The Case for Phase-Aware Scheduling of Parallelizable Jobs

Benjamin Berg, Justin Whitehouse, Benjamin Moseley, Weina Wang, Mor Harchol-Balter

Carnegie Mellon University
Computer Science Department

Carnegie Mellon University

Parallel Systems Have Resulted in Parallel Workloads

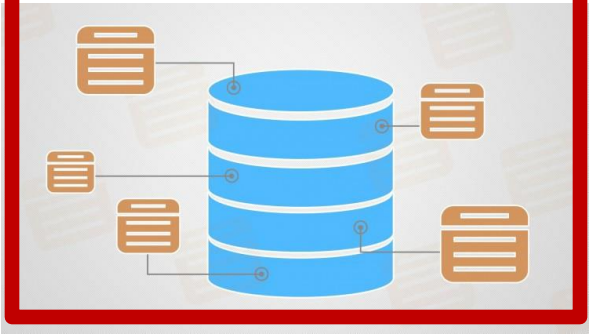
Multicore Server

Rack-Scale Computers

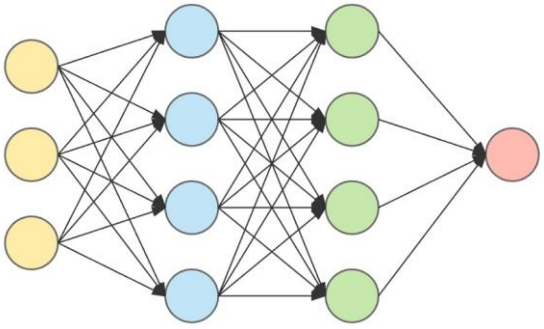
Data centers

How to run these workloads on these systems?

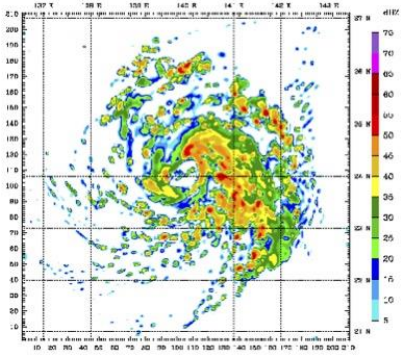
Databases



ML Training/Serving



Scientific Computing



Basic Model of a Database

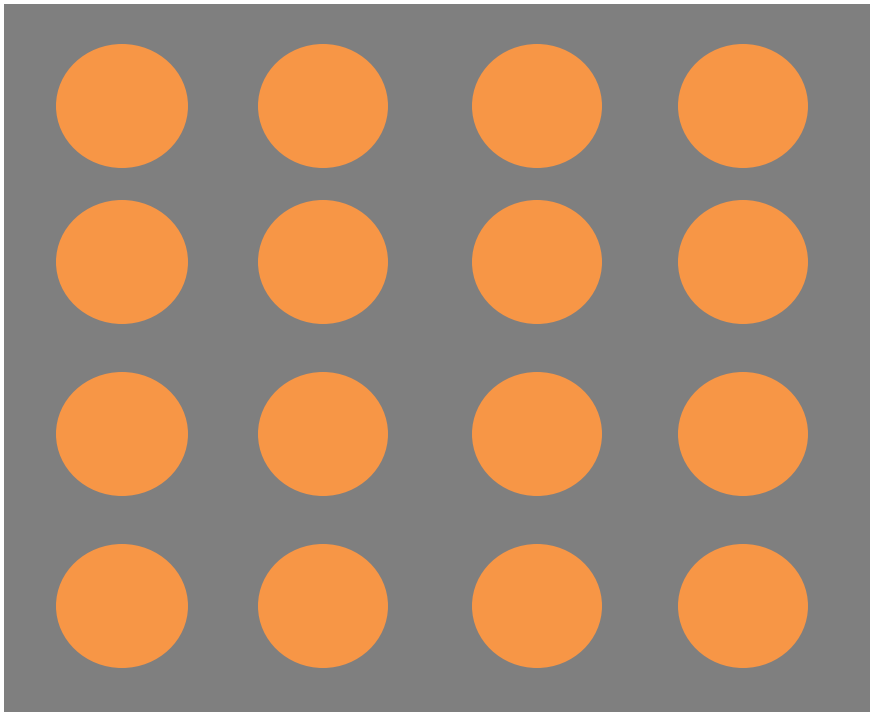
Q 1

Q 2

Q 3

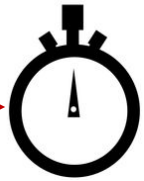
Q 4

λ queries/sec.



K=16 cores

- **How to model the benefit of running on additional cores?**
- **How to split cores between queries?**



Re

Goal:

Minimize mean response time, E[T]

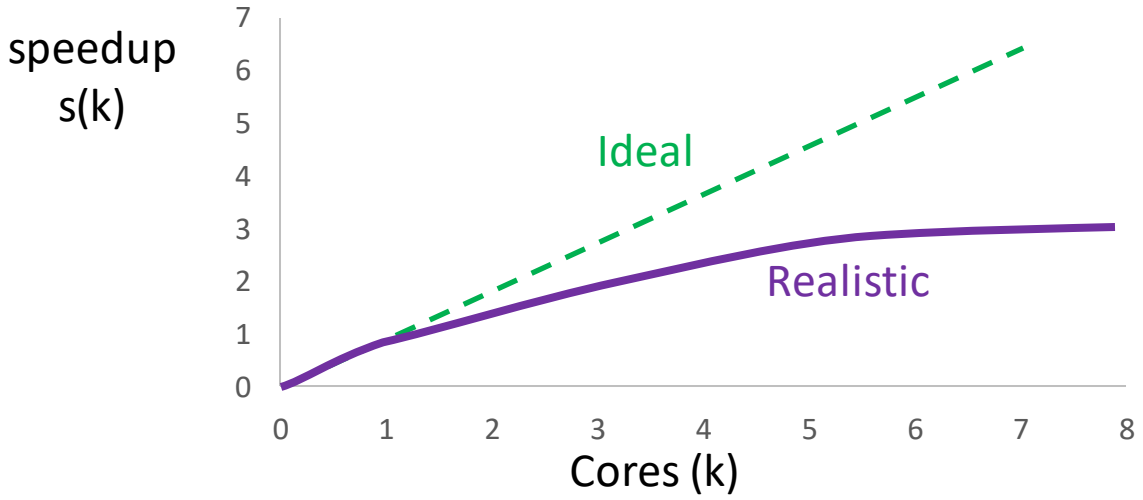
Carnegie Mellon University

Job Size

- **Job Size:** Running time on single core
- Reflects amount of "inherent work"
- E.g. number of tuples to be processed

How does running time change on k cores?

s(k): How many times faster on k cores than on 1 core

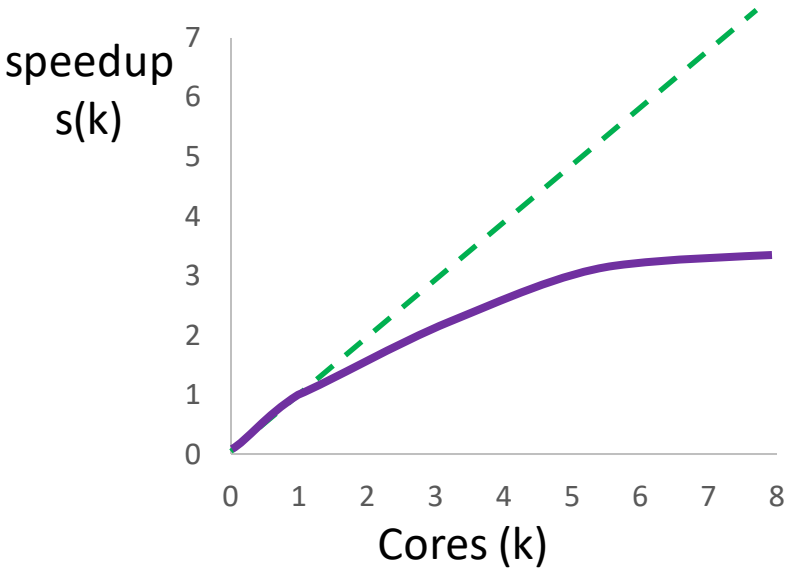


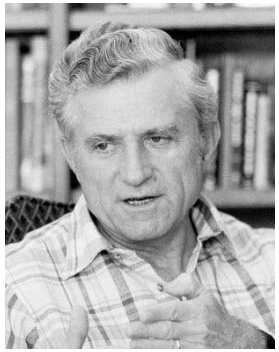
$$\frac{Query\ Size}{s(k)} = \textbf{running time on k cores}$$

Carnegie Mellon University

But, jobs change over time

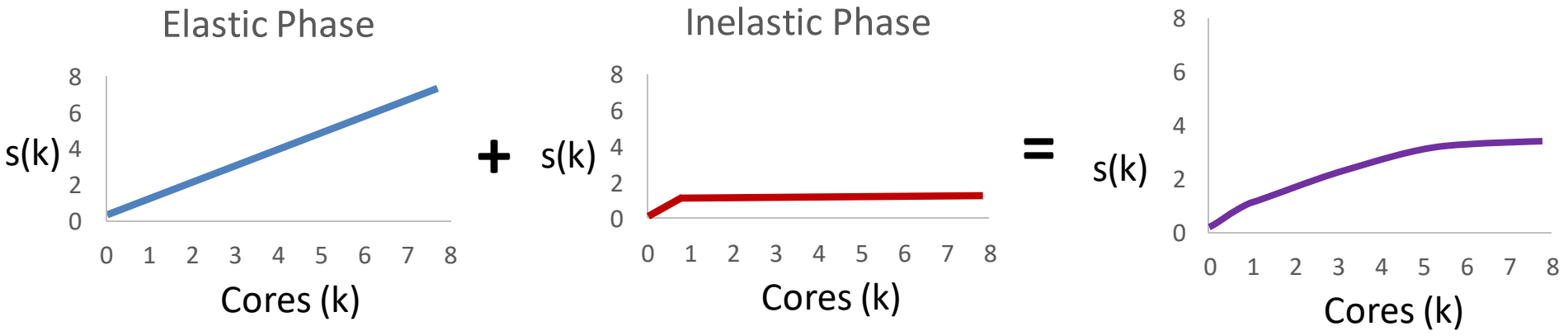
Beyond Speedup Functions

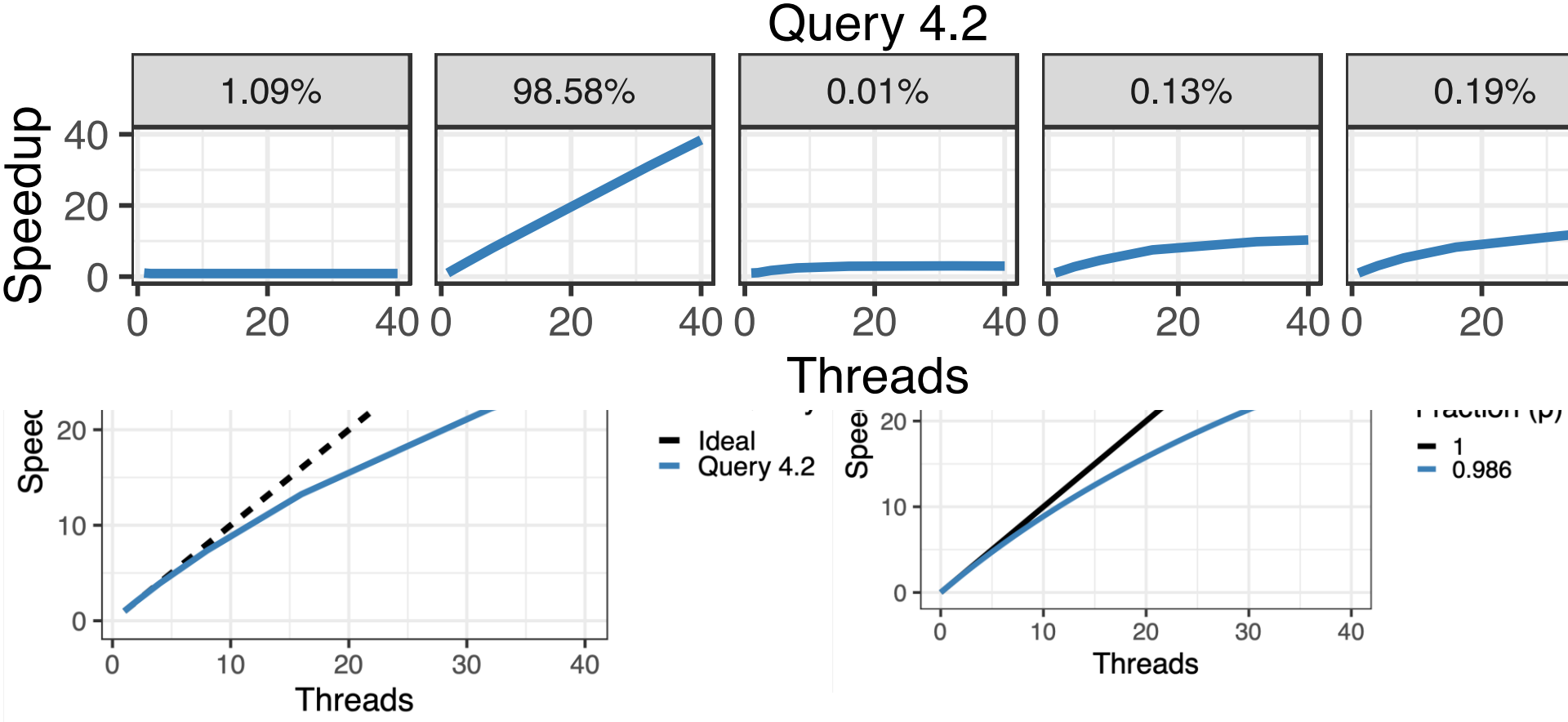


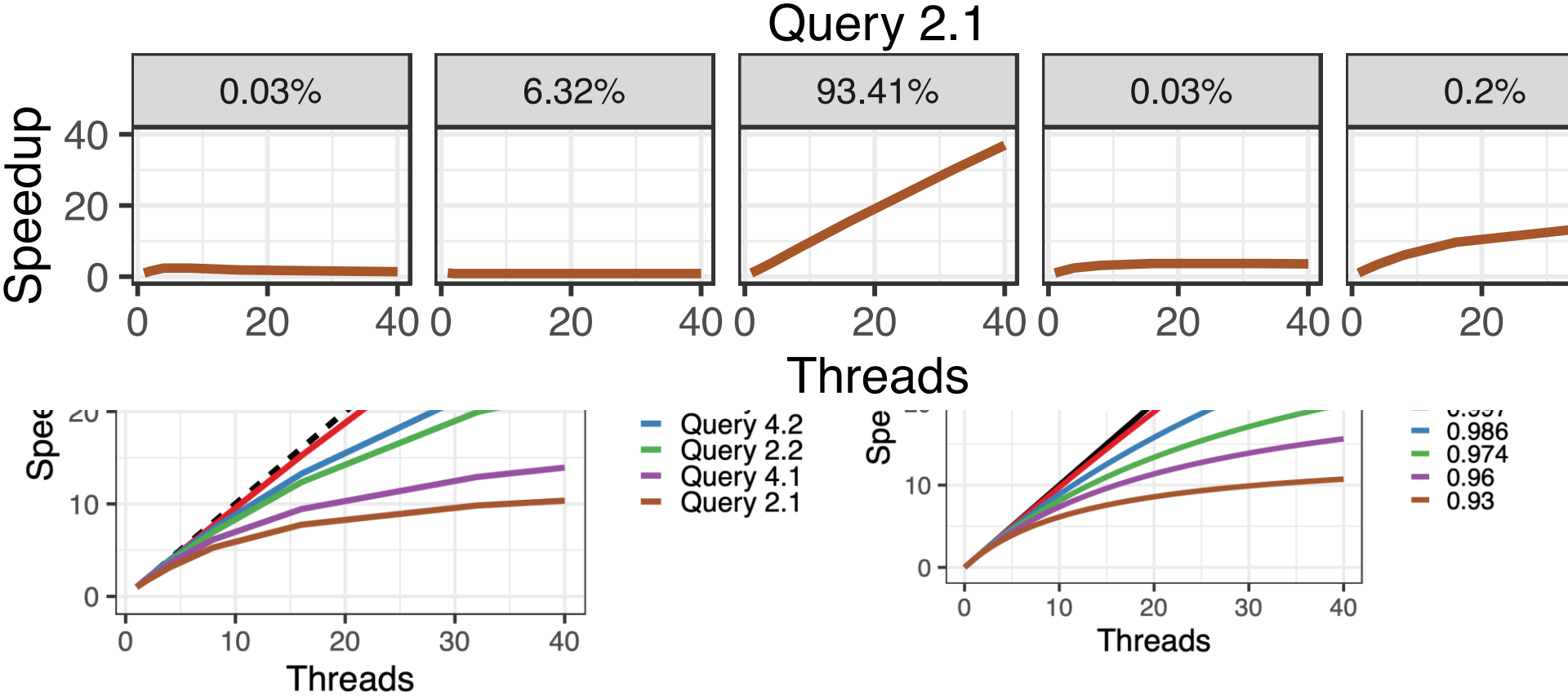


Amdahl's Law: Only p fraction of the job is parallelizable

Speedup functions measure ***average*** speedup



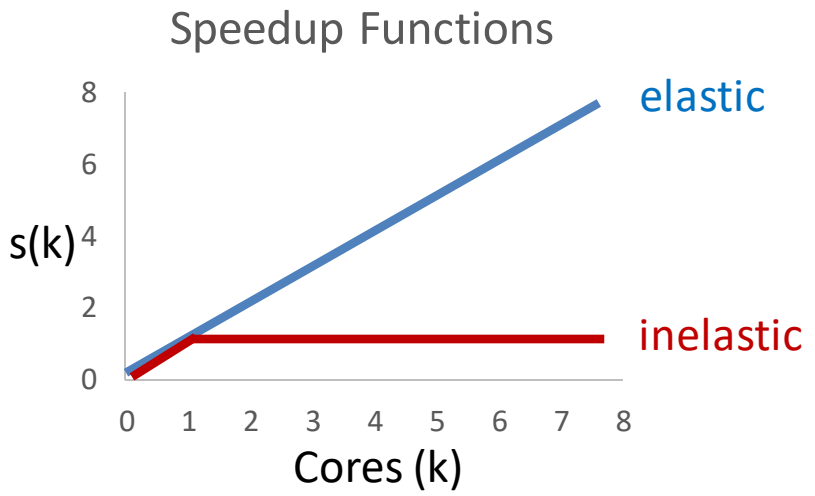

Query 4.2
1.09%
98.58%
0.01%
0.13%
0.19%
Speedup
40
20
0
0
20
40
Threads
Ideal
Query 4.2
Threads
1
0.986


Query 2.1
0.03%
6.32%
93.41%
0.03%
0.2%
Speedup
0
20
40
Threads
Query 4.2
Query 2.2
Query 4.1
Query 2.1
0.986
0.974
0.96
0.93

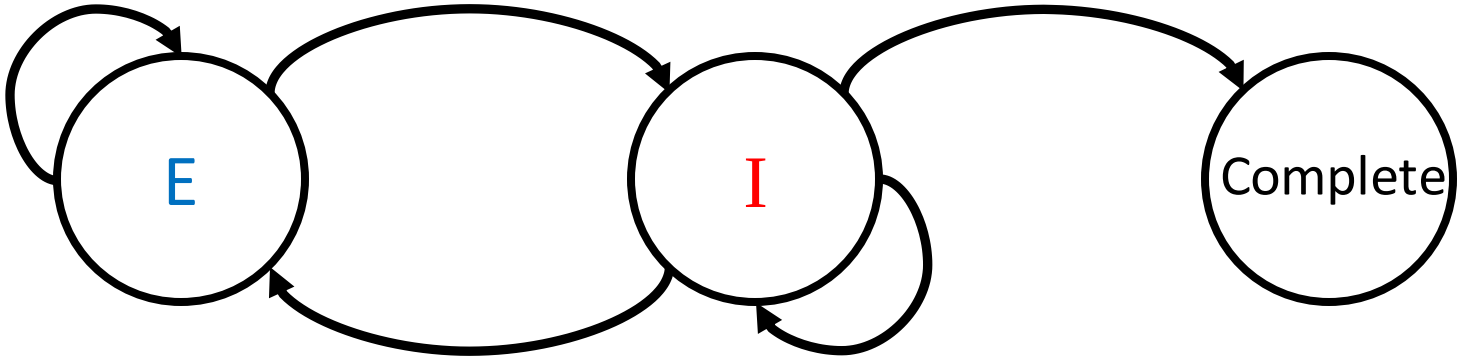
Modeling Jobs with Phases

A job is composed of multiple phases:

- Elastic phases
- Inelastic phases
- Completed phase



- Arbitrary rates/starting phase



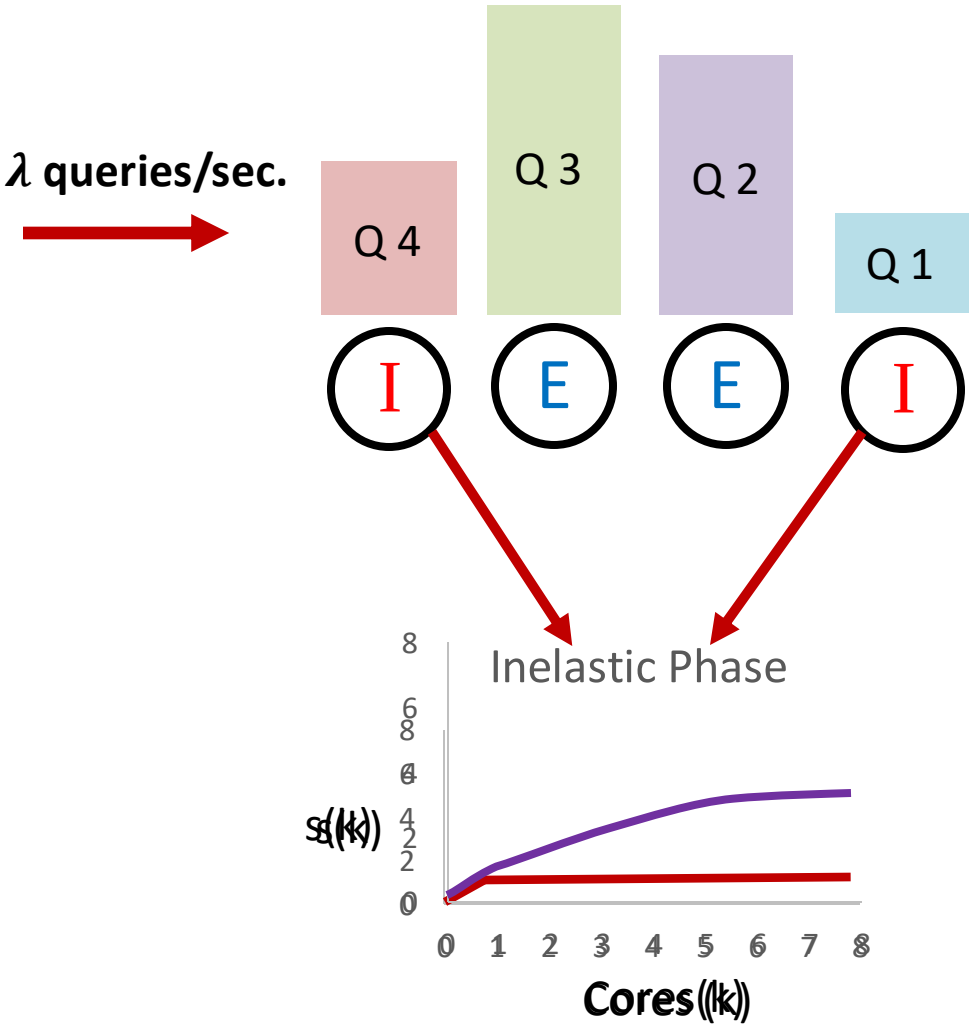
Idea: each job looks like a Continuous Time Markov Chain

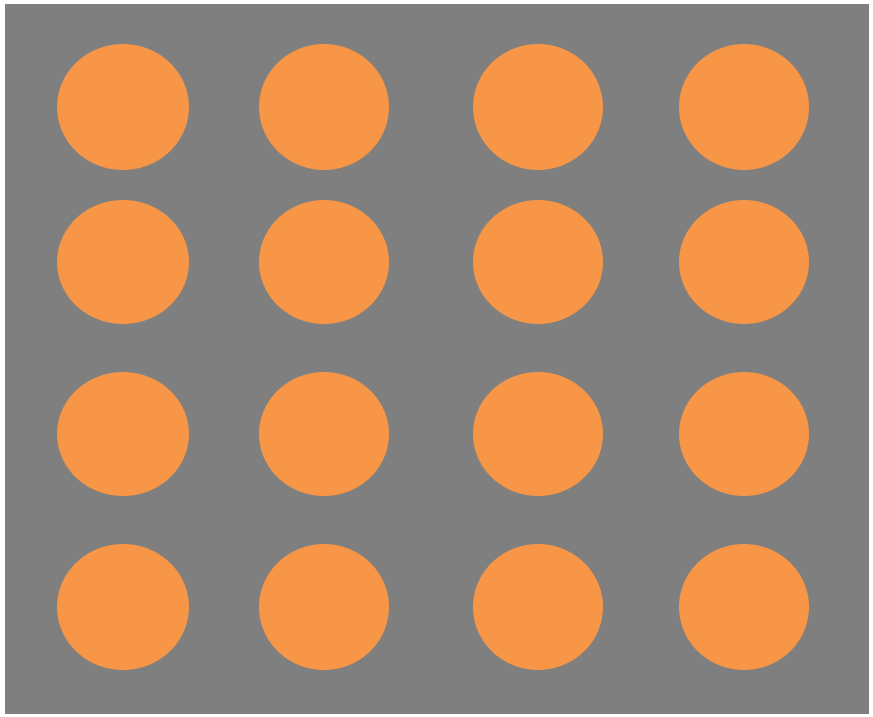
Carnegie Mellon University

How to split cores between queries?

How should a scheduler leverage phase information?

The Problem with Phase Unaware Scheduling

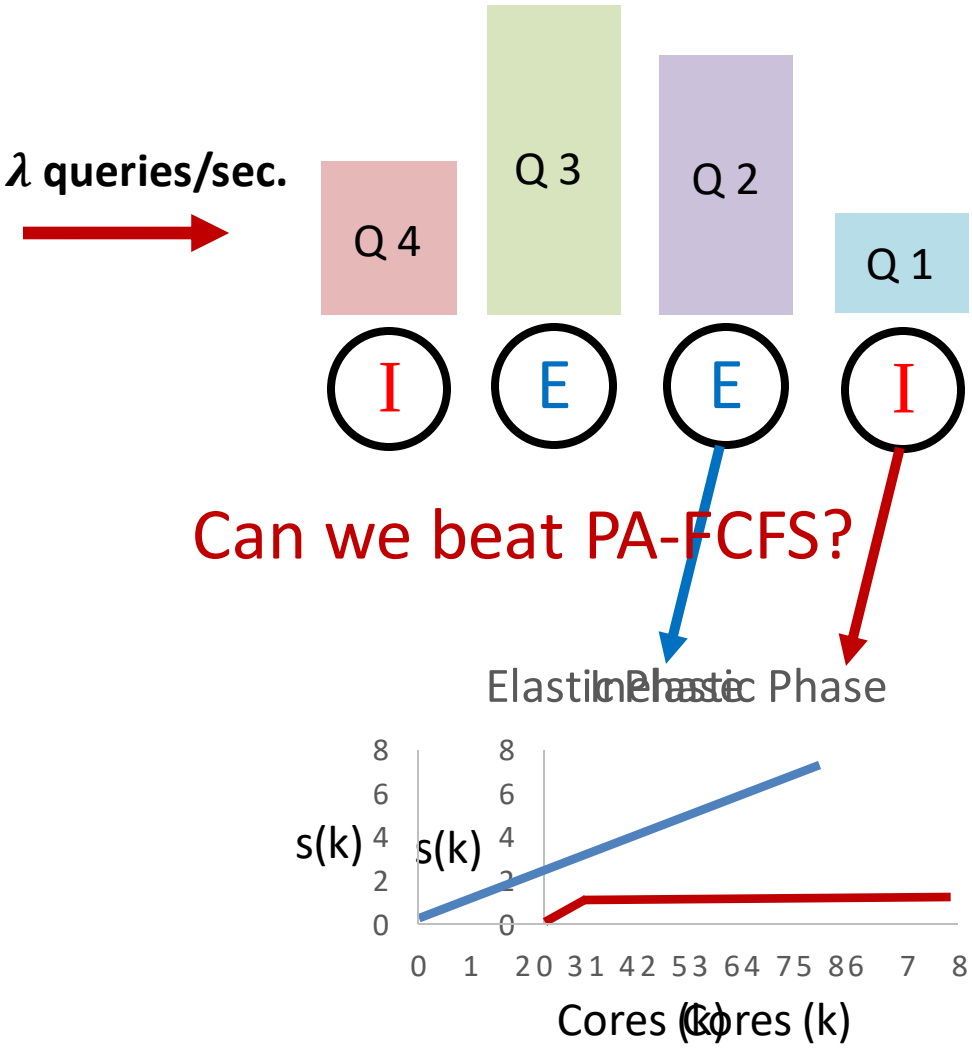


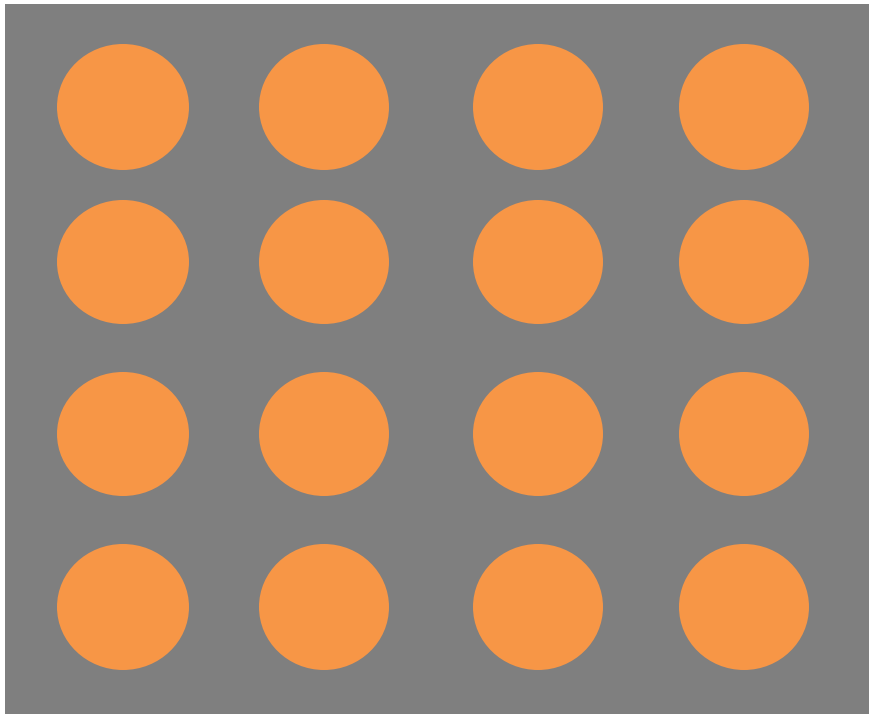


EQUI: divide cores equally at all times

Over-allocates to inelastic phases

Scheduling In Databases

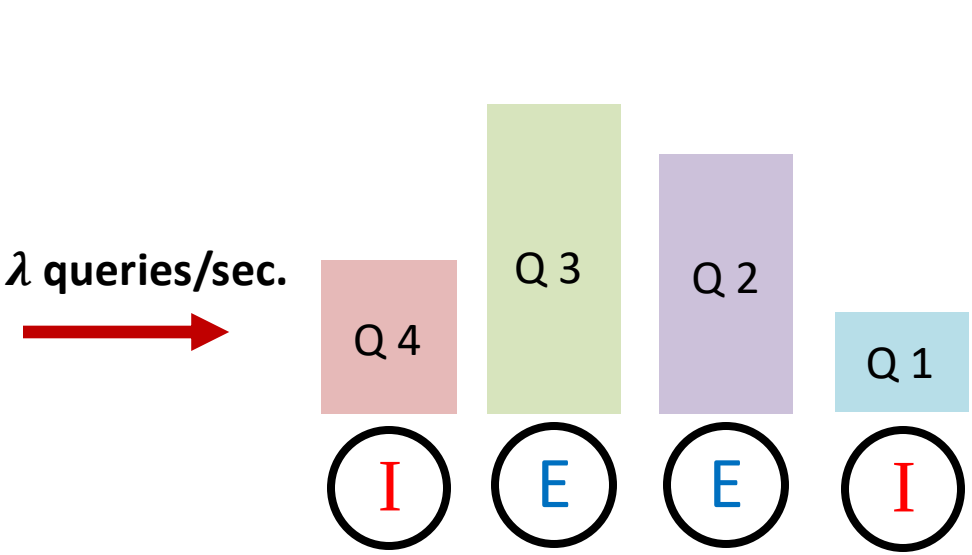


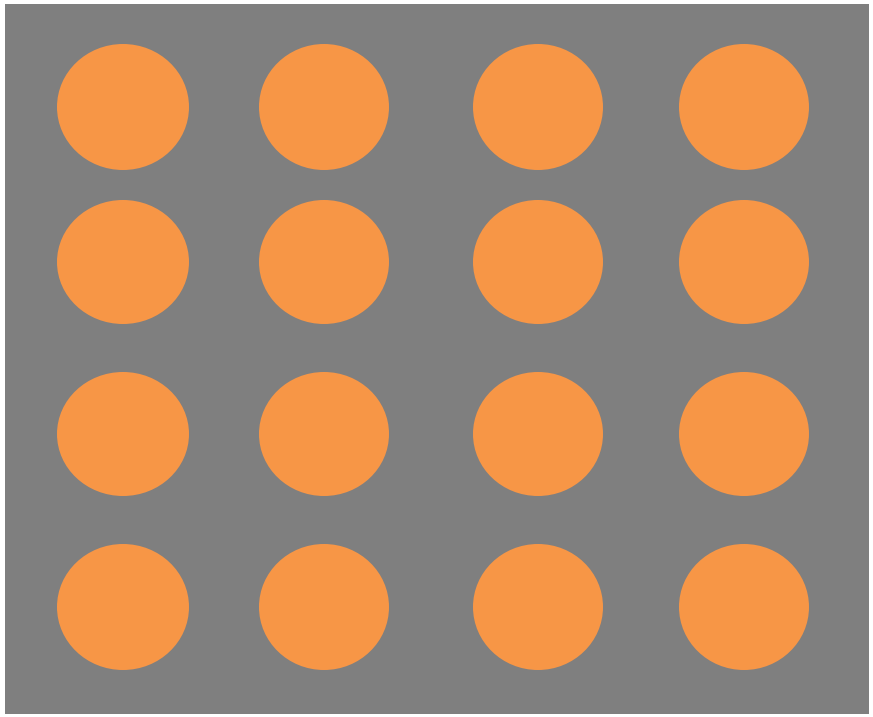


Phase-Aware First-come-first-served (PA-FCFS)

NoisePage database [VLDB 2021] using morsel-driven parallelism [SIGMOD 2014]

Improving phase Aware scheduling





Can we beat PA-FCFS?

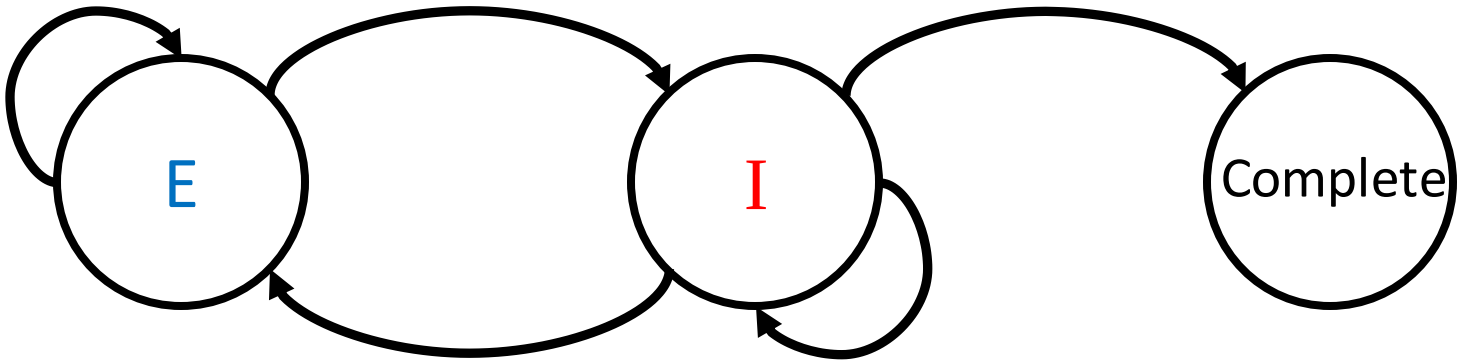
Two Key Ideas:

- **Defer Parallelizable Work**
- Work on shorter queries

Carnegie Mellon University

Proof: Deferring parallelizable work can be sufficient for optimality

- **Inelastic-First (IF) Policy:** give strict priority to inelastic phases

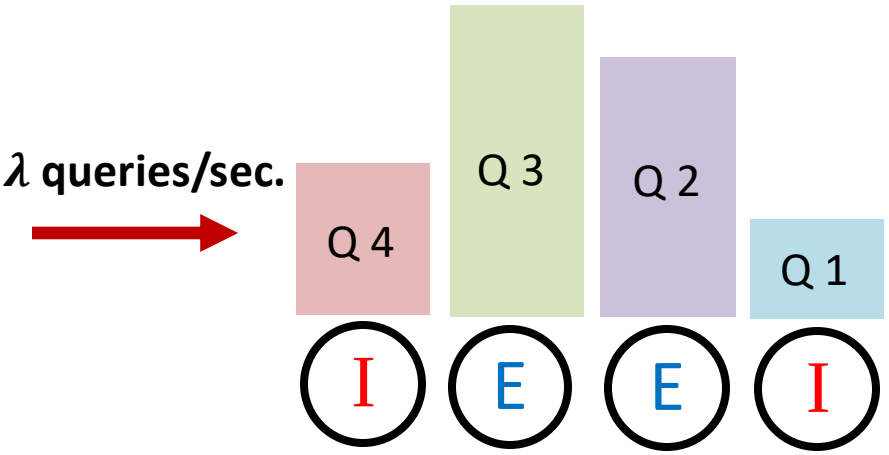


- Phase sizes unknown
- Elastic phases$\sim \exp(\mu_E)$, Inelastic phases$\sim \exp(\mu_I)$
- IF minimizes mean response time

What happens when we violate these assumptions?

Carnegie Mellon University

Incorporating Query Size



Two Key Ideas:

- **Defer Parallelizable Work**
- Work on shorter jobs

Known query structures?

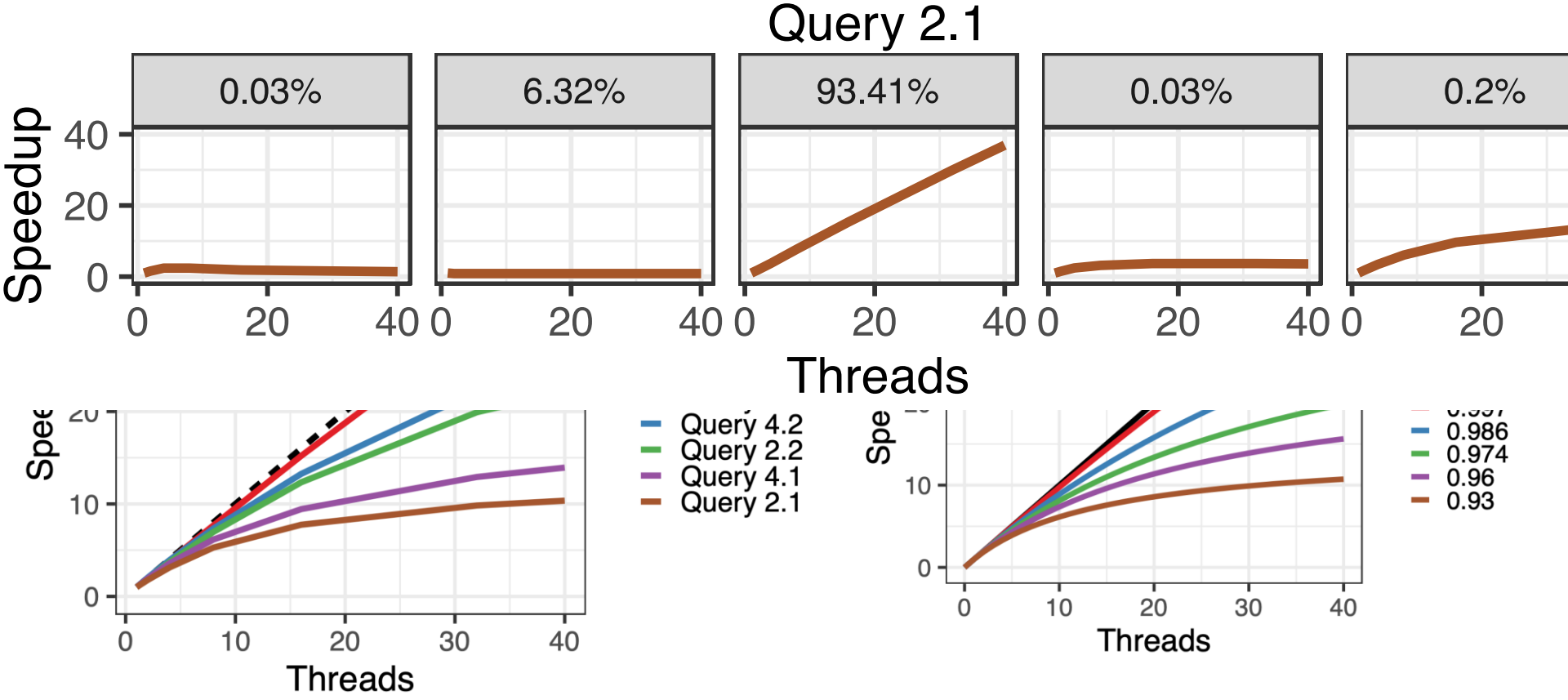
Phase-sizes known?

IF-SRPT

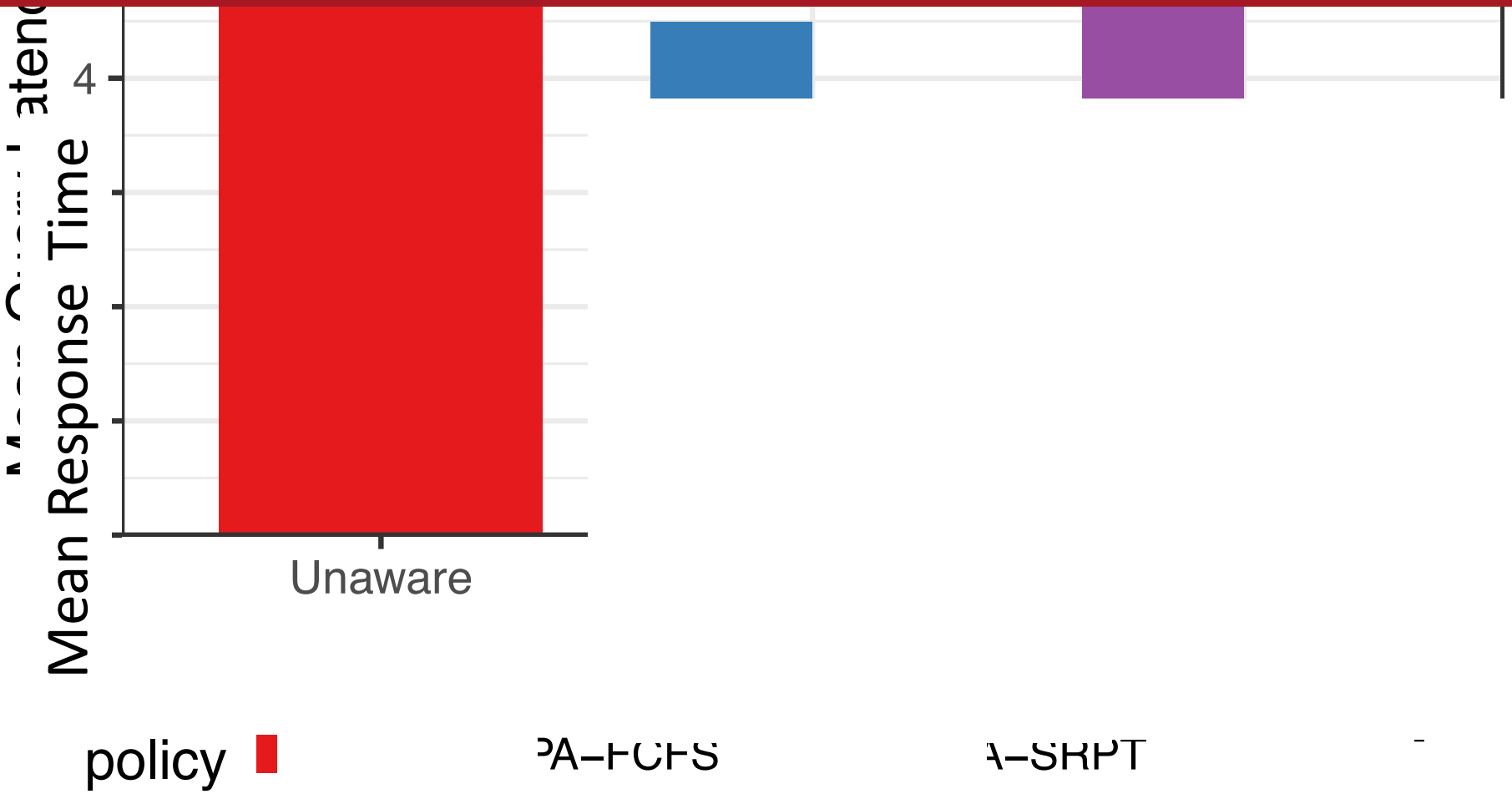
- Give priority to inelastic phases
- Process inelastic phases in SRPT order
- Process elastic phases in SRPT order

Why not Phase-Aware SRPT (PA-SRPT)?

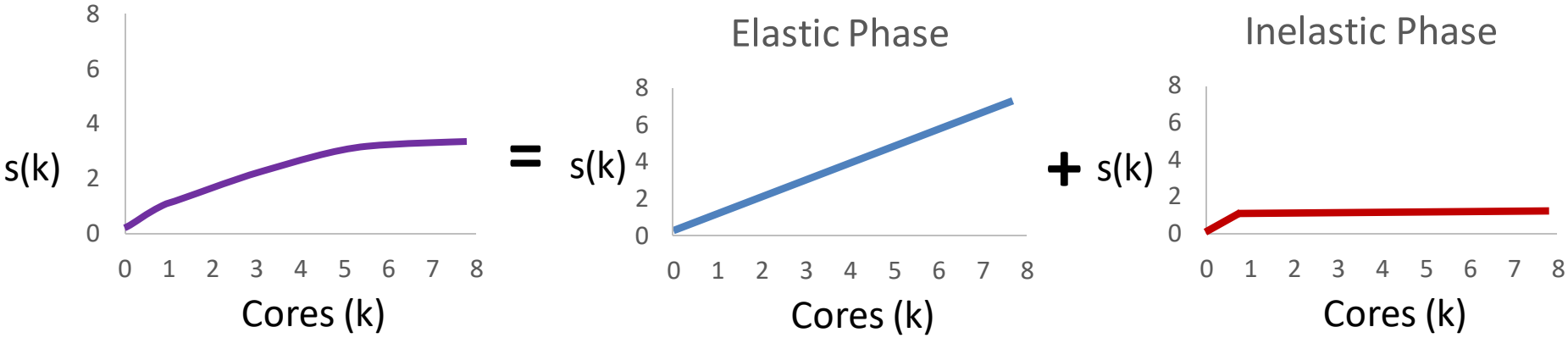
Carnegie Mellon University

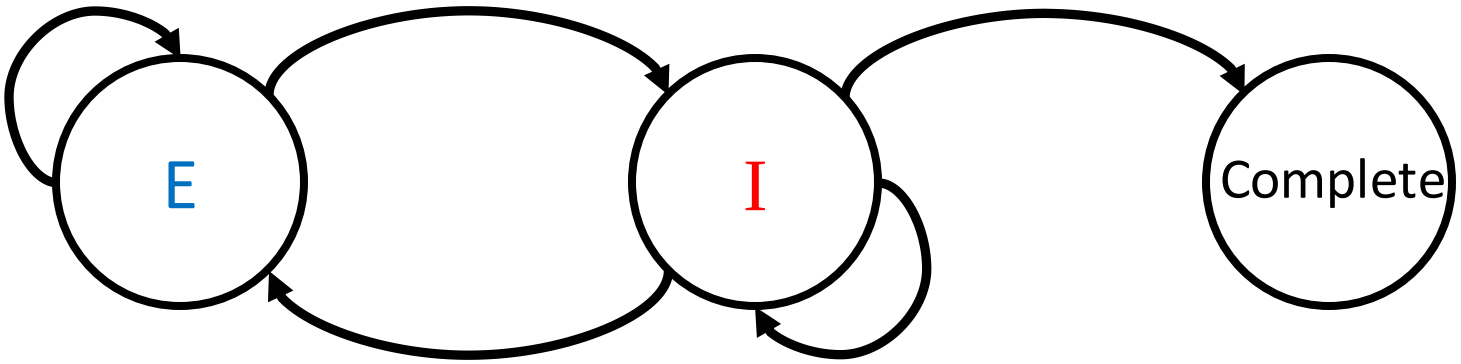

Query 2.1
0.03%
6.32%
93.41%
0.03%
0.2%
Speedup
0
20
40
Threads
Query 4.2
Query 2.2
Query 4.1
Query 2.1
0.986
0.974
0.96
0.93

Does IF work in Practice?



Conclusion



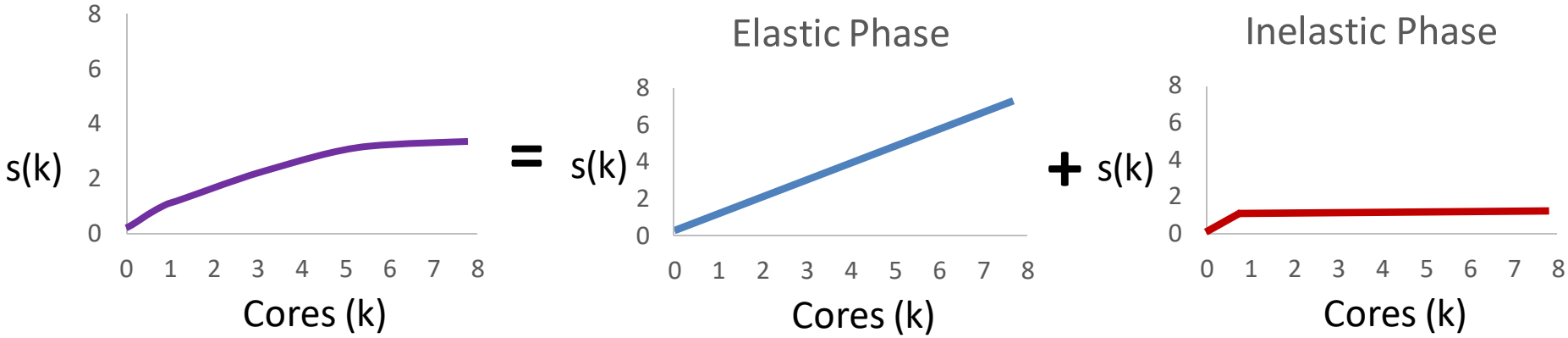


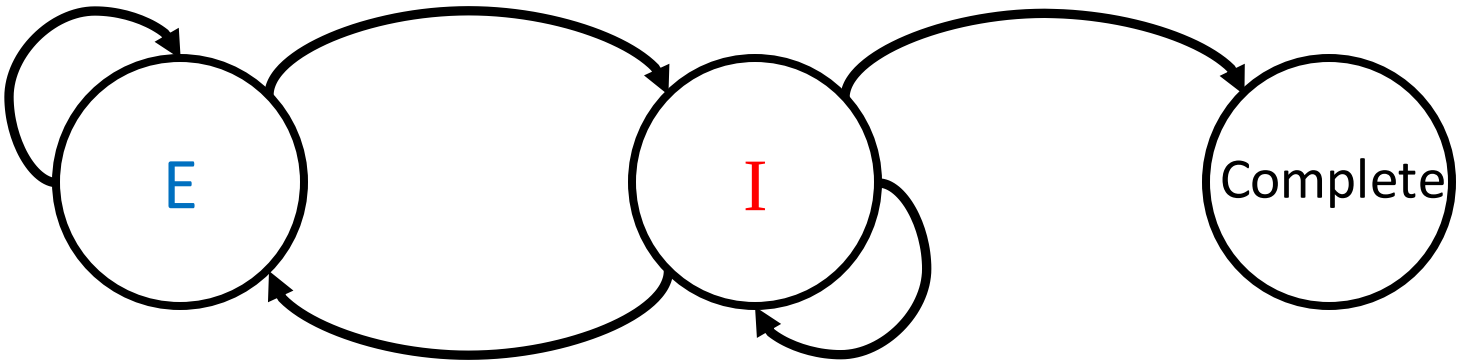
Two Key Ideas:

- **Defer Parallelizable Work**
- Work on shorter jobs

- IF can be optimal
- IF-SRPT performs well in practice

Questions?





Two Key Ideas:

- **Defer Parallelizable Work**
- Work on shorter jobs

- IF can be optimal
- IF-SRPT performs well in practice