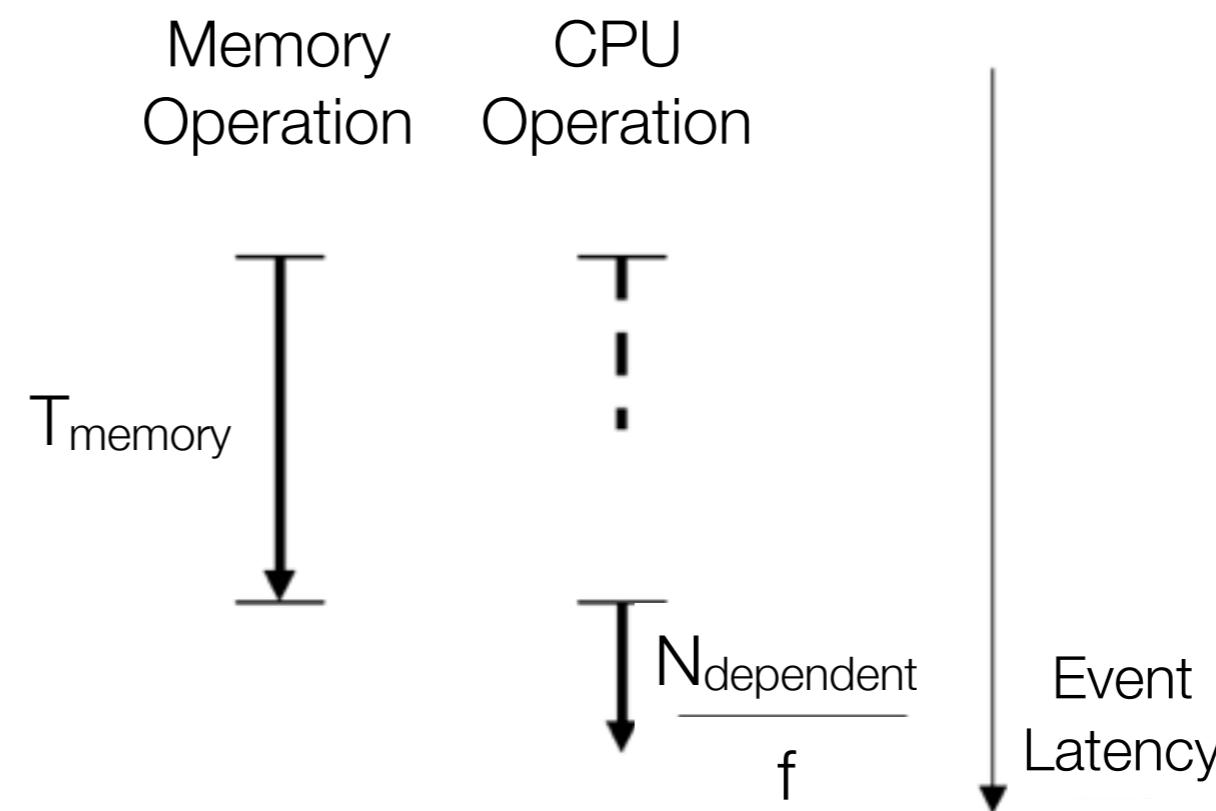
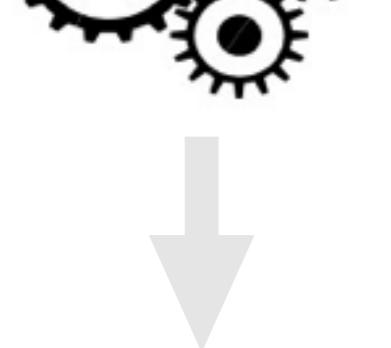
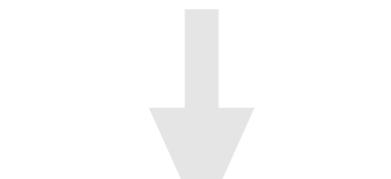
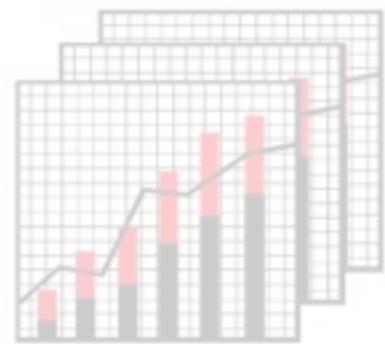
Predicting Event Latency



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



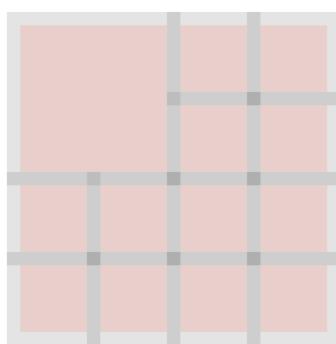
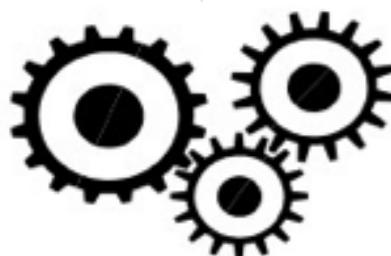
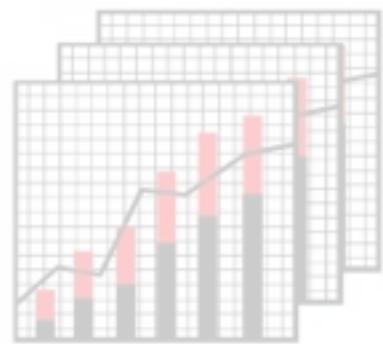
Predicting Event Latency



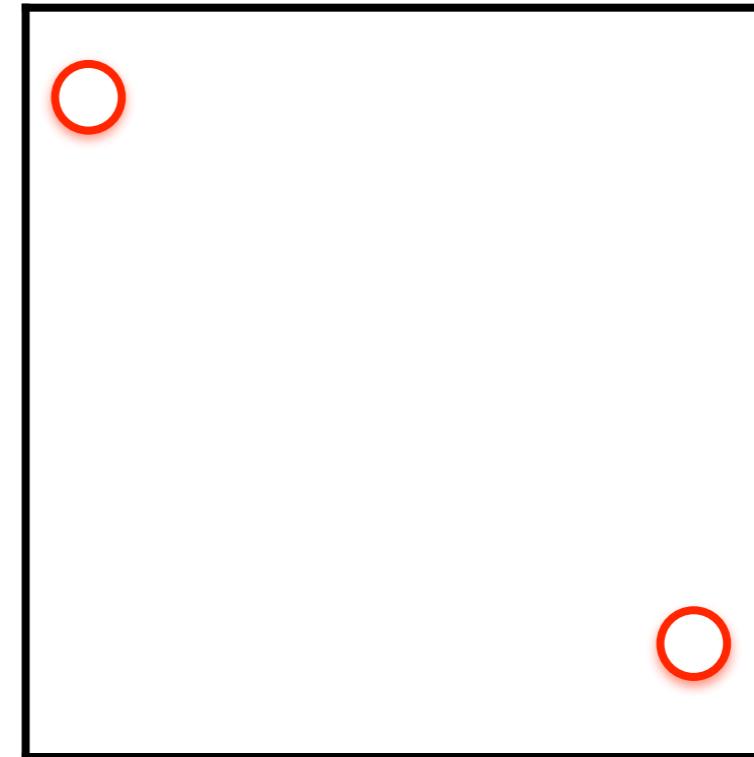
$$\text{Event Latency} = \boxed{T_{\text{memory}}} + \boxed{\frac{N_{\text{dependent}}}{f}}$$



Predicting Event Latency



Event Latency

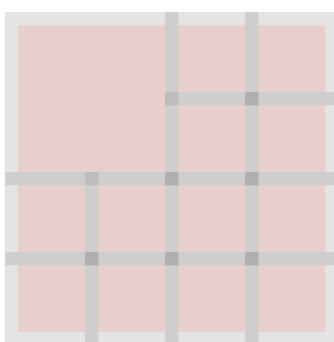
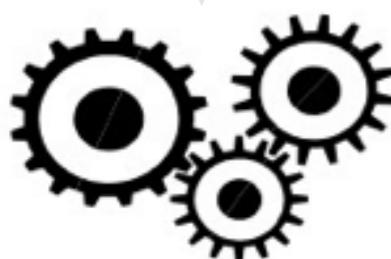
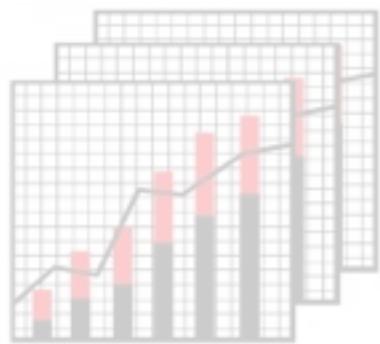


Frequency

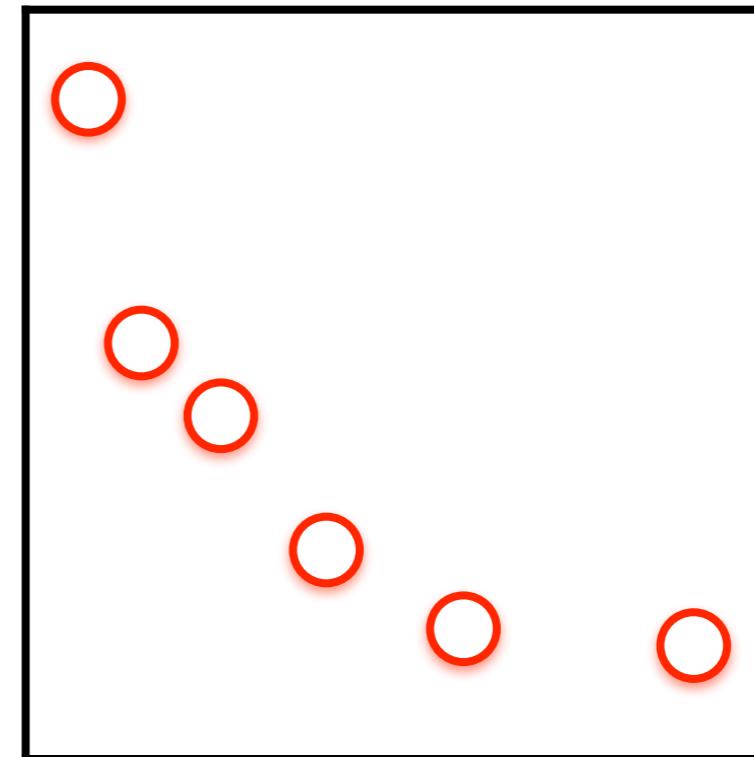
$$\text{Event Latency} = T_{\text{memory}} + N_{\text{dependent}} / f$$



Predicting Event Latency



Event Latency

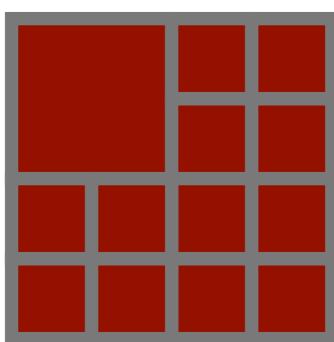
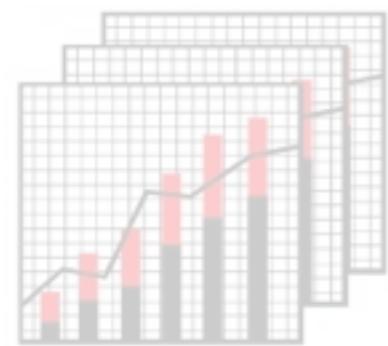


Frequency

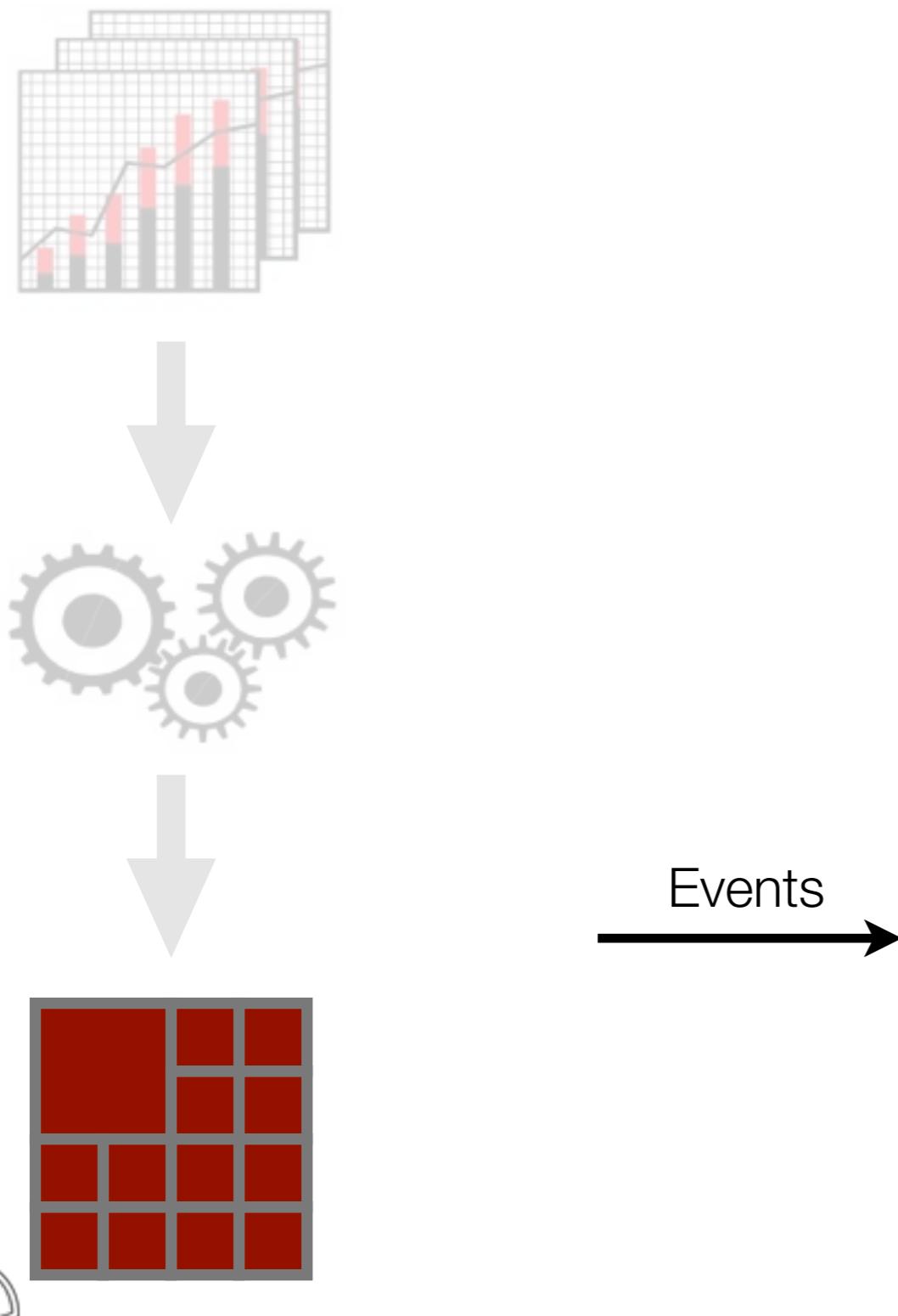
$$\text{Event Latency} = T_{\text{memory}} + N_{\text{dependent}} / f$$



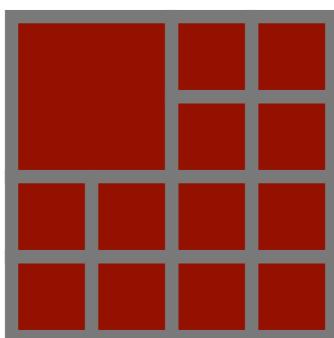
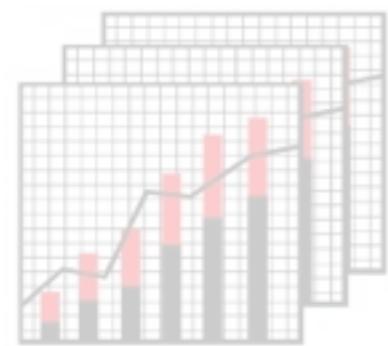
Event-based Scheduler



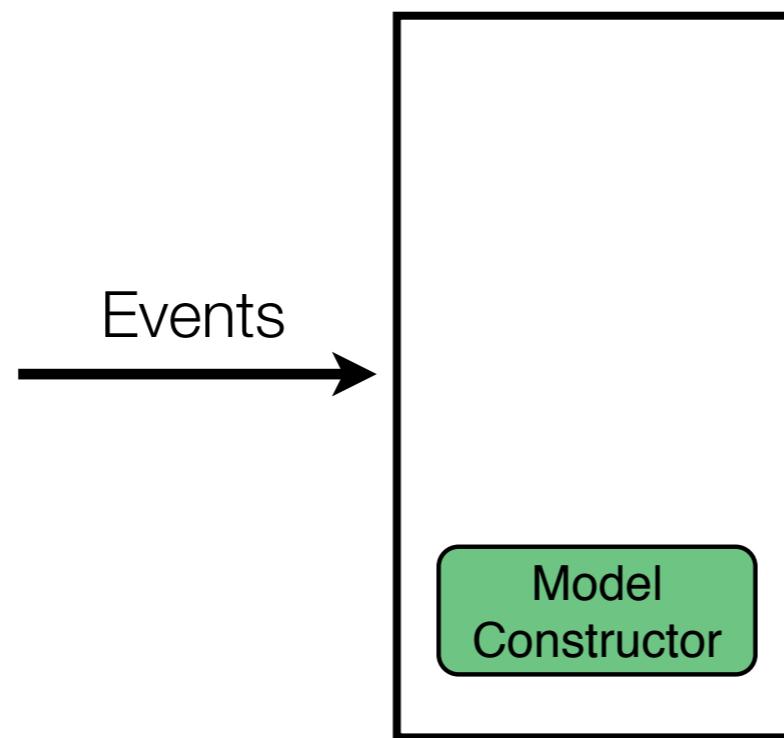
Event-based Scheduler



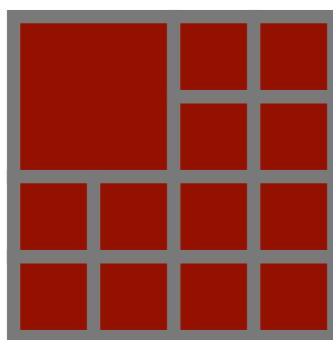
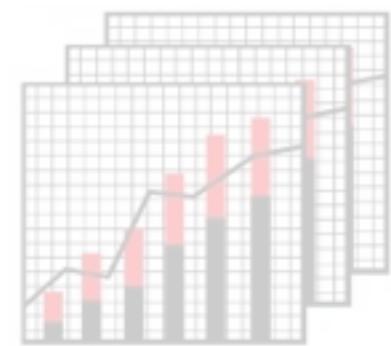
Event-based Scheduler



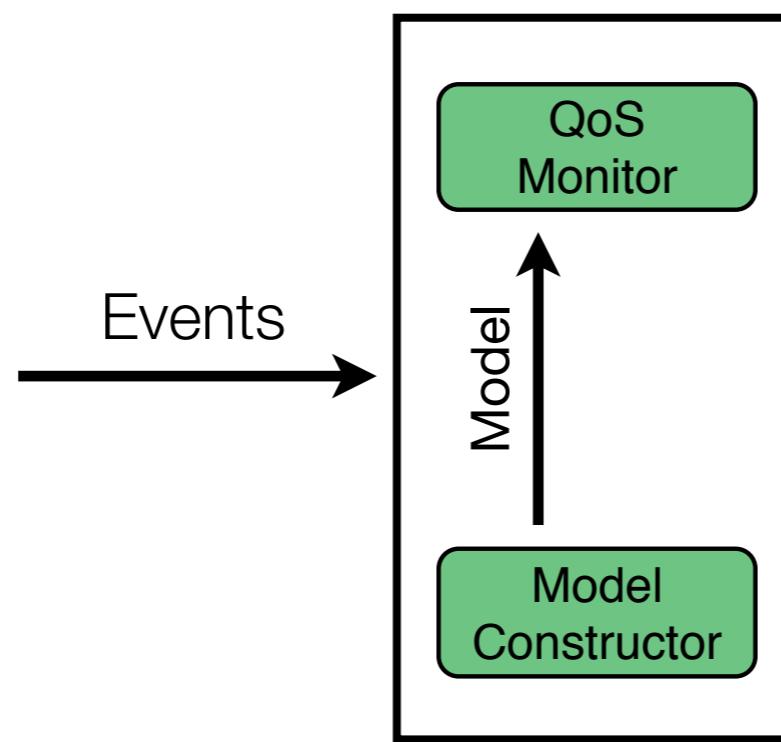
Event-Based Scheduler



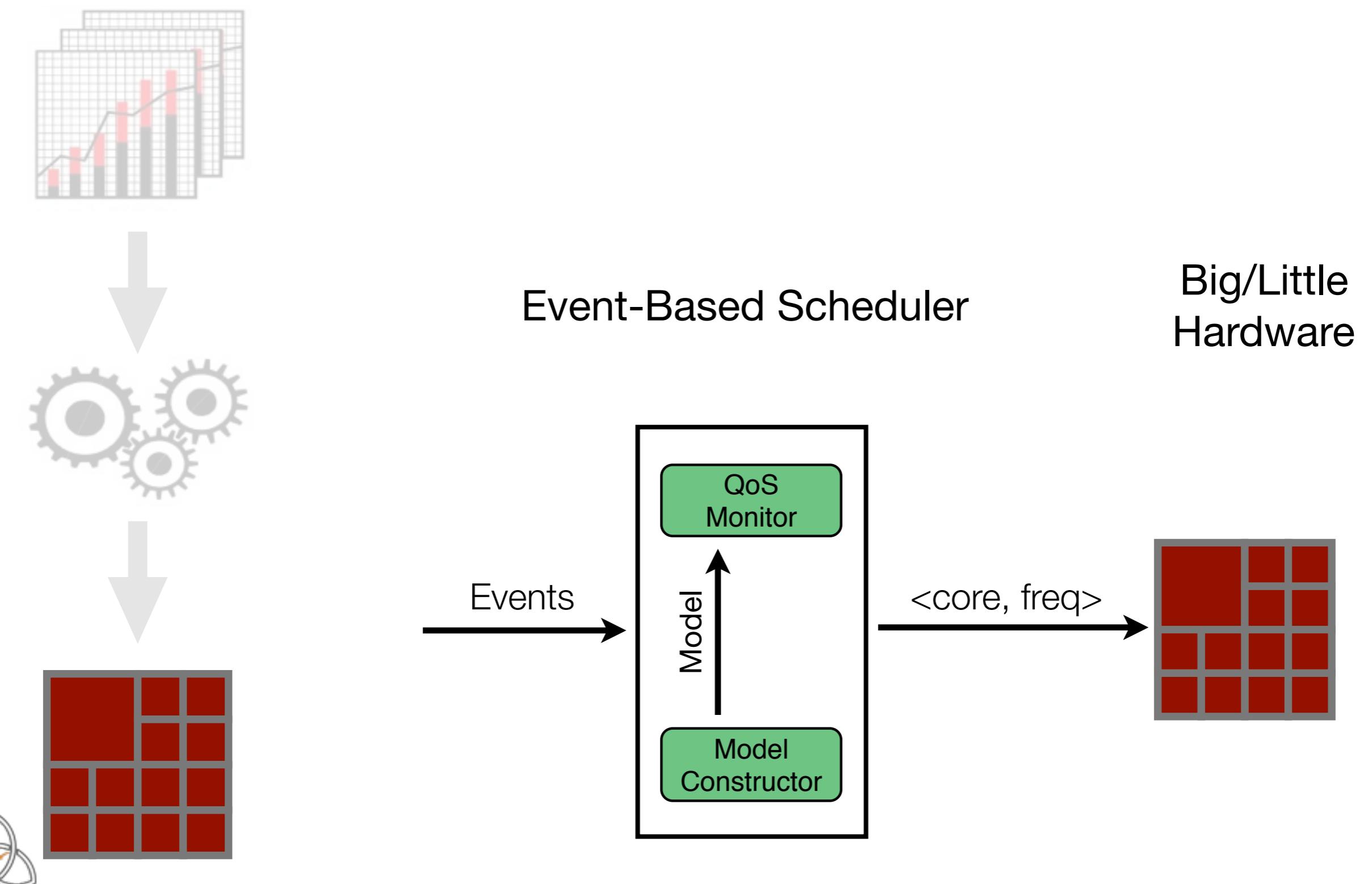
Event-based Scheduler



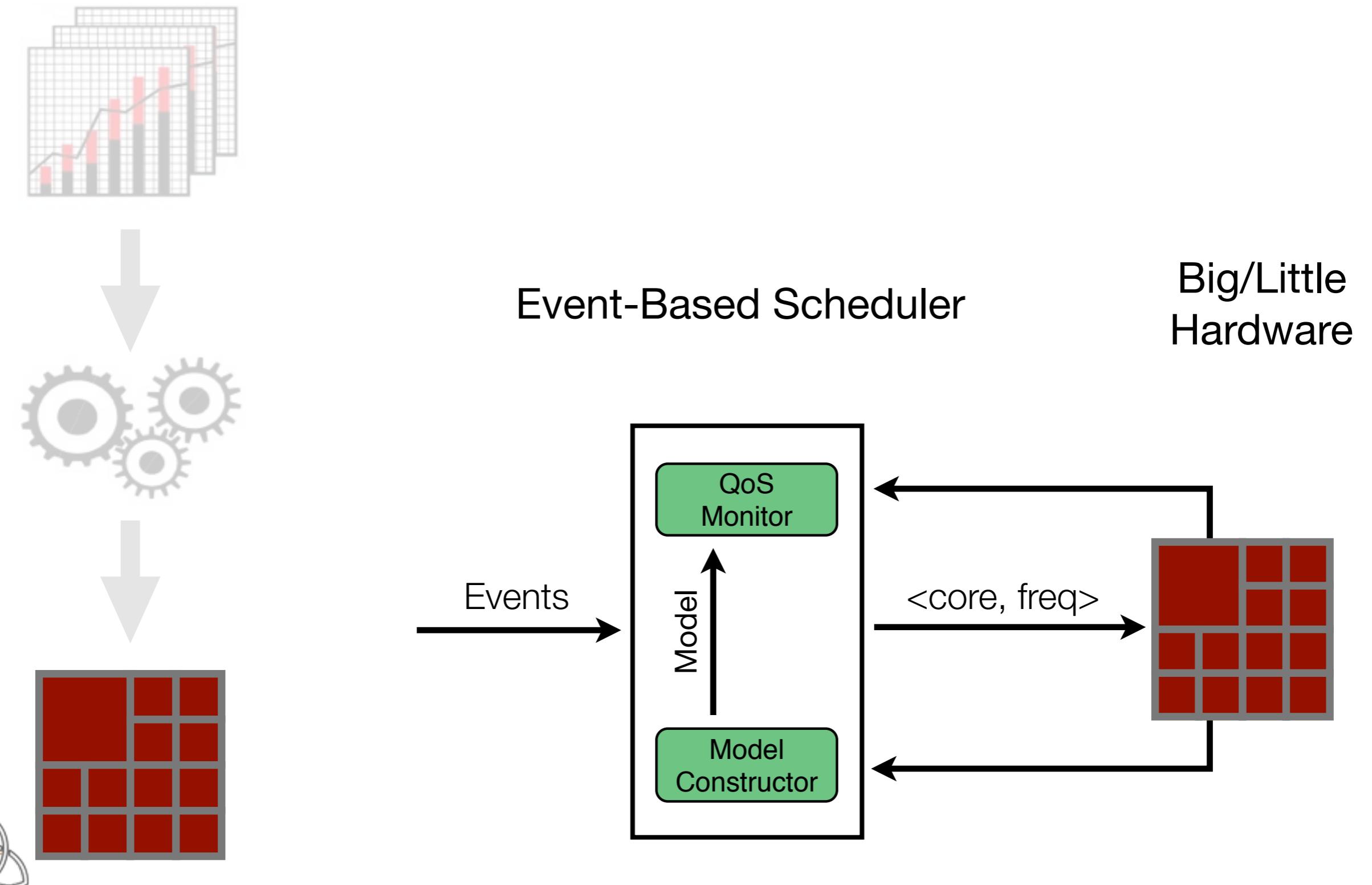
Event-Based Scheduler



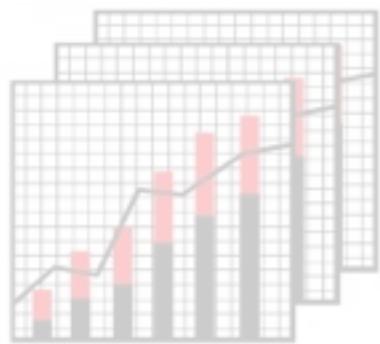
Event-based Scheduler



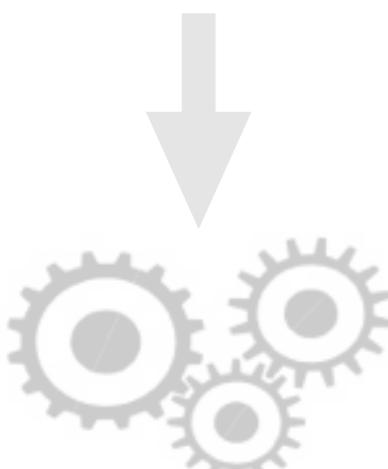
Event-based Scheduler



Event-based Scheduler

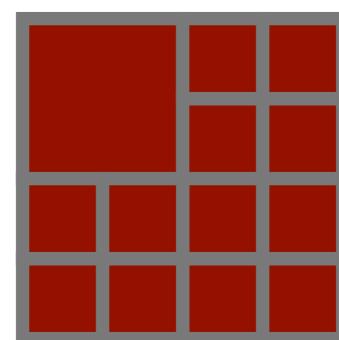
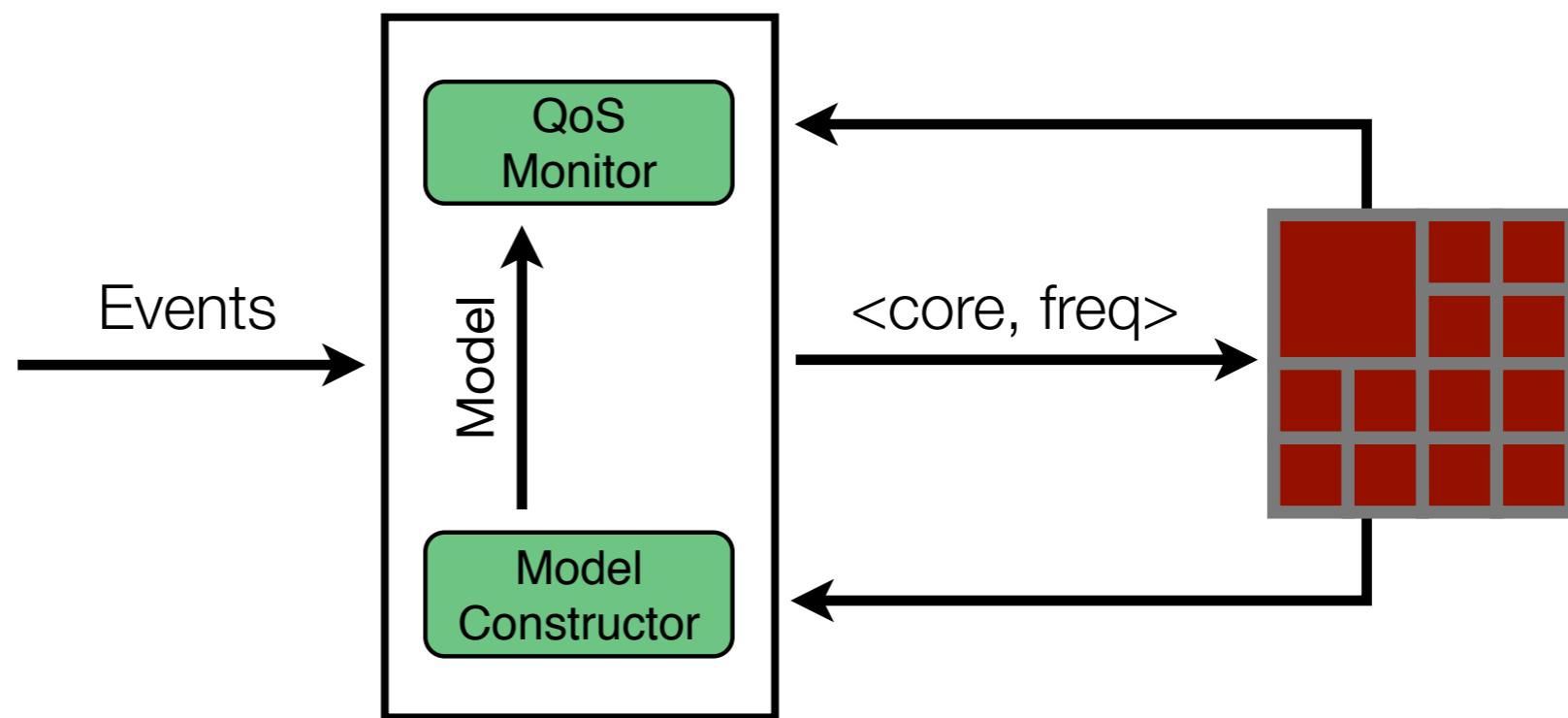


- ▶ Fine-tune the model when over or under-predict

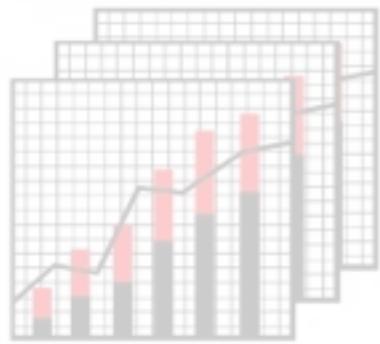


Event-Based Scheduler

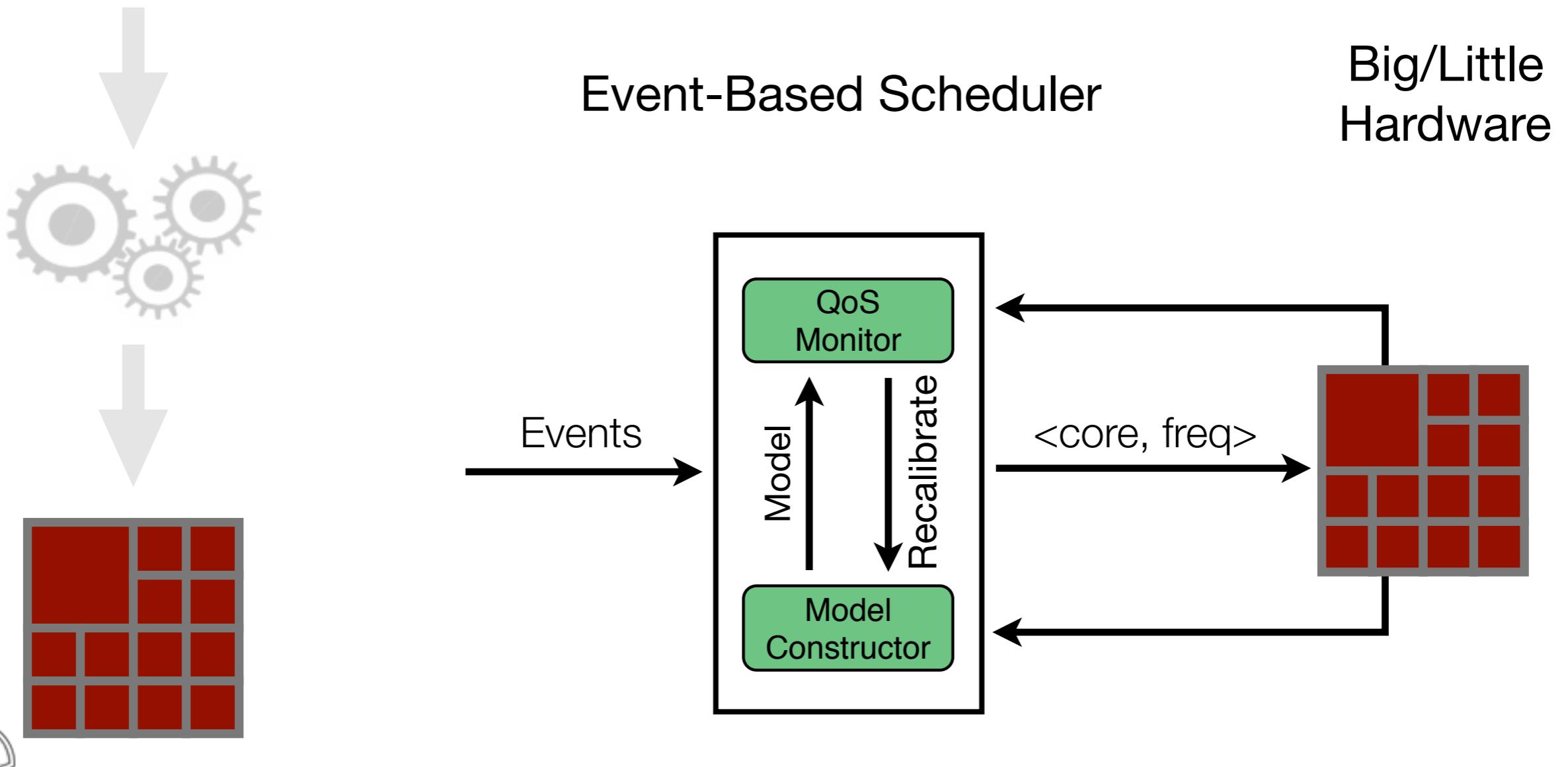
Big/Little Hardware



Event-based Scheduler



- ▶ Fine-tune the model when over or under-predict
- ▶ Recalibrate if it mispredicts too often



Evaluation

► Baseline Mechanisms

- ▷ Highest performance (**Perf**) — Standard to guarantee responsiveness
- ▷ Minimal energy (**Energy**) — Minimize energy consumption
- ▷ Interactive governor (**Interactive**) — Android default



Evaluation

► Baseline Mechanisms

- ▷ Highest performance (**Perf**) — Standard to guarantee responsiveness
- ▷ Minimal energy (**Energy**) — Minimize energy consumption
- ▷ Interactive governor (**Interactive**) — Android default

► Metrics

- ▷ Energy Savings
- ▷ QoS Violations



Evaluation

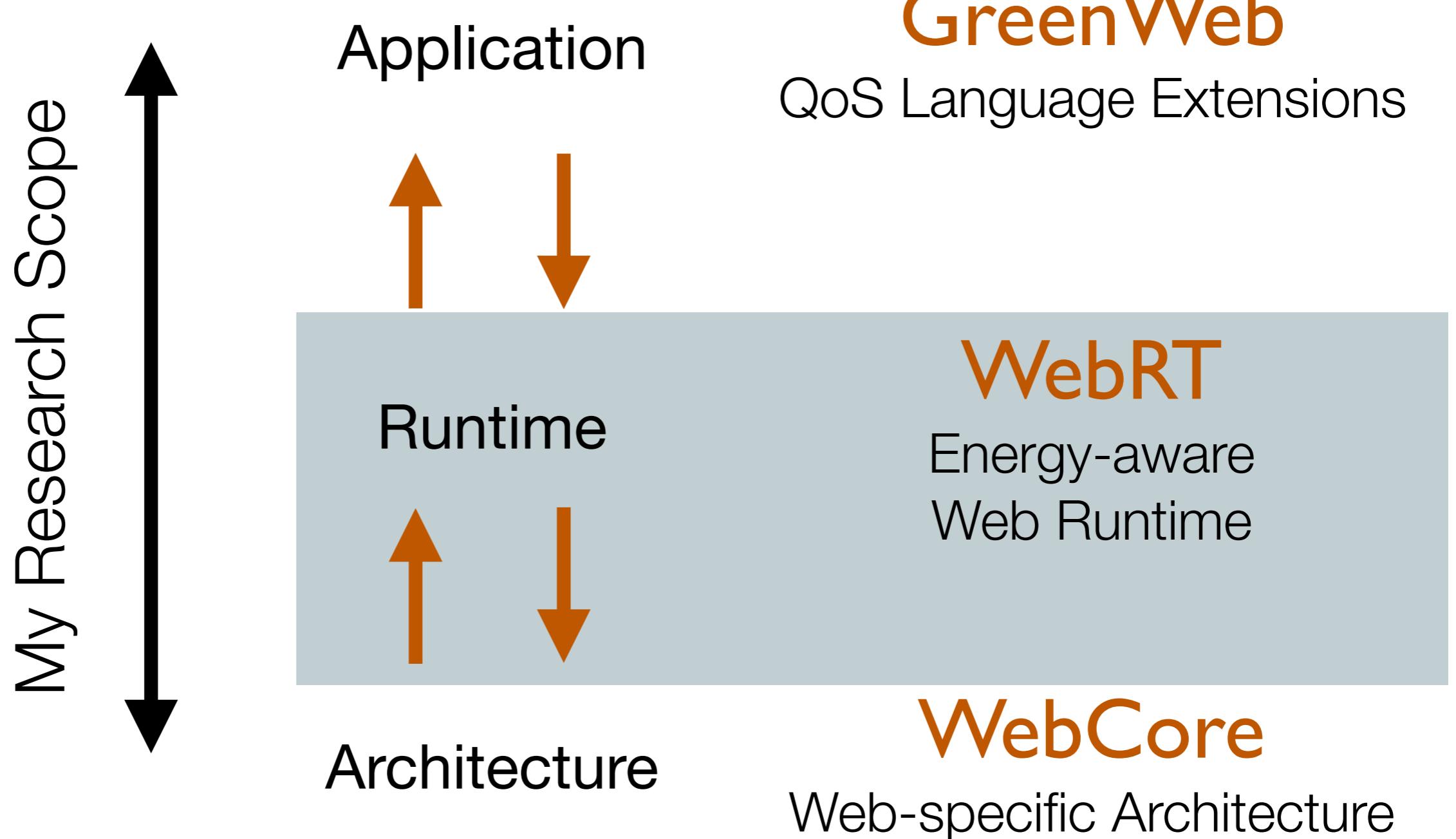
- ▶ Baseline Mechanisms
 - ▷ Highest performance (**Perf**) – Standard to guarantee responsiveness
 - ▷ Minimal energy (**Energy**) – Minimize energy consumption

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

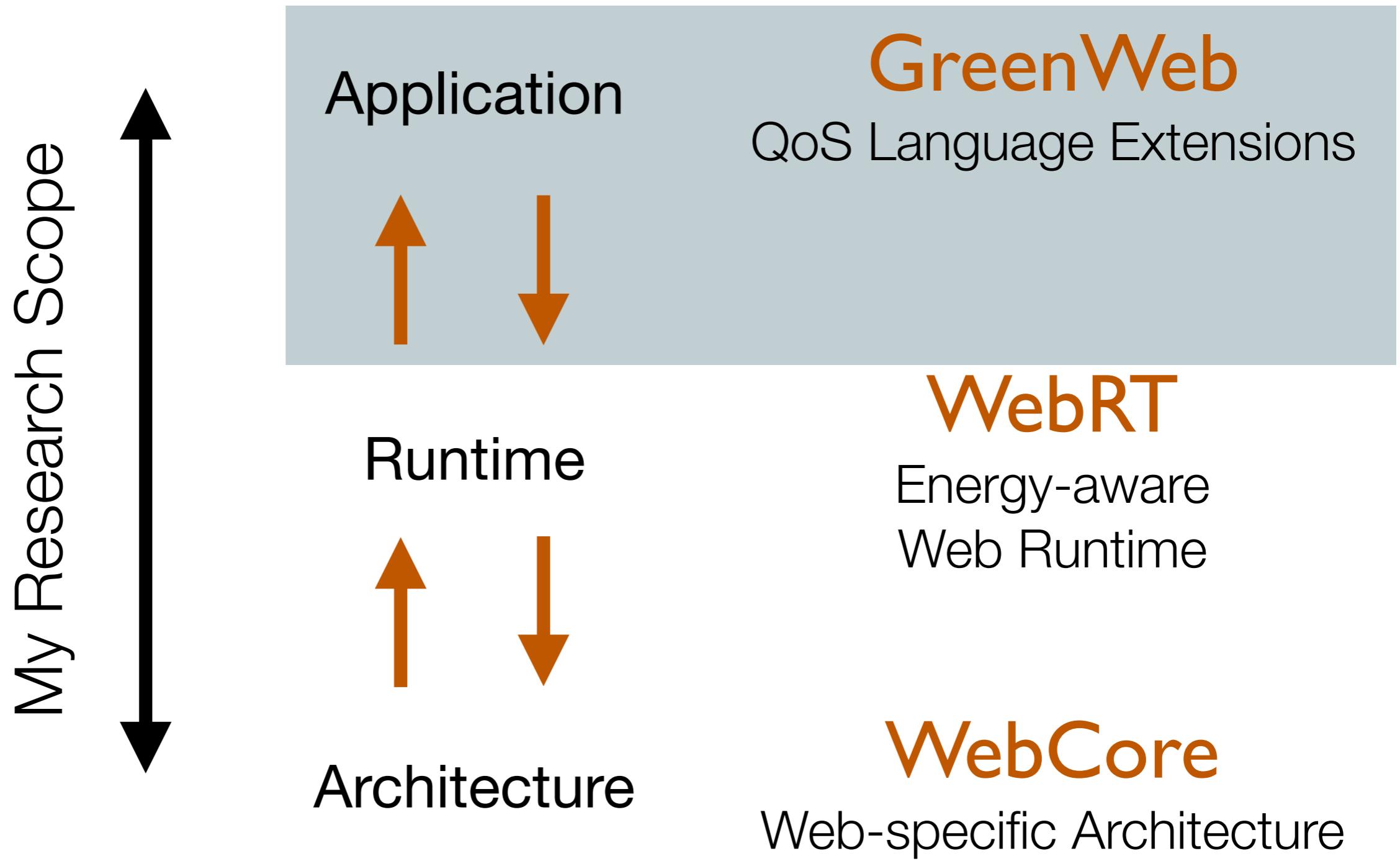
- ▷ Energy Savings
- ▷ QoS Violations



My Approach



My Approach



GreenWeb: QoS Web Language Extensions



GreenWeb: QoS Web Language Extensions

**Understanding
Mobile QoS**



GreenWeb: QoS Web Language Extensions

**Understanding
Mobile QoS**



**Abstracting
Mobile QoS**



GreenWeb: QoS Web Language Extensions

**Understanding
Mobile QoS**



**Abstracting
Mobile QoS**



**Expressing
Abstractions**



GreenWeb: QoS Web Language Extensions

**Understanding
Mobile QoS**



**Abstracting
Mobile**



Expressing



GreenWeb: QoS Web Language Extensions

Understanding Mobile QoS



Abstracting
Mobile



Expressing



GreenWeb: QoS Web Language Extensions

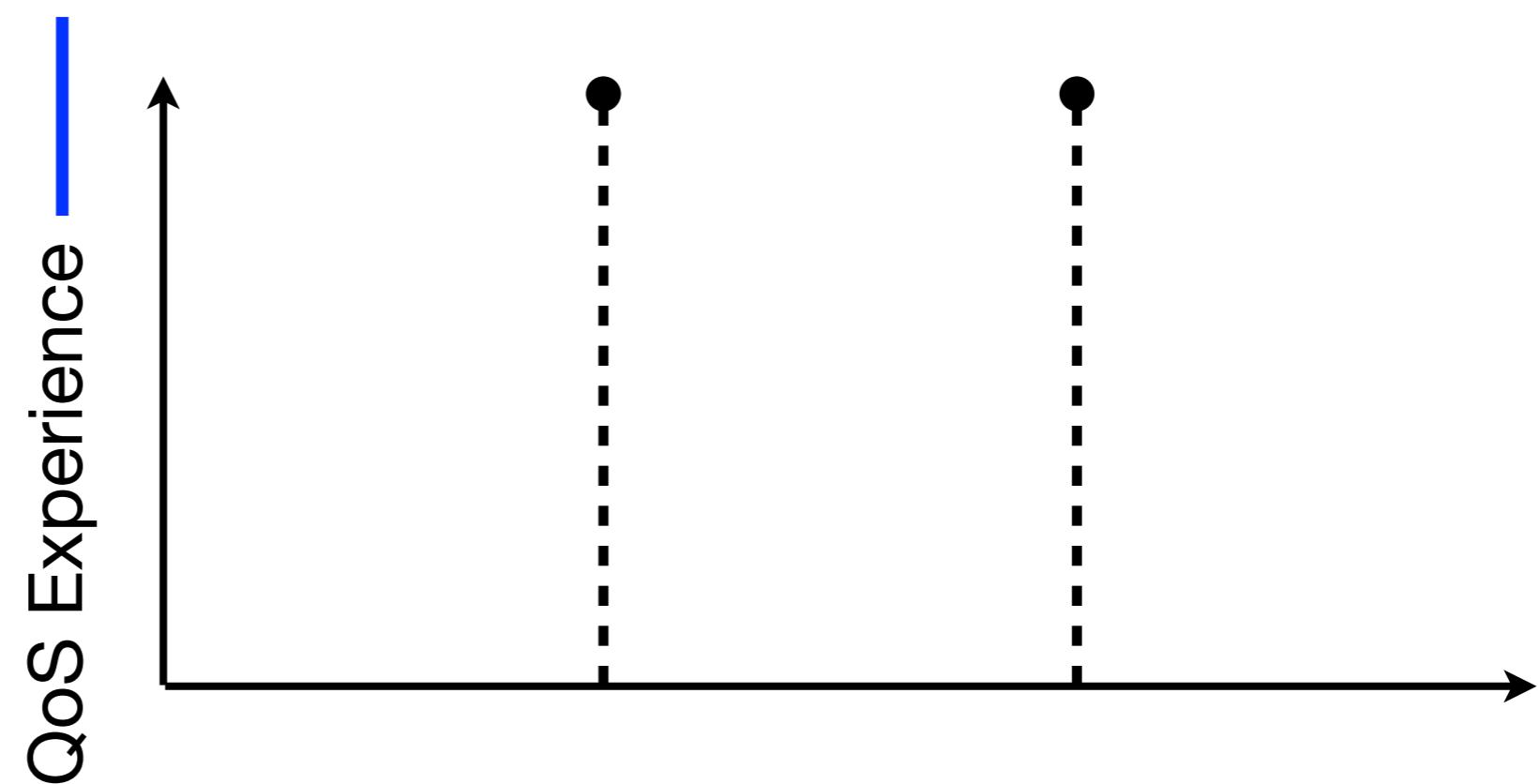
Understanding Mobile QoS



Abstracting
Mobile



Expressing



Performance Degradation



GreenWeb: QoS Web Language Extensions

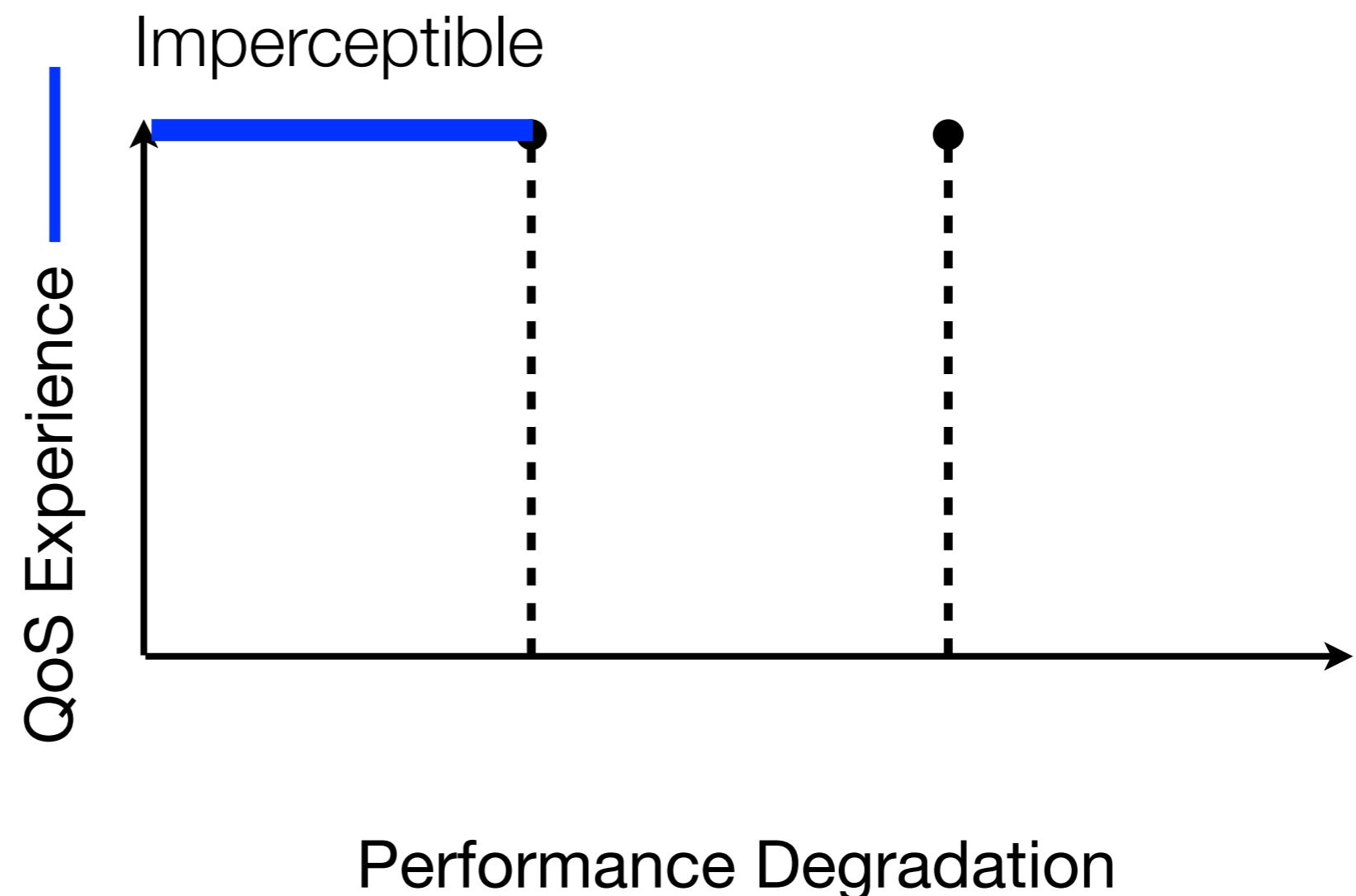
Understanding Mobile QoS



Abstracting
Mobile



Expressing



GreenWeb: QoS Web Language Extensions

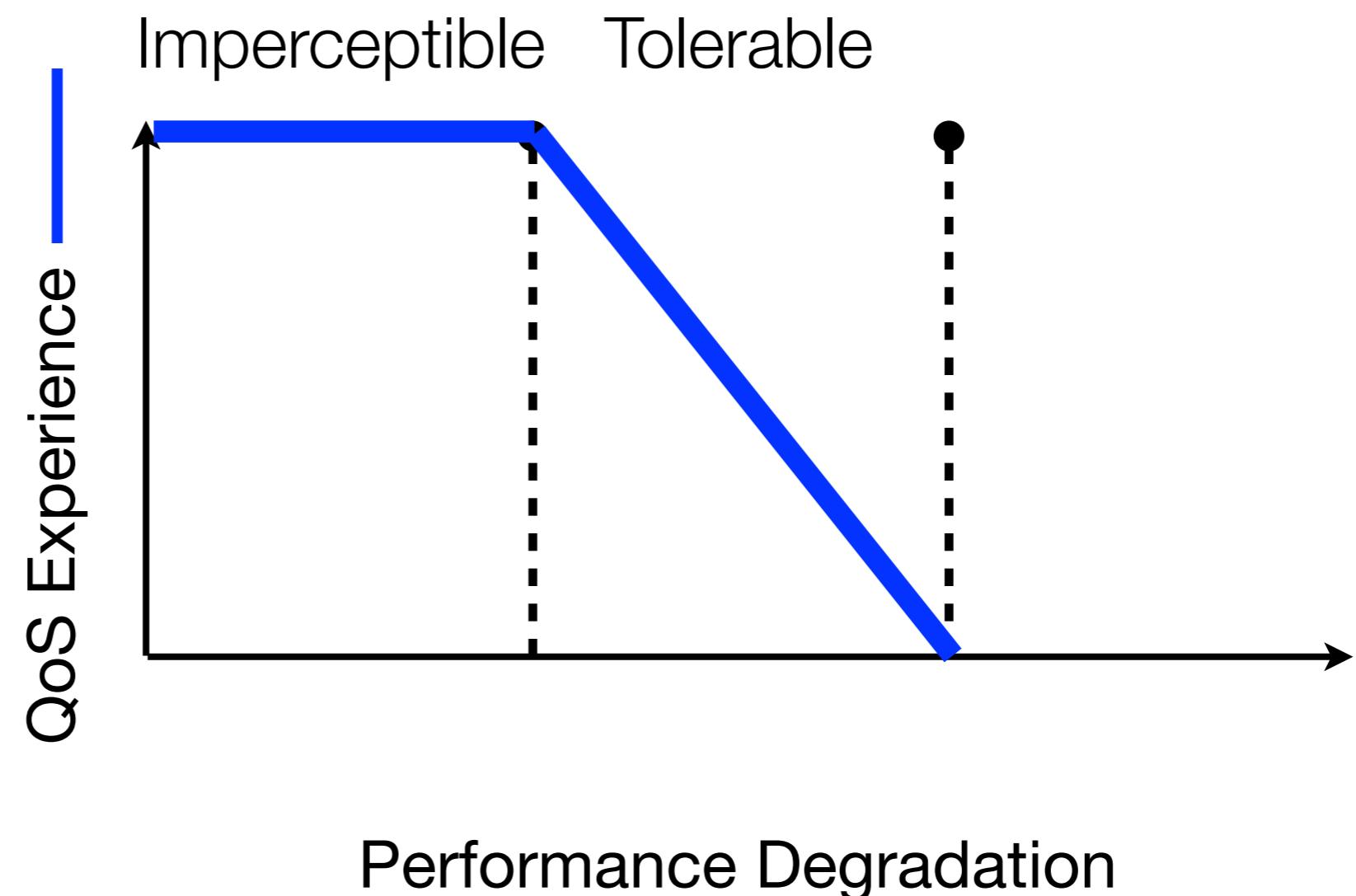
Understanding Mobile QoS



Abstracting
Mobile



Expressing



GreenWeb: QoS Web Language Extensions

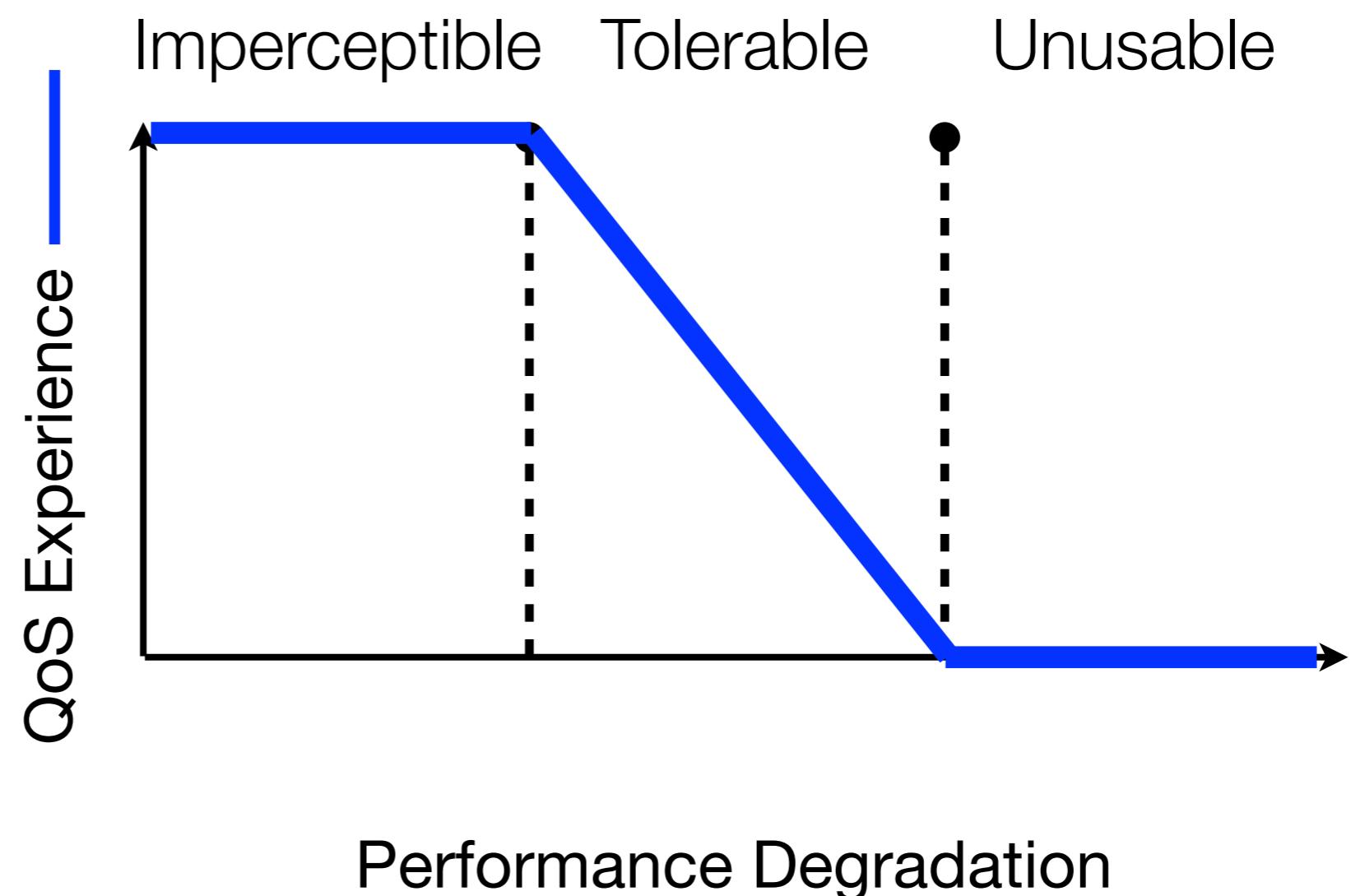
Understanding Mobile QoS



Abstracting
Mobile



Expressing

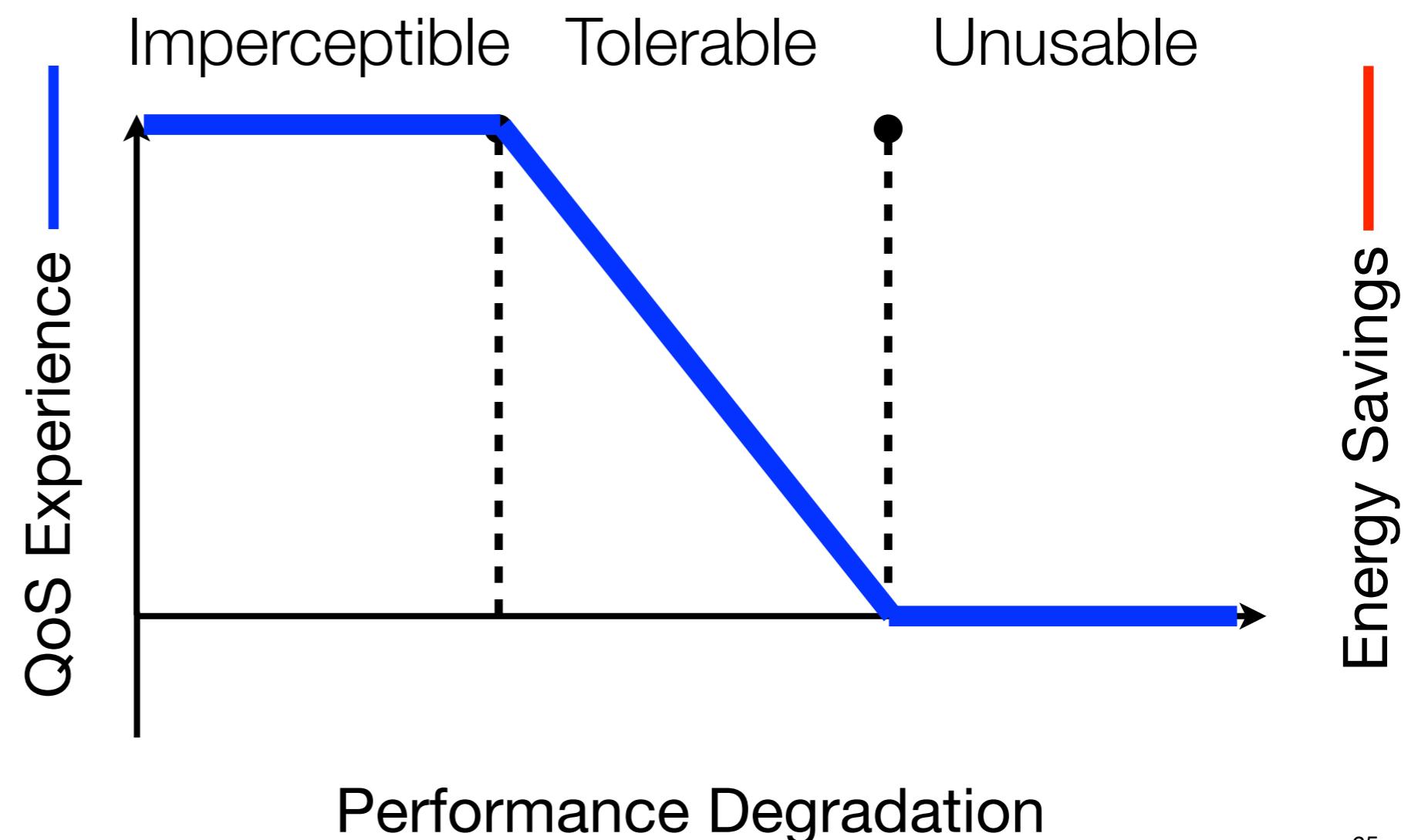


GreenWeb: QoS Web Language Extensions

Understanding Mobile QoS

Abstracting
Mobile

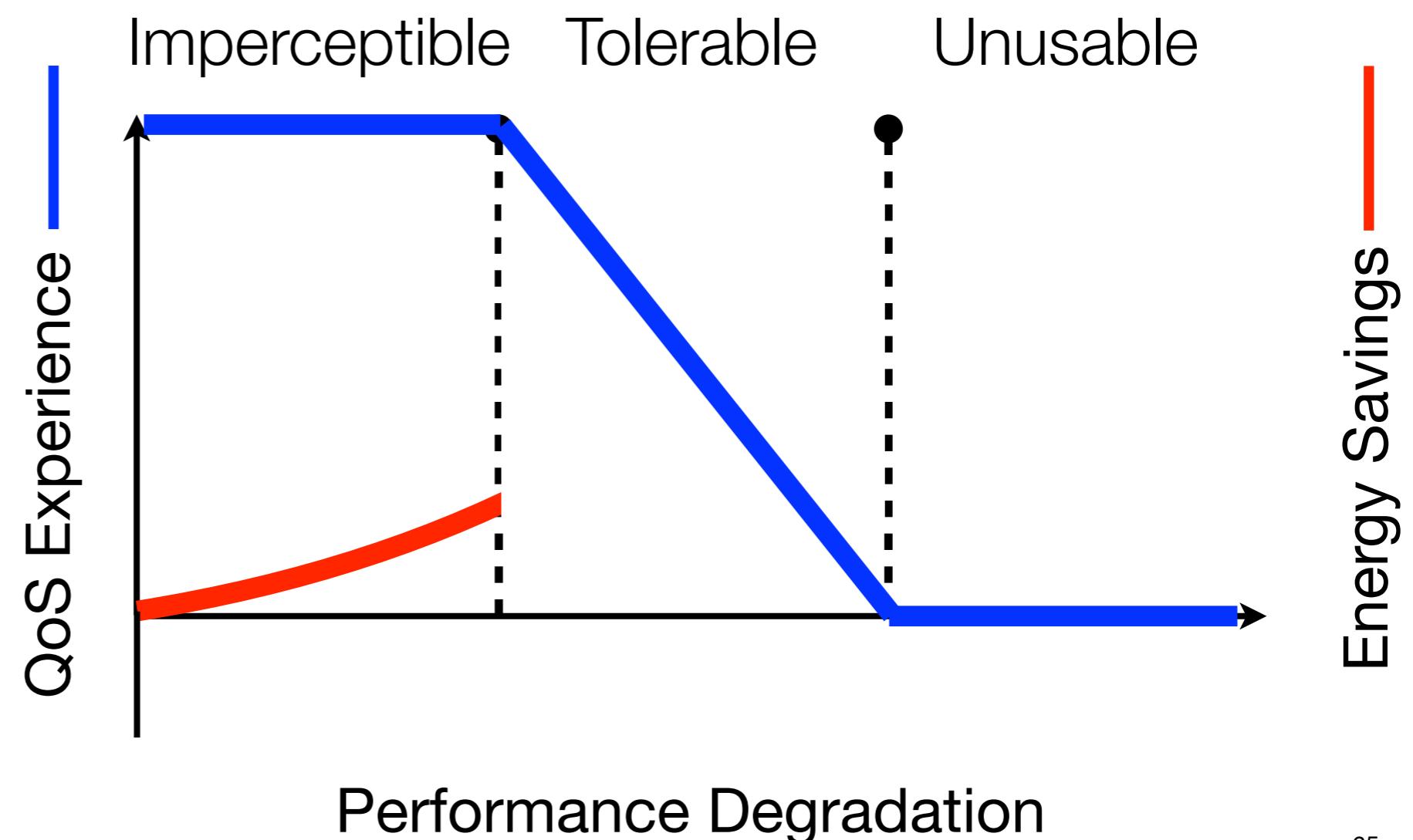
Expressing



GreenWeb: QoS Web Language Extensions

Understanding Mobile QoS

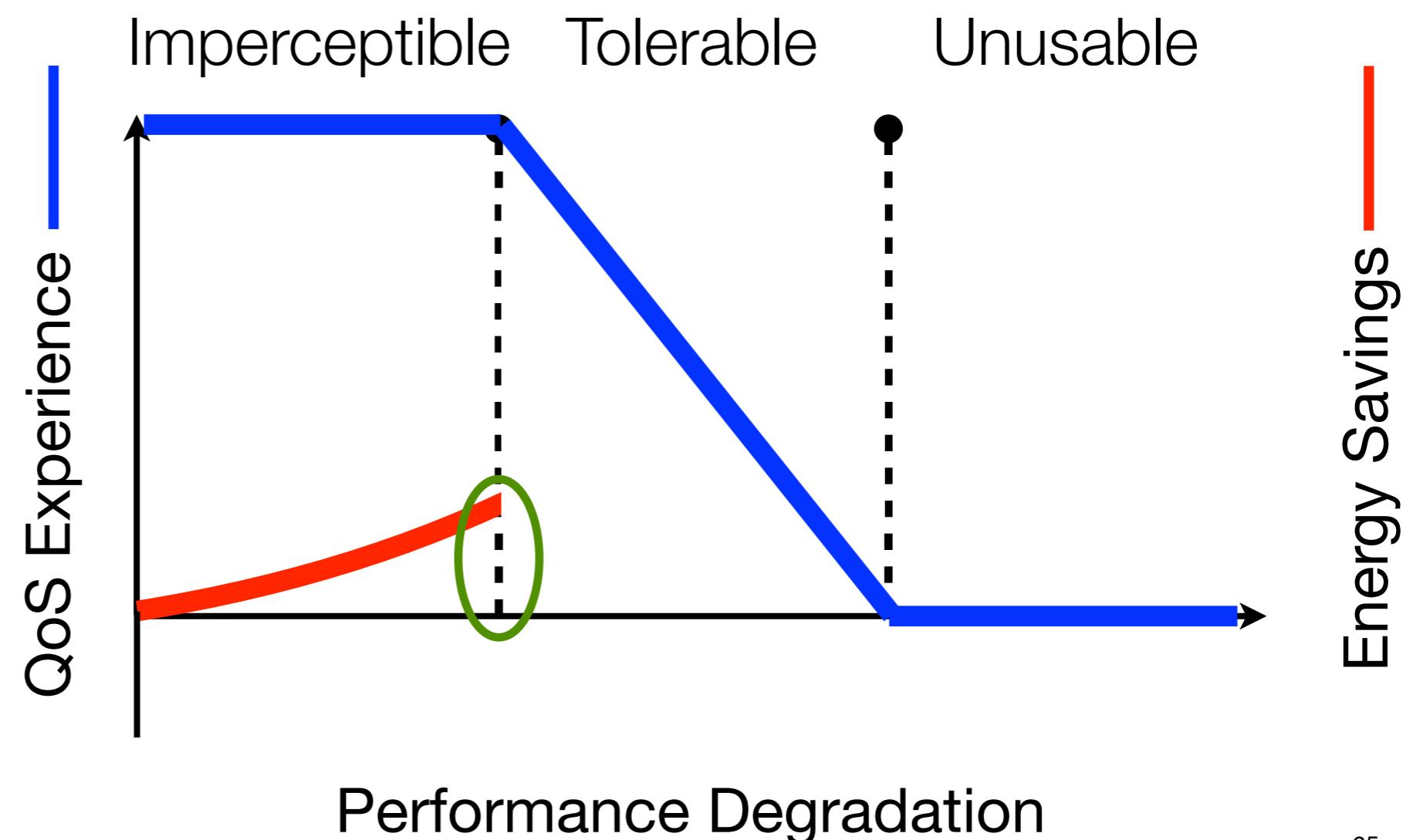
↓
Abstracting
Mobile
↓
Expressing



GreenWeb: QoS Web Language Extensions

Understanding Mobile QoS

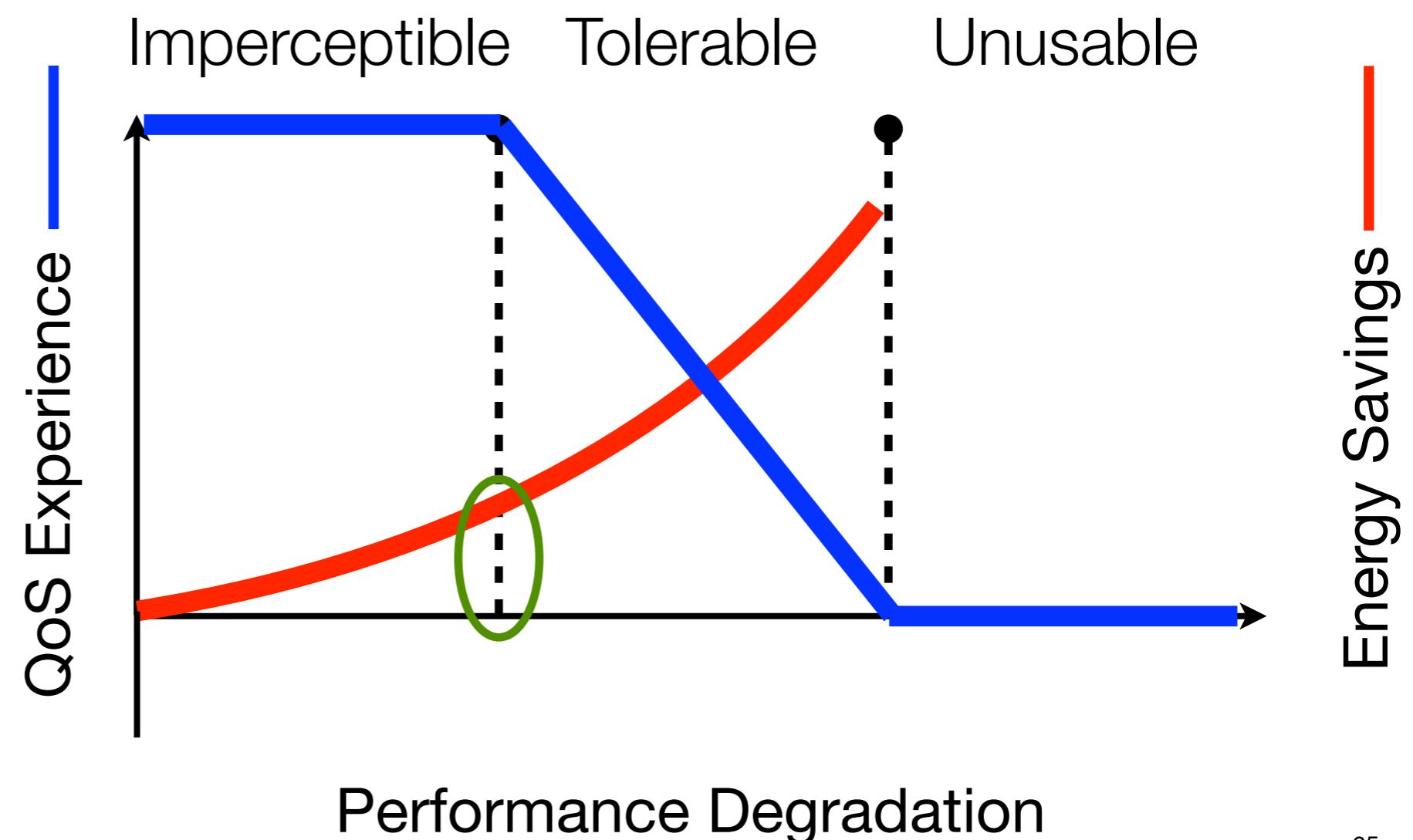
↓
Abstracting
Mobile
↓
Expressing



GreenWeb: QoS Web Language Extensions

Understanding Mobile QoS

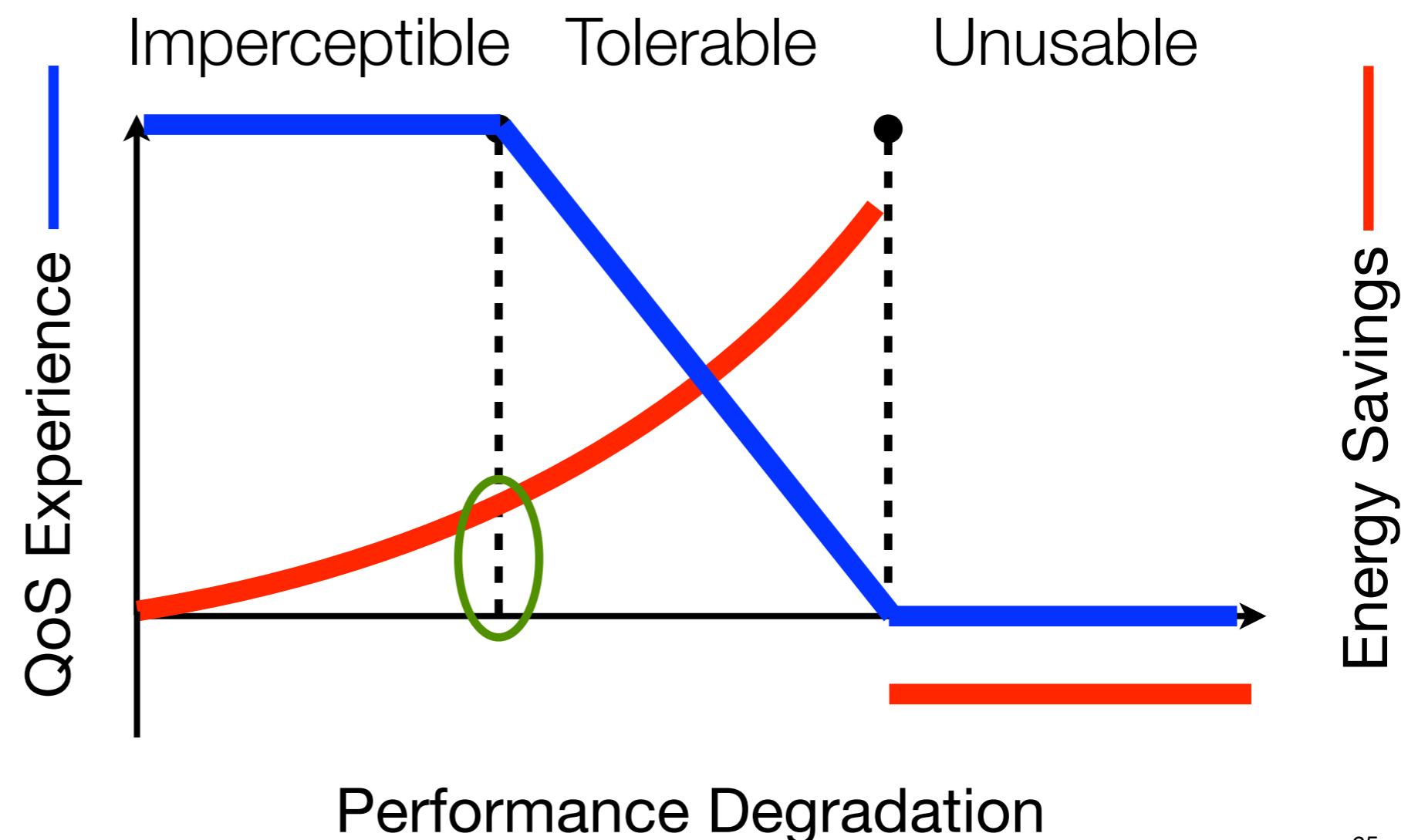
↓
Abstracting
Mobile
↓
Expressing



GreenWeb: QoS Web Language Extensions

Understanding Mobile QoS

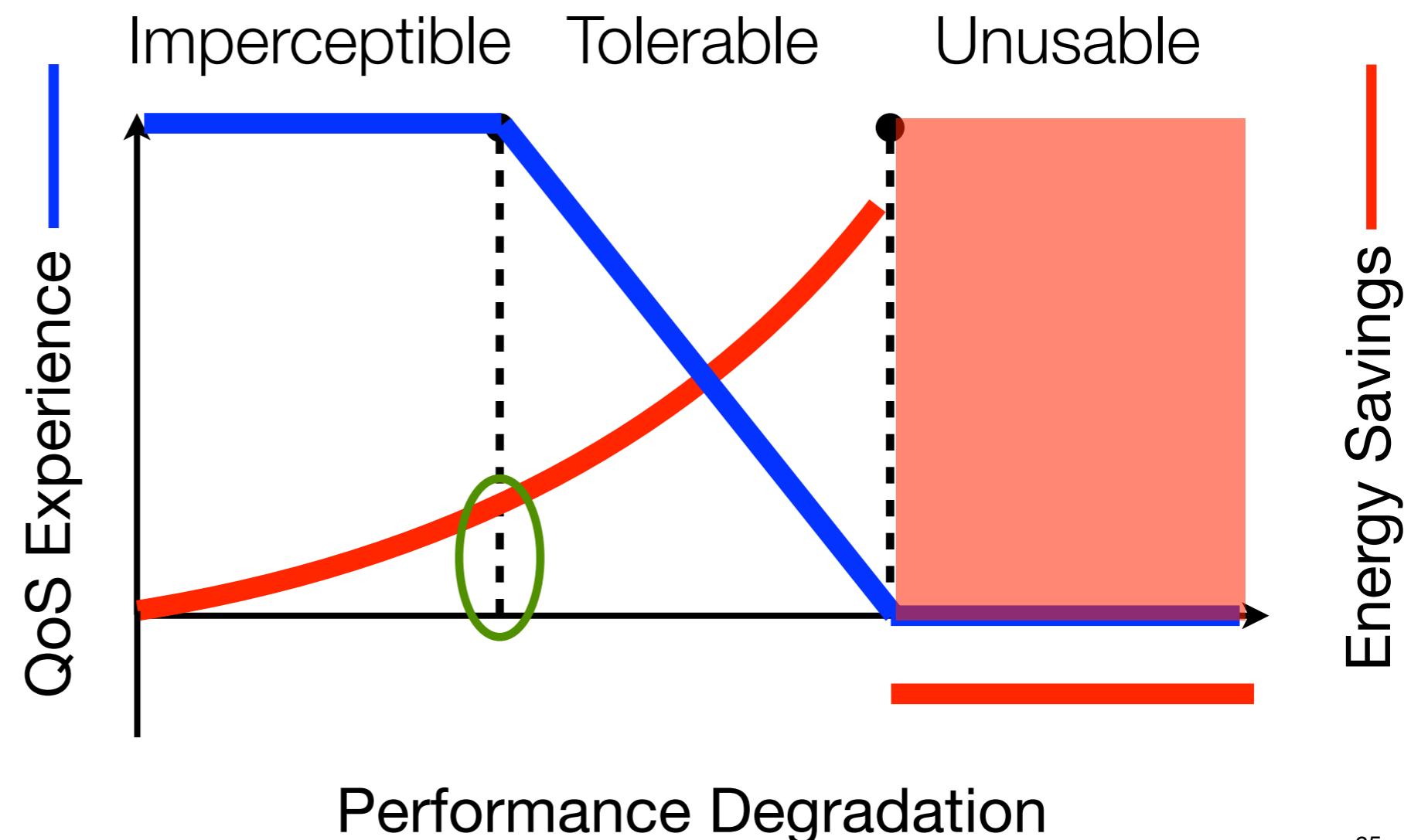
↓
Abstracting
Mobile
↓
Expressing



GreenWeb: QoS Web Language Extensions

Understanding Mobile QoS

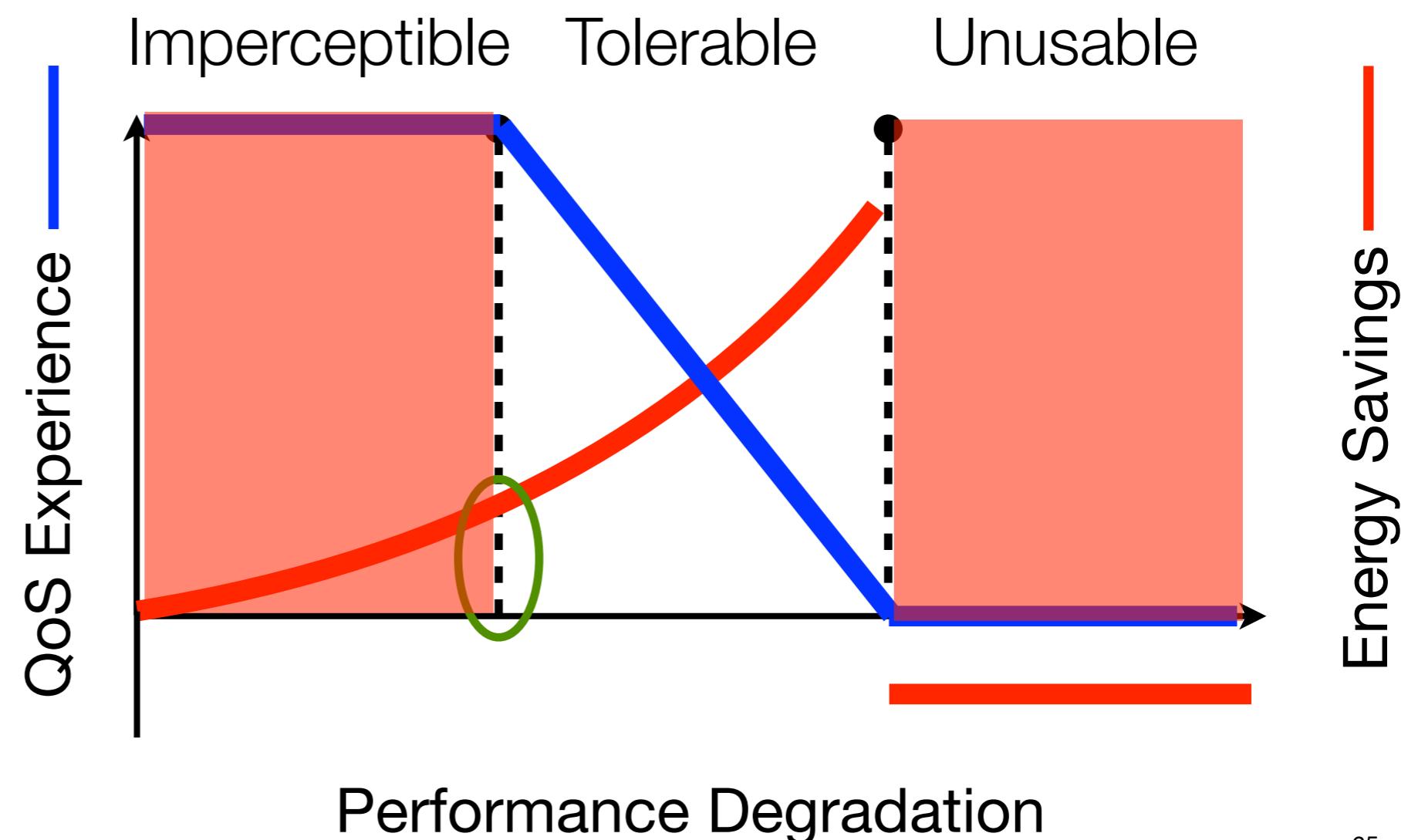
↓
Abstracting
Mobile
↓
Expressing



GreenWeb: QoS Web Language Extensions

Understanding Mobile QoS

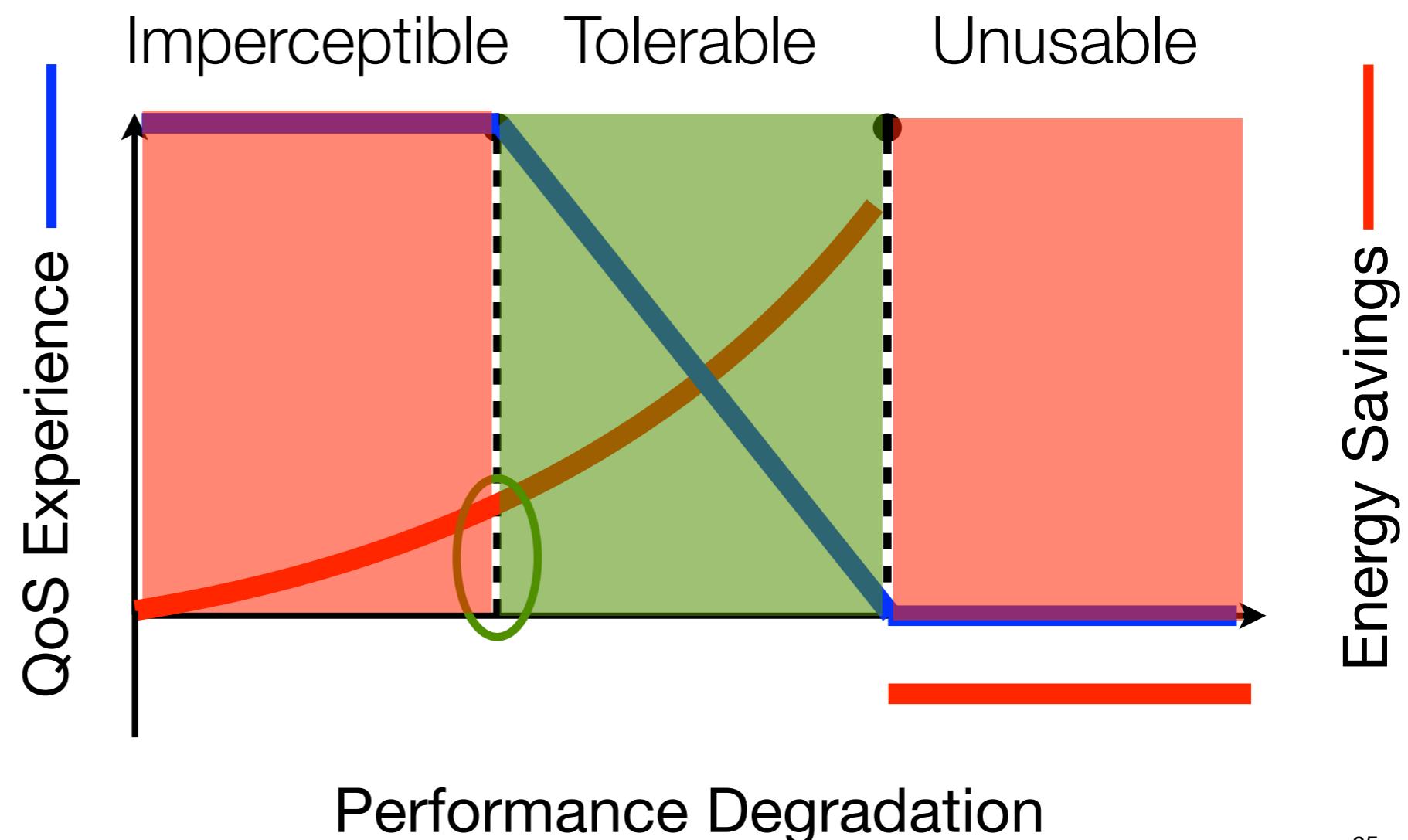
↓
Abstracting
Mobile
↓
Expressing



GreenWeb: QoS Web Language Extensions

Understanding Mobile QoS

↓
Abstracting
Mobile
↓
Expressing



GreenWeb: QoS Web Language Extensions

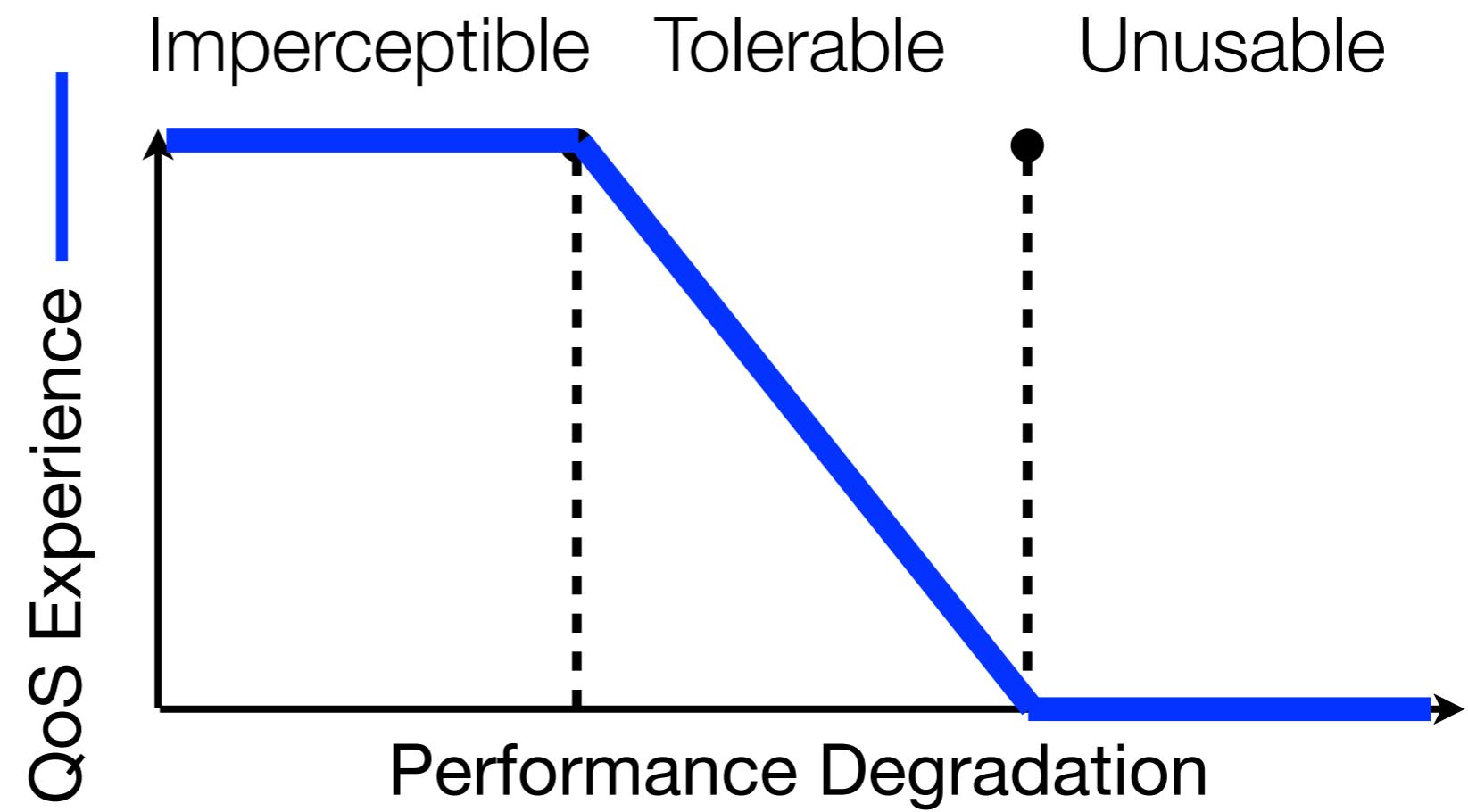
Understanding
Mobile QoS



Abstracting
Mobile QoS



Expressing
Abstractions



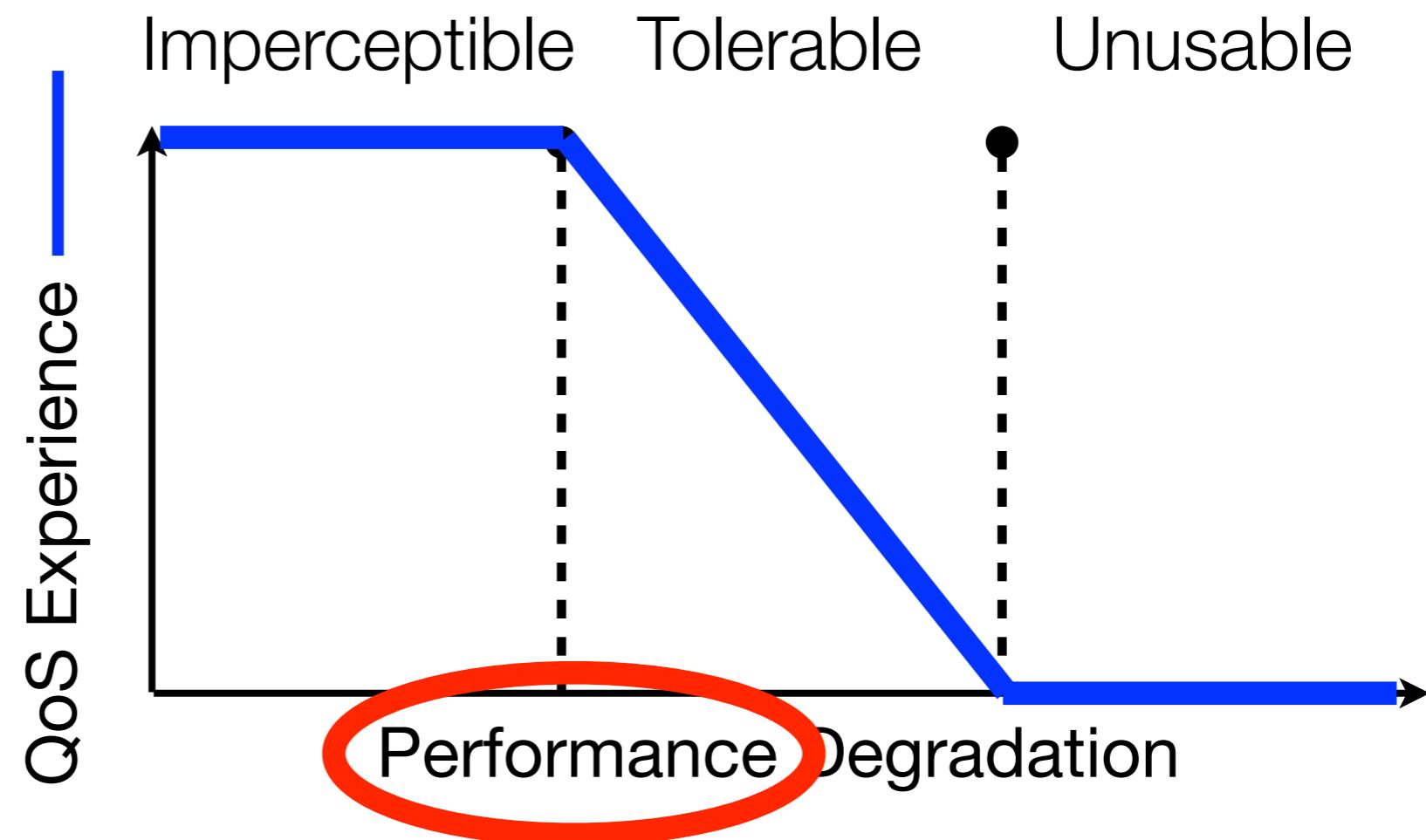
GreenWeb: QoS Web Language Extensions

Understanding
Mobile QoS

Abstracting
Mobile QoS

Expressing
Abstractions

- **QoS Type:** performance metric



GreenWeb: QoS Web Language Extensions

Understanding
Mobile QoS

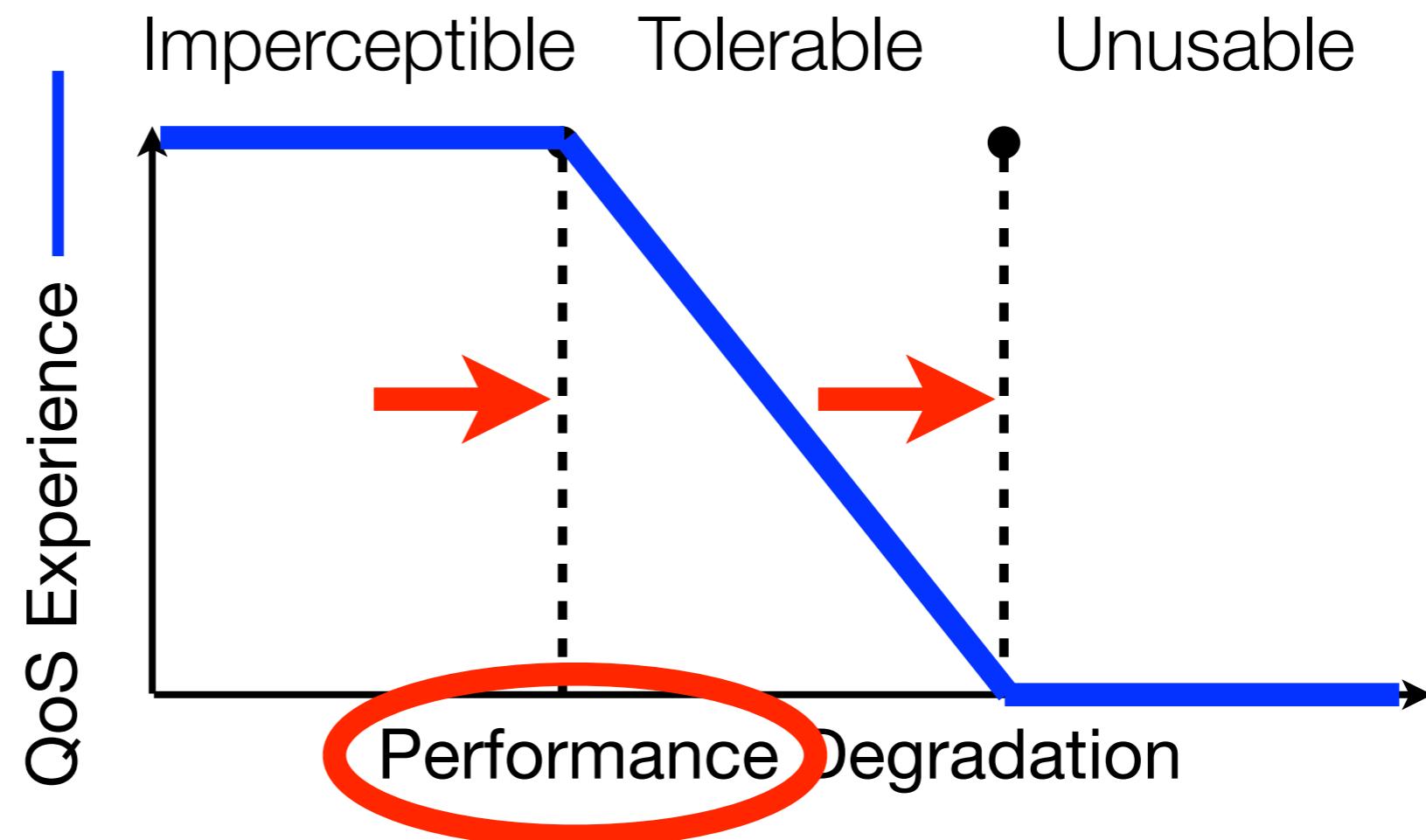


Abstracting
Mobile QoS



Expressing
Abstractions

- ▶ **QoS Type:** performance metric
- ▶ **QoS Target:** threshold performance values



GreenWeb: QoS Web Language Extensions

Understanding
Mobile QoS



Abstracting
Mobile



Expressing
Abstractions

- ▶ **QoS Type:** performance metric
- ▶ **QoS Target:** threshold performance values

Syntax (CSS Compatible)

```
element {event: Type, Target}
```

Semantics:

When **event** is triggered on **element**,
the QoS type and QoS target is **Type**
and **Target**, respectively.



Future Work

- ▶ Automatic GreenWeb Annotation
 - ▷ Empower the developers, but not overburden them!
- ▶ GreenWeb Composability
 - ▷ Can GreenWeb programs be safely integrated with other code?
 - ▷ How to compose comprehensive QoS abstractions?
- ▶ Integrating WebRT with GreenWeb
 - ▷ How can WebRT adapt to different QoS constraints?



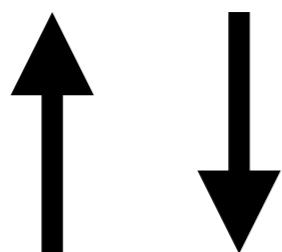
Timeline

Key Tasks	FEB	MAR	APR	MAY	JUNE	JULY	AUG
Program-level Composability Study <i>(Goal: Improve the composability and flexibility of GreenWeb extensions.)</i>							
Automatic Annotation System for GreenWeb <i>(Goal: Explore the feasibility of automatically applying GreenWeb annotations.)</i>							
WebRT Adaptivity Study <i>(Goal: Evaluate the sensitivity of WebRT with respect to different QoS constraints.)</i>							
Thesis Writing							

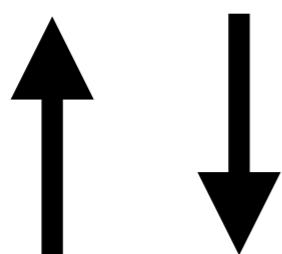


Retrospective: Three Principles Learnt

Application



Runtime

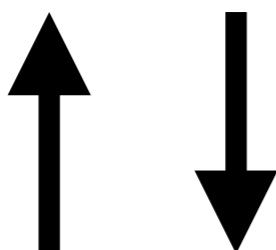


Architecture

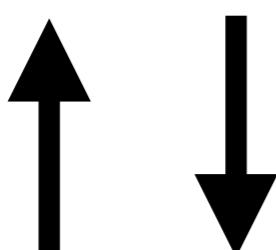


Retrospective: Three Principles Learnt

Application



Runtime



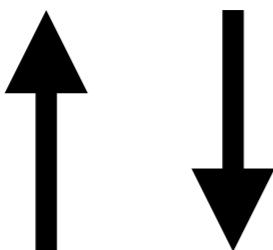
Architecture

- ▶ General-purpose vs. Specialization
 - ▶ WebCore combines general-purpose customization with domain specialization

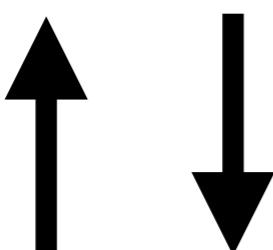


Retrospective: Three Principles Learnt

Application



Runtime



Architecture

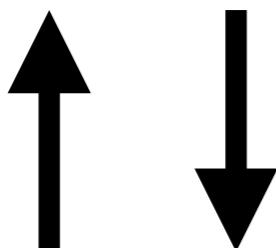
- ▶ **Exposing Hardware Complexities**
 - ▷ WebRT Leverages Core Type and Core Frequency

- ▶ **General-purpose vs. Specialization**
 - ▷ WebCore combines general-purpose customization with domain specialization



Retrospective: Three Principles Learnt

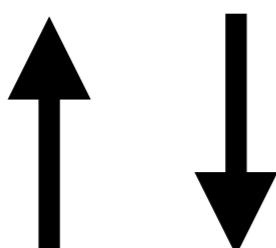
Application



▶ Empowering the Developers

- ▷ GreenWeb Language Extensions Provide QoS Abstractions

Runtime



▶ Exposing Hardware Complexities

- ▷ WebRT Leverages Core Type and Core Frequency

Architecture

▶ General-purpose vs. Specialization

- ▷ WebCore combines general-purpose customization with domain specialization



GreenWeb

[PLDI 2016] Yuhao Zhu, Vijay Janapa Reddi, “GreenWeb: Language Extensions for Energy-Efficient Mobile Web Computing”

[HPCA 2015] Yuhao Zhu, Matthew Halpern, Vijay Janapa Reddi, “Event-Based Scheduling for Energy-Efficient QoS (eQoS) in Mobile Web Applications”

[HPCA 2013] Yuhao Zhu, Vijay Janapa Reddi, “High-Performance and Energy-Efficient Mobile Web Browsing on Big/Little Systems”

[CAL 2012] Yuhao Zhu, Aditya Srikanth, Jingwen Leng, Vijay Janapa Reddi, “Exploiting Webpage Characteristics for Energy-Efficient Mobile Web Browsing” (Best of CAL)

[ISCA 2014] Yuhao Zhu, Vijay Janapa Reddi, “WebCore: Architectural Support for Mobile Web Browsing”

[IEEE MICRO 2015] Yuhao Zhu, Matthew Halpern, Vijay Janapa Reddi, “The Role of the CPU in Energy-Efficient Mobile Web Browsing”

[HPCA 2016] Matthew Halpern, Yuhao Zhu, Vijay Janapa Reddi, “Mobile CPU’s Rise to Power: Quantifying the Impact of Generational Mobile CPU Design Trends on Performance, Energy, and User Satisfaction”

[MICRO 2015] Yuhao Zhu, Daniel Richins, Matthew Halpern, Vijay Janapa Reddi, “Microarchitectural Implications of Event-driven Server-side Web Applications” (Top Picks Honorable Mention)

WebRT

Motivational Studies

Server Microarch

GPGPU & IP Routing Architecture

Tools

Reliability

- [DAC 2011] Yuhao Zhu, Yangdong Deng, Yubei Chen, “Hermes: An Integrated CPU/GPU Microarchitecture for IP Routing.”
- [DAC 2010] Bo Wang, Yuhao Zhu, Yangdong Deng, “Distributed Time, Conservative Parallel Logic Simulation on GPUs.”
- [TODAES 2011] Yuhao Zhu, Bo Wang, Yangdong Deng, “Massively Parallel Logic Simulation with GPUs.”
-
- [ISPASS 2015] Matthew Halpern, Yuhao Zhu, Ramesh Peri, and Vijay Janapa Reddi, “Mosaic: Cross-platform User-interaction Record and Replay for the Fragmented Android Ecosystem.”
-
- [IRPS 2014] Chen Zhou, Xiaofei Wang, Weichao Xu, Yuhao Zhu, Vijay Janapa Reddi, Chris Kim, “Estimation of Instantaneous Frequency Fluctuation in a Fast DVFS Environment Using an Empirical BTI Stress-Relaxation Model.”

Coursework

Name	Instructor	Semester	SUP	Grade
COMPILERS	Keshav Pingali	Fall 2010		A
ADV EMBED MICROCONTROL SYS	Mark McDermott	Spring 2011		A-
MEMORY MANAGEMENT	Kathryn McKinley	Spring 2011	Y	A
VLSI I	Jacob Abraham	Fall 2011		A-
COMP ARCH: PARALLISM/LOCLTY	Mattan Erez	Fall 2011		A
MICROARCHITECTURE	Yale Patt	Spring 2012		B
DYNAMIC COMPILATION	Vijay Janapa Reddi	Spring 2012		A-
COMP PERF EVAL/BENCHMARKING	Lizy John	Fall 2012		B+
PARALLEL COMP ARCHITECTURE	Derek Chiou	Spring 2013		B+
HUMAN COMPUT & CROWDSRCING	Matt Lease	Fall 2015	Y	A-



Thank you!