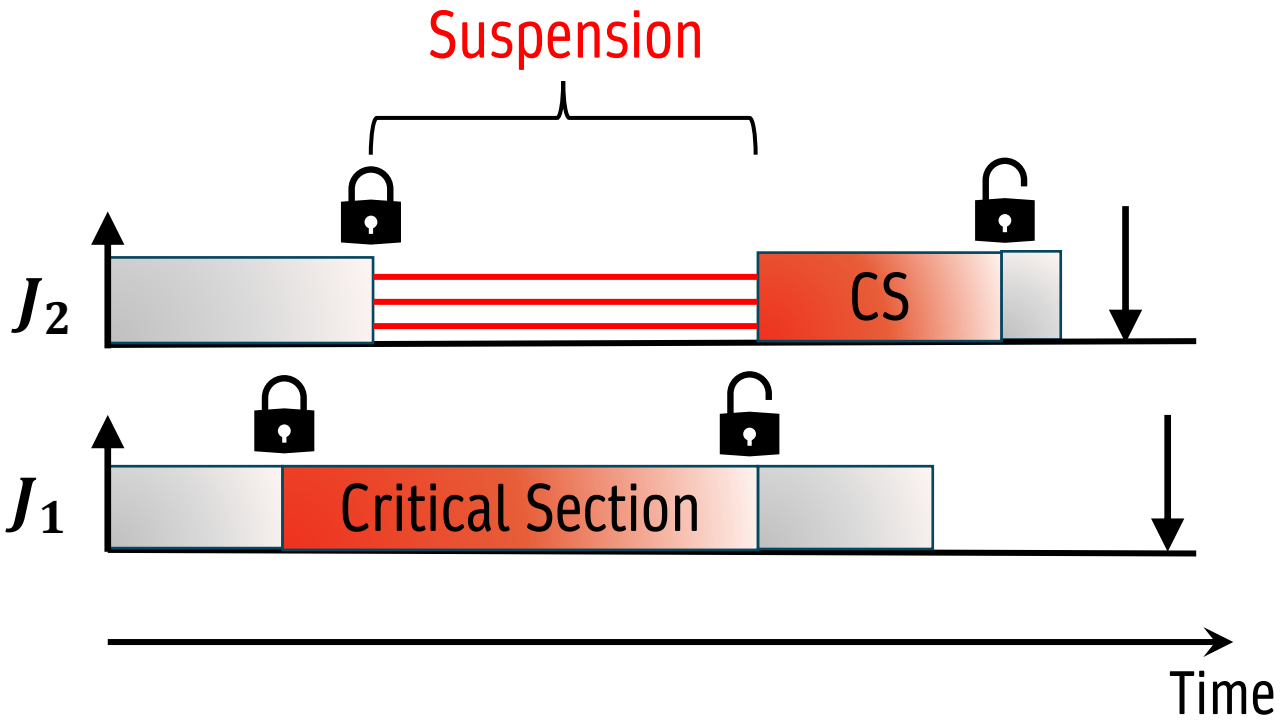
Open Problem Resolved: The "Two" in Existing Multiprocessor PI-Blocking Bounds is Fundamental

Shareef Ahmed and James H. Anderson

Background

Mutex locks

```
pthread_mutex_lock(…)
//critical section
pthread_mutex_unlock(…)
```

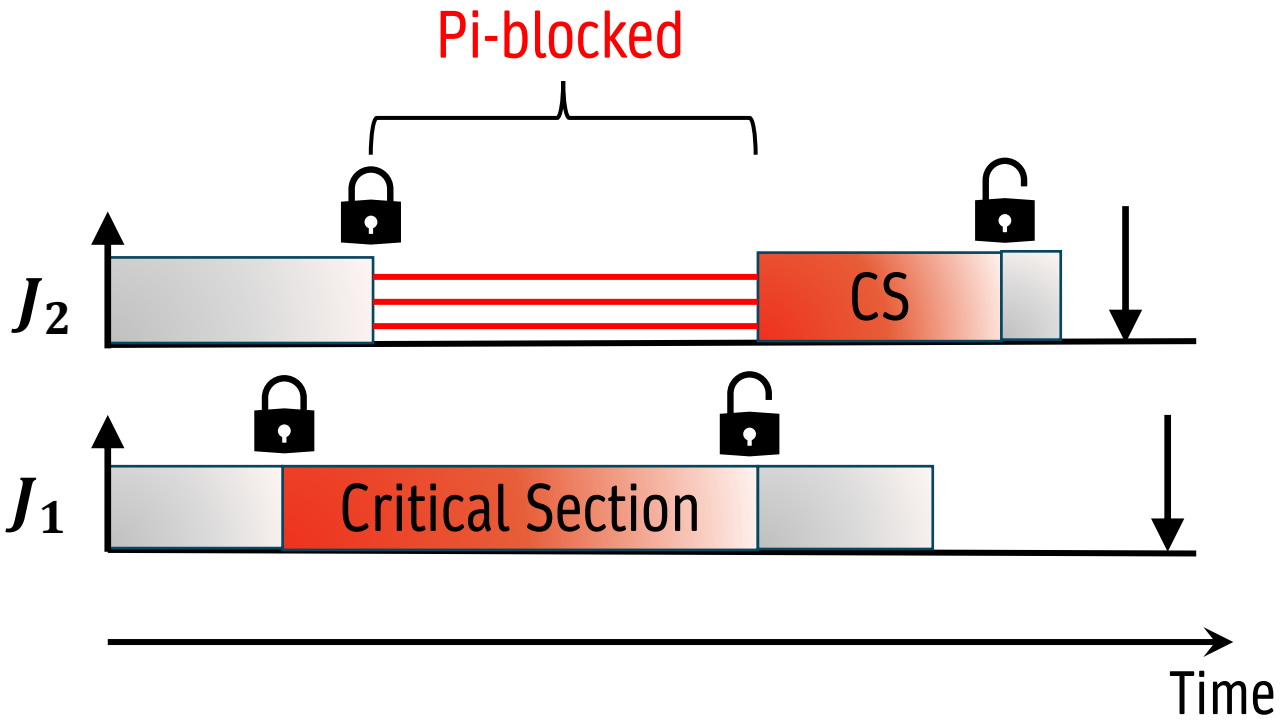


A locking protocol determines ordering of CSs

Background

Pi-blocking

A job is NOT scheduled when it should be

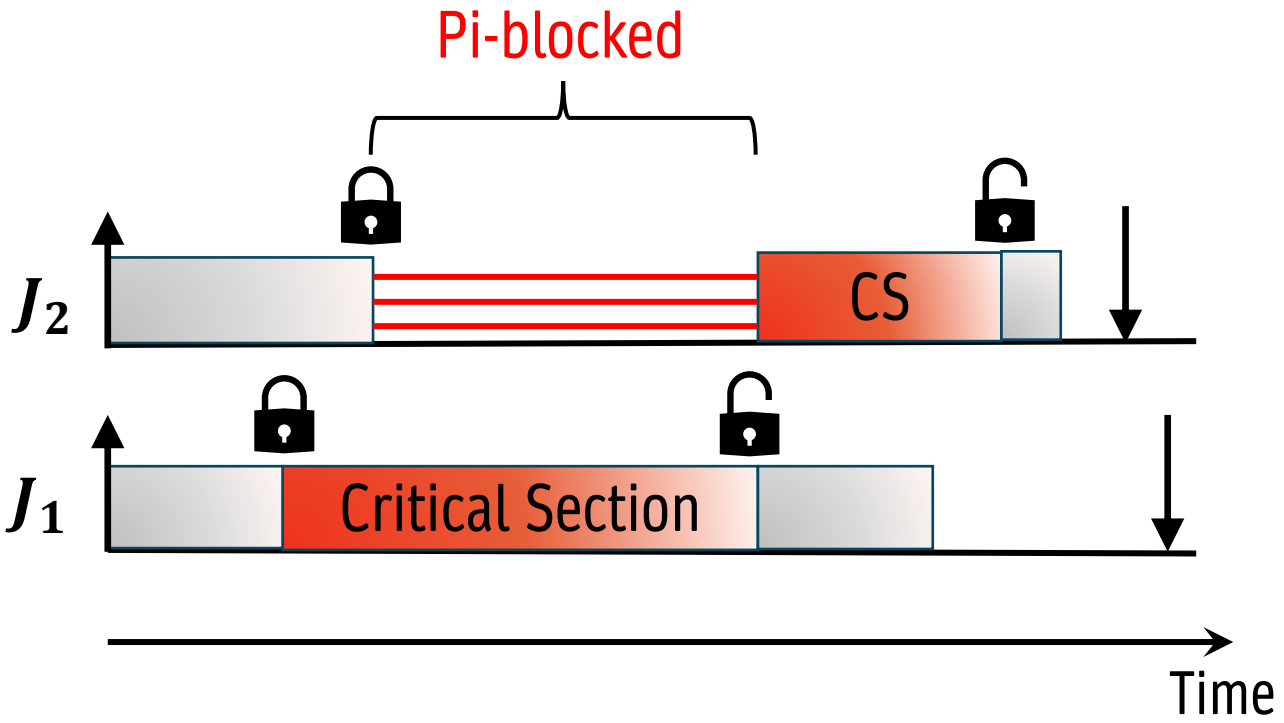


On a 2-processor platform, J_2 is pi-blocked when it is suspended

Background

Pi-blocking

A job is NOT scheduled when it should be



Real-time locking protocol goal: minimize the maximum pi-blocking

Background

Focus of this work

Pi-blocking

Suspension-oblivious

- Schedulability analysis assumes no suspension

Suspension-aware

- Schedulability analysis models suspension times explicitly

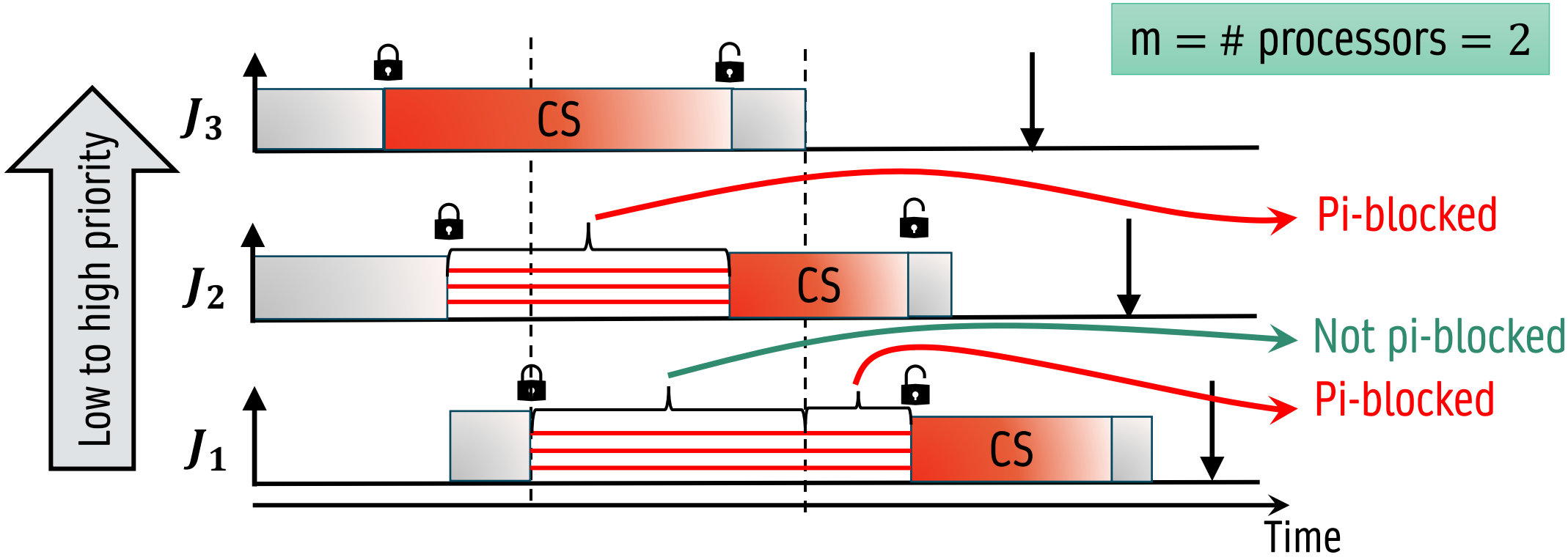
This paper:

- Global scheduling
- One shared resource and one CS per job

Background

Suspension-oblivious pi-blocking

A job is pi-blocked if it is one of the top-m-priority pending jobs, but NOT scheduled [Brandenburg & Anderson, 2010]



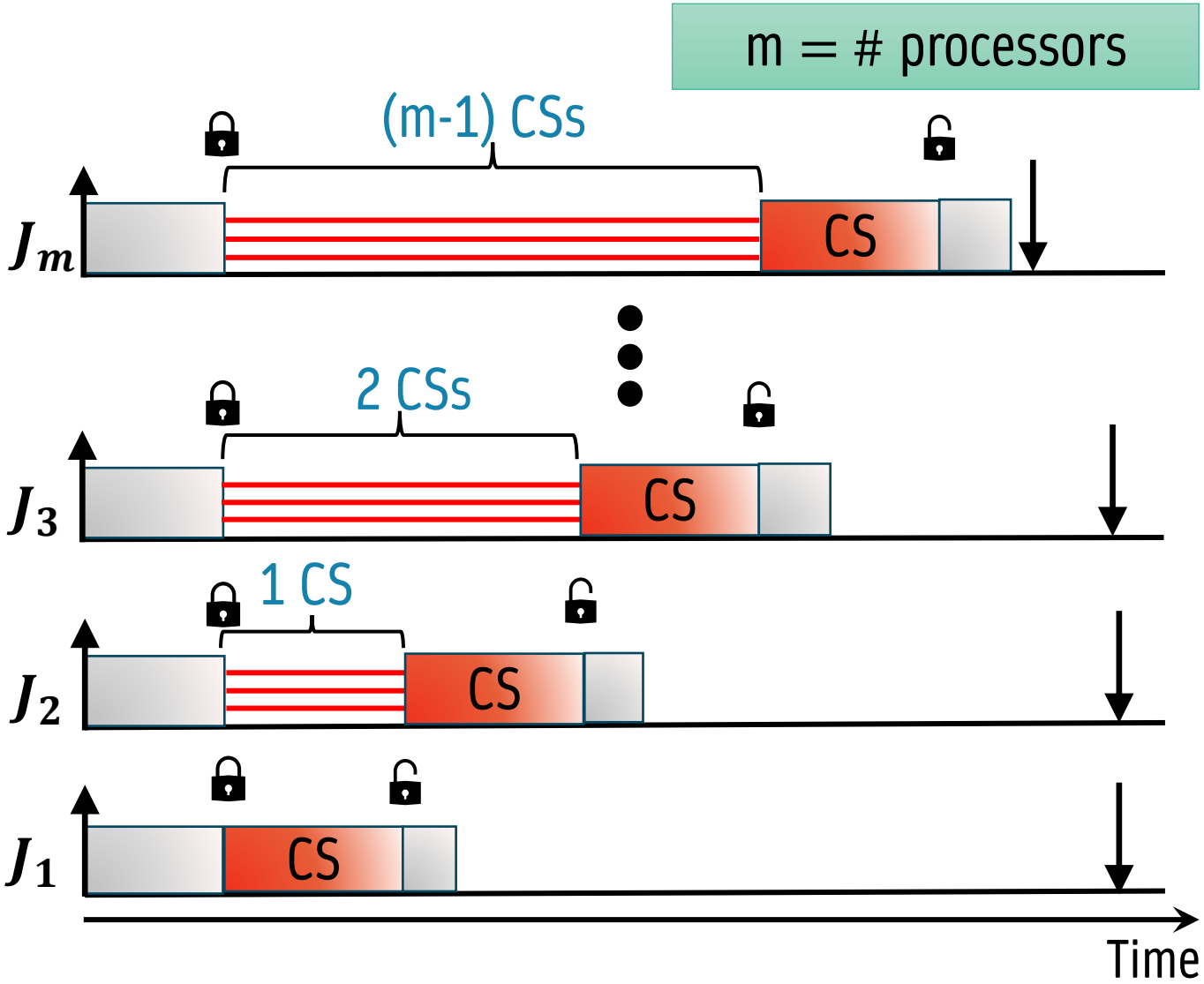
Prior Work

Real-time locking protocol goal: minimize the maximum pi-blocking

Prior Work: Lower Bound

S-oblivious pi-blocking lower bound (under any JLFP scheduler)

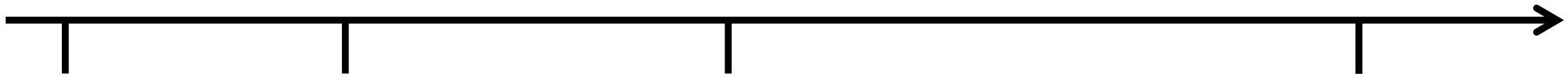
Max per-job pi-blocking ≥ (m-1) CS lengths [Brandenburg & Anderson, 2010]



Prior Work: Upper Bound

m = # processors

Protocol	G-OMLP	C-OMLP	OMIP	C-OLPF
Scheduling	Global JLFP	Clustered JLFP	Clustered JLFP	Clustered FIFO
Max per-job pi-blocking (one CS per task)	(2m-1) CS lengths	(2m-1) CS lengths	(2m-1) CS lengths	(m-1) CS lengths



2010 [Brandenburg & Anderson]

2011 [Brandenburg & Anderson]

2013 [Brandenburg]

2023 [Ahmed & Anderson]

Prior Work

m = # processors

FIFO scheduling

Pi-blocking L.B.	Pi-blocking U.B.
(m-1) CS lengths	(m-1) CS lengths

An optimal locking protocol (C-OLPF) is known

Non-FIFO scheduling

Pi-blocking L.B.	Pi-blocking U.B.
(m-1) CS lengths	(2m-1) CS lengths

Asymptotically optimal locking protocols (G-OMLP, C-OMLP, OMIP) are known, all with the same (2m-1) CS length pi-blocking bound

Focus of this paper

This Paper

Contribution 1

m = # processors

G-EDF, G-FP scheduling

Pi-blocking L.B.	Pi-blocking U.B.
(2m-2) CS lengths	(2m-1) CS lengths

∃ system where a job's pi-blocking ≥ (2m-2) CS lengths under any locking protocol

The OMLP, C-OMLP, OMIP are **almost optimal** under G-EDF, G-FP

This Paper

Contribution 2

m = # processors

G-EDF, G-FP scheduling

∃ system where a job's pi-blocking ≥ (2m-1) CS lengths - ϵ under any **reorder-bounded** locking protocol

- A **non reorder-bounded** protocol can be overhead-heavy
- Known protocols are reorder-bounded

Rest of The Talk

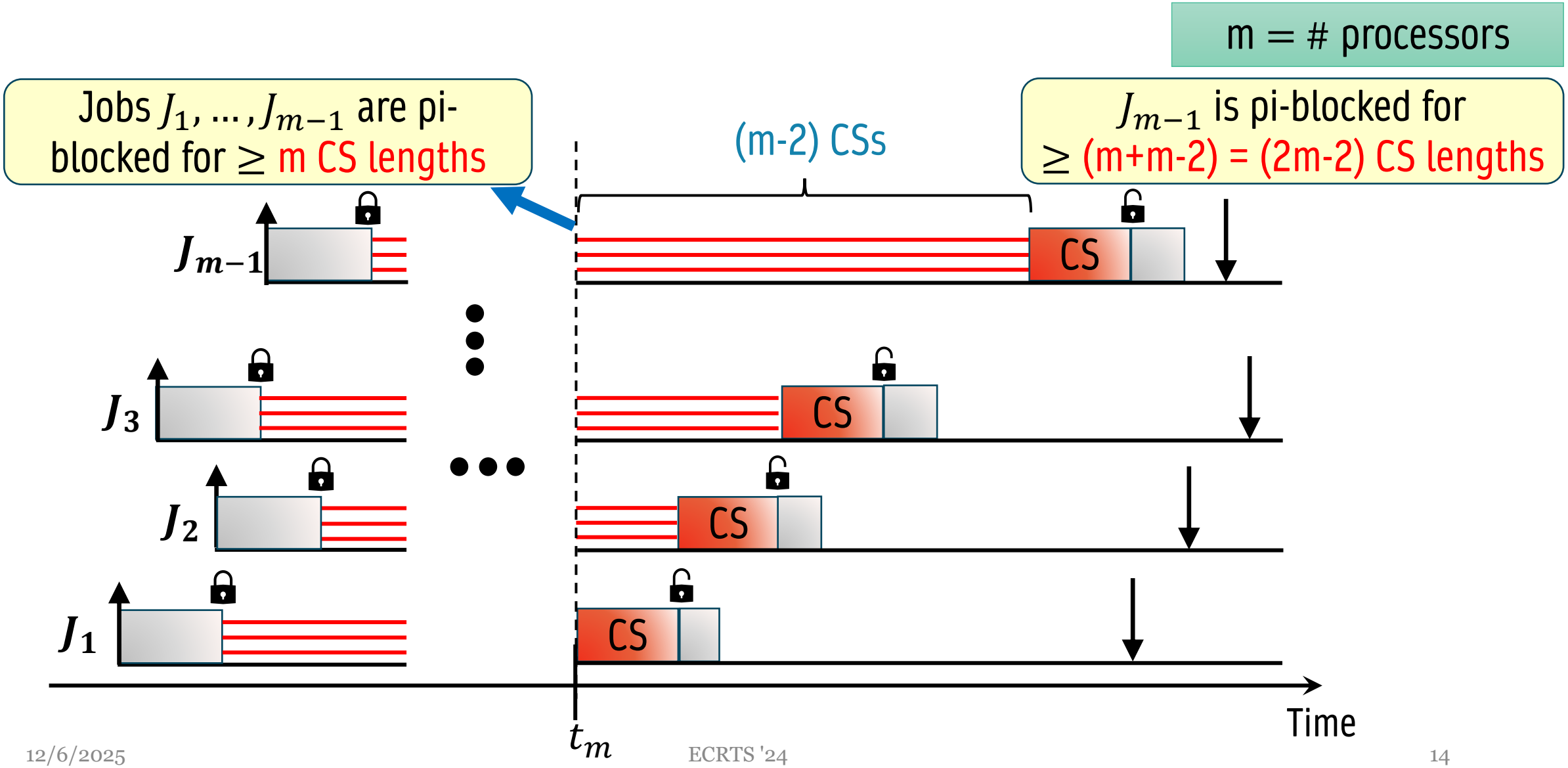
Contribution 1

m = # processors

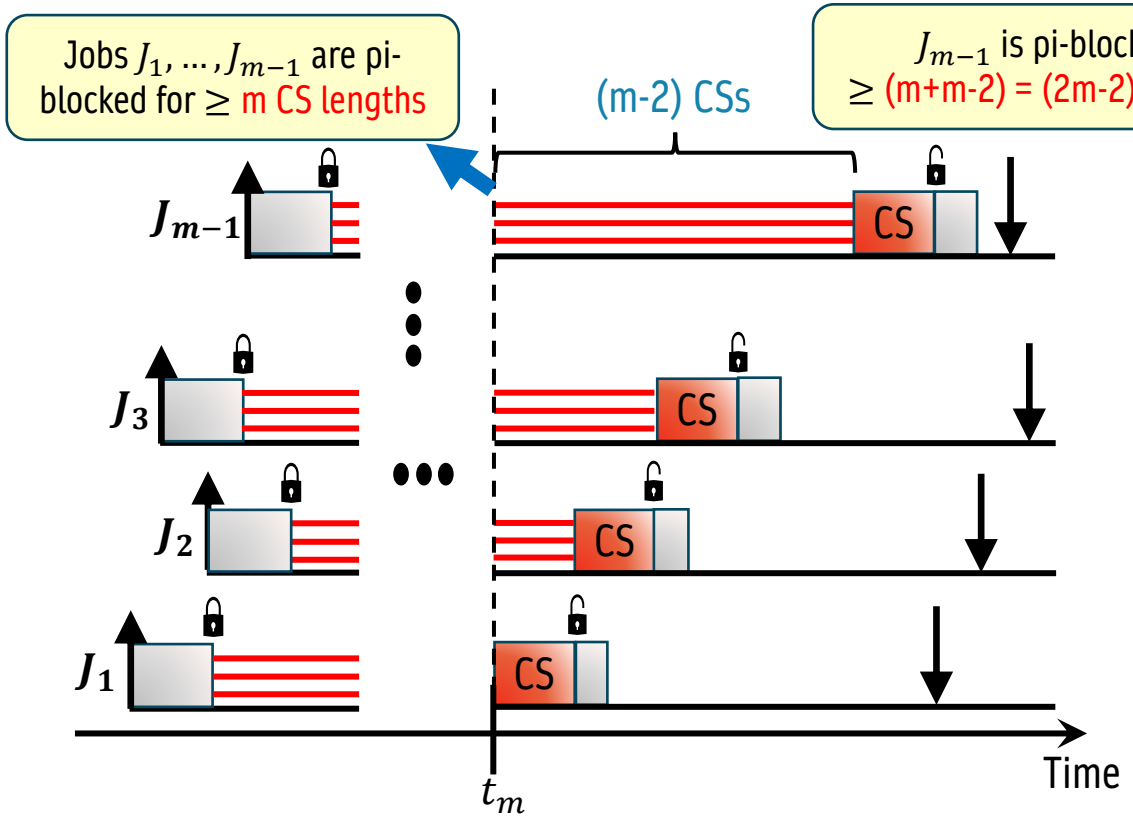
G-EDF, G-FP scheduling

∃ system where a job's pi-blocking ≥ (2m-2) CS lengths under any locking protocol

Lower Bound: Key Idea



Lower Bound: Key Idea



m = # processors

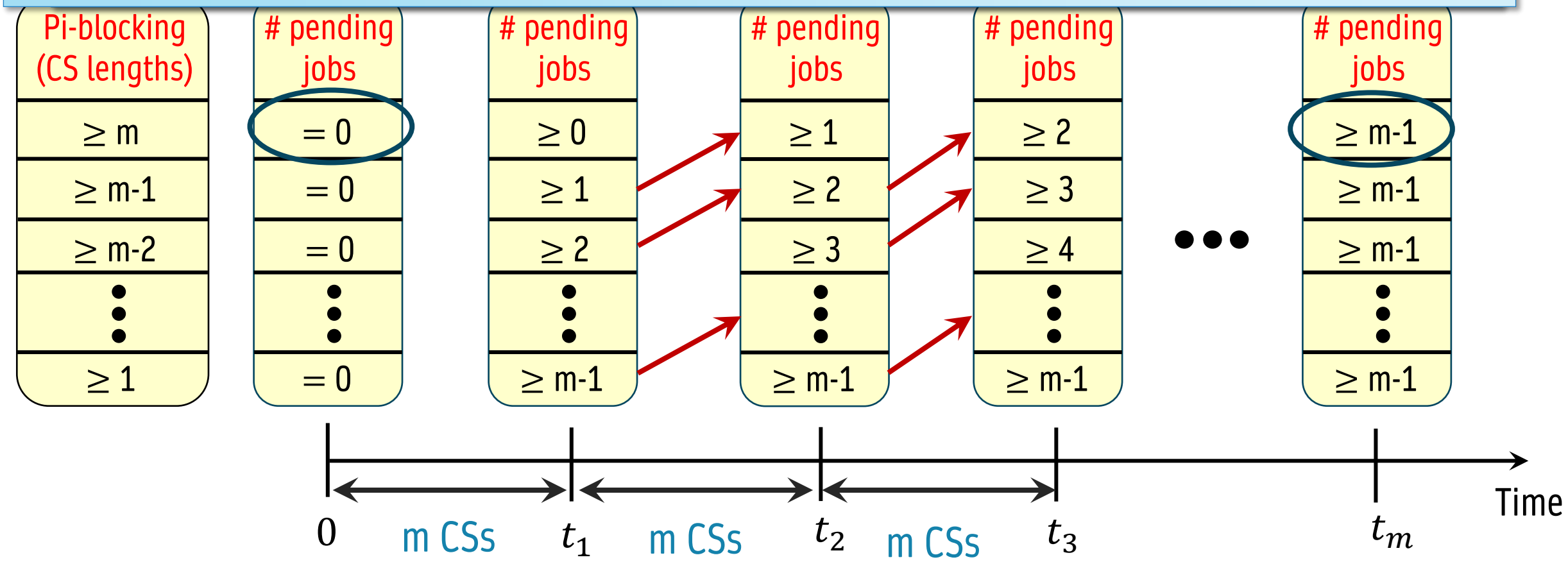
Goal

There exists a job release pattern causing (m-1) pending jobs with ≥ m CS lengths pi-blocking

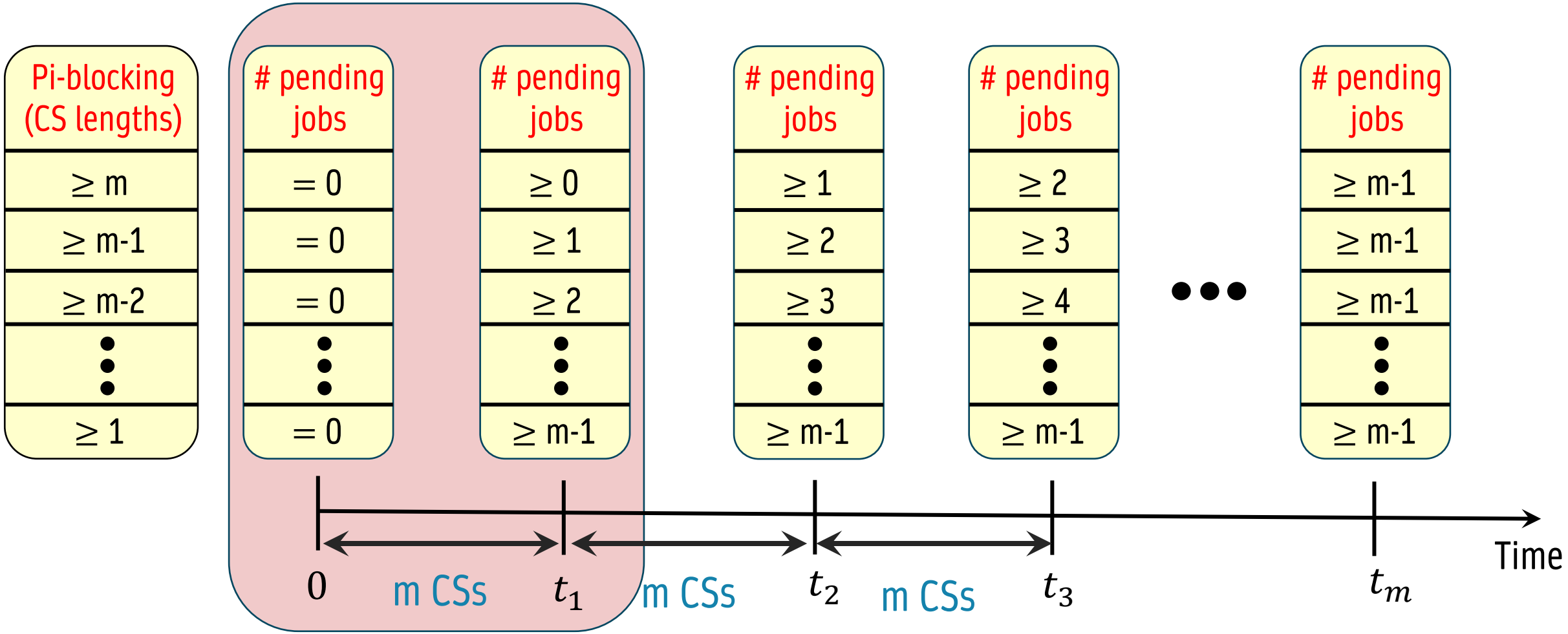
A job suffers pi-blocking for ≥ (2m-2) CS lengths

Lower Bound: Key Idea

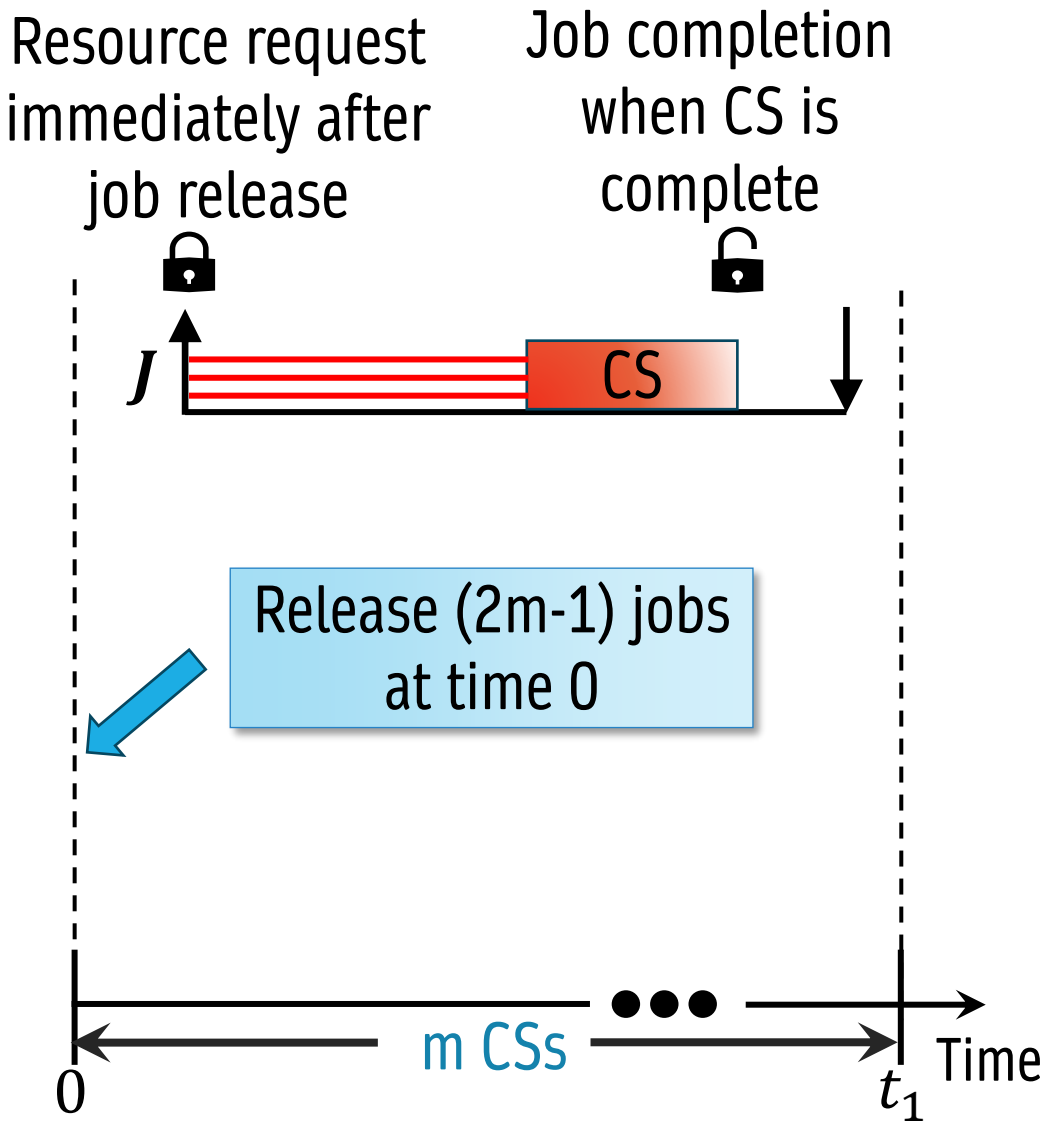
There are **at least** m-1 pending jobs with ≥ m CS lengths of pi-blocking at time t_m



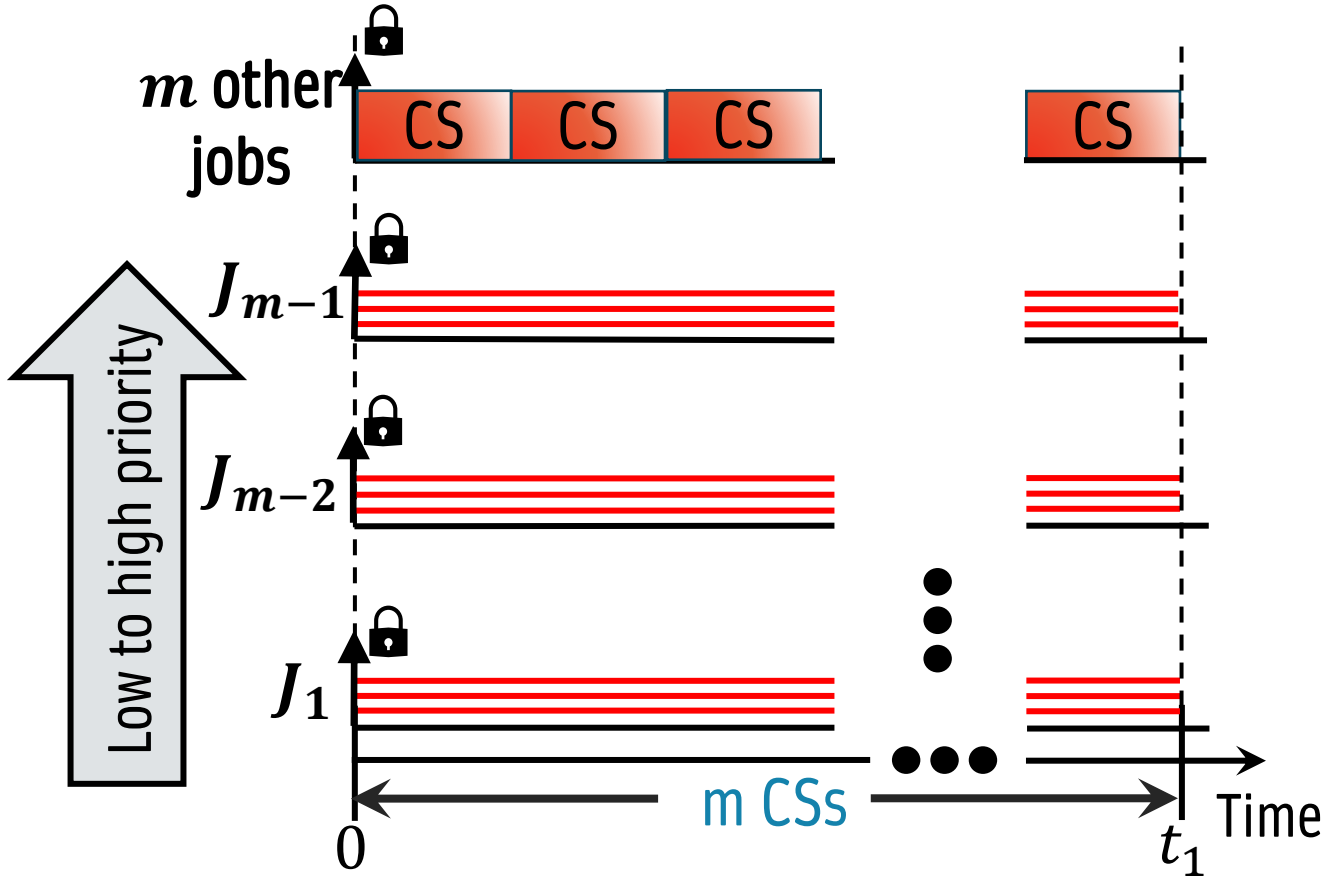
Lower Bound: Execution Scenarios



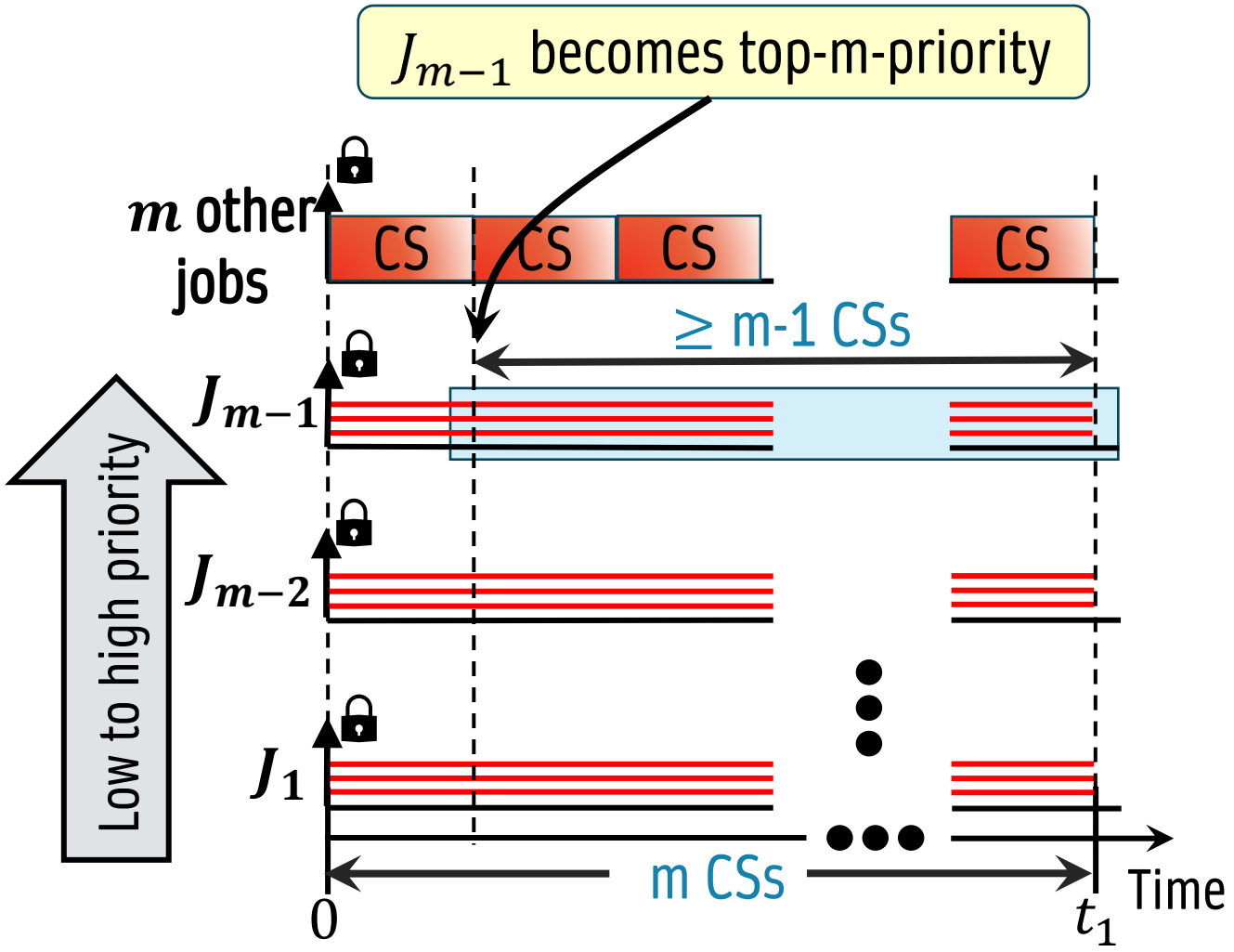
Lower Bound: Execution Scenarios



Lower Bound: Execution Scenarios

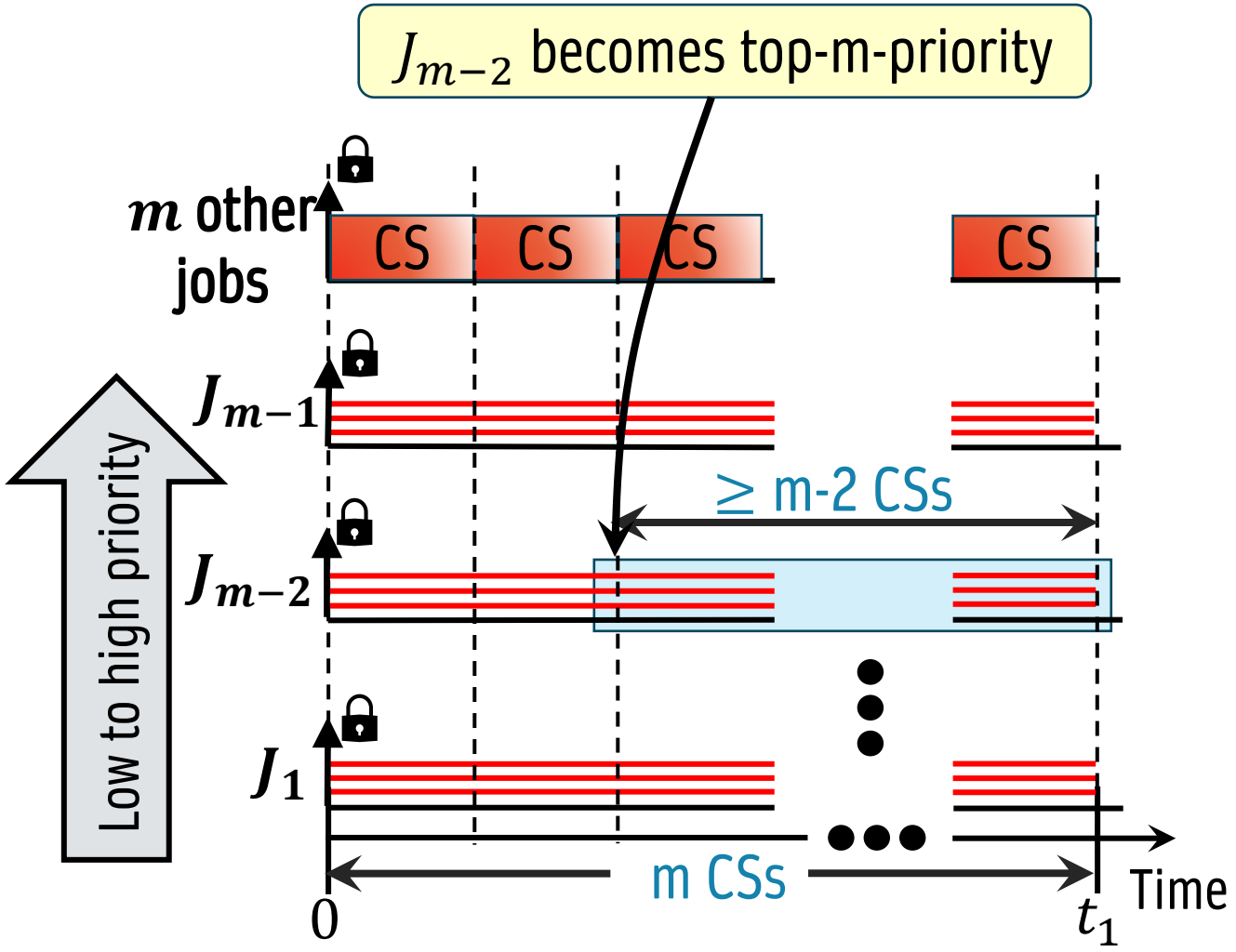


Lower Bound: Execution Scenarios



J_{m-1} is pi-blocked for ≥ m-1 CS lengths

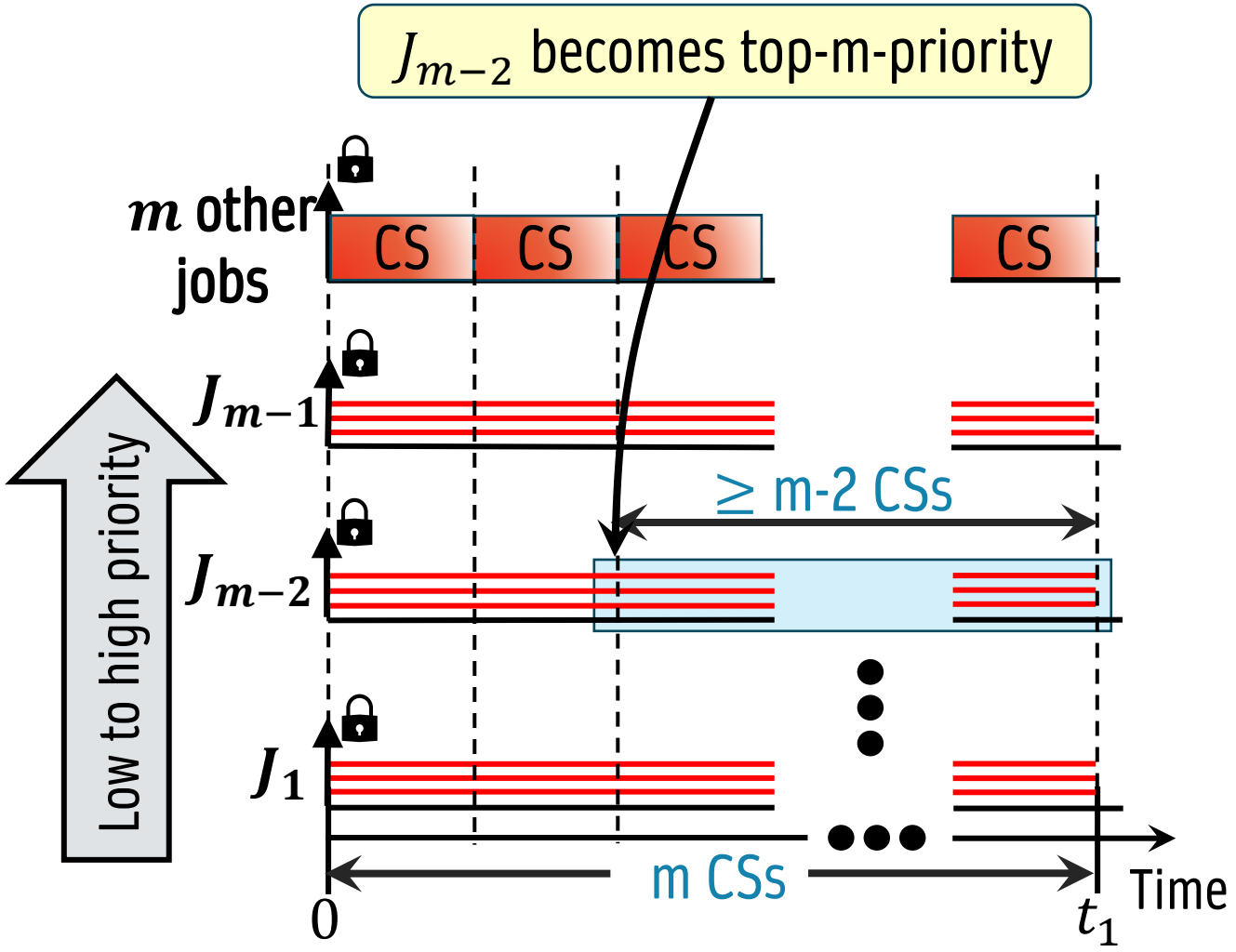
Lower Bound: Execution Scenarios



J_{m-1} is pi-blocked for ≥ m-1 CS lengths

J_{m-2} is pi-blocked for ≥ m-2 CS lengths

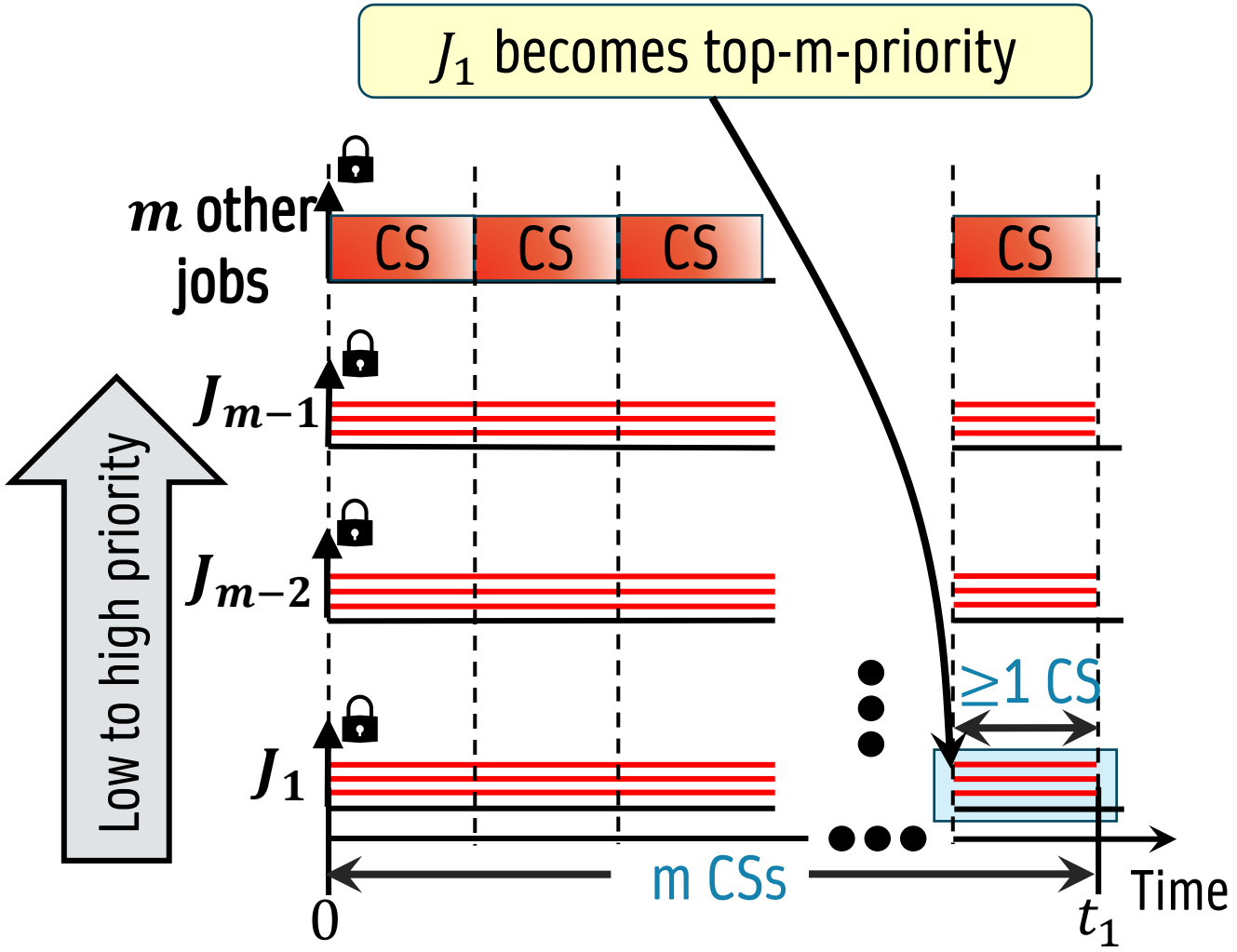
Lower Bound: Execution Scenarios



J_{m-1} is pi-blocked for ≥ m-1 CS lengths

J_{m-2} is pi-blocked for ≥ m-2 CS lengths

Lower Bound: Execution Scenarios

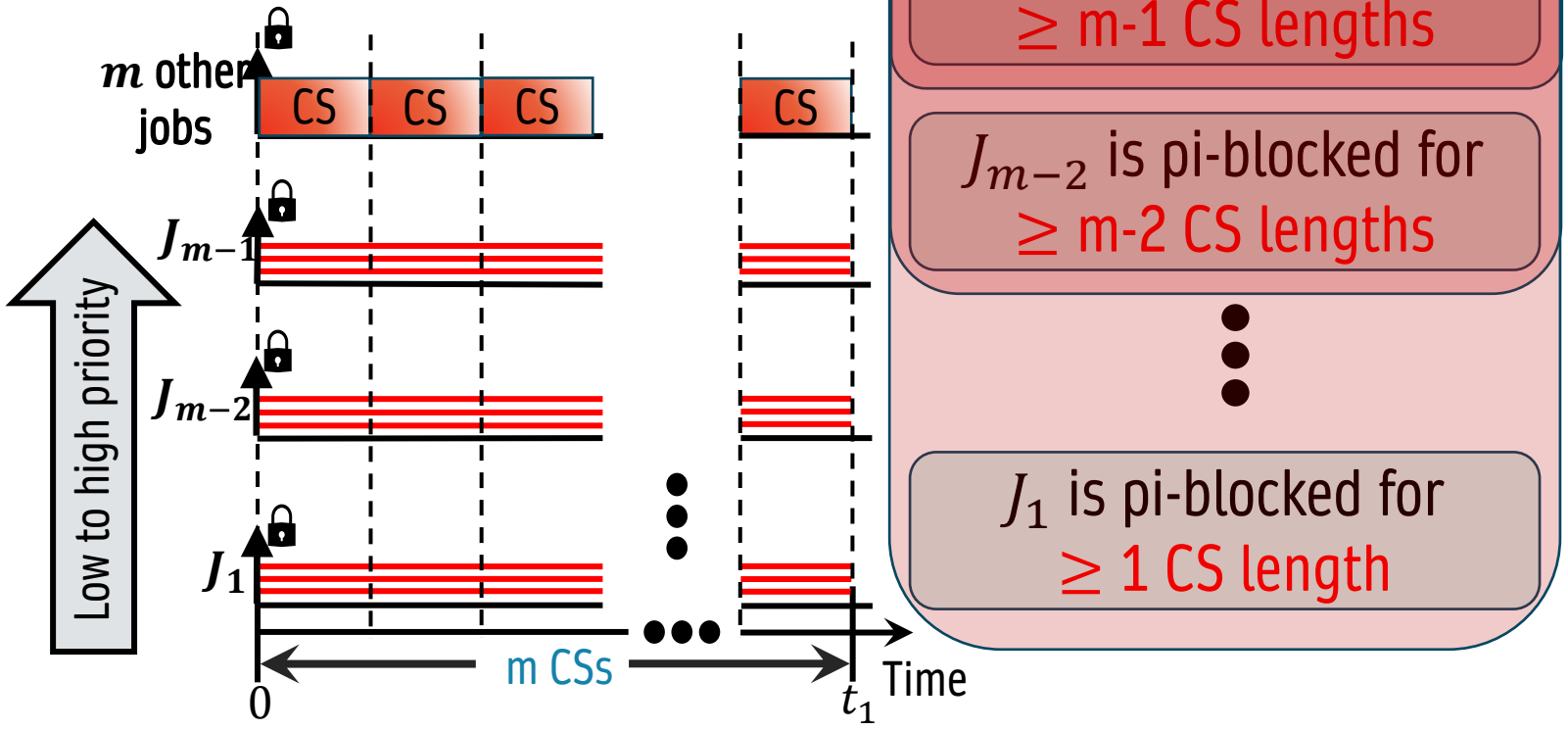


J_{m-1} is pi-blocked for ≥ m-1 CS lengths

J_{m-2} is pi-blocked for ≥ m-2 CS lengths

J_1 is pi-blocked for ≥ 1 CS length

Lower Bound: Execution Scenarios



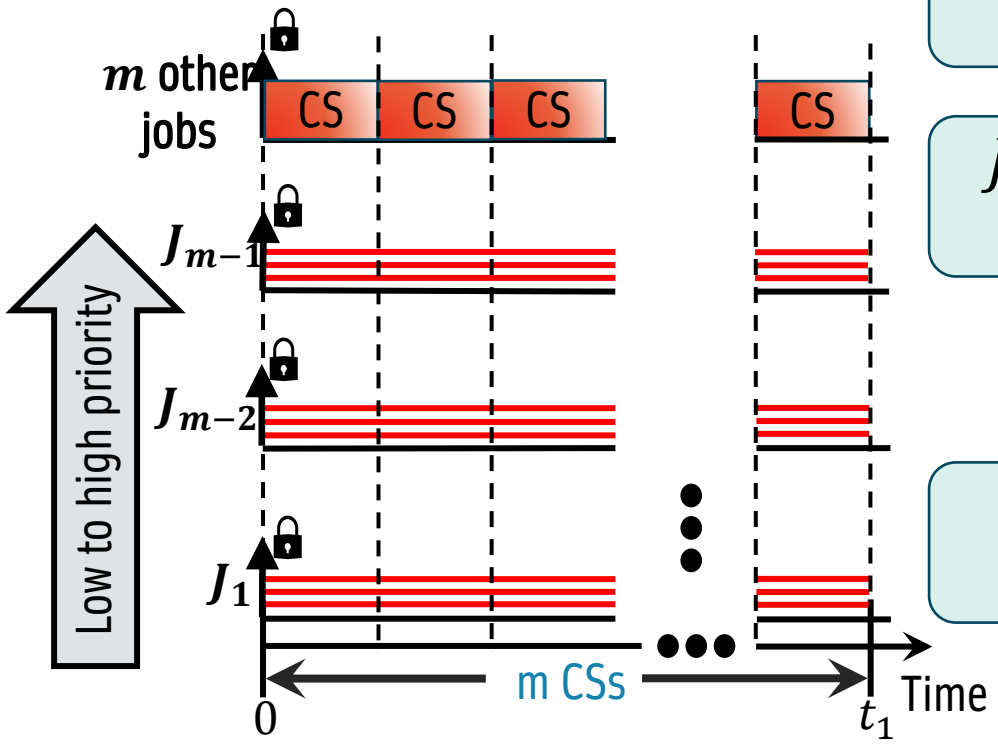
J_{m-1} is pi-blocked for ≥ m-1 CS lengths

J_{m-2} is pi-blocked for ≥ m-2 CS lengths

J_1 is pi-blocked for ≥ 1 CS length

Pi-blocking (CS lengths)	# pending jobs	Jobs
≥ m		
≥ m-1		J_{m-1}
≥ m-2		J_{m-1}, J_{m-2}
⋮	⋮	⋮
≥ 1		$J_{m-1}, J_{m-2}, \ldots, J_1$

Lower Bound: Execution Scenarios



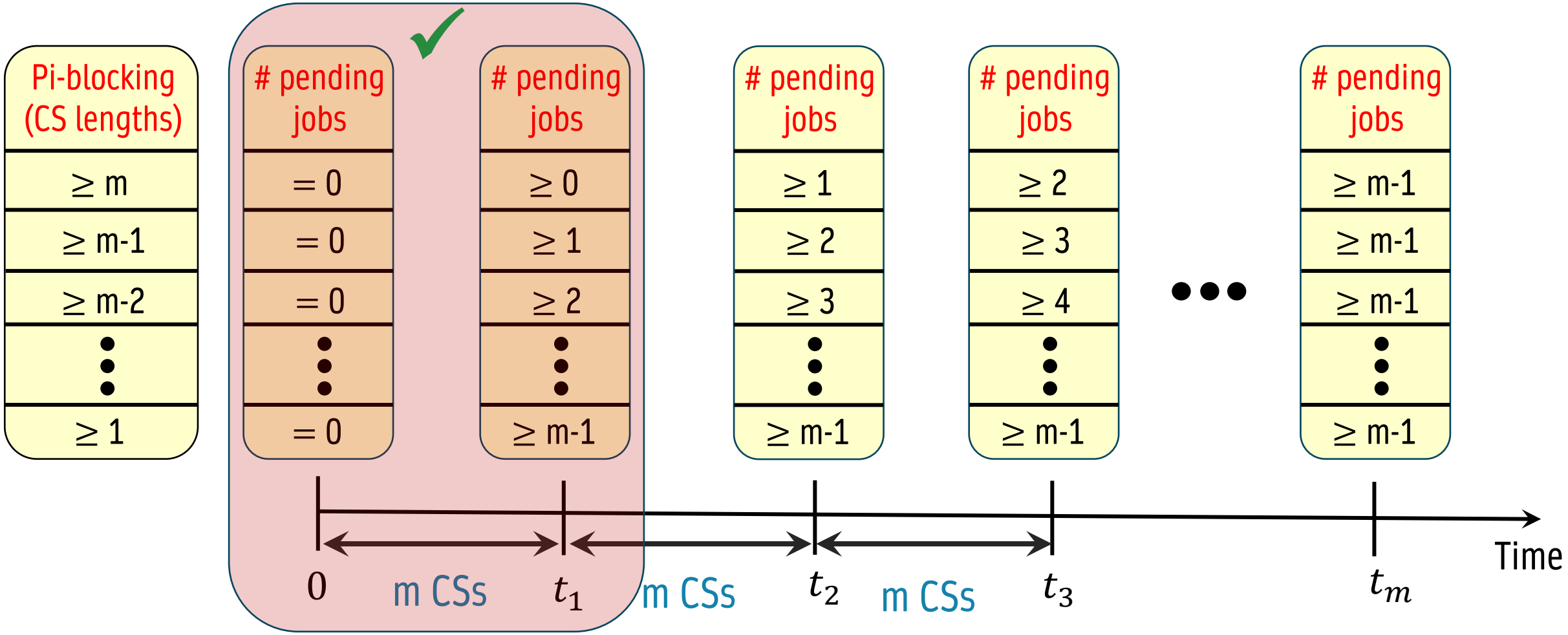
J_{m-1} is pi-blocked for ≥ m-1 CS lengths

J_{m-2} is pi-blocked for ≥ m-2 CS lengths

J_1 is pi-blocked for ≥ 1 CS length

Pi-blocking (CS lengths)	# pending jobs	Jobs
≥ m	≥ 0	
≥ m-1	≥ 1	J_{m-1}
≥ m-2	≥ 2	J_{m-1}, J_{m-2}
⋮	⋮	⋮
≥ 1	≥ m-1	$J_{m-1}, J_{m-2}, \ldots, J_1$

Lower Bound: Key Idea



Lower Bound: Execution Scenarios

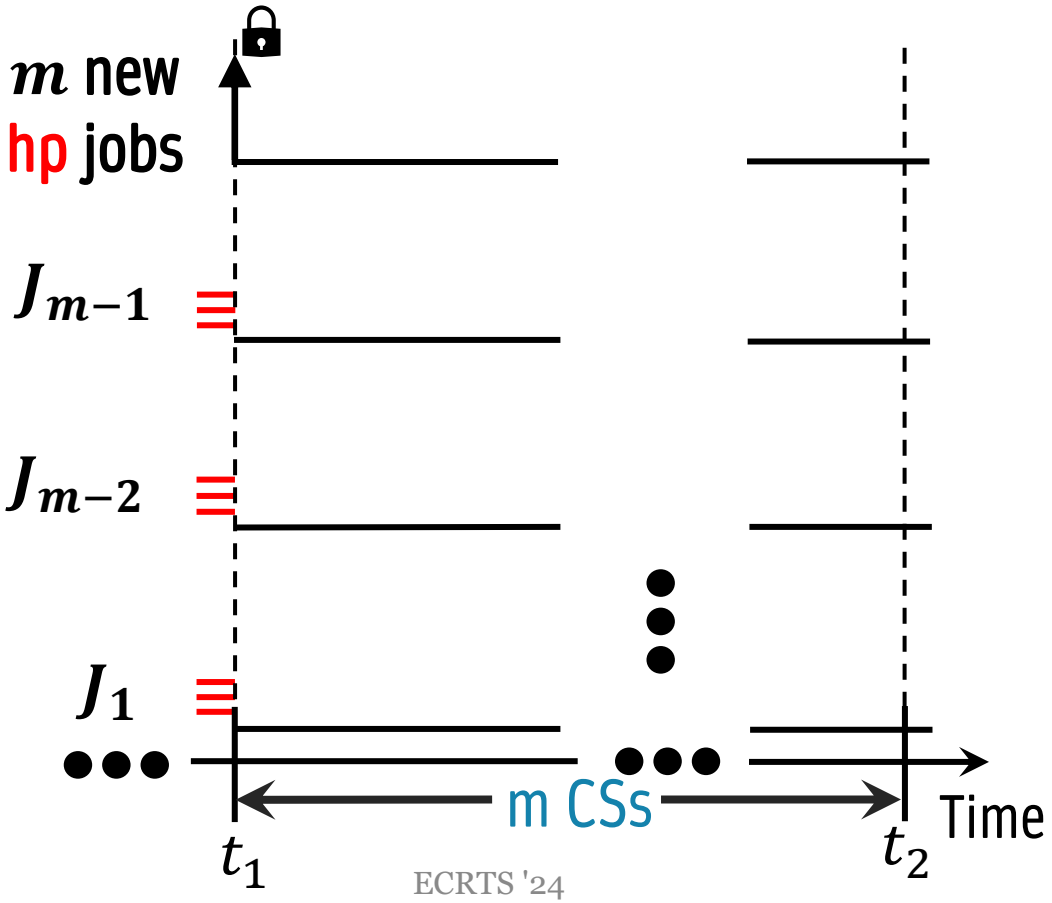
Case 1: all CSs of new jobs execute in $[t_1, t_2)$

Case 2: some CSs of old jobs execute in $[t_1, t_2)$

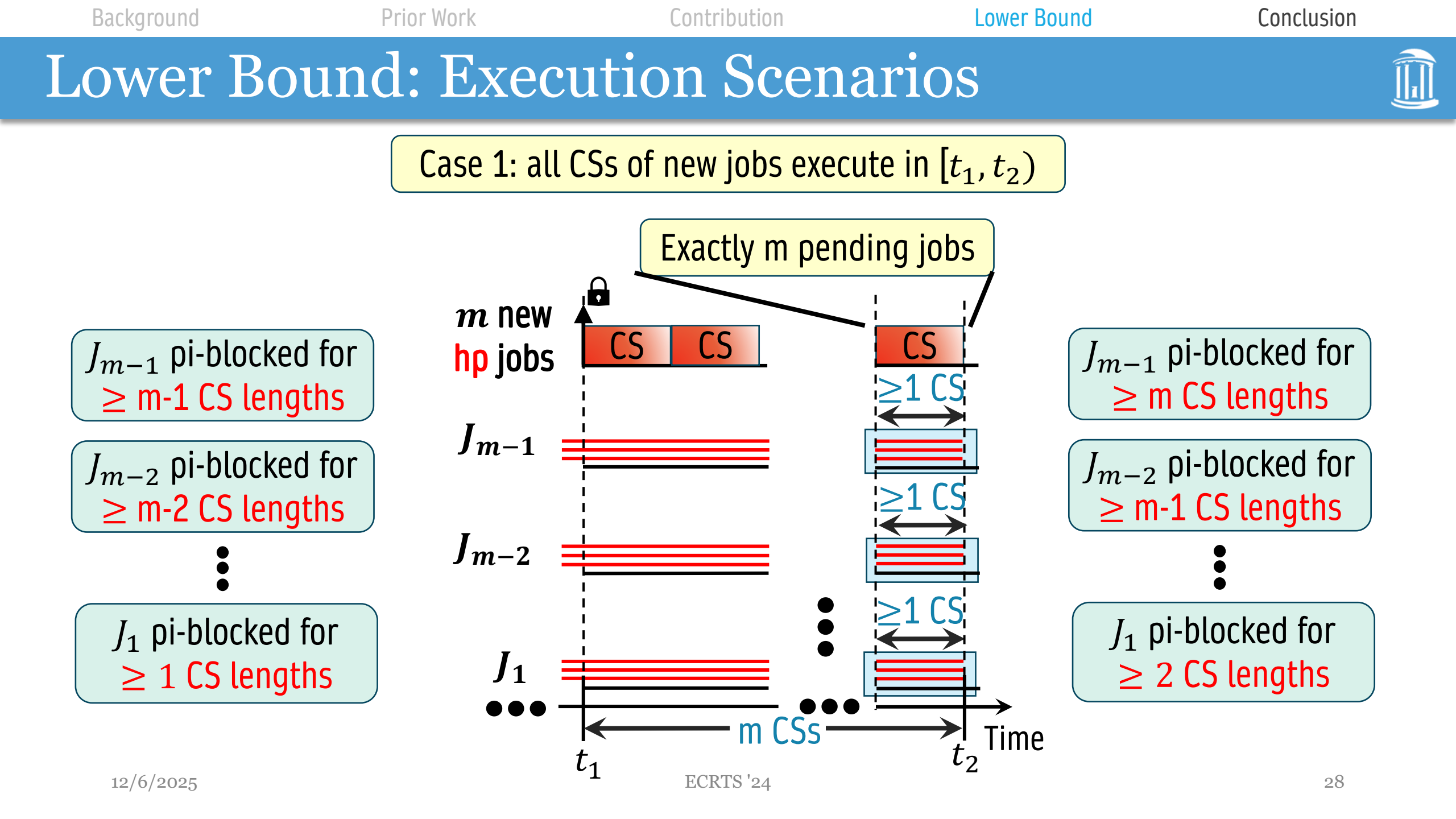
J_{m-1} pi-blocked for ≥ m-1 CS lengths

J_{m-2} pi-blocked for ≥ m-2 CS lengths

J_1 pi-blocked for ≥ 1 CS lengths

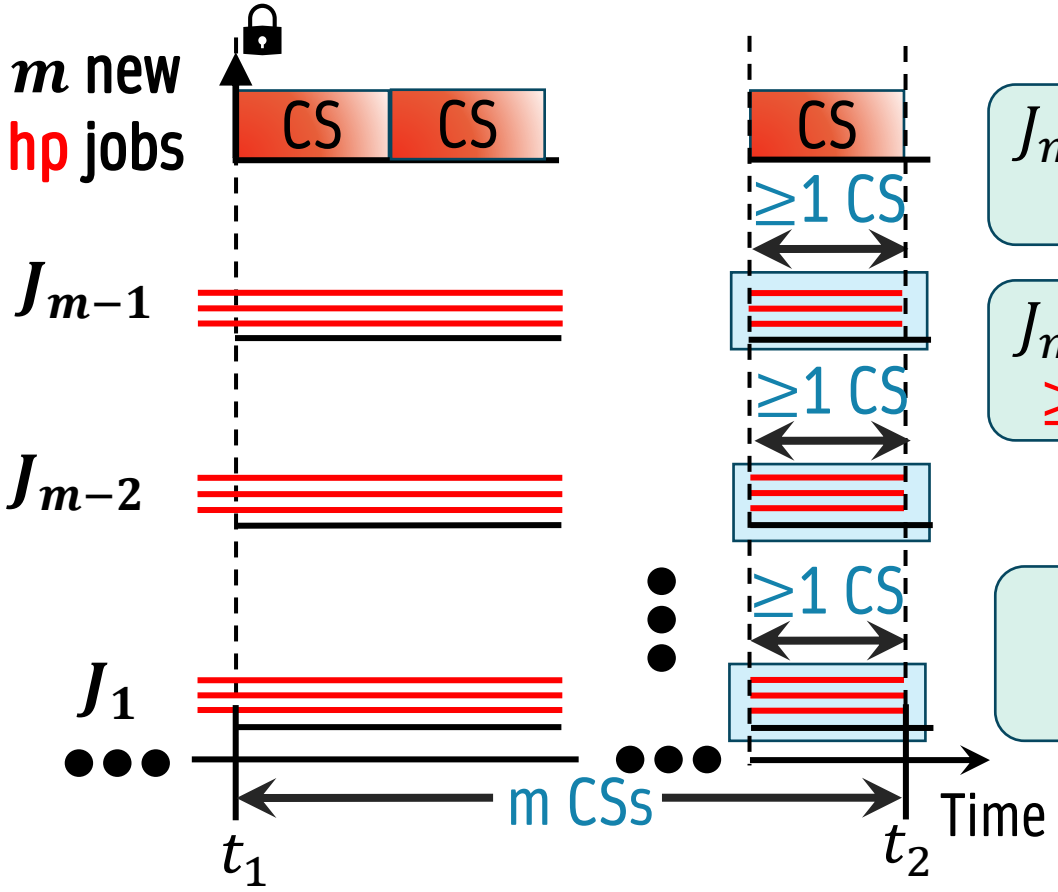


Lower Bound: Execution Scenarios



Lower Bound: Execution Scenarios

Case 1: all CSs of new jobs execute in $[t_1, t_2)$



J_{m-1} pi-blocked for ≥ m CS lengths

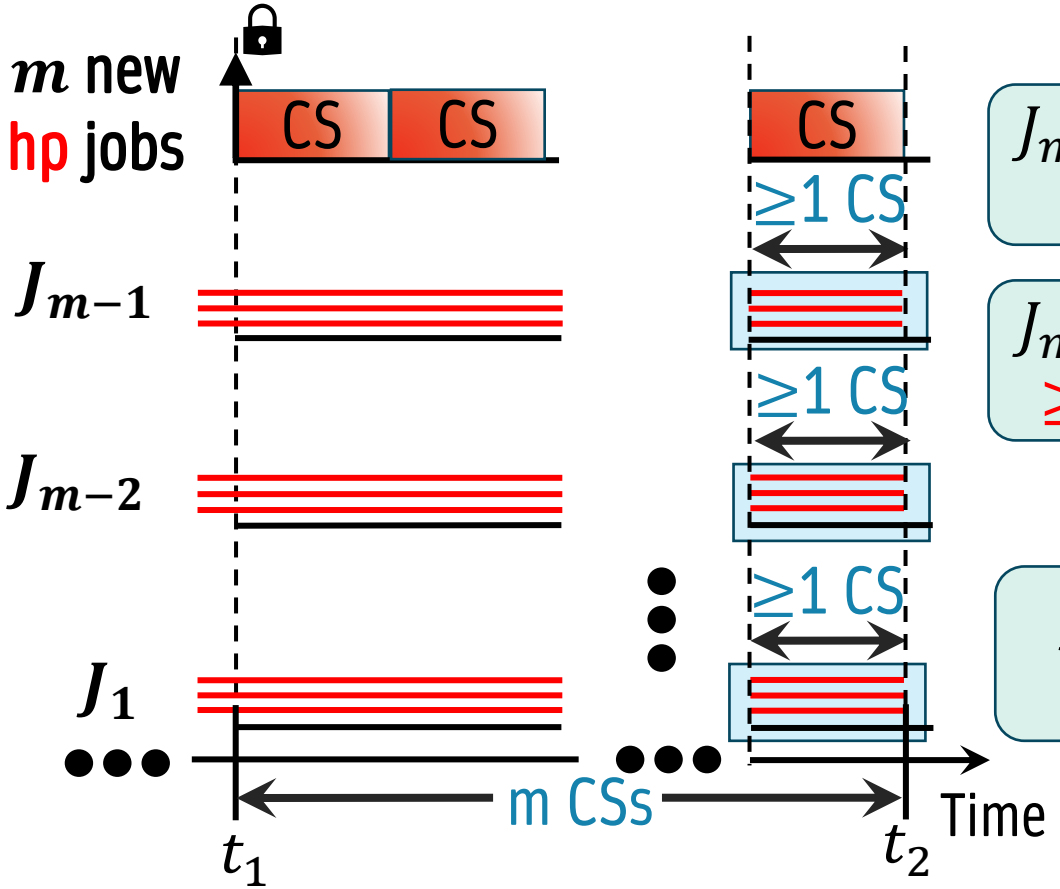
J_{m-2} pi-blocked for ≥ m-1 CS lengths

J_1 pi-blocked for ≥ 2 CS lengths

Pi-blocking (CS lengths)	# pending jobs	Jobs
≥ m		J_{m-1}
≥ m-1		J_{m-1}, J_{m-2}
≥ m-2		$J_{m-1}, \ldots, J_{m-3}$
⋮	⋮	⋮
≥ 1		$J_{m-1}, \ldots, J_1$

Lower Bound: Execution Scenarios

✓ Case 1: all CSs of new jobs execute in $[t_1, t_2)$



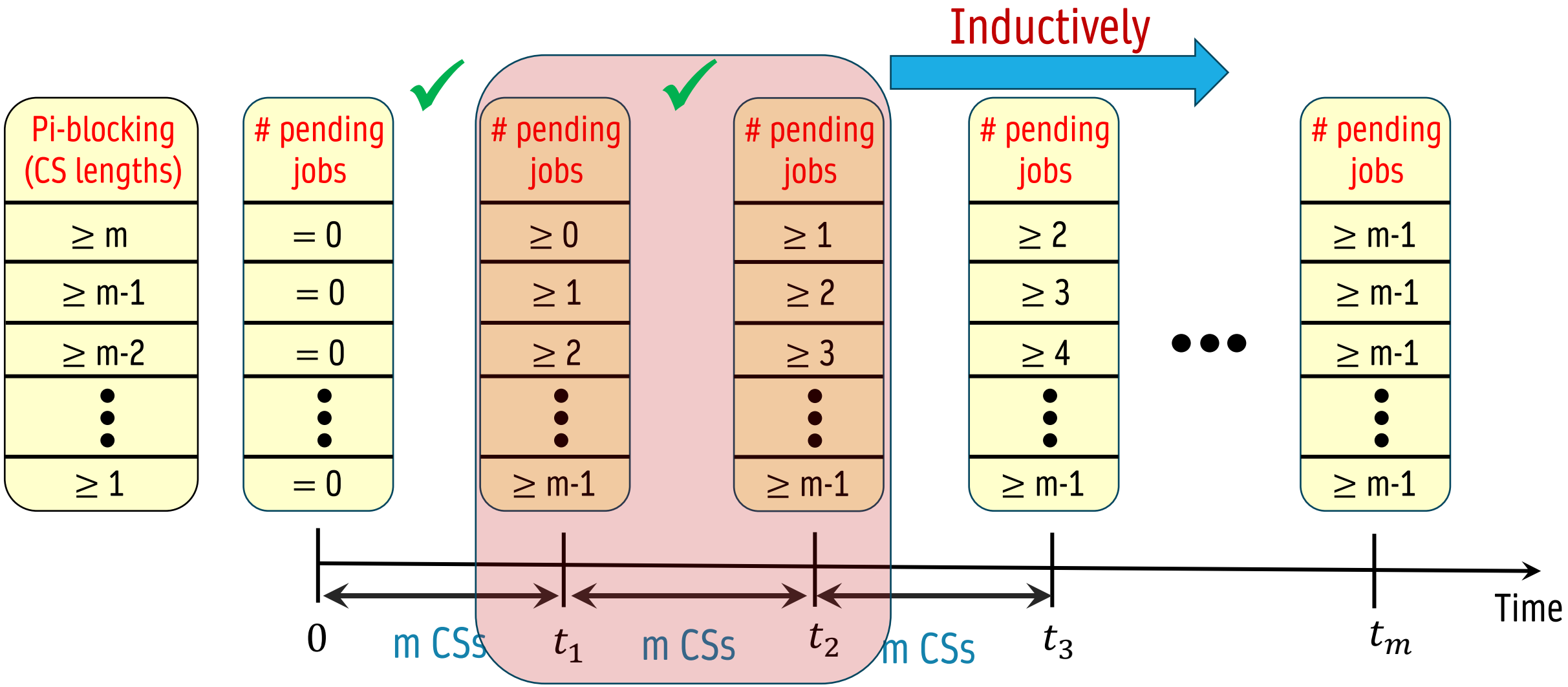
J_{m-1} pi-blocked for ≥ m CS lengths

J_{m-2} pi-blocked for ≥ m-1 CS lengths

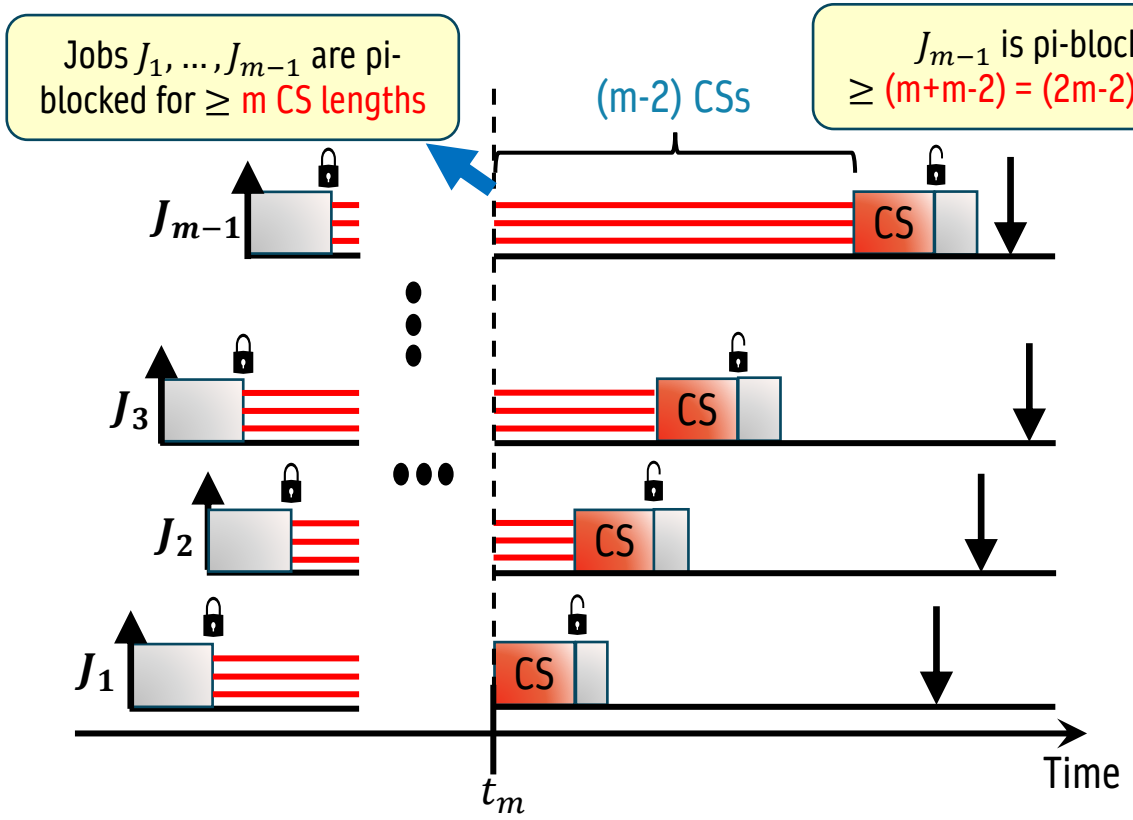
J_1 pi-blocked for ≥ 2 CS lengths

Pi-blocking (CS lengths)	# pending jobs	Jobs
≥ m	≥ 1	J_{m-1}
≥ m-1	≥ 2	J_{m-1}, J_{m-2}
≥ m-2	≥ 3	$J_{m-1}, \ldots, J_{m-3}$
⋮	⋮	⋮
≥ 1	≥ m-1	$J_{m-1}, \ldots, J_1$

Lower Bound: Key Idea



Lower Bound: Key Idea



m = # processors

✓ Goal

There exists a job release pattern causing (m-1) pending jobs with ≥ m CS lengths pi-blocking

A job suffers pi-blocking for ≥ (2m-2) CS lengths

Conclusion

- The "**two**" in (2m-1) pi-blocking bounds is **fundamental** under G-EDF, G-FP scheduling
- The G-OMLP, C-OMLP, OMIP are **almost optimal** under G-EDF, G-FP scheduling
- Future work:
  - Is the "two" fundamental under **partitioned** or **clustered** scheduling too?

Thank You