THE UNIVERSITY *of* NORTH CAROLINA *at* CHAPEL HILL

Making OpenVX Really "Real Time"

Ming Yang[1], Tanya Amert[1], Kecheng Yang[1,2], Nathan Otterness[1], James H. Anderson[1], F. Donelson Smith[1], and Shige Wang[3]

[1]The University of North Carolina at Chapel Hill
[2]Texas State University
[3]General Motors Research

700 ms

OpenVX™

A new approach for OpenVX™ graph scheduling

Shorter response time

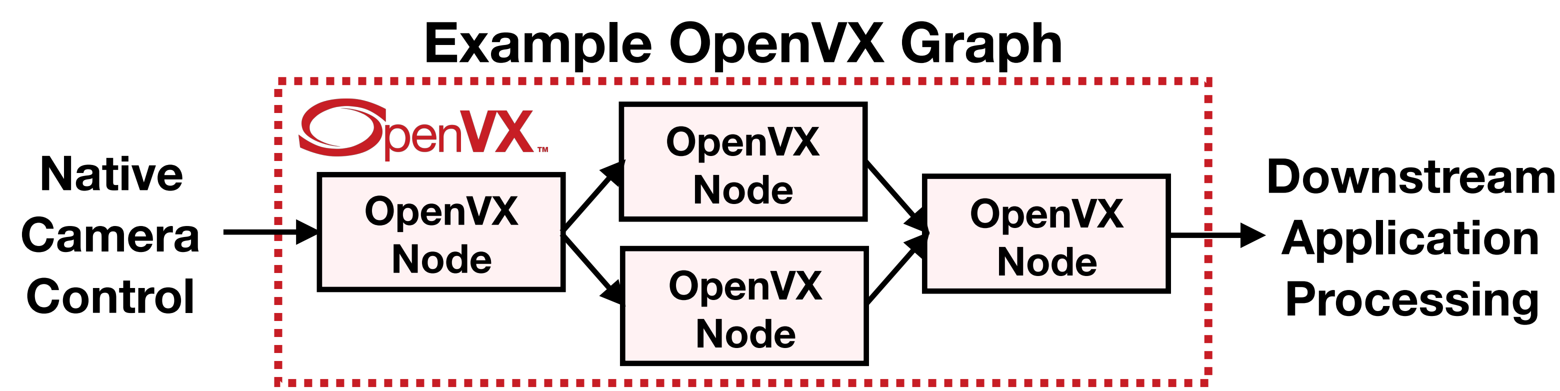
+

Less capacity loss

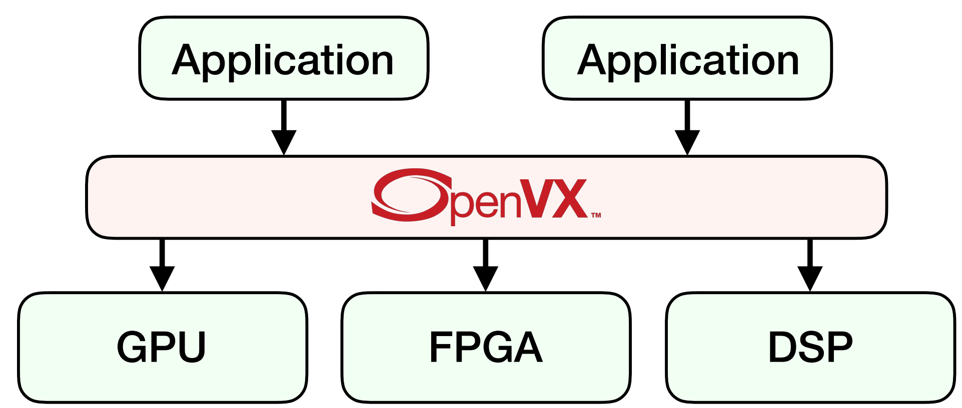
1. State of the art
2. Our approach
3. Future work

OpenVX™

Graph-based architecture



Portability to diverse hardware



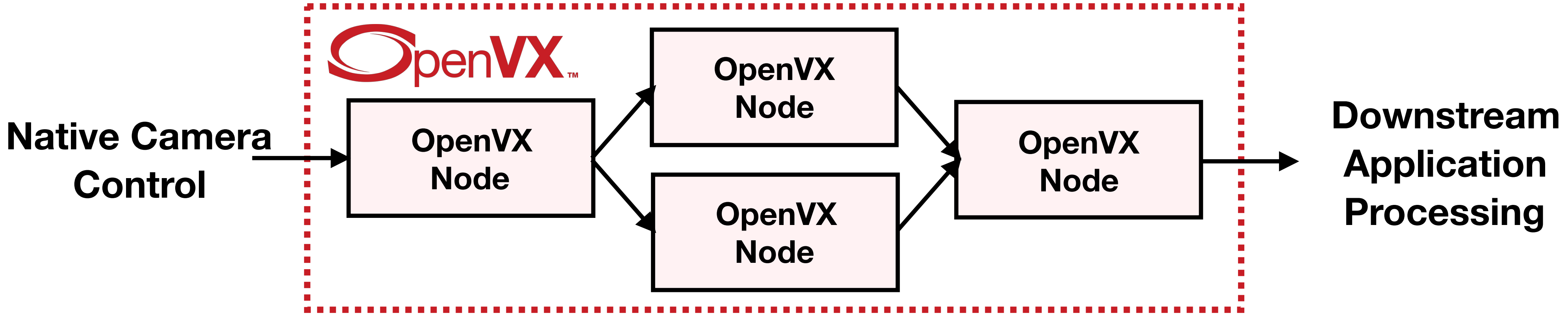
Does OpenVX really target “real-time” processing?

Source: https://www.khronos.org/openvx/

Does OpenVX really target “real-time” processing?

1. It lacks real-time concepts
2. Entire graphs = monolithic schedulable entities

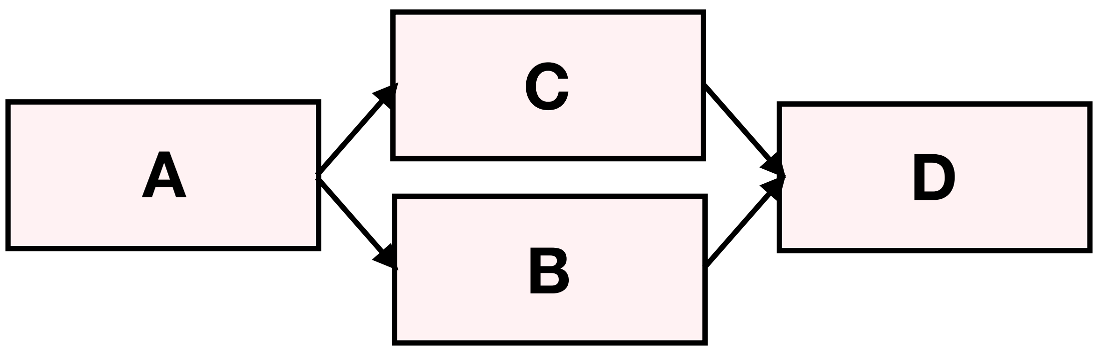
Example OpenVX Graph



Source: https://www.khronos.org/openvx/

Does OpenVX really target “real-time” processing?

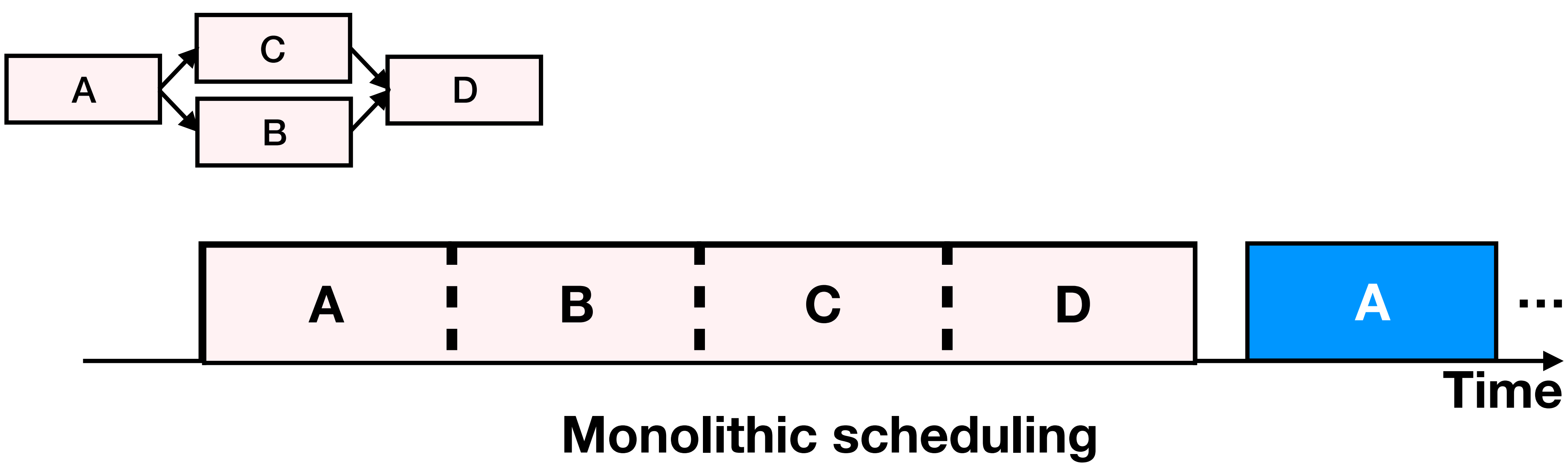
1. It lacks real-time concepts
2. Entire graphs = monolithic schedulable entities



Source: https://www.khronos.org/openvx/

Does OpenVX really target “real-time” processing?

1. It lacks real-time concepts
2. Entire graphs = monolithic schedulable entities

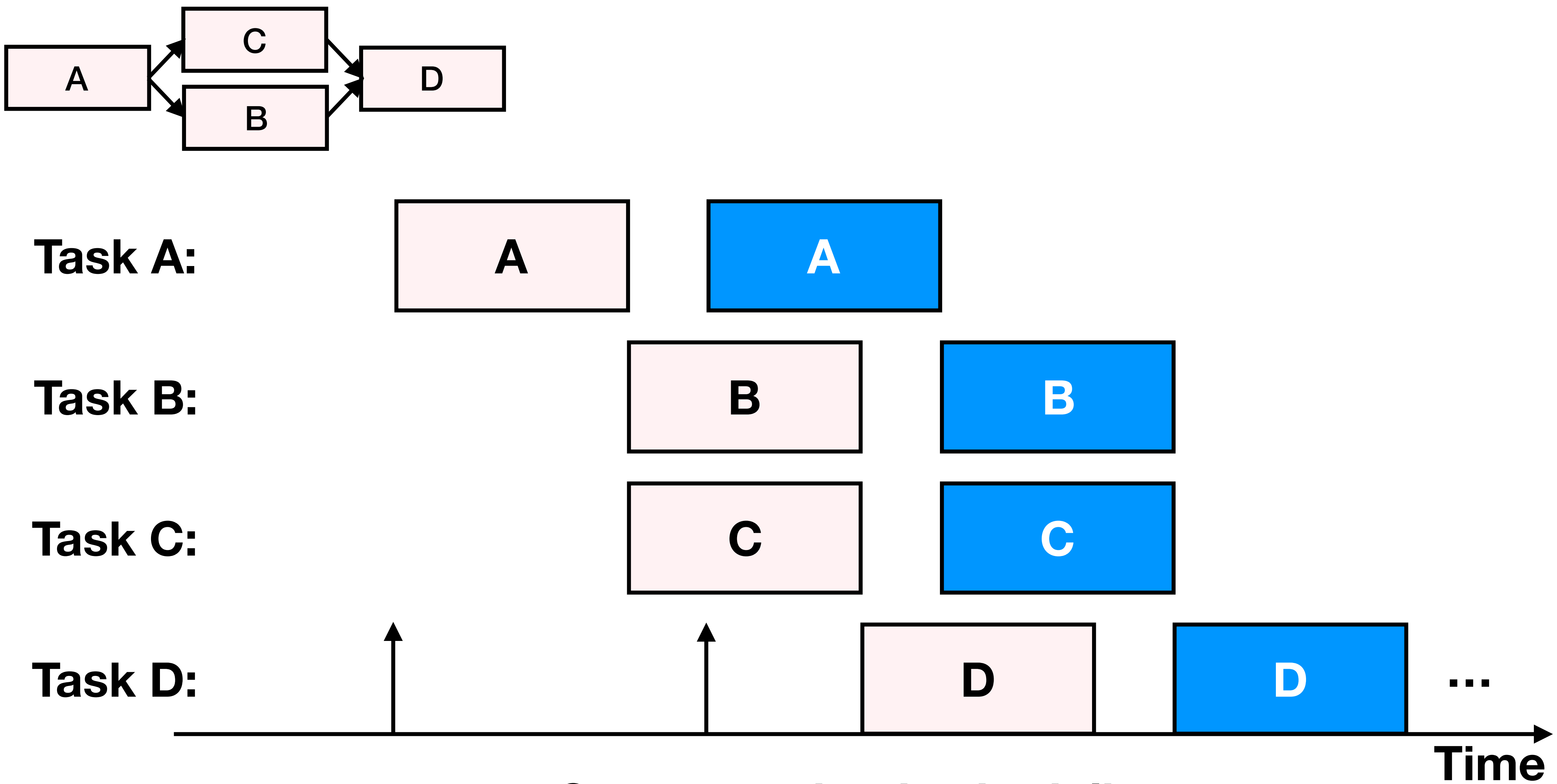


Source: https://www.khronos.org/openvx/

Prior Work

Coarse-grained scheduling

- OpenVX nodes = schedulable entities [23, 51]



Coarse-grained scheduling

Prior Work

Coarse-grained scheduling

- OpenVX nodes = schedulable entities [23, 51]

Remaining problems:

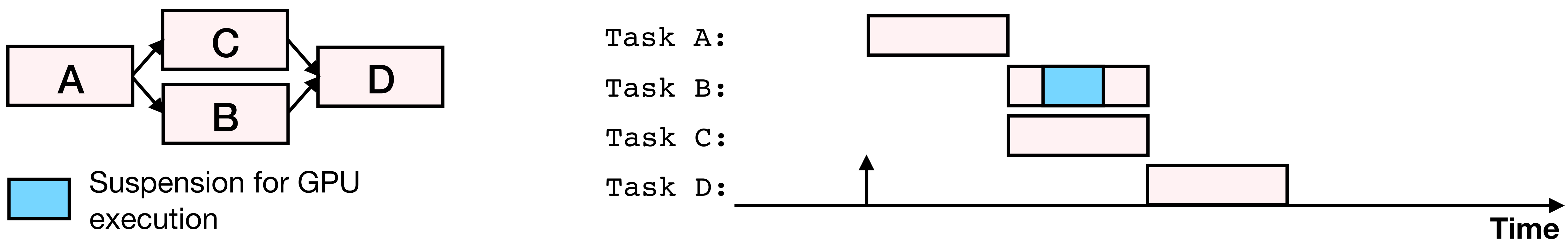
1. More parallelism to be explored
2. Suspension-oblivious analysis was applied and causes capacity loss.

Fine-Grained Scheduling

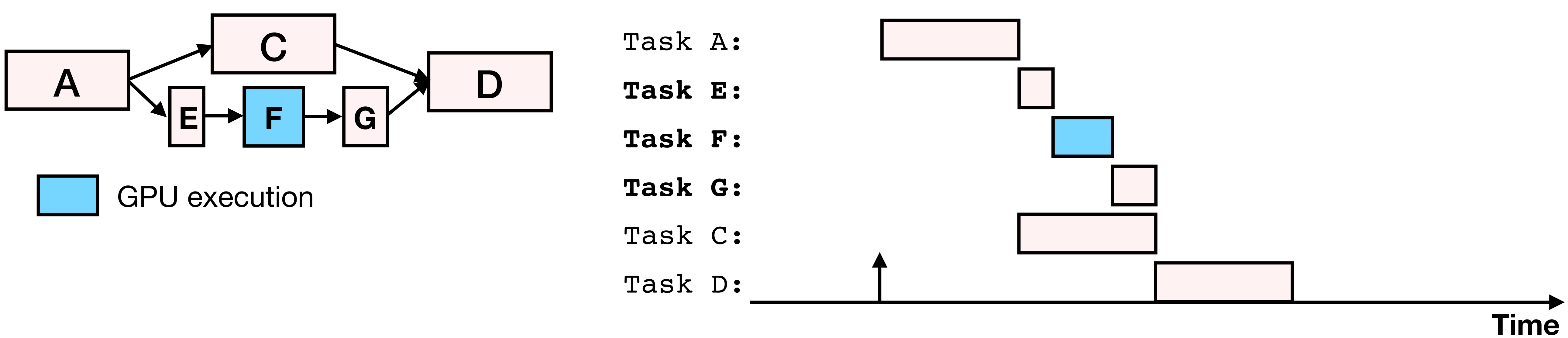
This Work

1. Coarse-grained vs. fine-grained
2. Response-time bounds analysis
3. Case study

1. Coarse-grained vs. fine-grained
2. Response-time bounds analysis
3. Case study



Coarse-Grained Scheduling



Fine-Grained Scheduling

1. Coarse-grained vs. fine-grained
2. Response-time bounds analysis
3. Case study

Deriving Response-Time Bounds for a DAG*

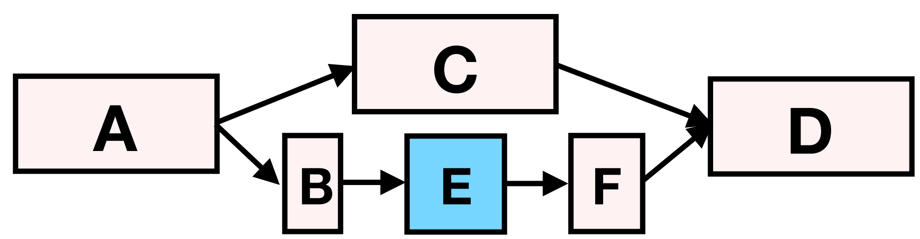
Step 1: Schedule the nodes as sporadic tasks

Step 2: Compute bounds for every node

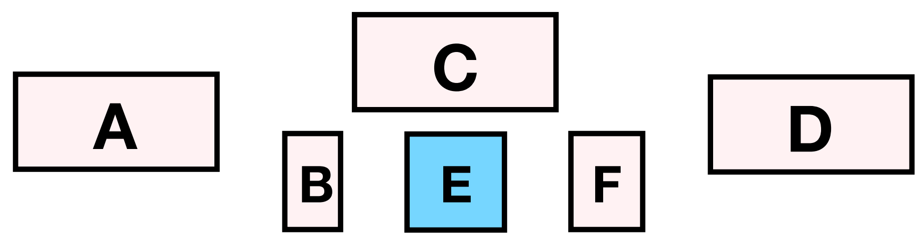
Step 3: Sum the bounds of nodes on the critical path

* *C. Liu and J. Anderson, "Supporting Soft Real-Time DAG-based Systems on Multiprocessors with No Utilization Loss," in RTSS, 2013.*

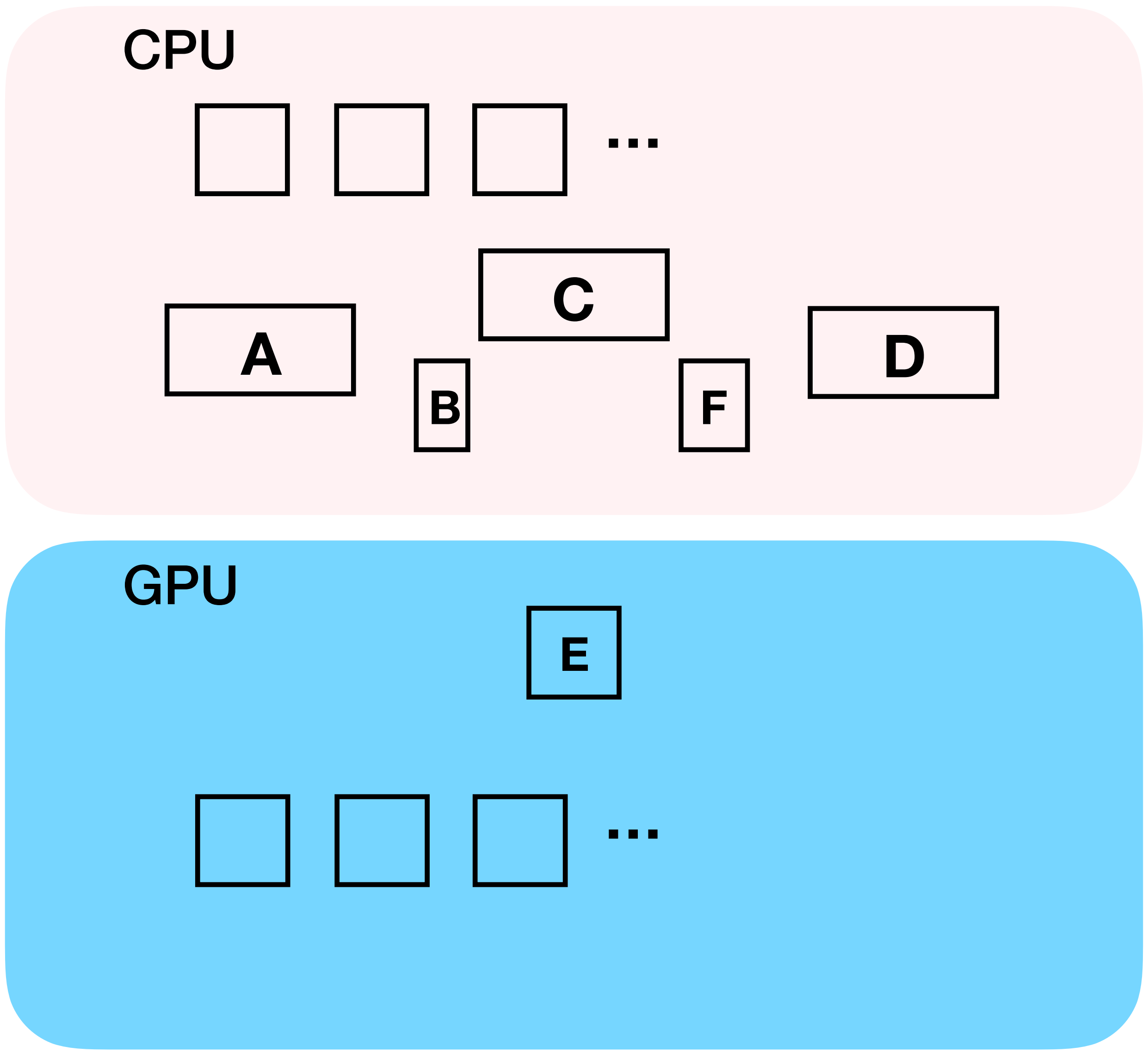
Deriving Response-Time Bounds for a DAG



Deriving Response-Time Bounds for a DAG



Deriving Response-Time Bounds for a DAG



Need a response-time bound analysis for GPU tasks

A system model of GPU Tasks

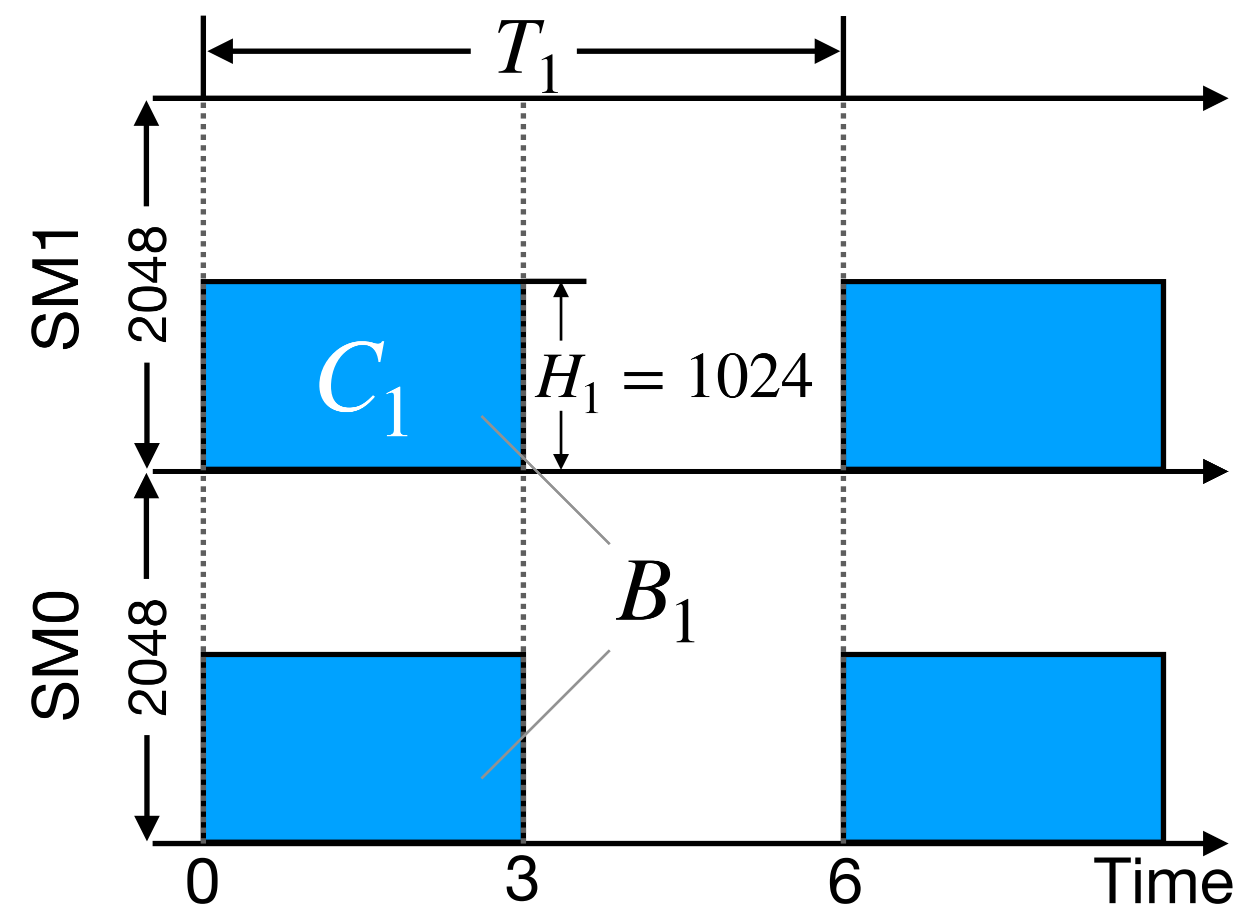
Per-block worst-case workload

Number of blocks

$$\tau_i = (C_i,\ T_i,\ B_i,\ H_i)$$

Period

Number of threads per block (or block size)



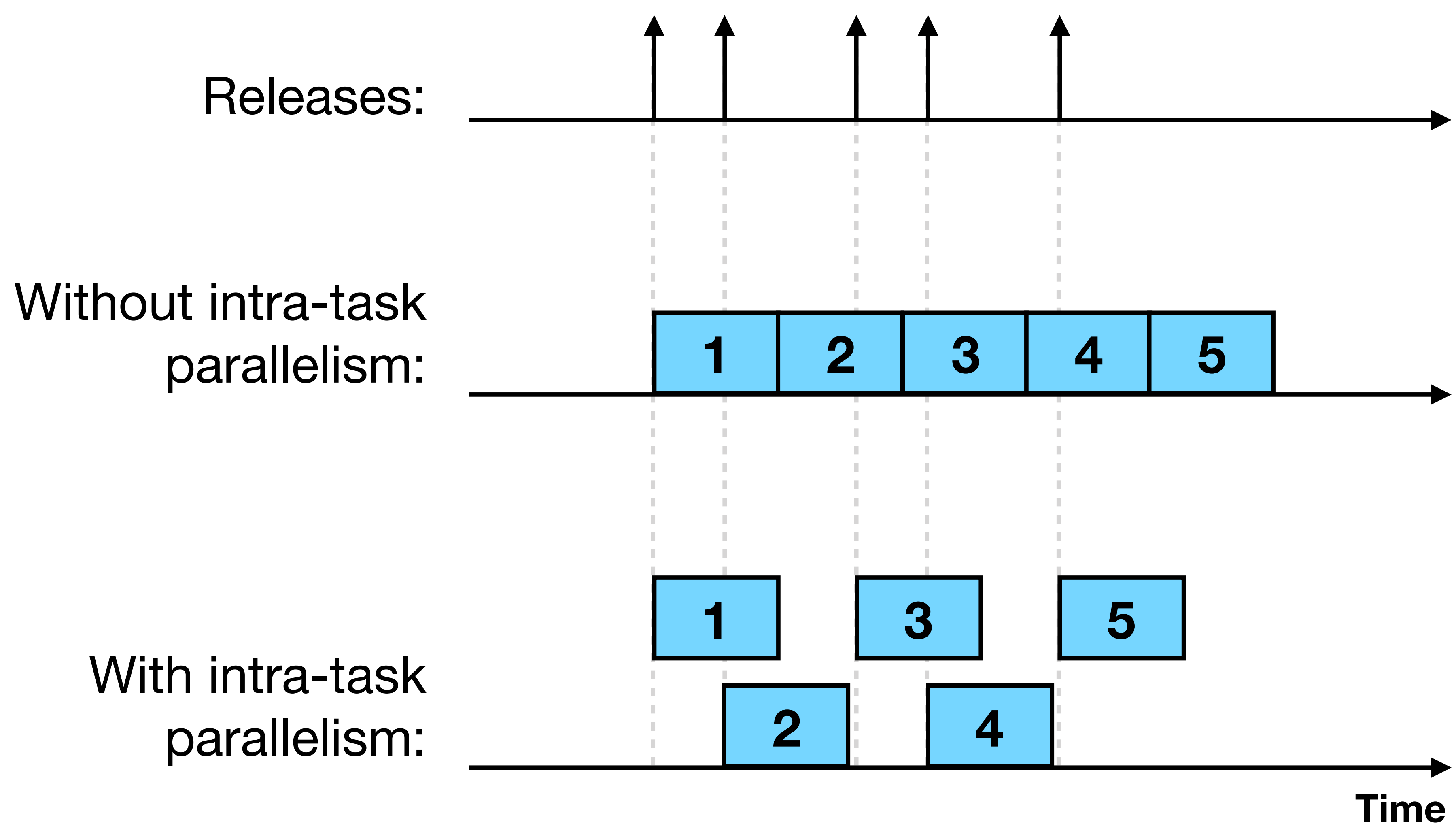
$$\tau_1 = (3076,6,2,1024)$$

Response-Time Bounds Proof Sketch

1. We first show the necessity of a total utilization bound and intra-task parallelism via counterexamples.

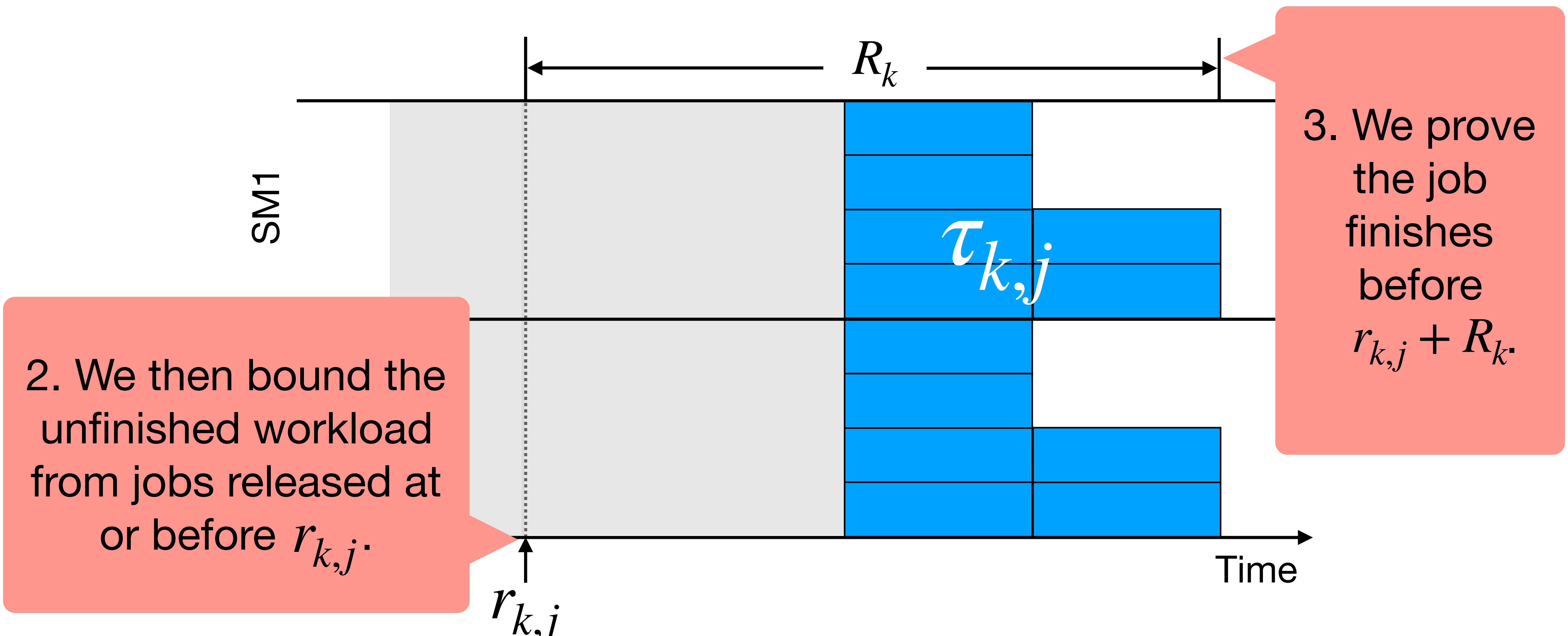
Response-Time Bounds Proof Sketch

1. We first show the necessity of a total utilization bound and intra-task parallelism via counterexamples.



Response-Time Bounds Proof Sketch

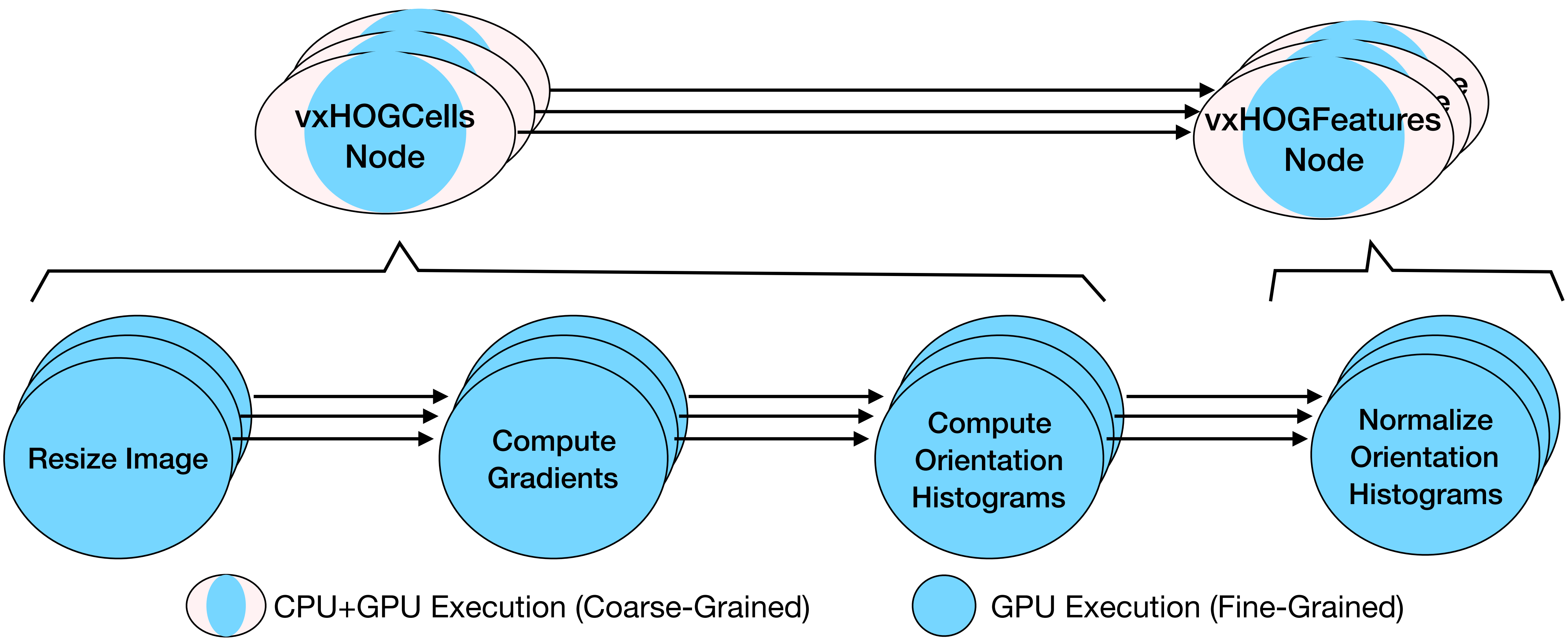
1. We first show the necessity of a total utilization bound and intra-task parallelism via counterexamples.



1. Coarse-grained vs. fine-grained
2. Response-time bounds analysis
3. Case study

Case Study: Comparing Fine-Grained/ Coarse-Grained/Monolithic Scheduling

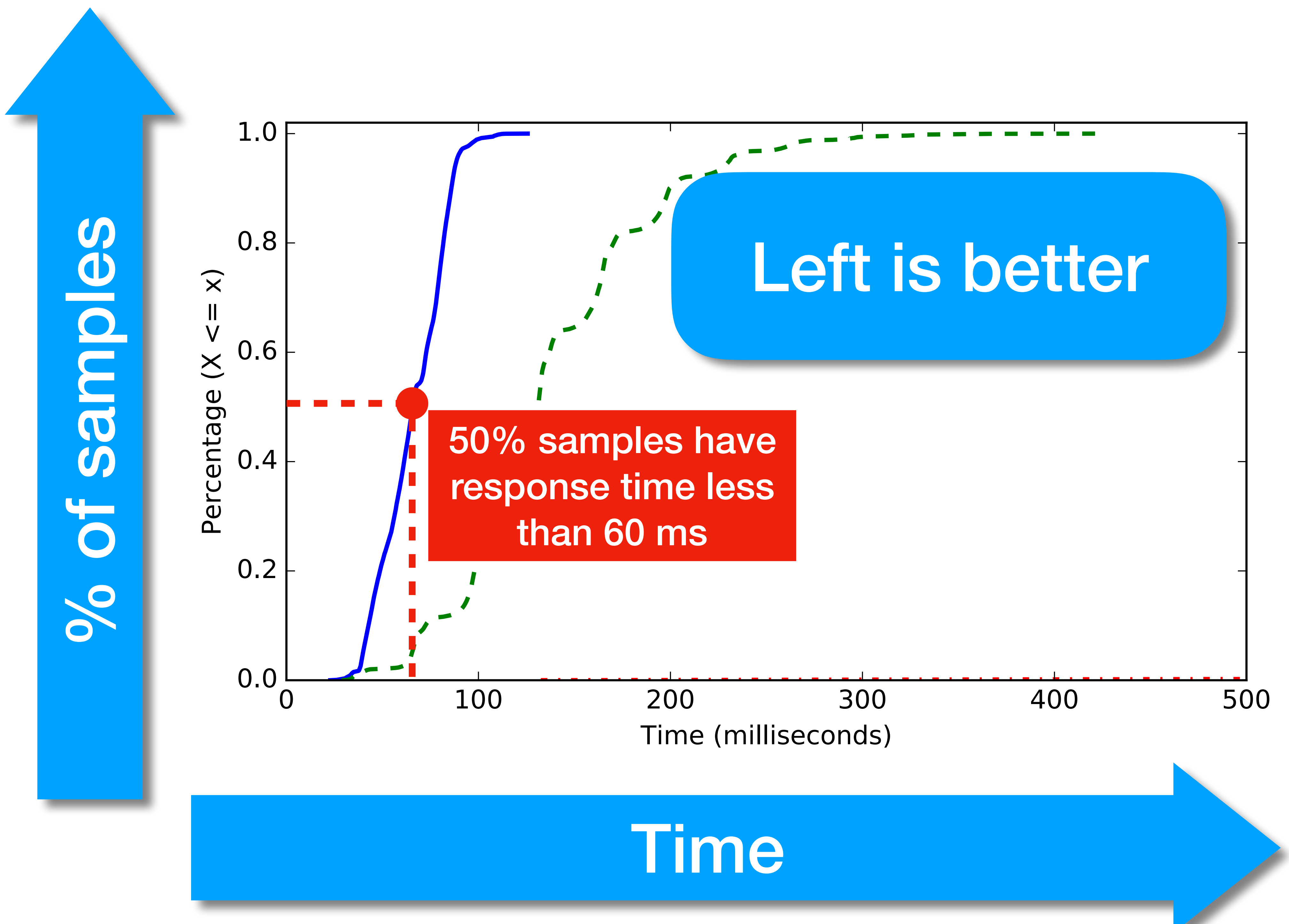
- Application: Histogram of Oriented Gradients (HOG)



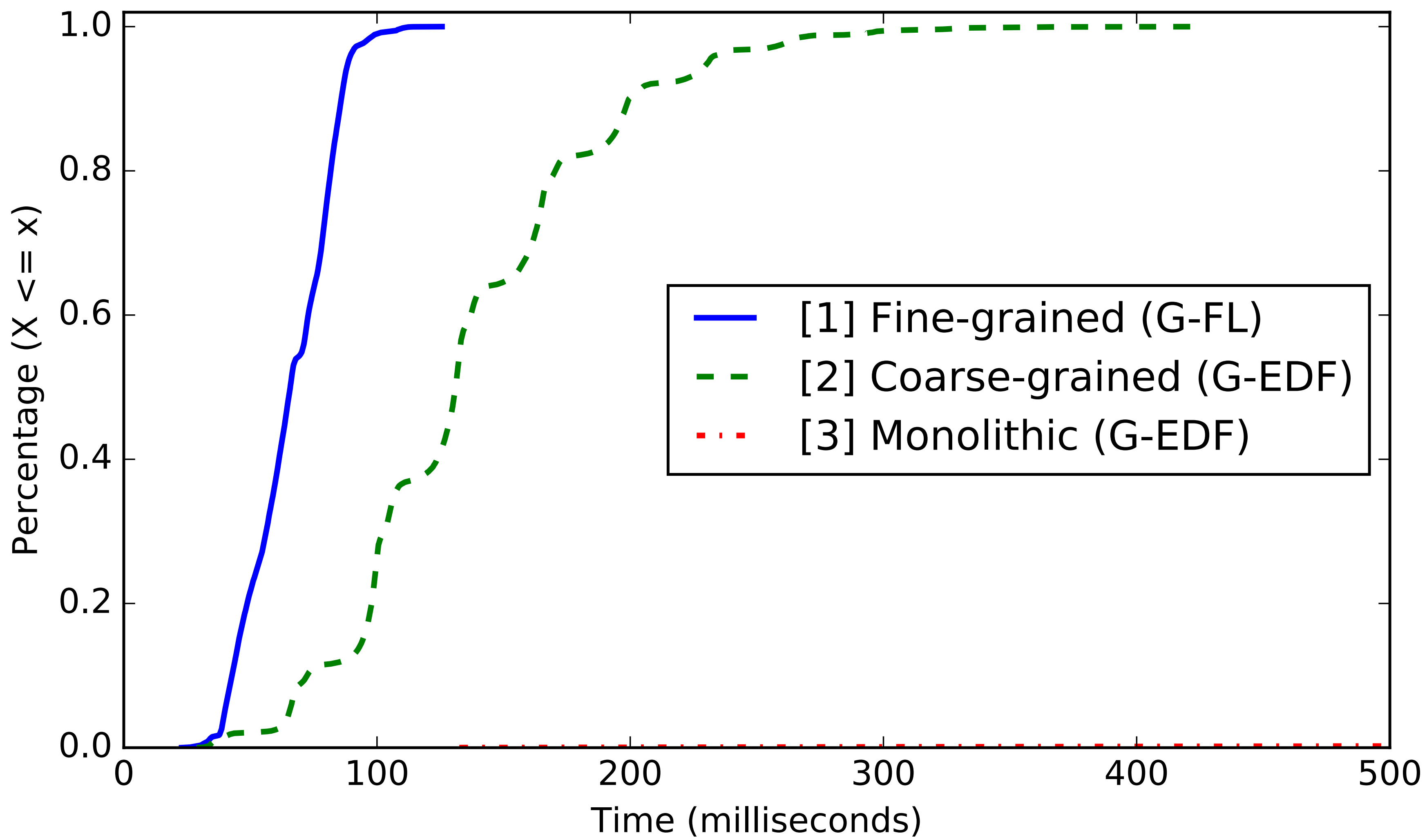
Case Study: Comparing Fine-Grained/ Coarse-Grained/Monolithic Scheduling

- Application: Histogram of Oriented Gradients (HOG)
  - 6 instances
  - 33 ms period
  - 30,000 samples
- Platform: NVIDIA Titan V GPU + Two eight-core Intel CPUs.
- Schedulers: **G-EDF**, **G-FL** (fair-lateness)

Case Study: Comparing Fine-Grained/ Coarse-Grained/Monolithic Scheduling



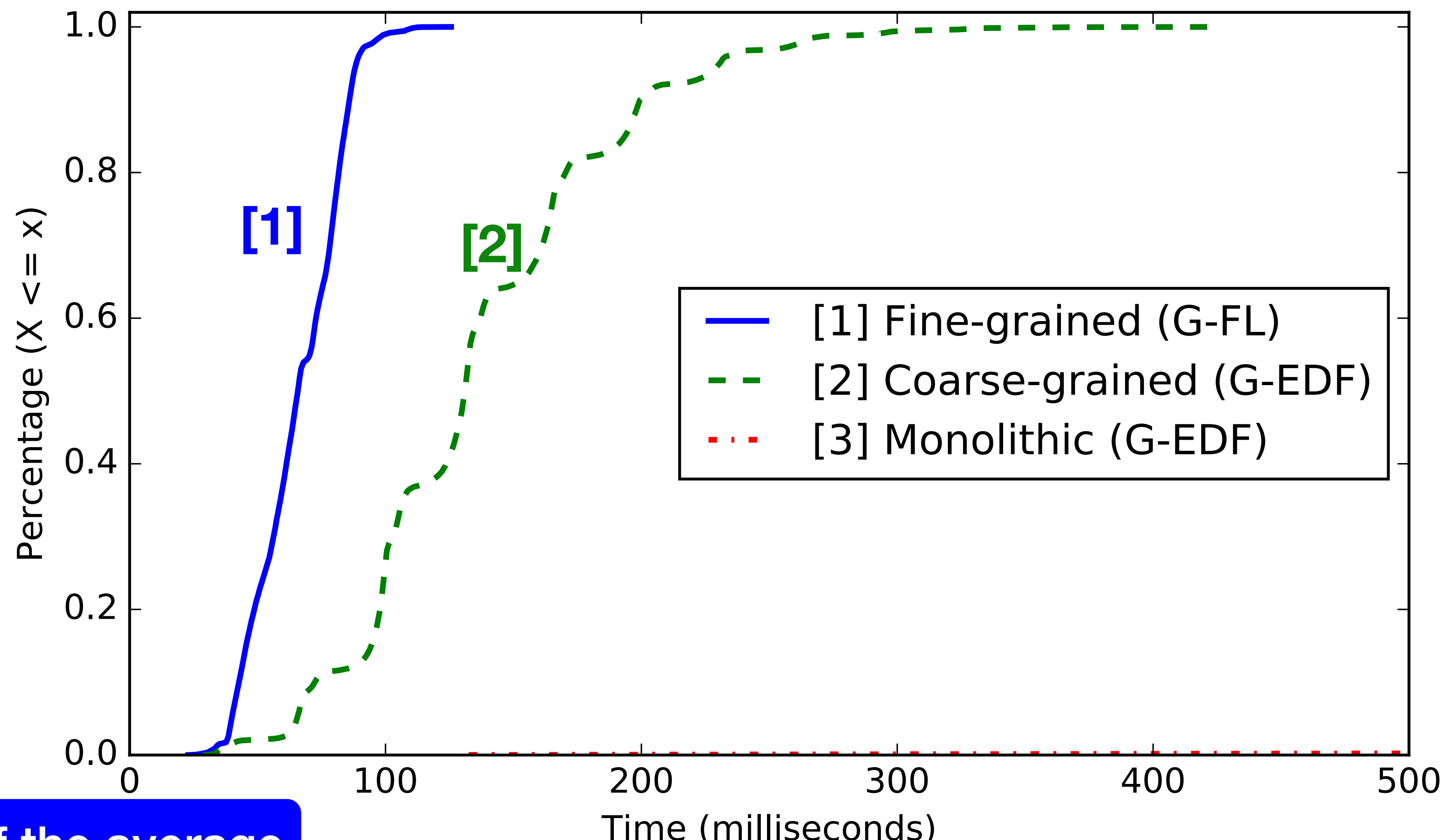
Case Study: Comparing Fine-Grained/ Coarse-Grained/Monolithic Scheduling



FL: fair-lateness

	[1] Fine-grained (G-FL)	[2] Coarse-grained (G-EDF)	[3] Monolithic (G-EDF)
Average Response Time (ms)	65.99	136.57	84669.47
Maximum Response Time (ms)	125.66	427.07	170091.06

Case Study: Comparing Fine-Grained/ Coarse-Grained/Monolithic Scheduling

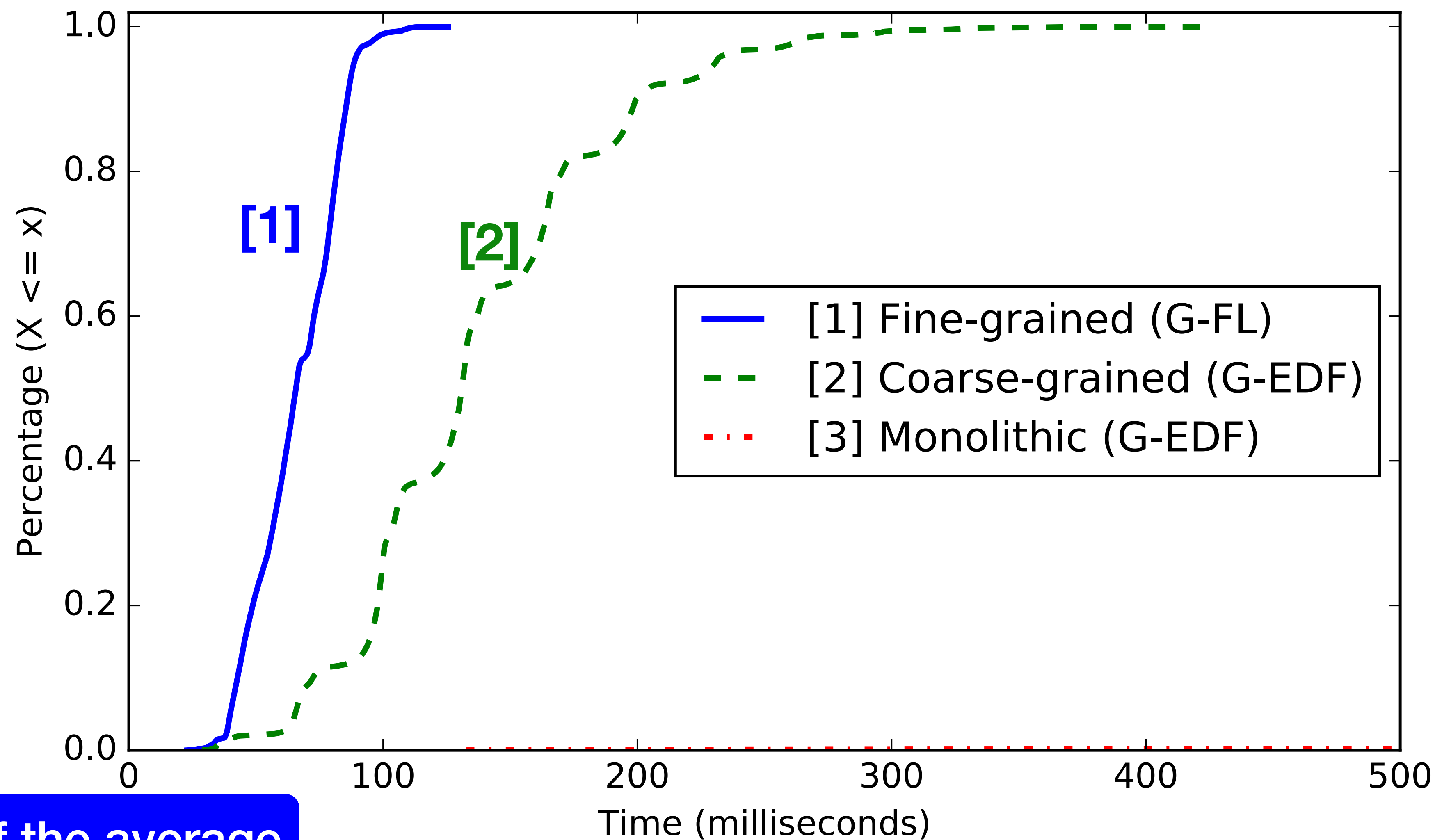


FL: fair-lateness

Half the average response time

	[1] Fine-grained (G-FL)	[2] Coarse-grained (G-EDF)	[3] Monolithic (G-EDF)
Average Response Time (ms)	**65.99**	136.57	84669.47
Maximum Response Time (ms)	125.66	427.07	170091.06

Case Study: Comparing Fine-Grained/ Coarse-Grained/Monolithic Scheduling



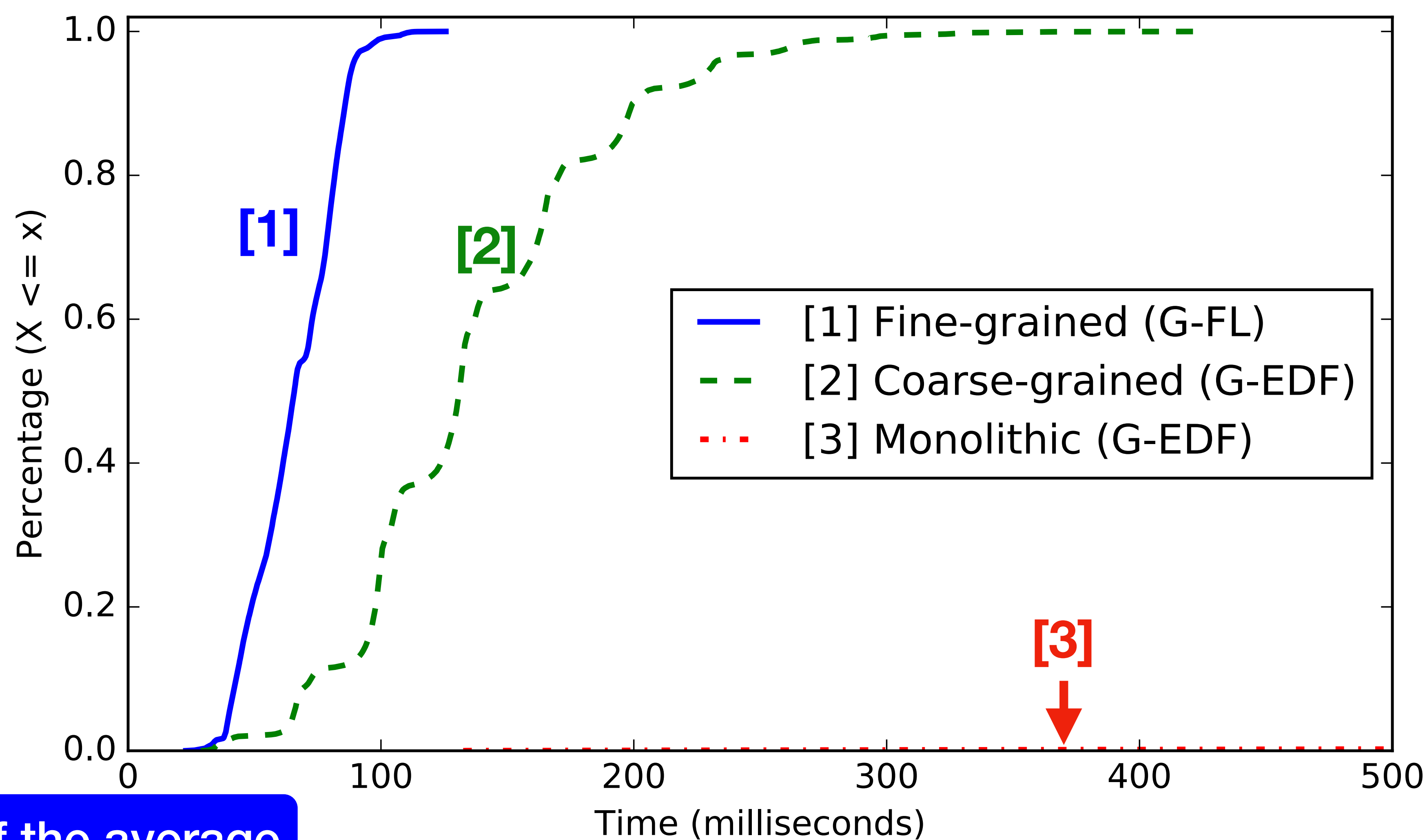
FL: fair-lateness

Half the average response time

	[1] Fine-grained (G-FL)	[2] Coarse-grained (G-EDF)	[3] Monolithic (G-EDF)
Average Response Time (ms)	**65.99**	136.57	84669.47
Maximum Response Time (ms)	**125.66**	427.07	170091.06

One-third the maximum response time

Case Study: Comparing Fine-Grained/ Coarse-Grained/Monolithic Scheduling



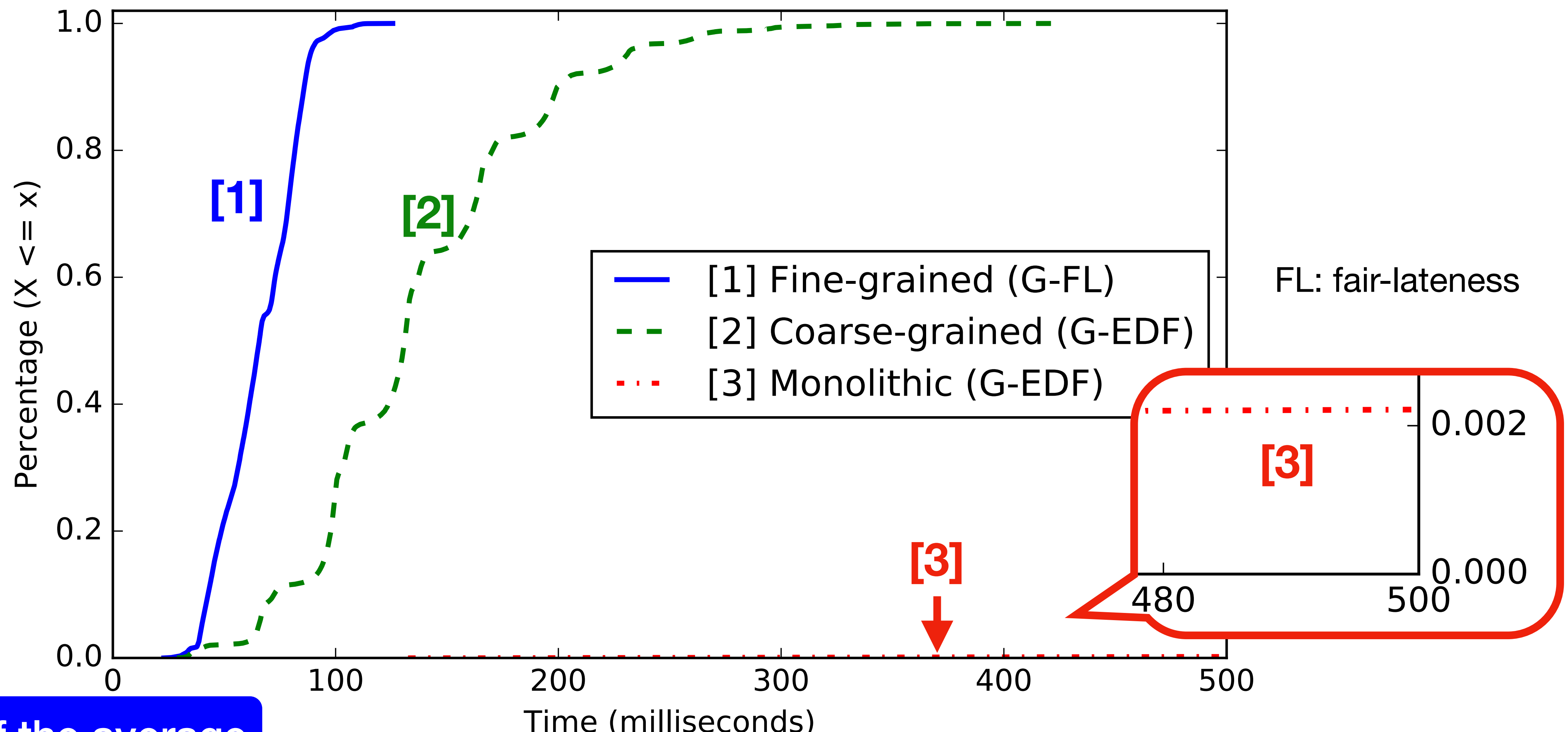
FL: fair-lateness

Half the average response time

	[1] Fine-grained (G-FL)	[2] Coarse-grained (G-EDF)	[3] Monolithic (G-EDF)
Average Response Time (ms)	**65.99**	136.57	84669.47
Maximum Response Time (ms)	**125.66**	427.07	170091.06

One-third the maximum response time

Case Study: Comparing Fine-Grained/ Coarse-Grained/Monolithic Scheduling

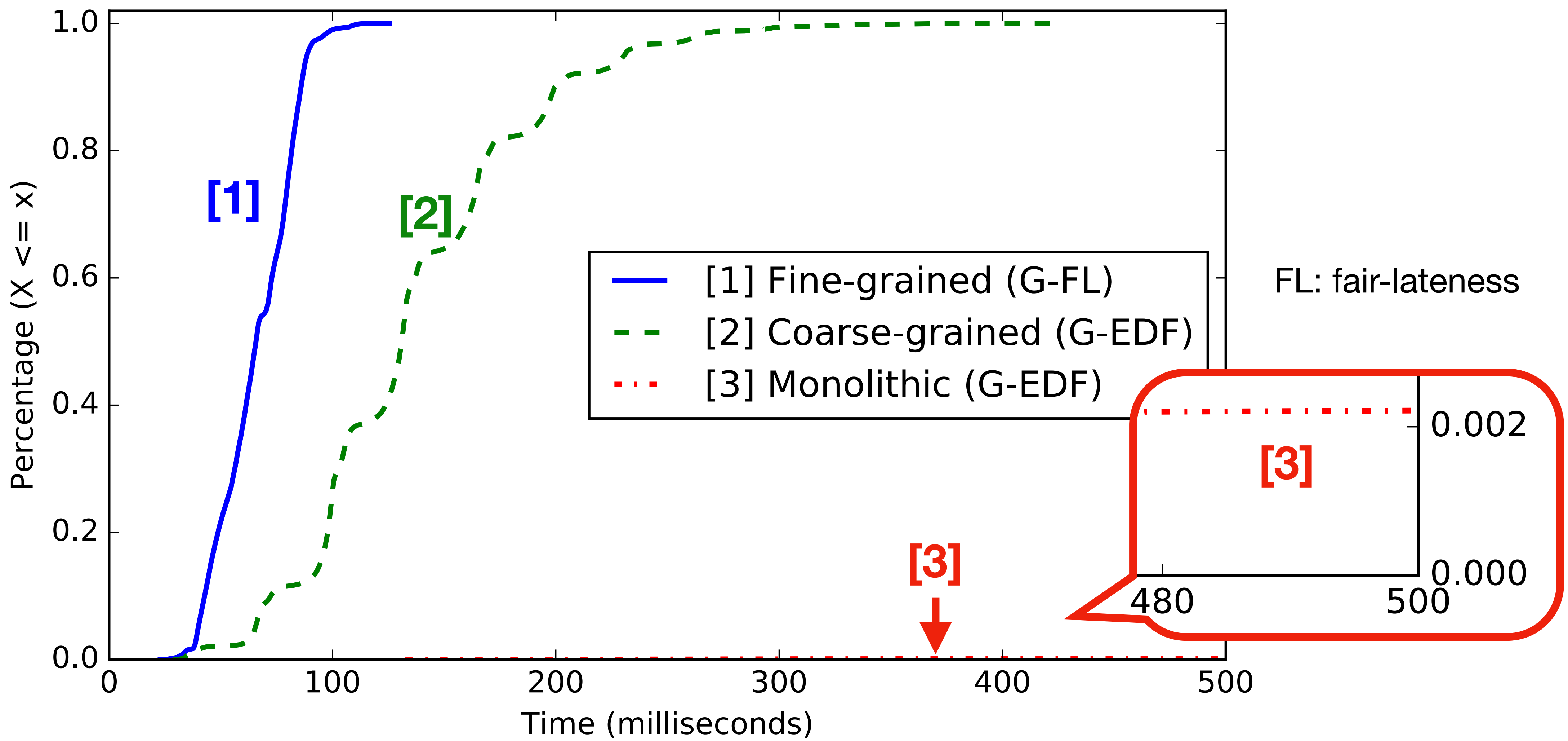


	[1] Fine-grained (G-FL)	[2] Coarse-grained (G-EDF)	[3] Monolithic (G-EDF)
Average Response Time (ms)	**65.99**	136.57	84669.47
Maximum Response Time (ms)	**125.66**	427.07	170091.06

Half the average response time

One-third the maximum response time

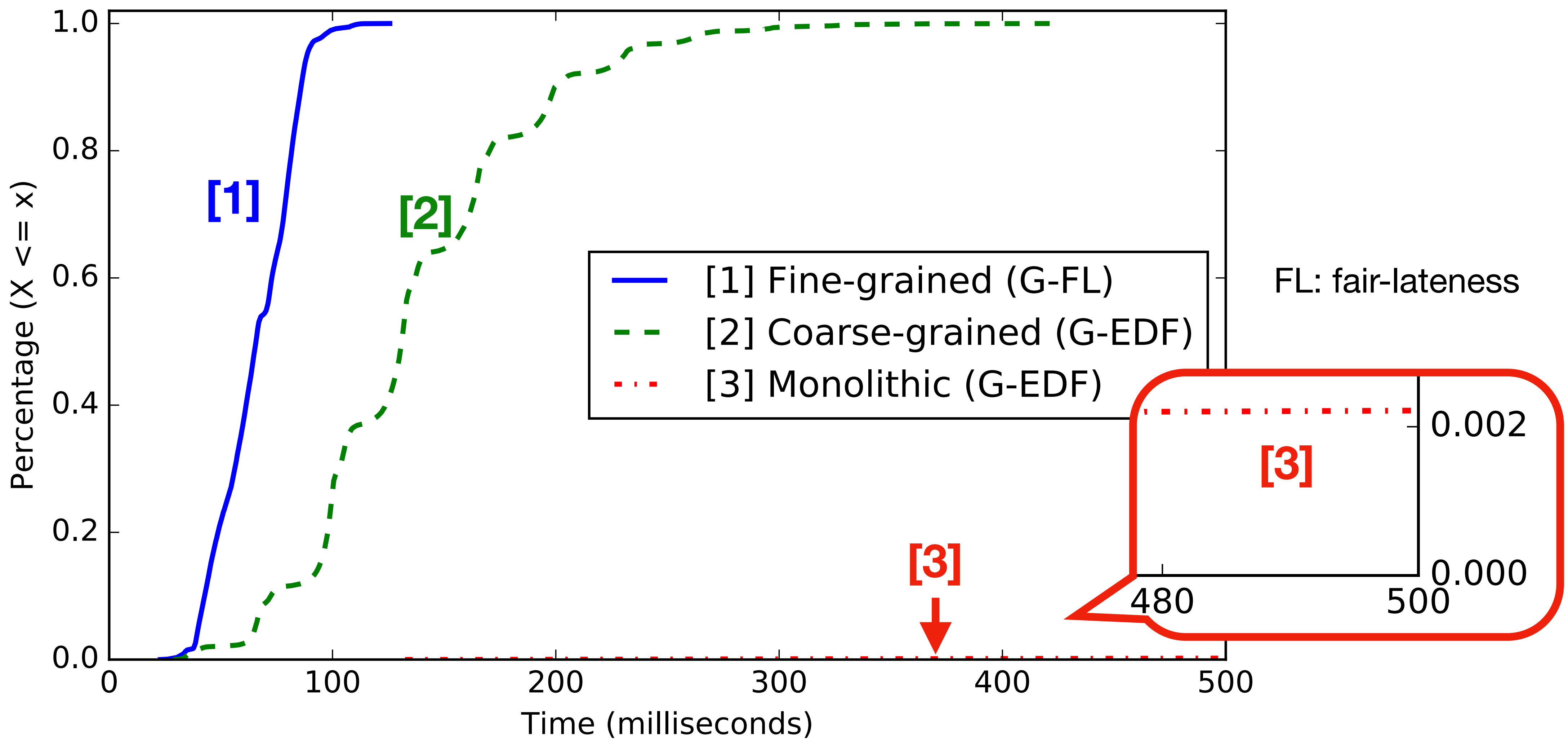
Case Study: Comparing Fine-Grained/ Coarse-Grained/Monolithic Scheduling



FL: fair-lateness

	[1] Fine-grained (G-FL)	[2] Coarse-grained (G-EDF)	[3] Monolithic (G-EDF)
Average Response Time (ms)	**65.99**	136.57	84669.47
Maximum Response Time (ms)	**125.66**	427.07	170091.06
Analytical Bound (ms)			N/A

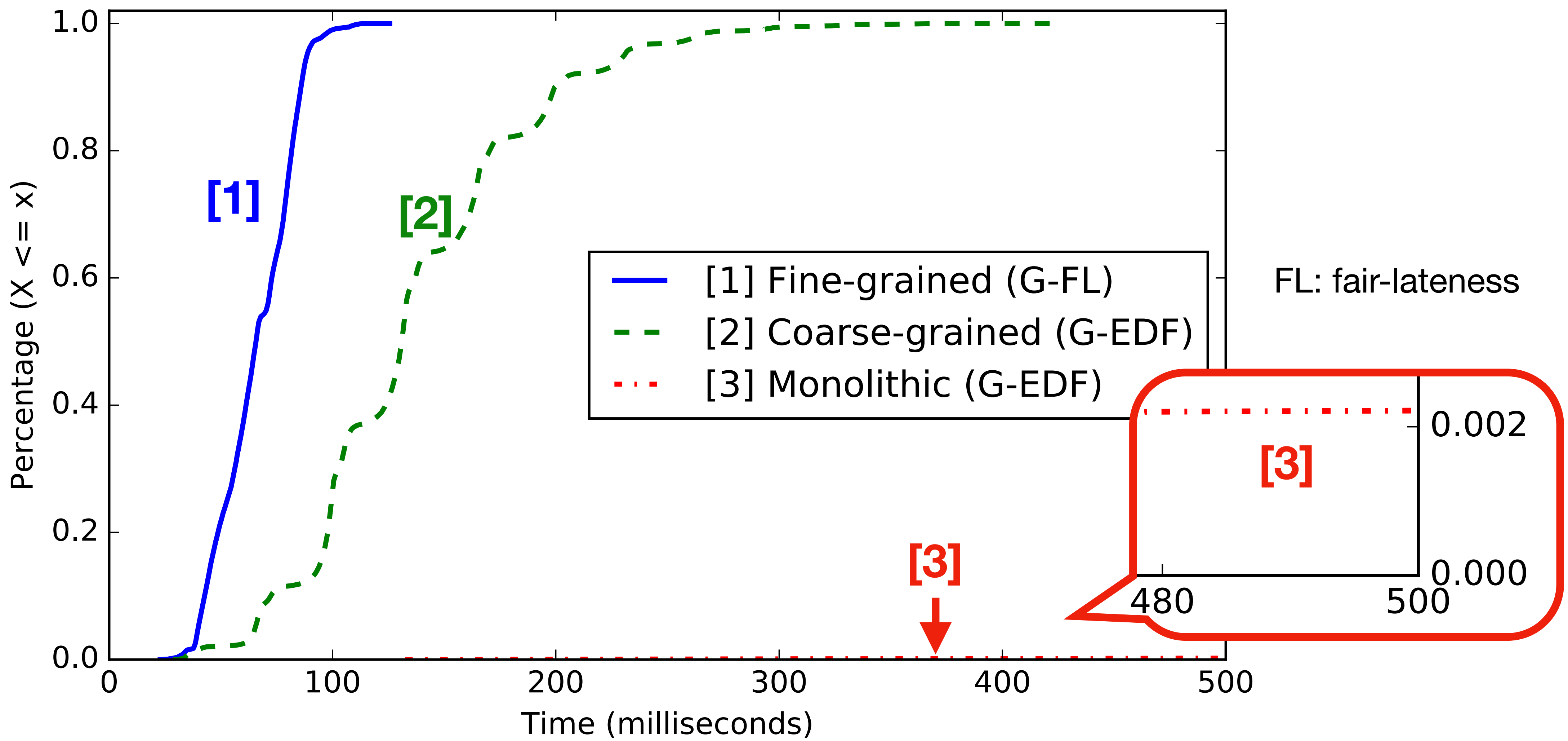
Case Study: Comparing Fine-Grained/ Coarse-Grained/Monolithic Scheduling



FL: fair-lateness

	[1] Fine-grained (G-FL)	[2] Coarse-grained (G-EDF)	[3] Monolithic (G-EDF)
Average Response Time (ms)	**65.99**	136.57	84669.47
Maximum Response Time (ms)	**125.66**	427.07	170091.06
Analytical Bound (ms)		N/A	N/A

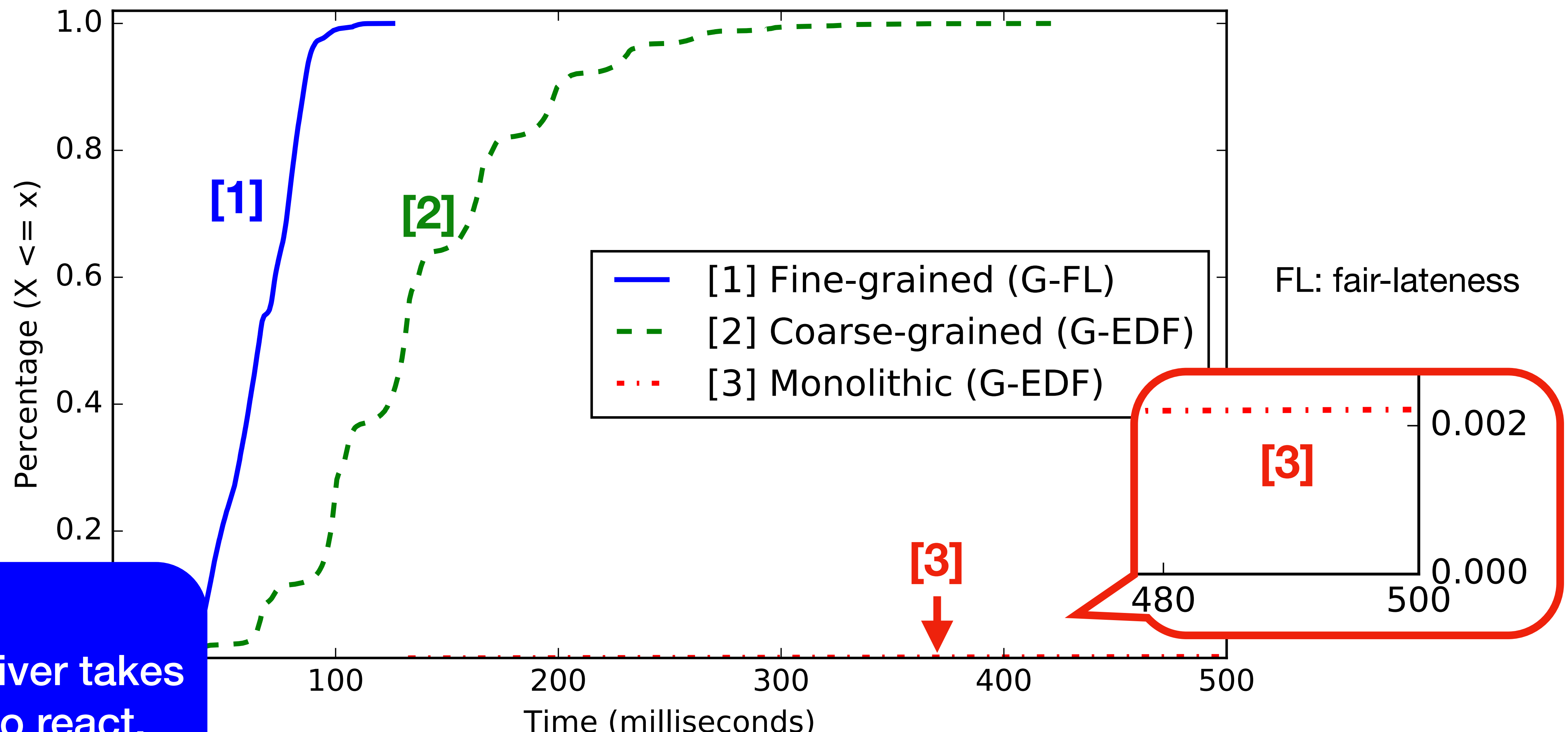
Case Study: Comparing Fine-Grained/Coarse-Grained/Monolithic Scheduling



FL: fair-lateness

	[1] Fine-grained (G-FL)	[2] Coarse-grained (G-EDF)	[3] Monolithic (G-EDF)
Average Response Time (ms)	**65.99**	136.57	84669.47
Maximum Response Time (ms)	**125.66**	427.07	170091.06
Analytical Bound (ms)	**542.39**	N/A	N/A

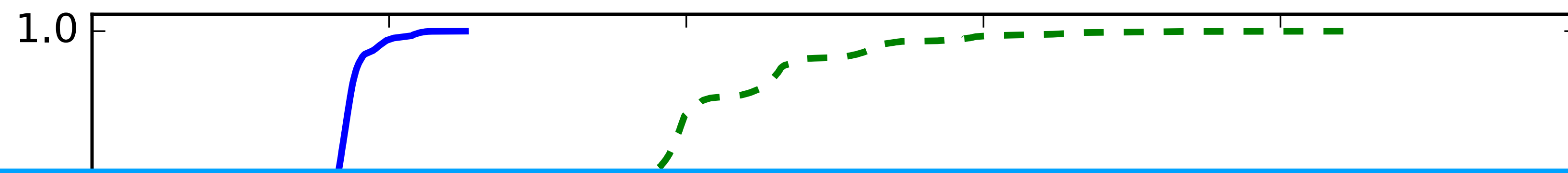
Case Study: Comparing Fine-Grained/ Coarse-Grained/Monolithic Scheduling



An alert driver takes 700 ms to react.

	[1] Fine-grained (G-FL)	[2] Coarse-grained (G-EDF)	[3] Monolithic (G-EDF)
Average Response Time (	**65.99**	136.57	84669.47
Maximum Response Time (ms)	**125.66**	427.07	170091.06
Analytical Bound (ms)	**542.39**	N/A	N/A

Case Study: Comparing Fine-Grained/ Coarse-Grained/Monolithic Scheduling



- Fair-lateness-based scheduler is beneficial as it reduced node response times by up to 9.9%.
- Overheads of supporting fine-grained scheduling was 14.15%.

			(G-EDF)
Average [illegible]			[illegible].47
Maximum Response Time (ms)	125.66	427.07	170091.06
Analytical Bound (ms)	542.39	N/A	N/A

Conclusions

1. Fine-grained scheduling
2. Response-time bounds analysis for GPU tasks
3. Case study

Future Work

1. Cycles in the graph

2. Other resource constraints

3. Schedulability studies

Thanks!