Dynamic Query Re-planning using QOOP

Kshiteej Mahajan[w], Mosharaf Chowdhury[m], Aditya Akella[w], Shuchi Chawla[w]

WISCONSIN
UNIVERSITY OF WISCONSIN–MADISON

THE UNIVERSITY OF MICHIGAN
ARTES · SCIENTIA · VERITAS
1817

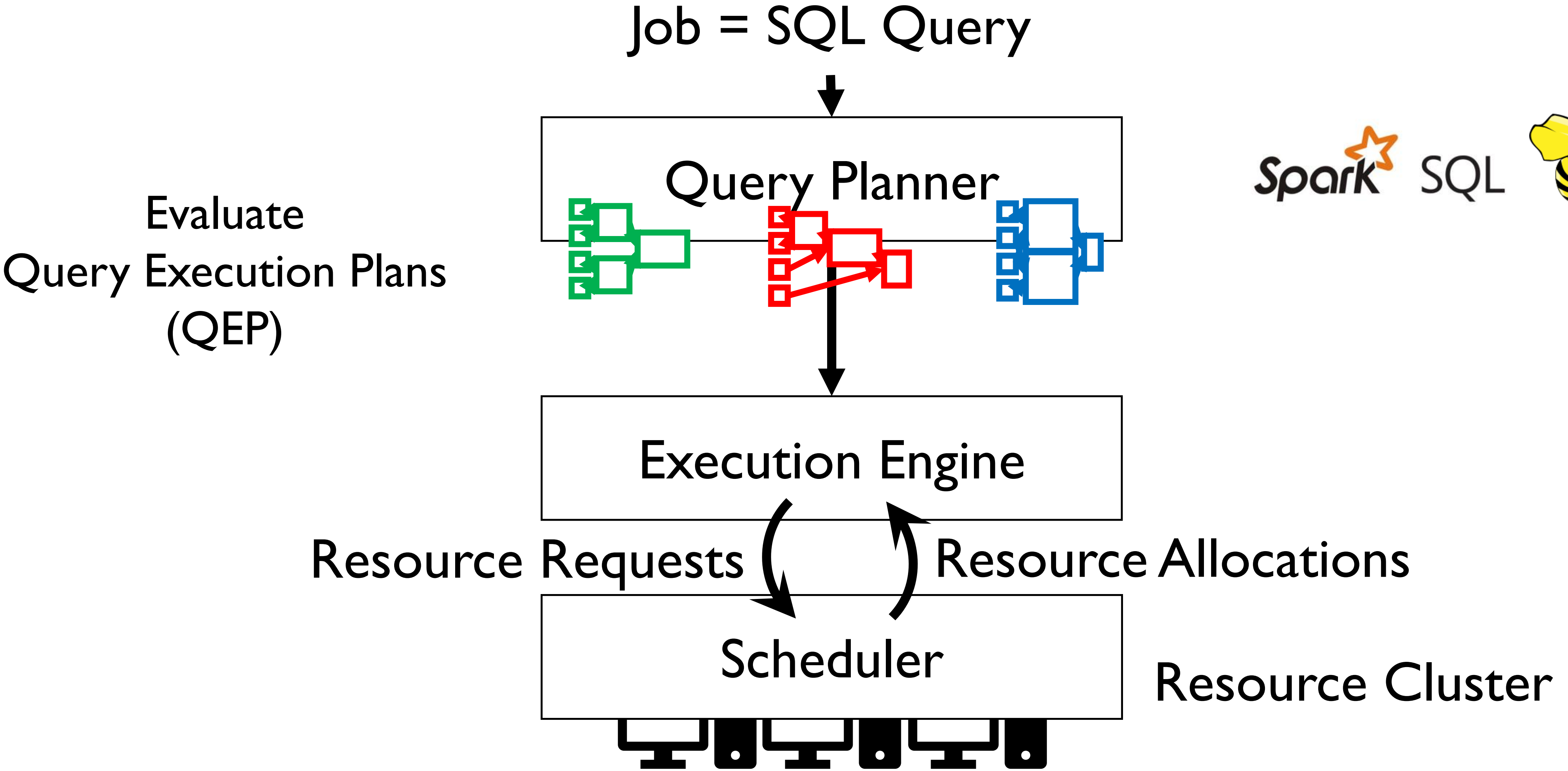
What is QOOP?

- QOOP is a **distributed data analytics system** that performs well under **resource volatilities**
- Core Ideas –
  - Re-architect the data analytics system stack
  - Enable Dynamic Query Re-planning
  - Simplify Scheduler

Agenda

- **Overview**
  - **Distributed Data Analytics Systems**
  - **Resource Volatilities**
- Overcoming Inefficiency #1
  - Static Query Planner
  - QOOP's Dynamic QEP Switching
- Overcoming Inefficiency #2
  - Complex and Opaque Scheduler
  - QOOP's Scheduler Choice
- Implementation
- Evaluation

Overview – Distributed Data Analytics



Job = SQL Query

Query Planner

Evaluate Query Execution Plans (QEP)

Execution Engine

Resource Requests

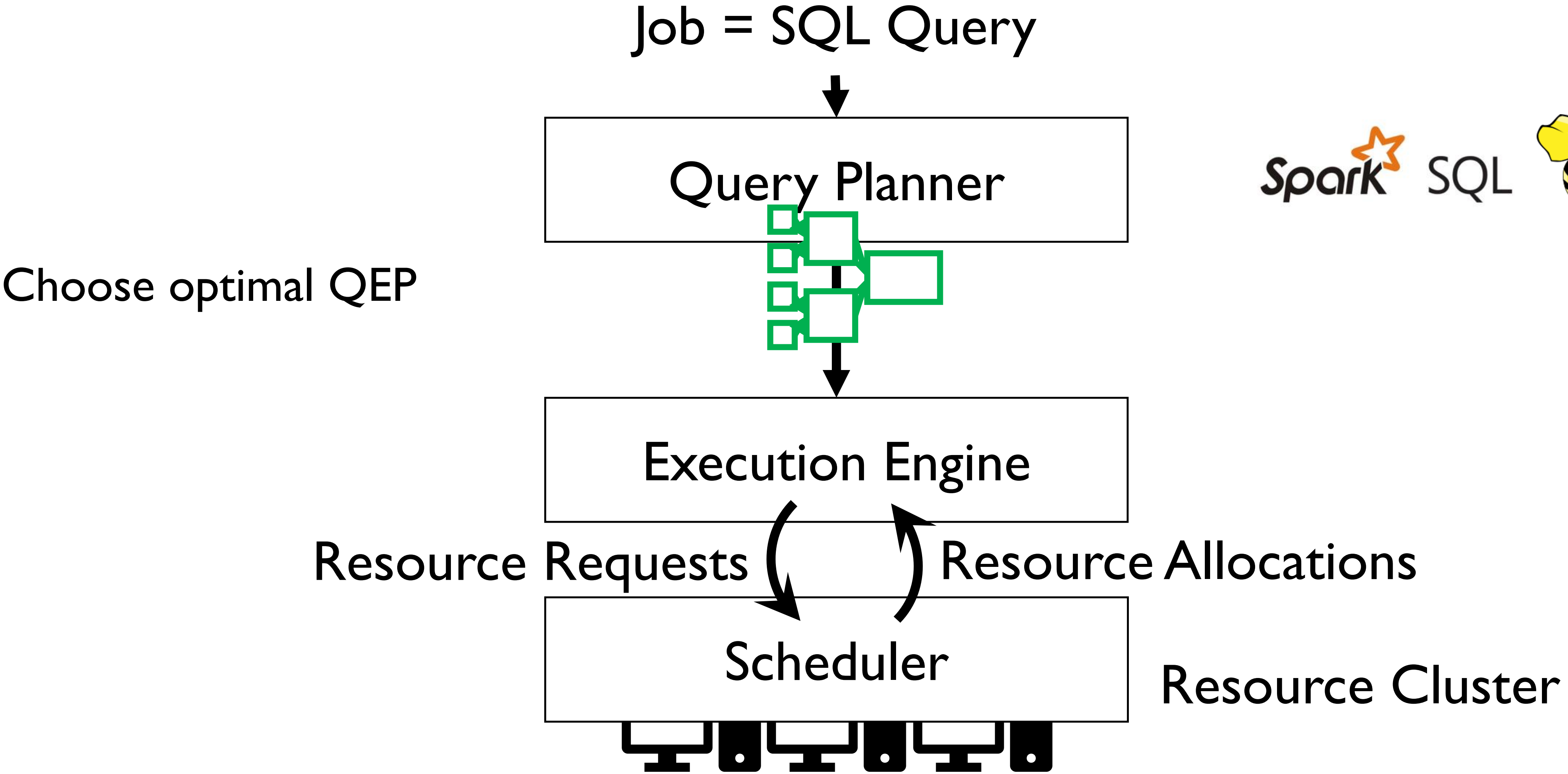
Resource Allocations

Scheduler

Resource Cluster

Spark SQL HIVE

Overview – Distributed Data Analytics



Job = SQL Query

Query Planner

Choose optimal QEP

Execution Engine

Resource Requests

Resource Allocations

Scheduler

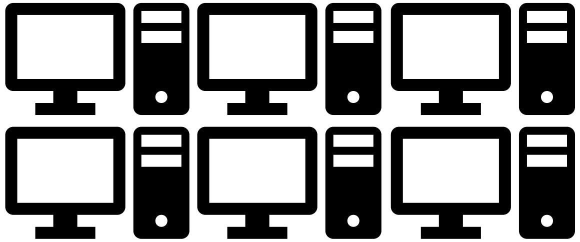
Resource Cluster

Overview – Resource Volatilities

Job = SQL Query

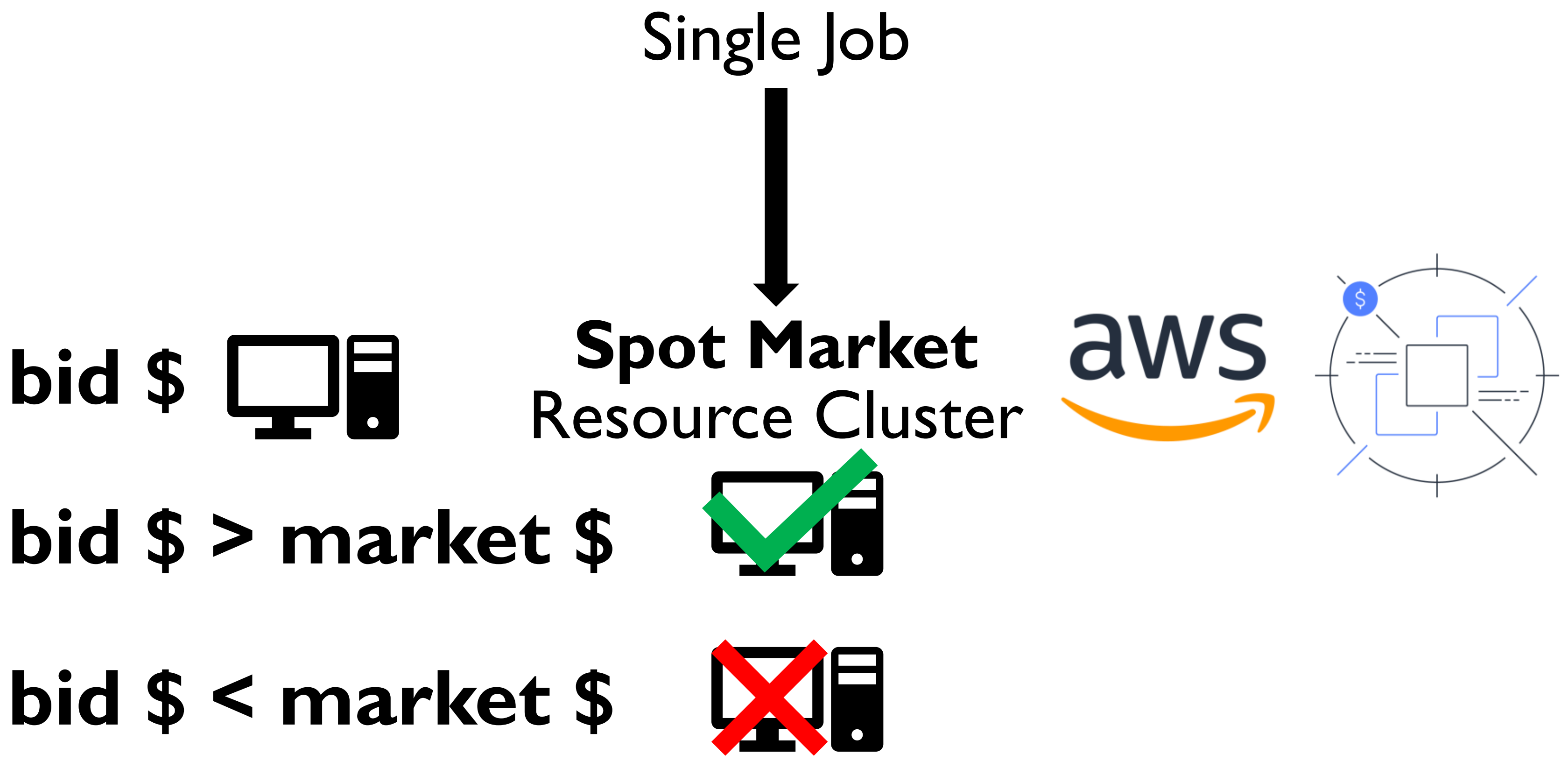
Resource Share

~~more or less fixed~~ **significantly** changes over time



Resource Volatilities

Overview – Resource Volatility; Spot Market



Single Job

Spot Market
Resource Cluster

aws

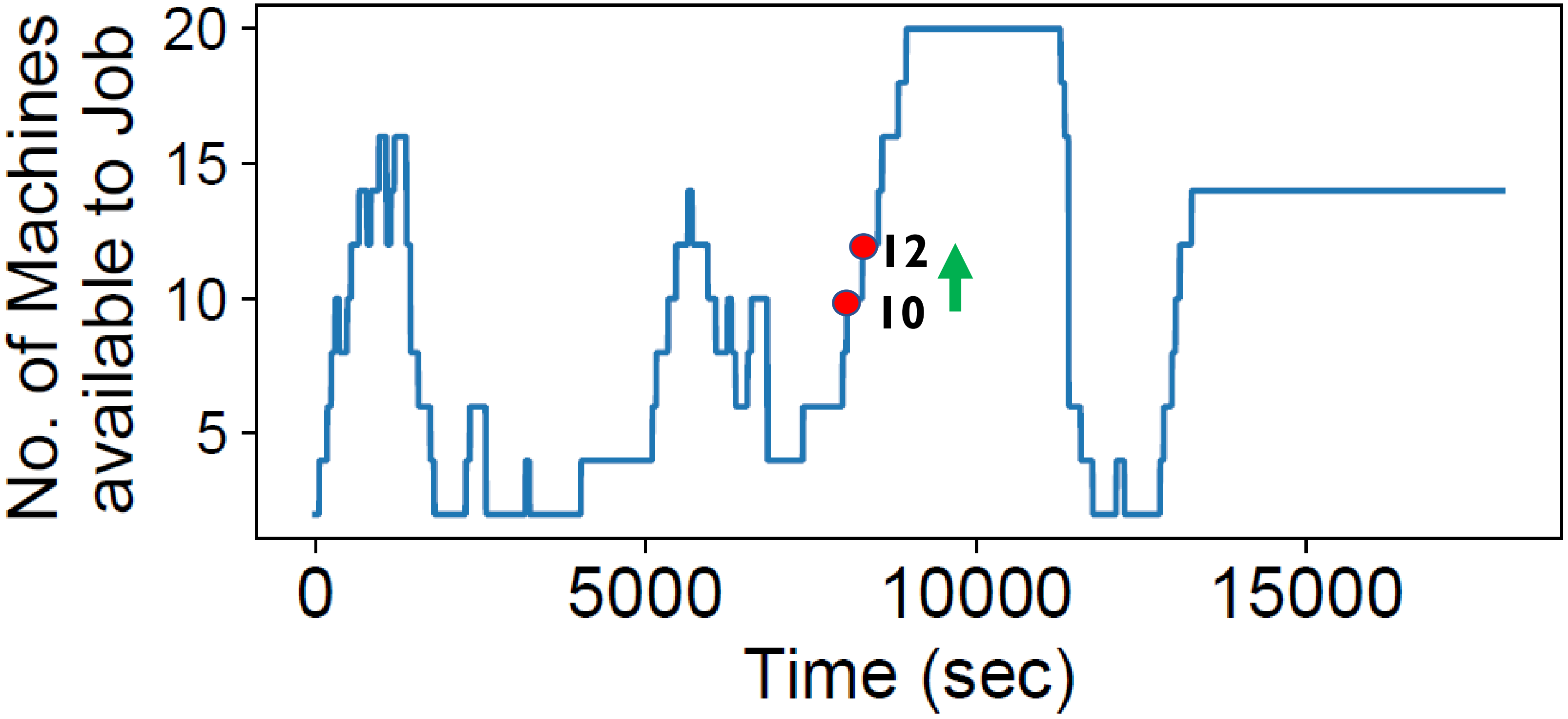
bid $

bid $ > market $ ✓

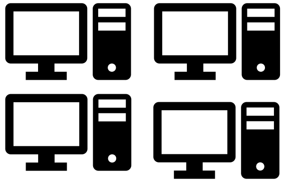
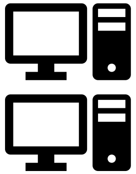
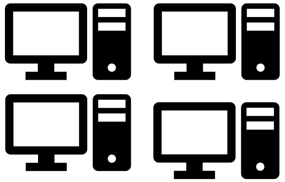
bid $ < market $ ✗

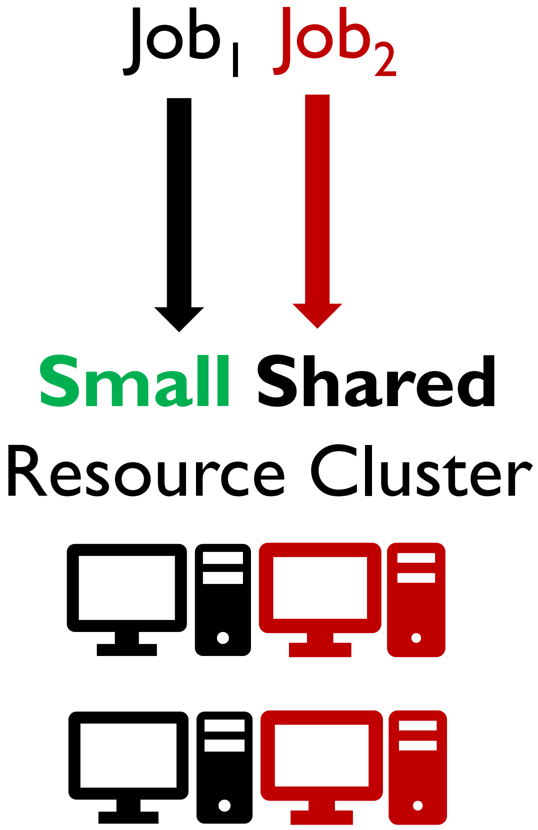
Overview – Resource Volatility; Spot Market

- Fixed budget cost-saving bidding strategy in AWS Spot Market
- 20% resource volatile event – 20% change in #machines over time
- 50 such events in a 5-hour span



Overview – Resource Volatility; Small Cluster

Resource Contention	**Job$_1$'s Resource Share**
Only Job$_1$	
Job$_2$ enters	
Job$_2$ exits	



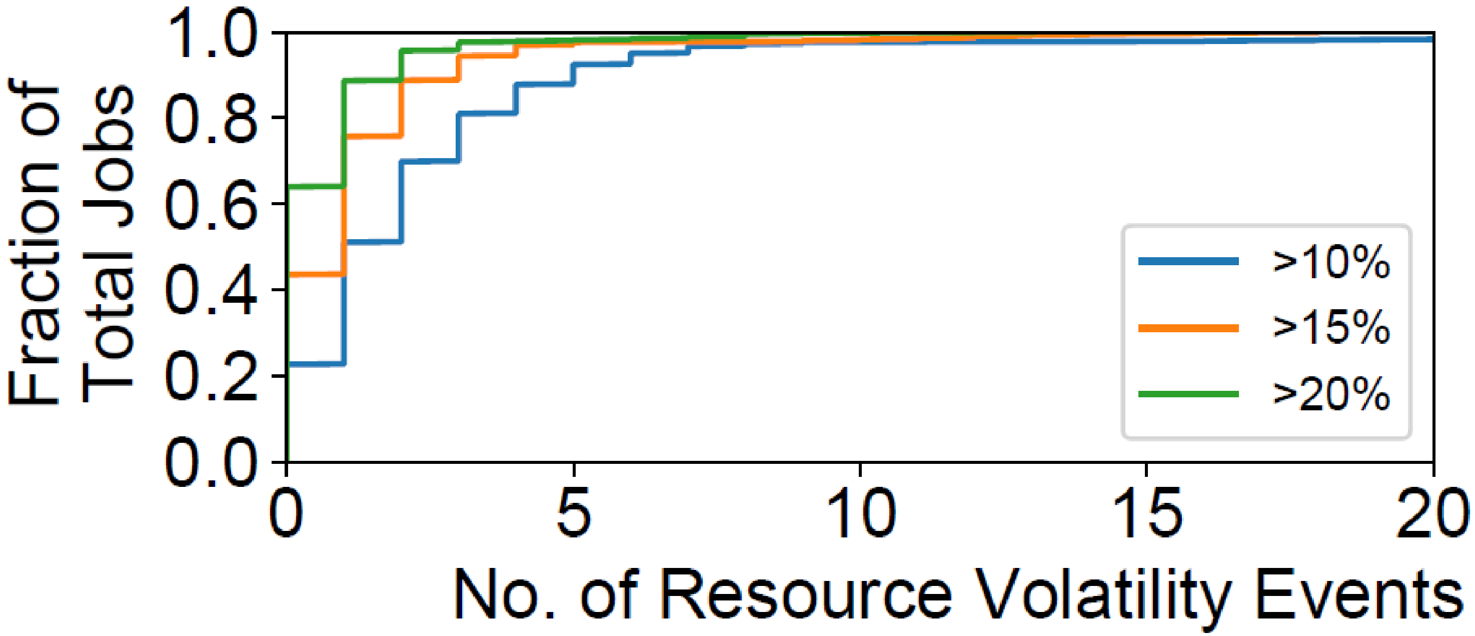
Job$_1$ Job$_2$

Small Shared

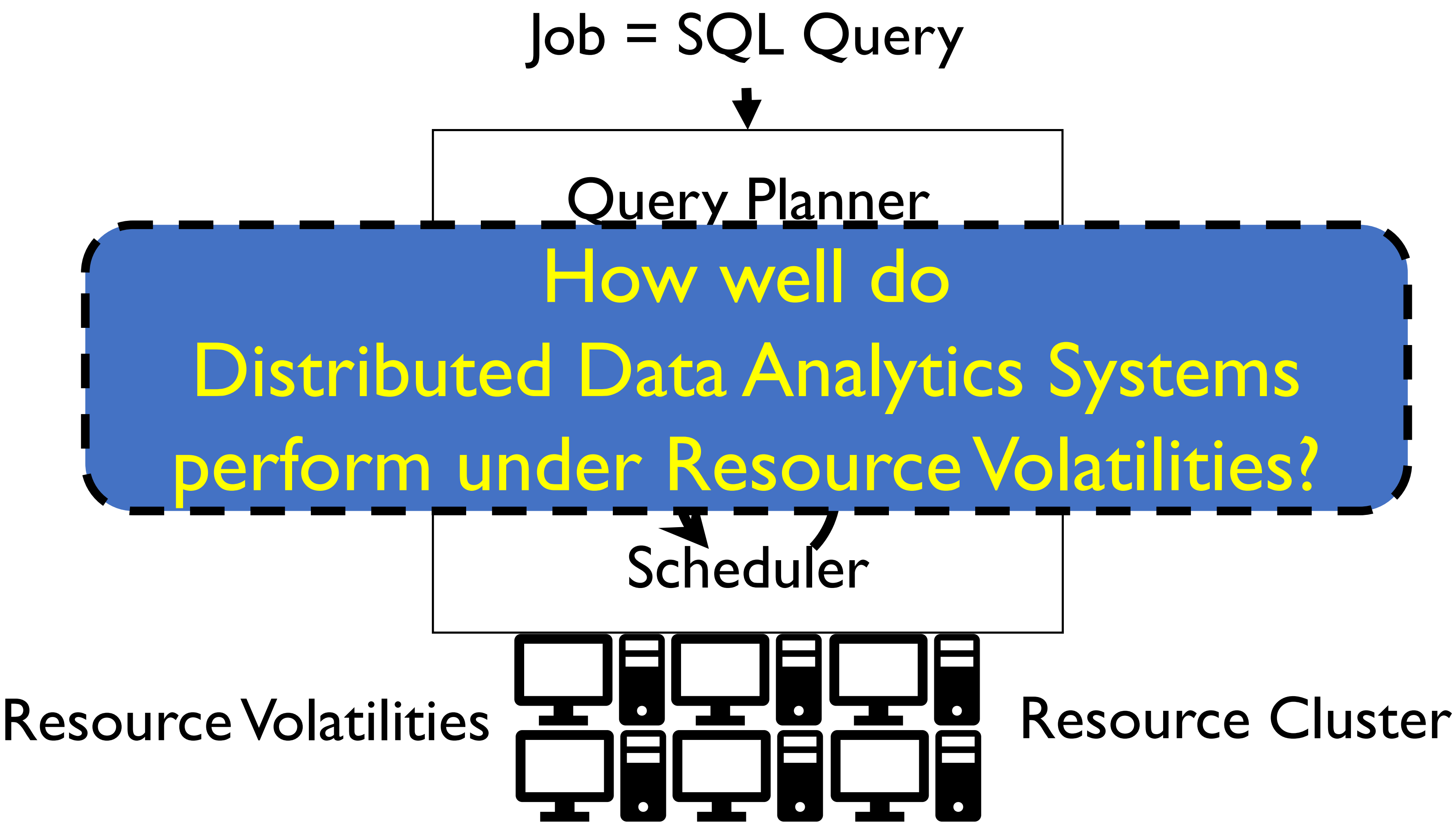
Resource Cluster

Overview – Resource Volatility; Small Cluster

- TPC-DS online workload + Carbyne (OSDI'16) scheduler managing 600 cores
- 38% queries experience at least one 20% resource volatility event

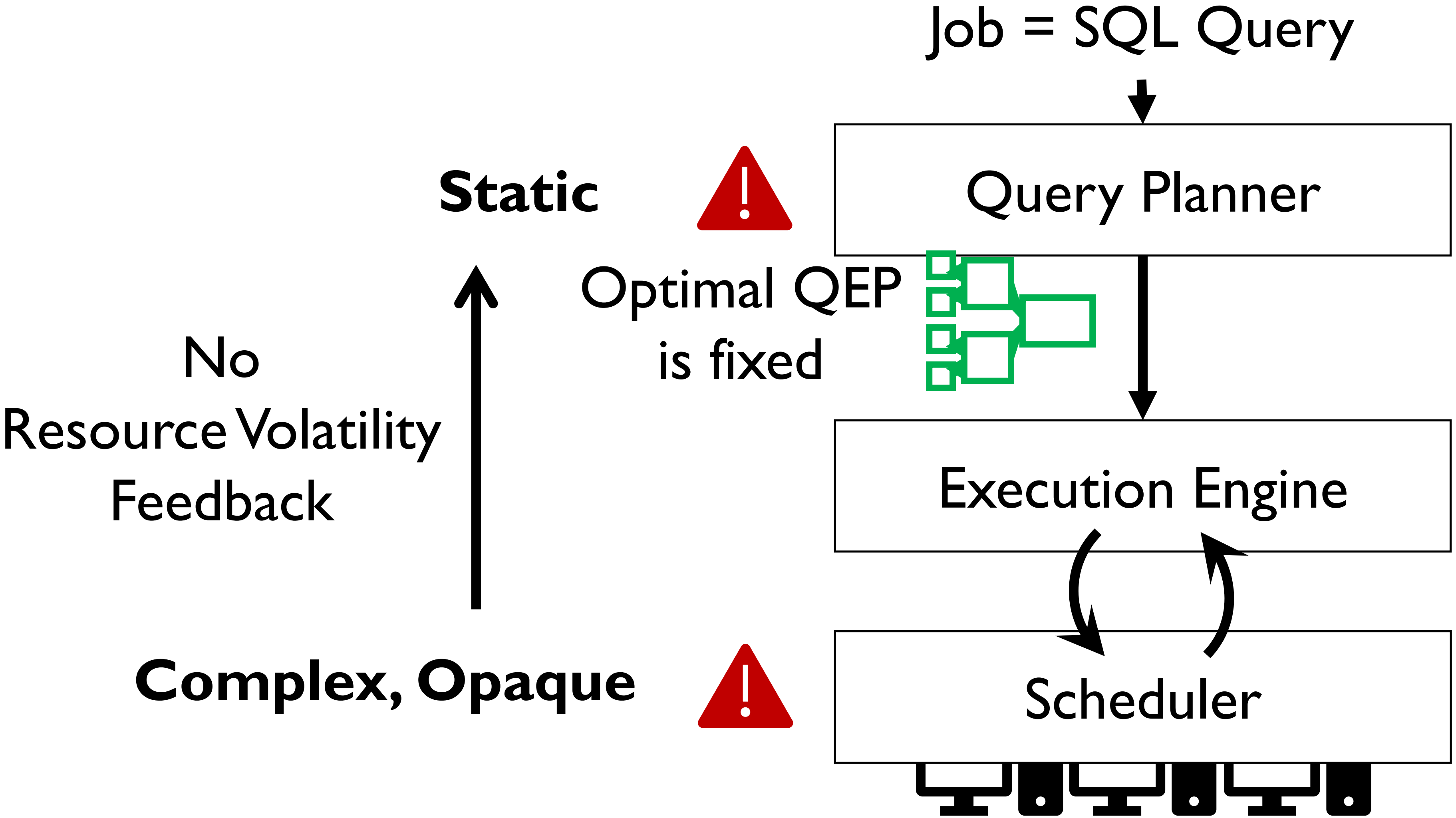


Motivating QOOP



How well do
Distributed Data Analytics Systems
perform under Resource Volatilities?

Motivating QOOP



Job = SQL Query

Query Planner

Optimal QEP is fixed

Execution Engine

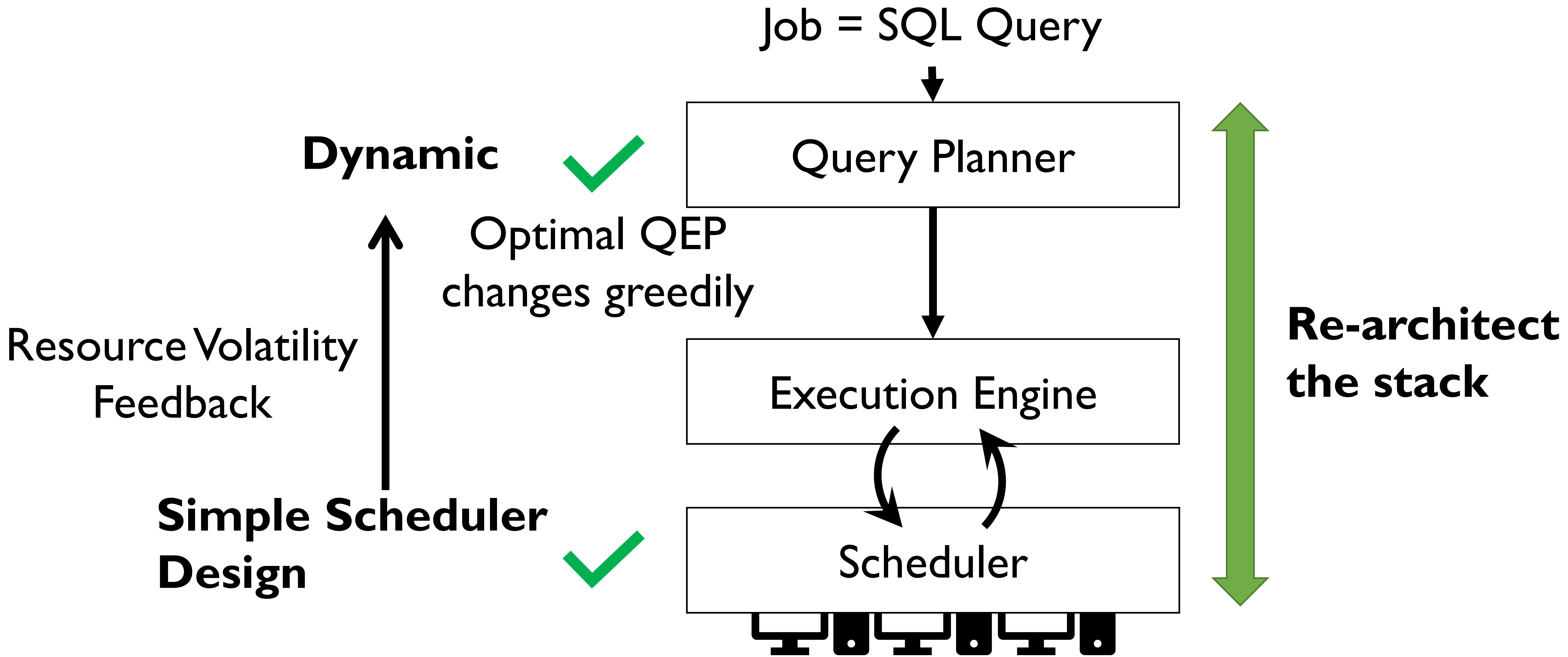
Scheduler

Static

No Resource Volatility Feedback

Complex, Opaque

Motivating QOOP



Job = SQL Query

Query Planner

Execution Engine

Scheduler

Dynamic ✓

Optimal QEP changes greedily

Resource Volatility Feedback

Simple Scheduler Design ✓

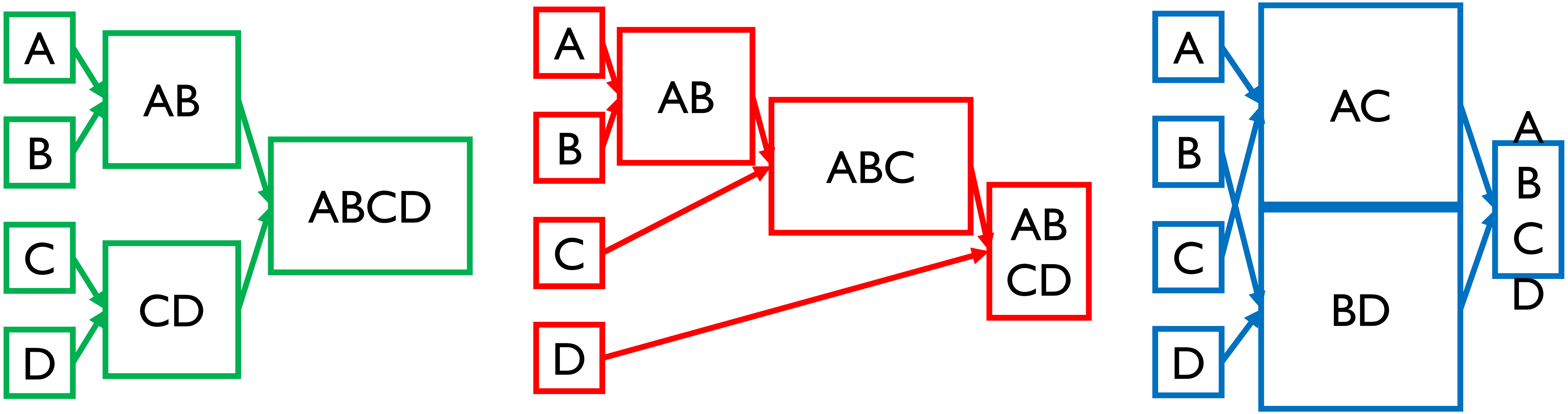
Re-architect the stack

Agenda

- Overview
  - Distributed Data Analytics Systems
  - Resource Volatilities
- **Overcoming Inefficiency #1**
  - **Static Query Planner**
  - **QOOP’s Dynamic QEP Switching**
- Overcoming Inefficiency #2
  - Complex and Opaque Scheduler
  - QOOP’s Scheduler Choice
- Implementation
- Evaluation

Static Query Planner; Example

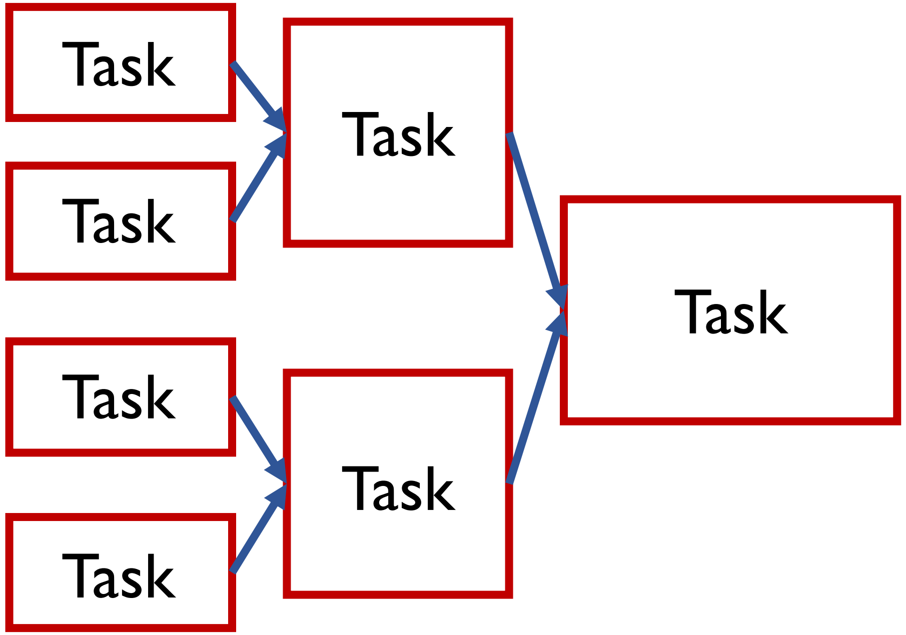
A join B join C join D



Three alternate Query Execution Plans (QEP's)
each with different join order

Static Query Planner; Terminology

What is a QEP?

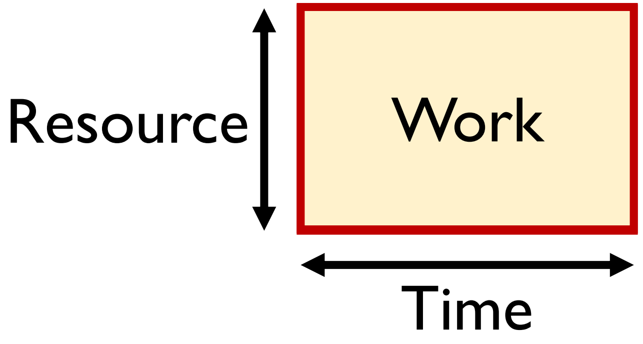


Directed Acyclic Graph (DAG)

Vertex: Task

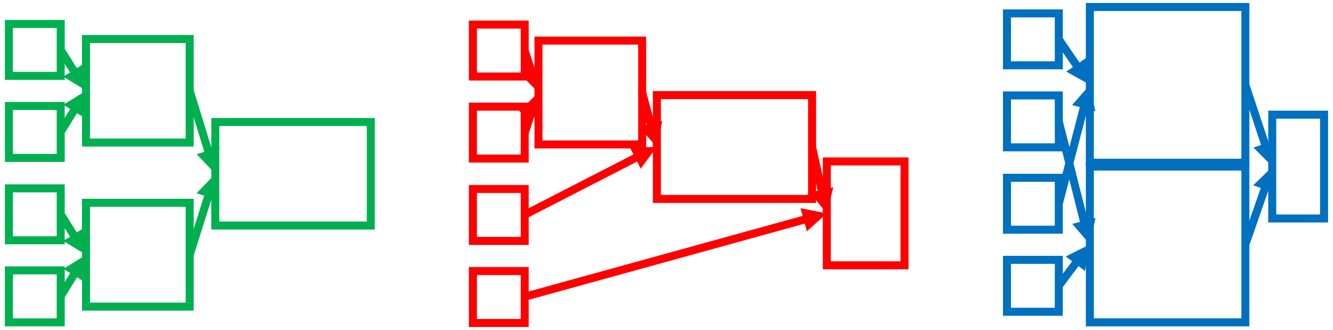
Edge: Dependency

What is a Task?



Static Query Planner; Example

A join B join C join D

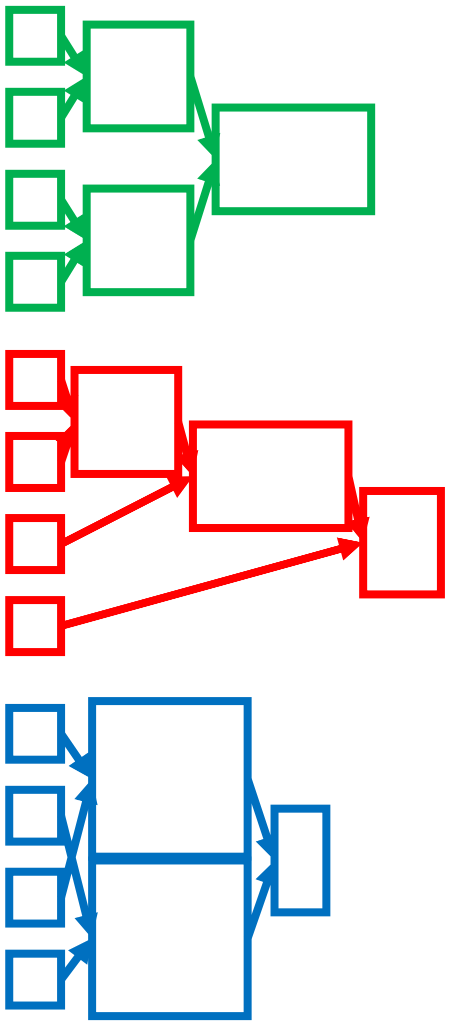


Choose an “optimal” QEP

Optimal – reduce query running time

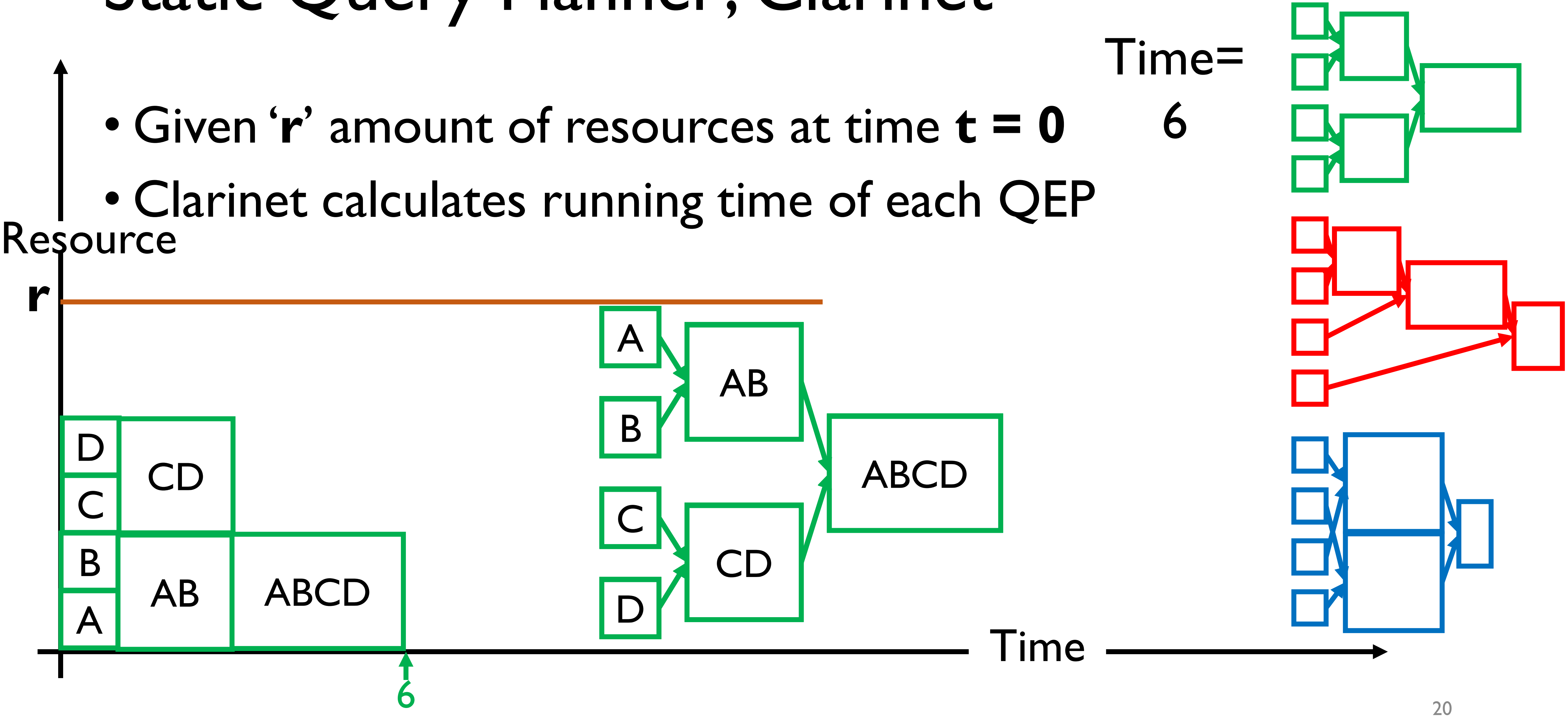
Static Query Planner; Clarinet

- **Clarinet (OSDI '16) Query Planner**
- Estimates network IO, memory, and compute resources just before job execution
- Estimates running time of each QEP by simulating execution
- Chooses QEP with minimum estimated running time



Static Query Planner; Clarinet

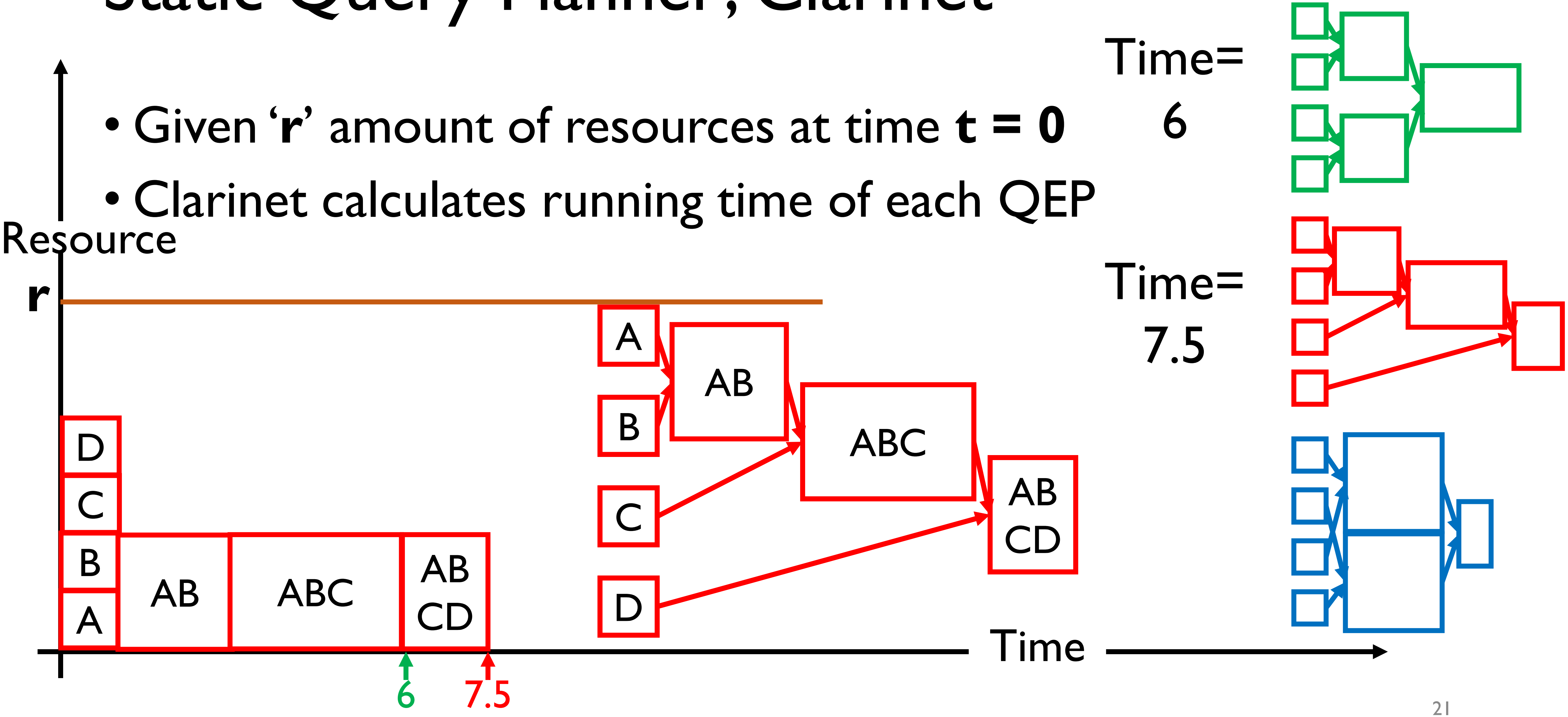
- Given ‘**r**’ amount of resources at time **t = 0**
- Clarinet calculates running time of each QEP



Time=
6

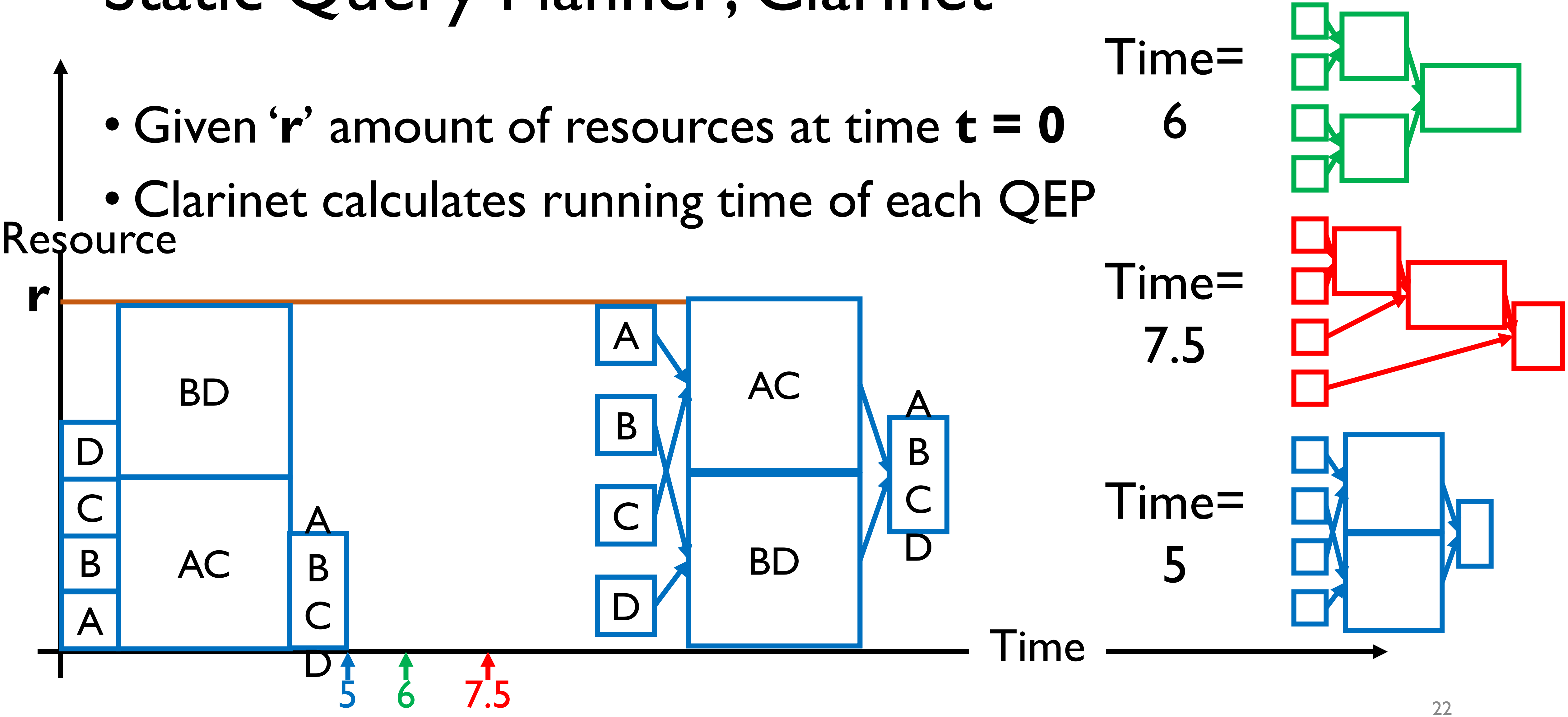
Static Query Planner; Clarinet

- Given ‘**r**’ amount of resources at time **t = 0**
- Clarinet calculates running time of each QEP



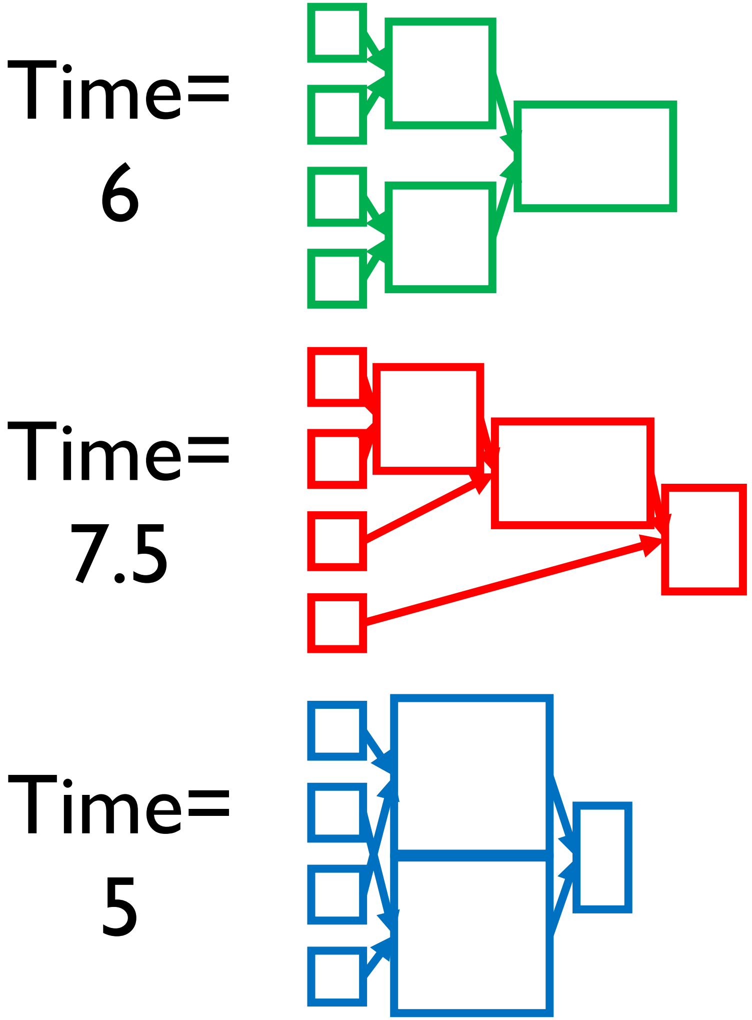
Static Query Planner; Clarinet

- Given ‘**r**’ amount of resources at time **t = 0**
- Clarinet calculates running time of each QEP



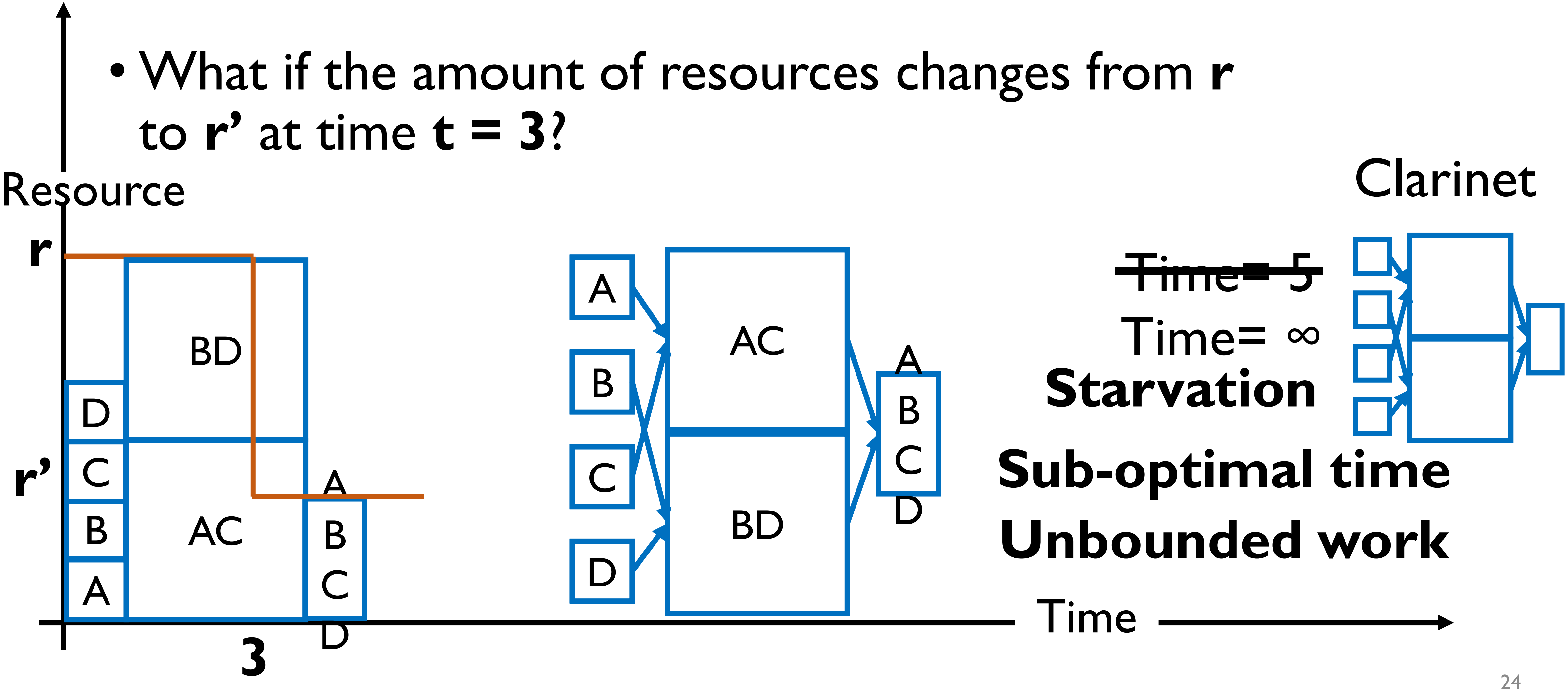
Static Query Planner

- Given ‘**r**’ amount of resources at time **t = 0**
- Clarinet calculates running time of each QEP
- Clarinet chooses Blue Plan
- However this choice is **static** and does not change during job’s lifetime



Static Query Planner; Bad Outcomes

- What if the amount of resources changes from **r** to **r’** at time **t = 3**?



Clarinet

~~Time= 5~~

Time= ∞

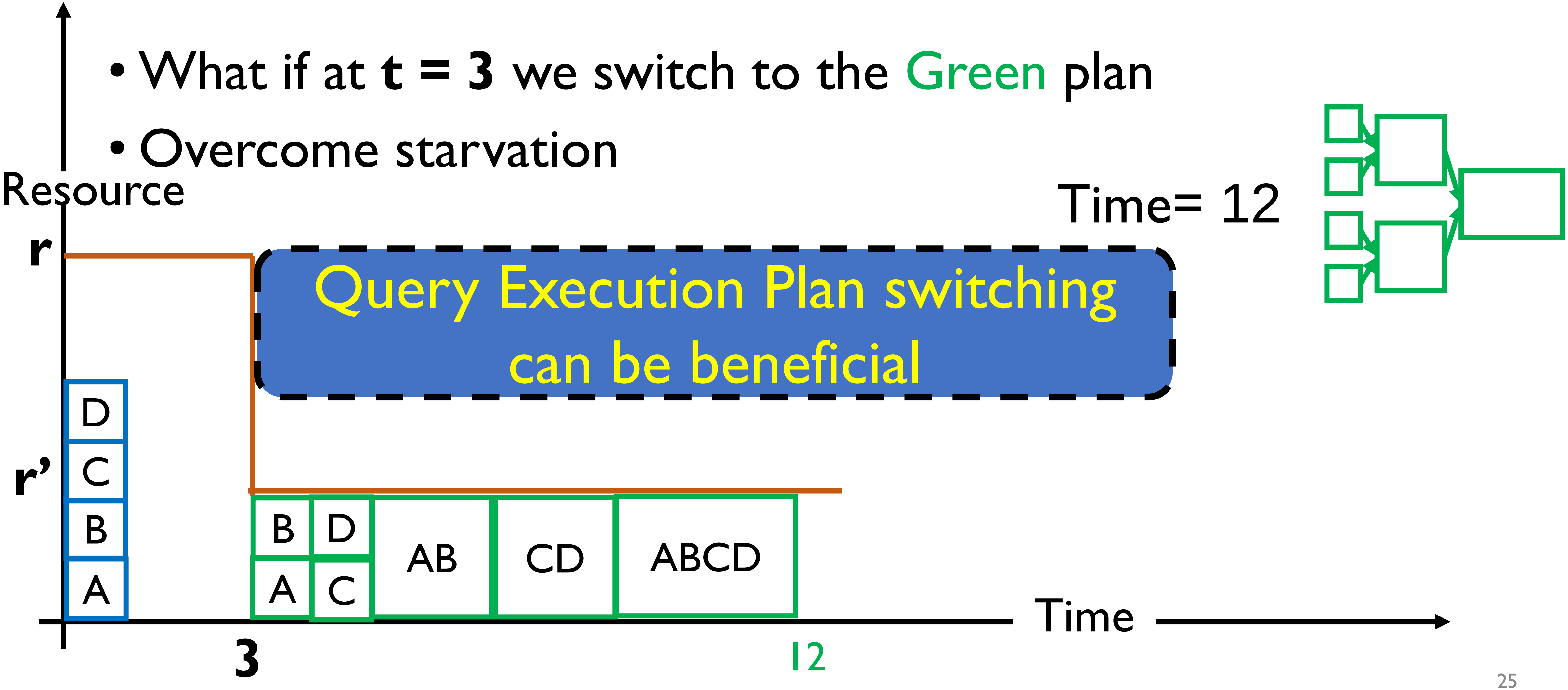
Starvation

Sub-optimal time

Unbounded work

Motivating QOOP's Dynamic QEP switching

- What if at **t = 3** we switch to the Green plan
- Overcome starvation



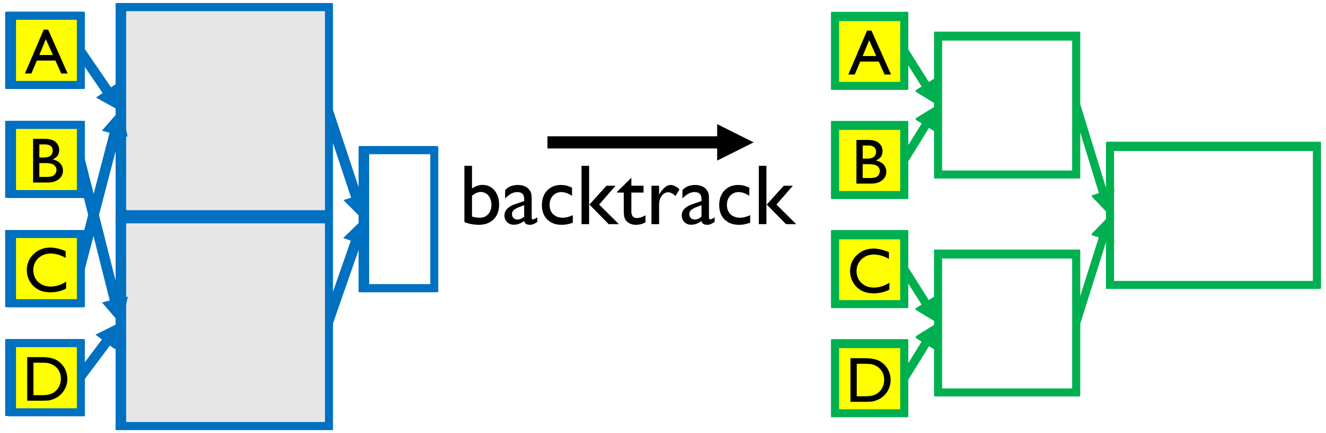
Query Execution Plan switching can be beneficial

QOOP – Dynamic QEP switching

- **Static QEP** – under adversarial resource volatilities can lead to **bad outcomes**
  - Sub-Optimal behavior
  - Starvation
  - Unbounded work
- To overcome – QOOP proposes **dynamic QEP switching** –
  - **Backtracking**
  - **Checkpointing**
  - **Greedy behavior**

Dynamic QEP switching; Backtracking

- Switch from the Blue QEP to the Green QEP
- Backtracking – sacrifice current work and redo work in prior stages



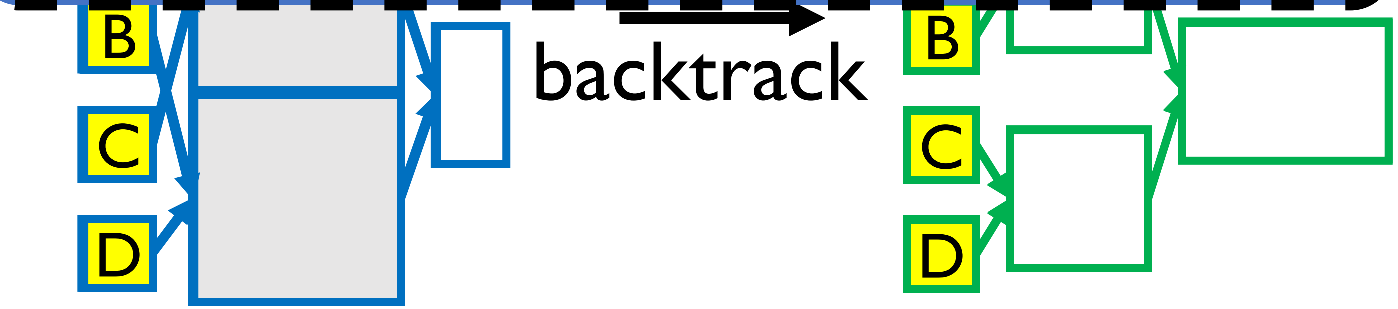
sacrifice partial work

repeat work from prior stages

Dynamic QEP switching; Backtracking

- Switch from the Blue QEP to the Green QEP
- Backtracking – sacrifice current work and redo work in prior stages

Only re-plan future work?



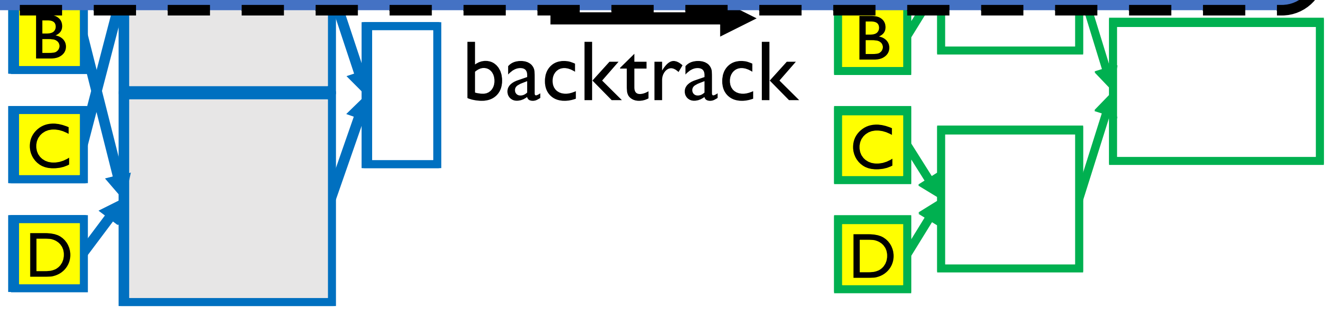
sacrifice partial work

repeat work from prior stages

Dynamic QEP switching; Backtracking

- Switch from the Blue QEP to the Green QEP
- Backtracking – sacrifice current work and redo work in prior stages

Backtracking essential to avoid bad outcomes



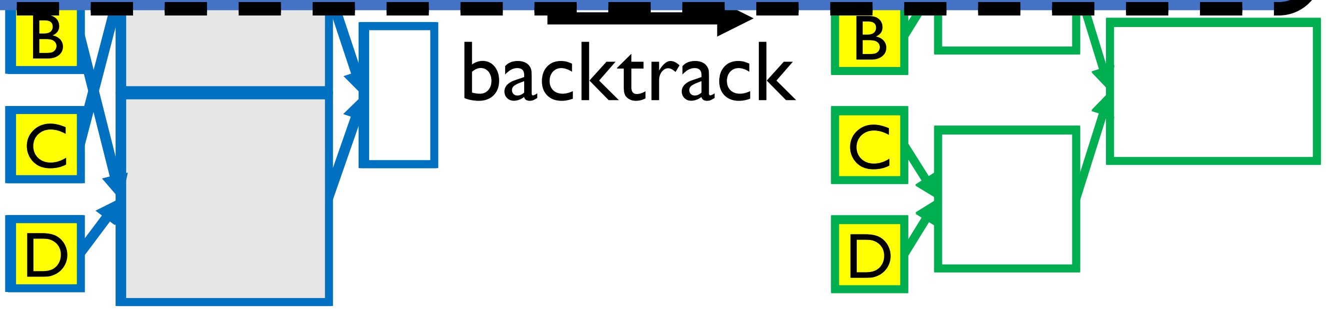
sacrifice partial work

repeat work from prior stages

Dynamic QEP switching; Backtracking

- Switch from the Blue QEP to the Green QEP
- Backtracking – sacrifice current work and redo work in prior stages

What if we keep repeating work in an unbounded manner?

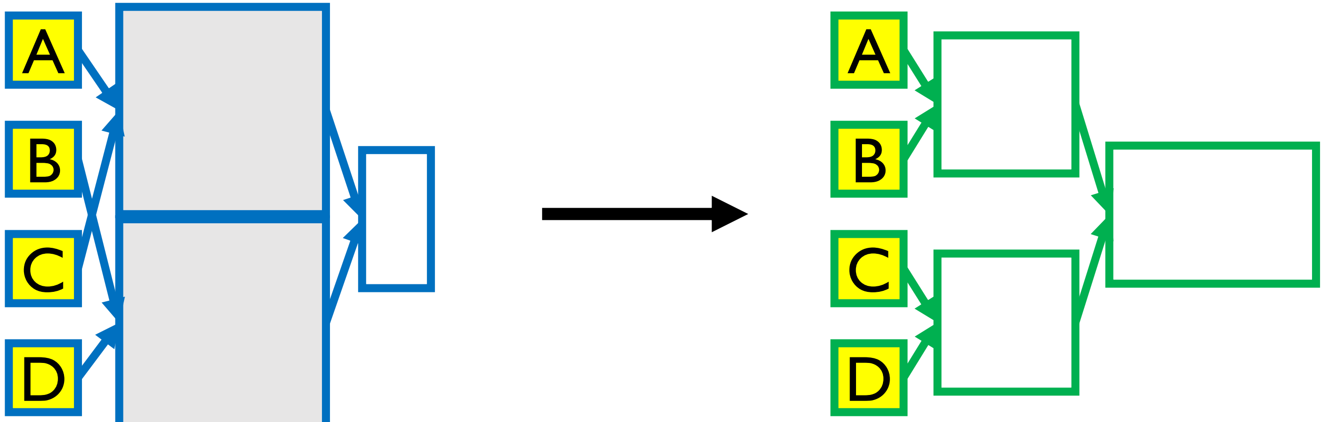


sacrifice partial work

repeat work from prior stages

Dynamic QEP switching; Checkpointing

- Checkpoint and resume from checkpoints to bound work
- Switch to Green QEP resumes from checkpoint



bound repeated work and resume from checkpoint

Dynamic QEP switching; Checkpointing

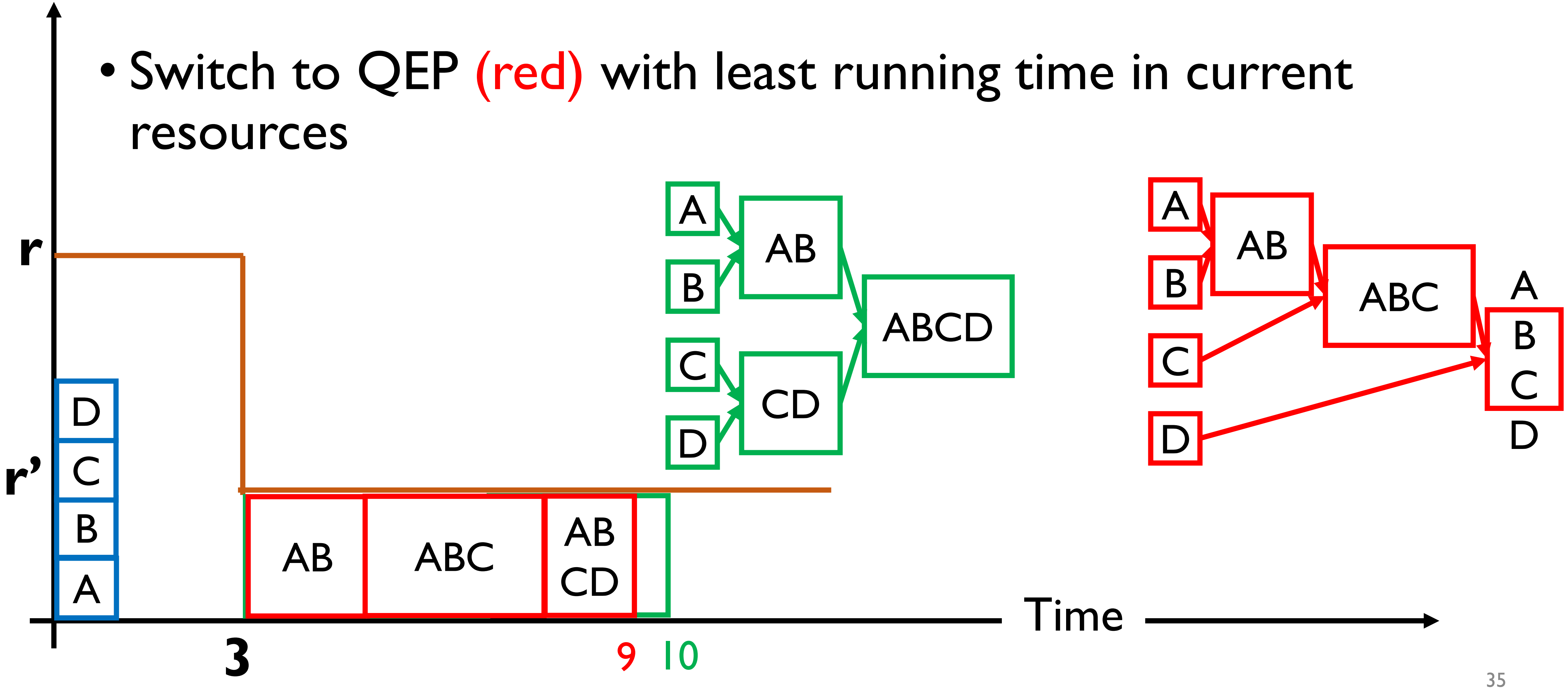
- Checkpoint and resume from checkpoints to bound work
- Switch to Green QEP resumes from checkpoint

A
A
Which QEP to switch to?
D
D

bound repeated work and resume from checkpoint

Dynamic QEP switching; Greedy

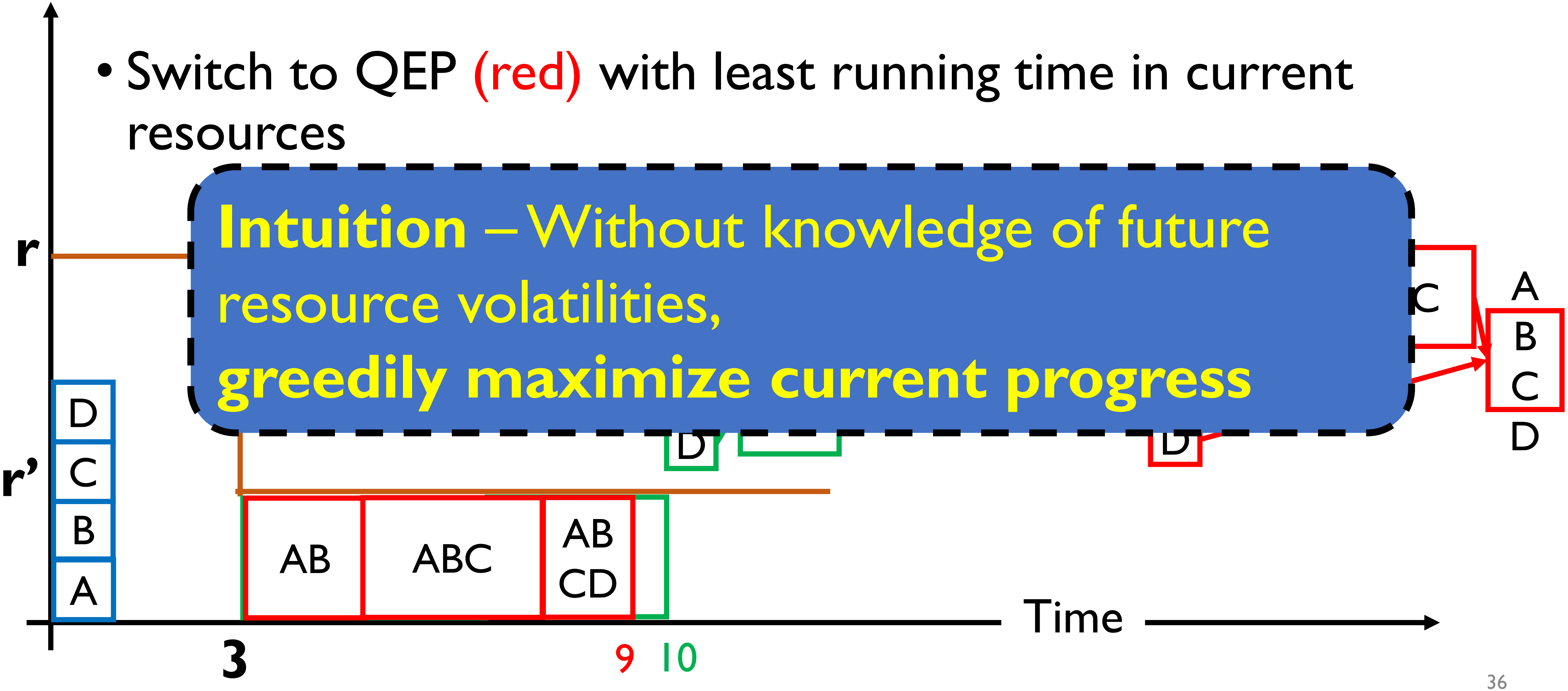
- Switch to QEP (red) with least running time in current resources



Dynamic QEP switching; Greedy

- Switch to QEP (red) with least running time in current resources

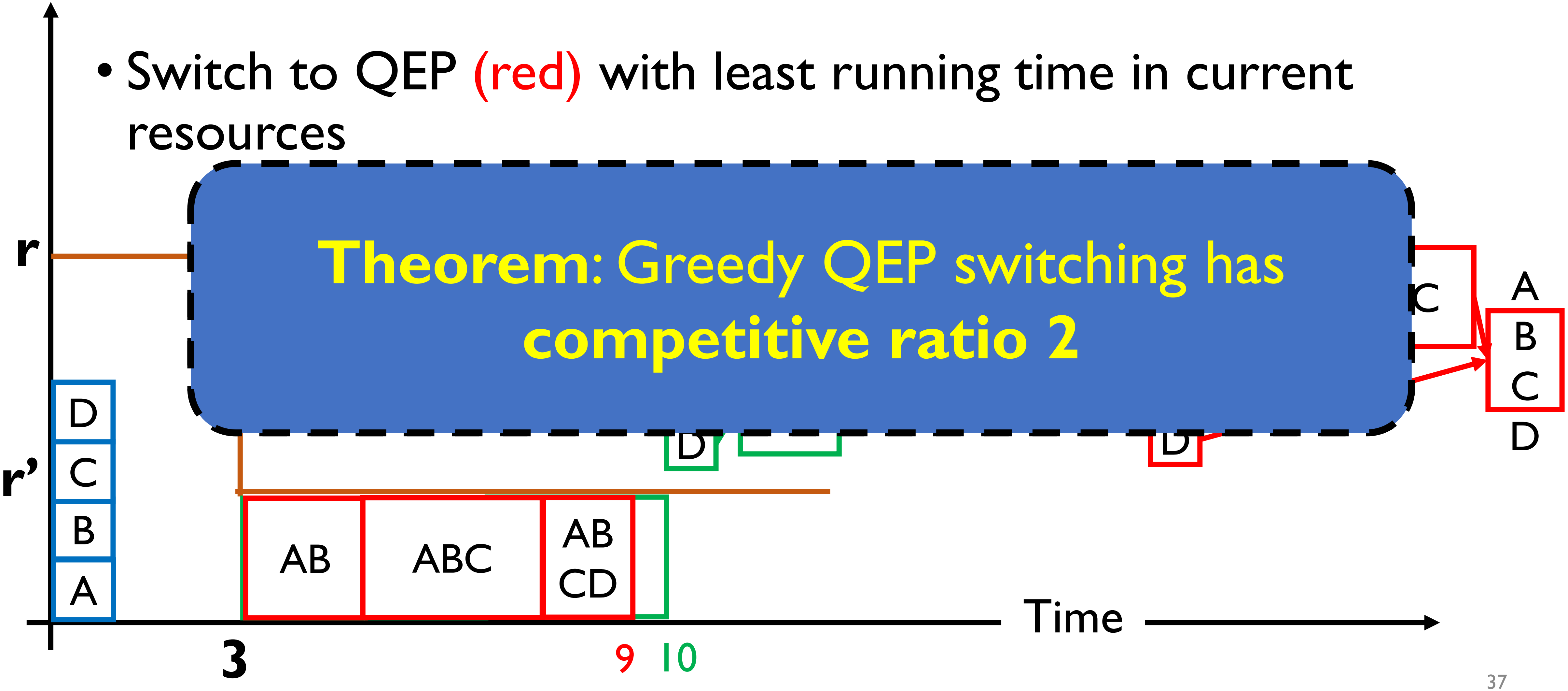
Intuition – Without knowledge of future resource volatilities, **greedily maximize current progress**



Dynamic QEP switching; Greedy

- Switch to QEP (red) with least running time in current resources

Theorem: Greedy QEP switching has **competitive ratio 2**

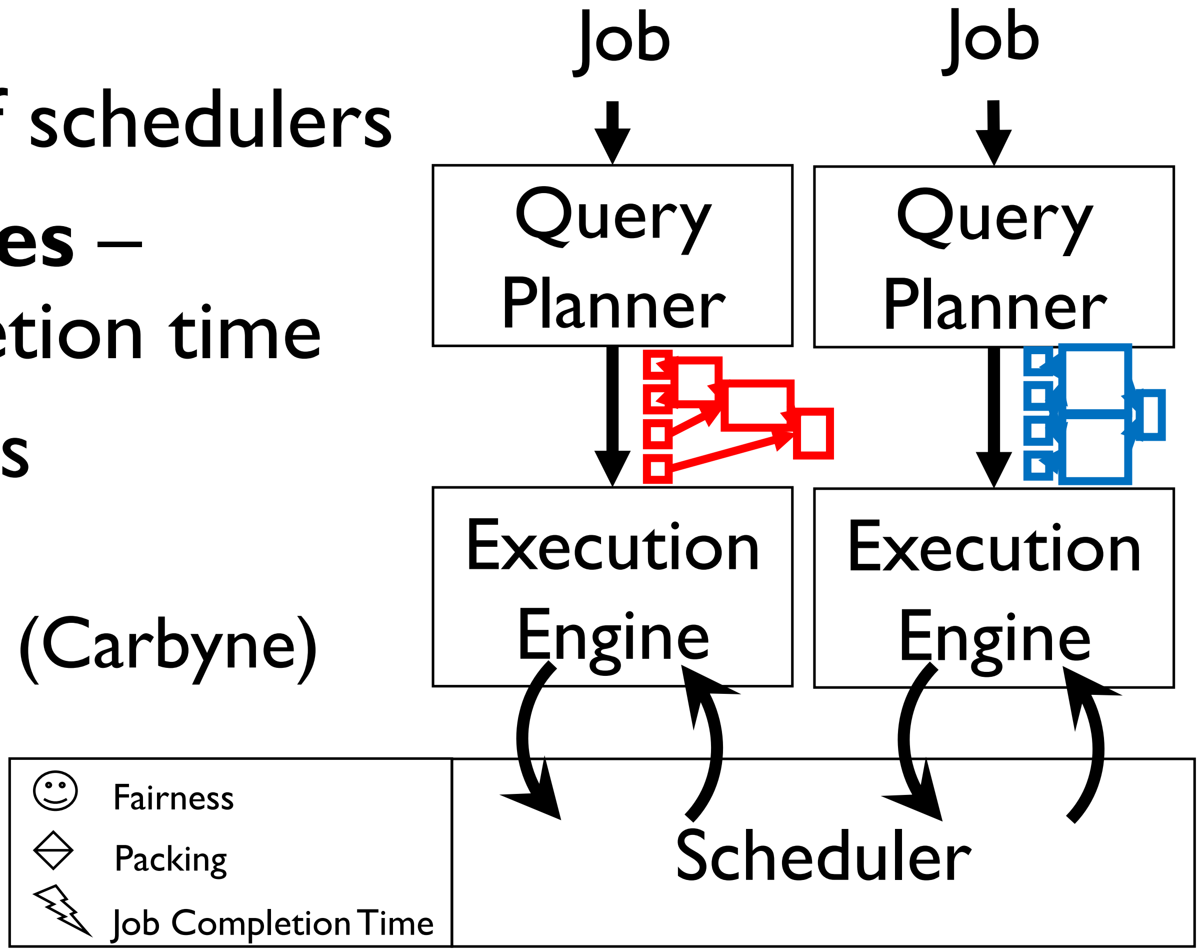


Agenda

- Overview
  - Distributed Data Analytics Systems
  - Resource Volatilities
- Overcoming Inefficiency #1
  - Static Query Planner
  - QOOP's Dynamic QEP Switching
- **Overcoming Inefficiency #2**
  - **Complex and Opaque Scheduler**
  - **QOOP's Scheduler Choice**
- Implementation
- Evaluation

Complex and Opaque Schedulers

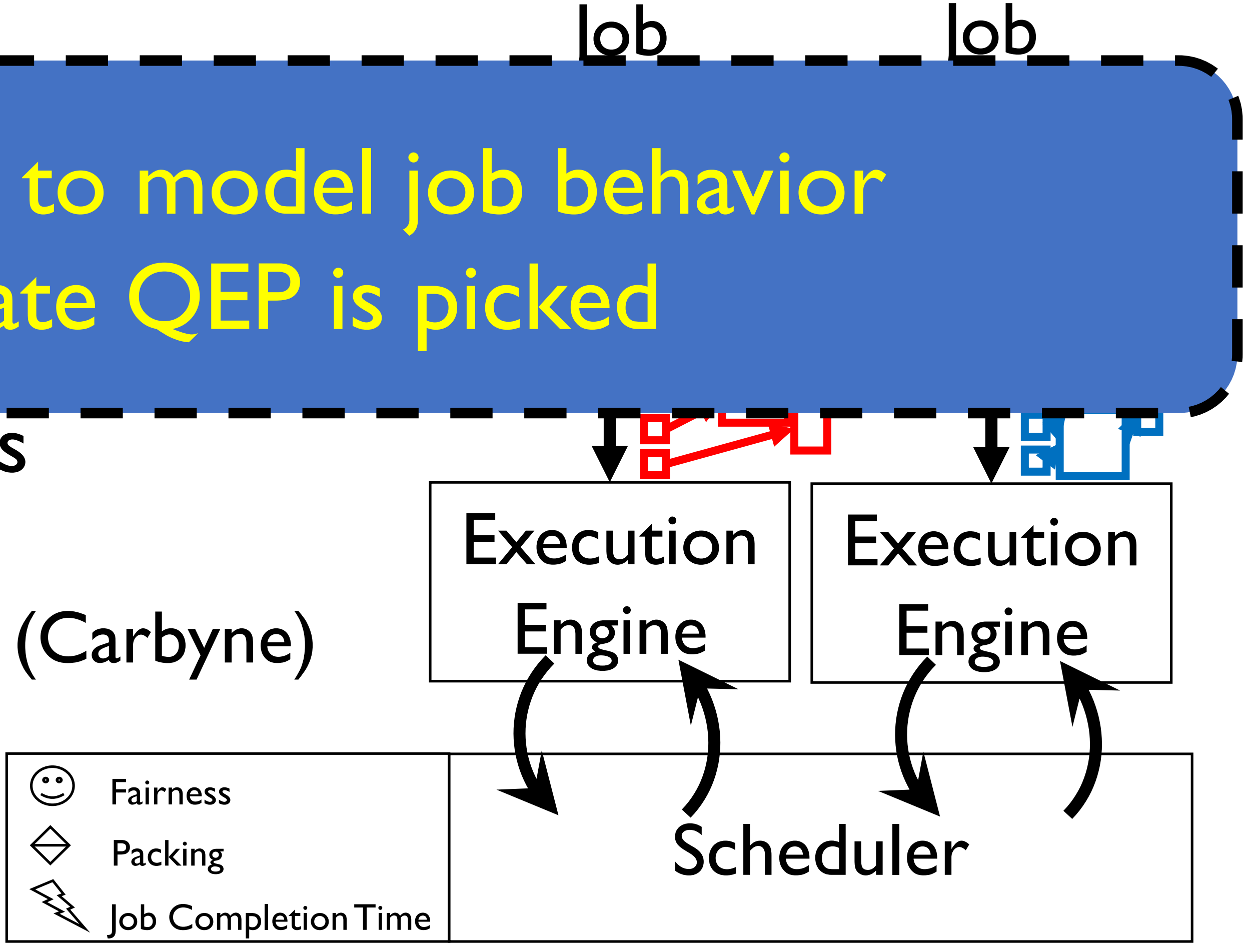
- **Increasing complexity** of schedulers
- Manage **multiple objectives** – fairness, packing, job completion time
- **QEP-dependent** heuristics
  - Task Size – better fit (Tetris)
  - Dependencies – critical path (Carbyne)



Complex and Opaque Schedulers

Opaque – Hard to model job behavior
if an alternate QEP is picked

- **QEP-dependent** heuristics
  - Task Size – better fit (Tetris)
  - Dependencies – critical path (Carbyne)

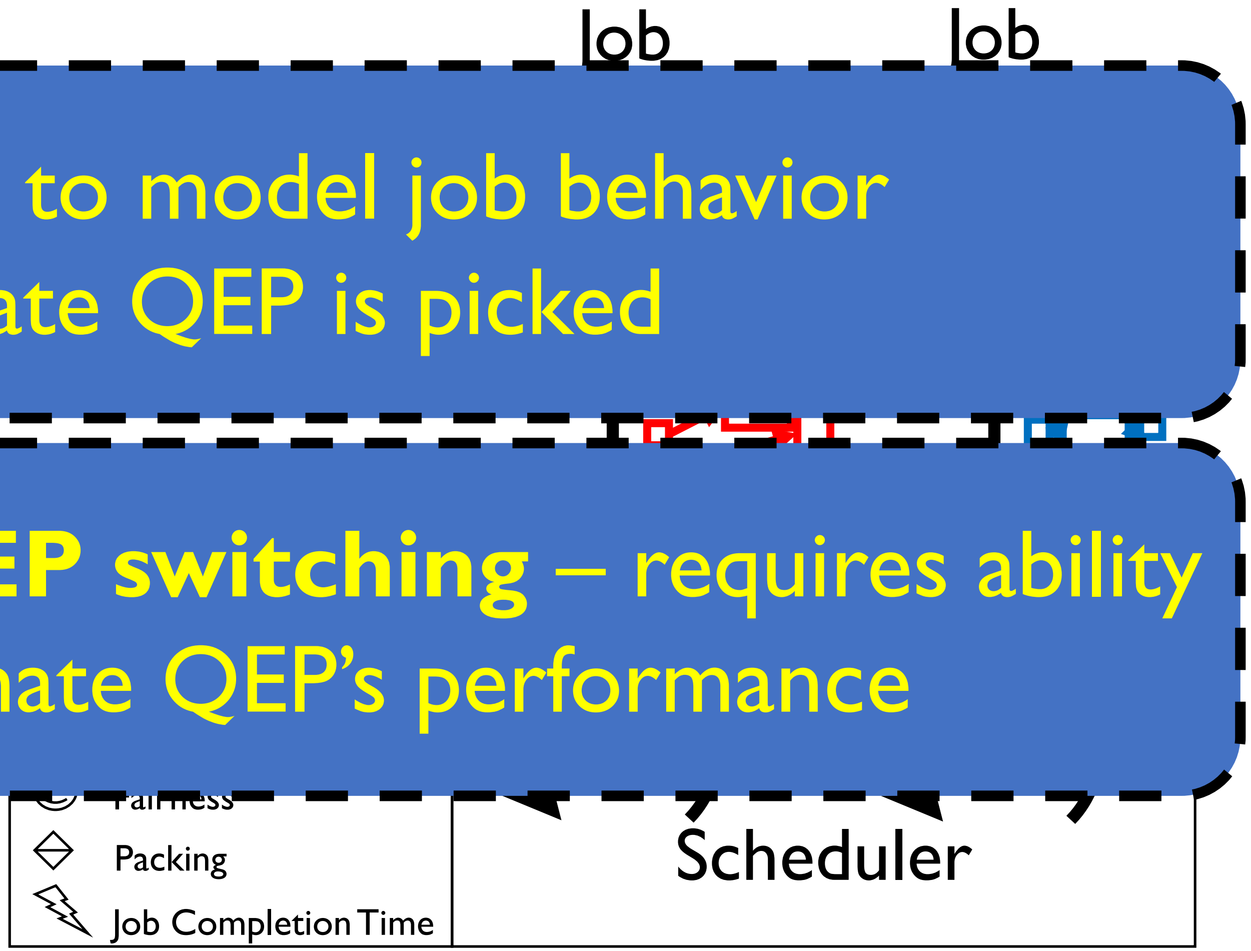


Complex and Opaque Schedulers

Opaque – Hard to model job behavior if an alternate QEP is picked

- **QEP dependent heuristics**

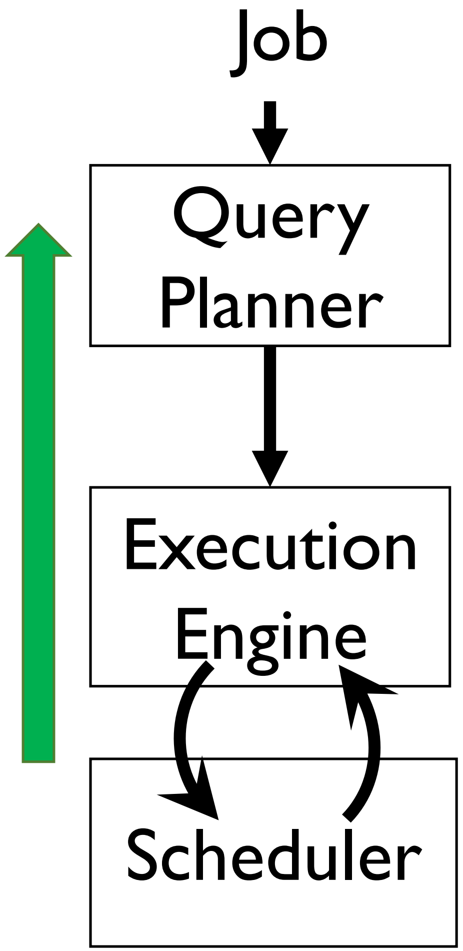
Obstructs **Dynamic QEP switching** – requires ability to estimate alternate QEP's performance



QOOP's Scheduler Choice

- We go back to a simple **QEP independent scheduler –** simple max-min fair scheduler
- Each job gets a fair **resource share guarantee**
- Enables **feedback** about resource volatilities
- Supports **dynamic QEP switching**

Resource Share = Total Resources / # Active Queries



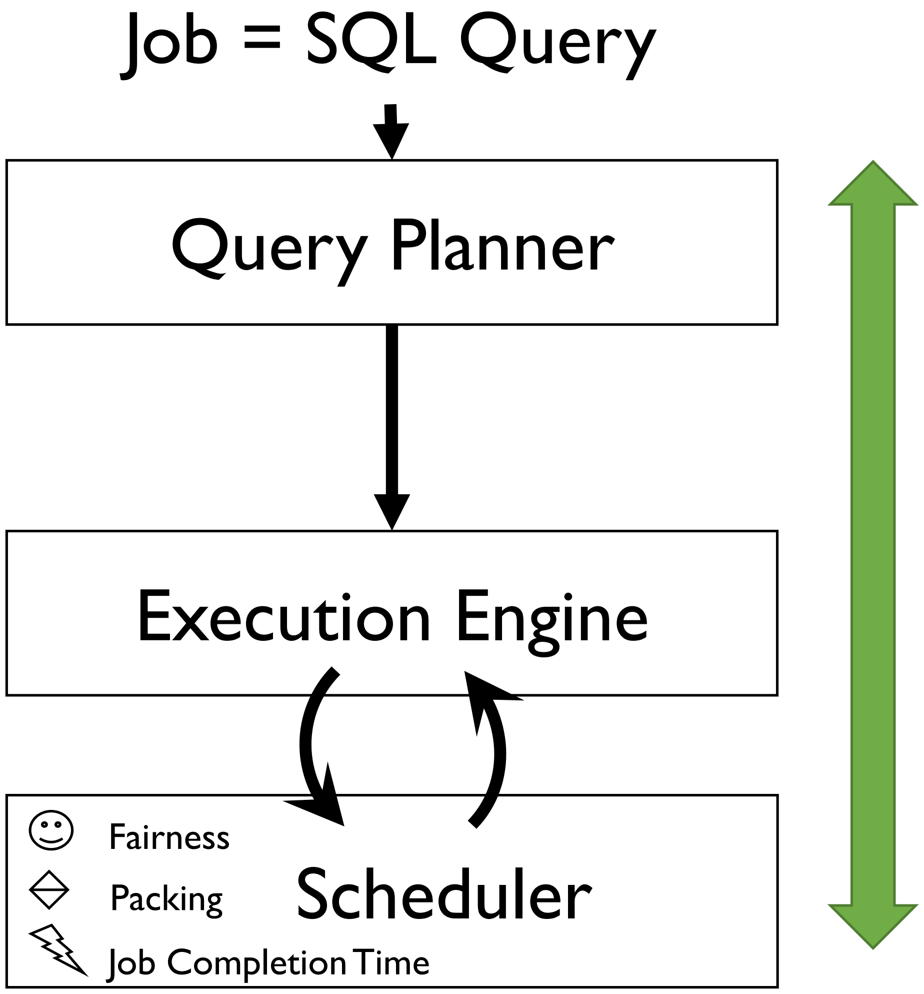
QOOP Overall Design

Dynamic Greedy dynamic QEP switching ✓

Resource Volatility feedback

= change in resource share

Simple Scheduler Design ✓



Re-architect the stack

Agenda

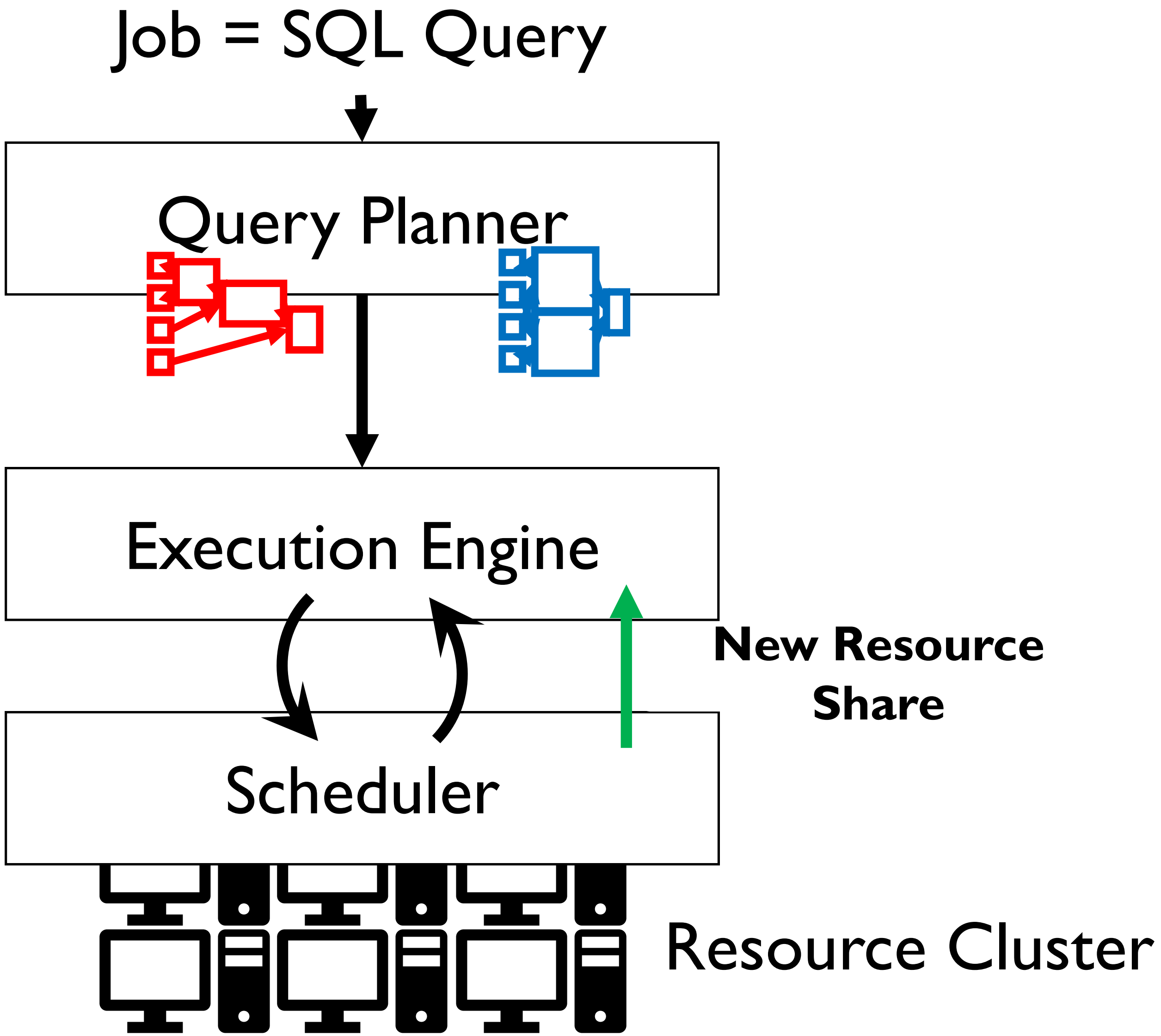
- Overview
  - Distributed Data Analytics Systems
  - Resource Volatilities
- Overcoming Inefficiency #1
  - Static Query Planner
  - QOOP's Dynamic QEP Switching
- Overcoming Inefficiency #2
  - Complex and Opaque Scheduler
  - QOOP's Scheduler Choice
- **Implementation**
- **Evaluation**

QOOP Implementation

Hive – Cache multiple QEP's and send to Tez

Tez – estimate runtime of QEP's and greedy switch

YARN – simple max-min fair with feedback



QOOP Evaluation

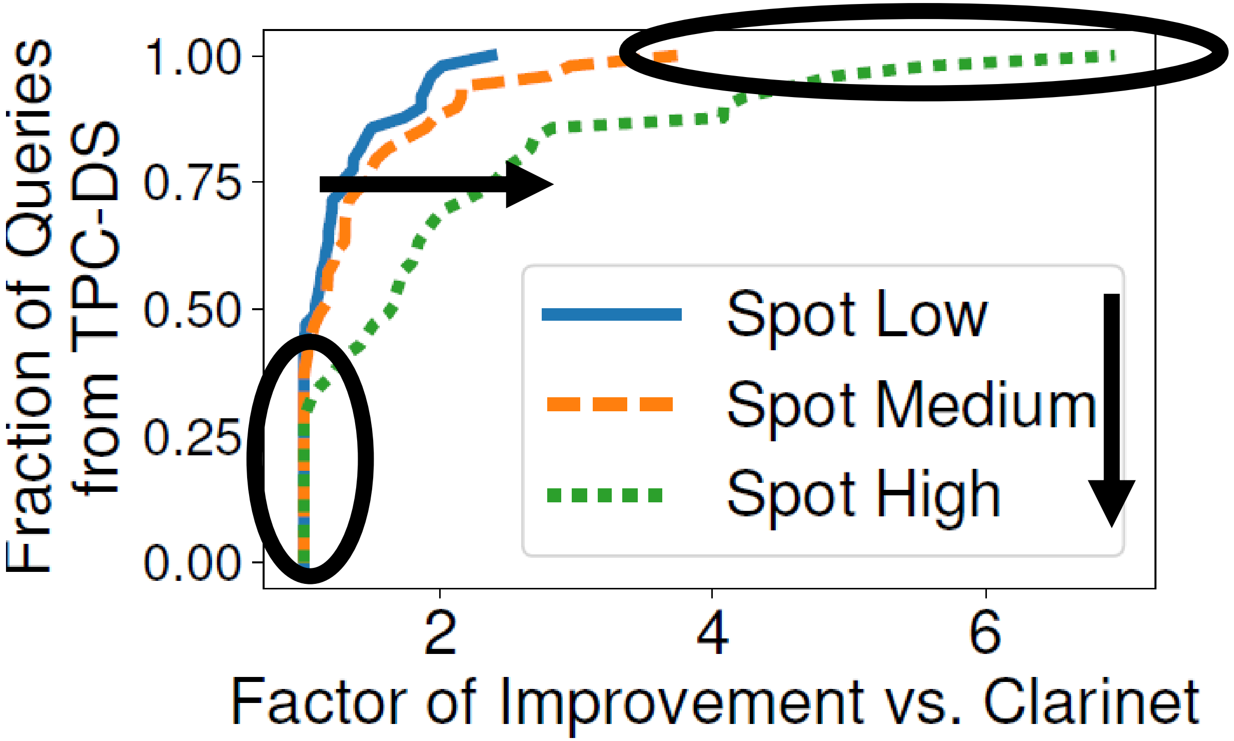
- Testbed –
    - 20 bare-metal servers
- Micro-benchmark Workload –
    - Single Query under different spot market resource volatility regimes

Regime	Volatility%
Low	< 10%
Medium	10% - 20%
High	> 20%

- Macro-benchmark Workload –
    - 200 queries randomly drawn from TPC-DS
    - Online arrival of queries following Poisson process

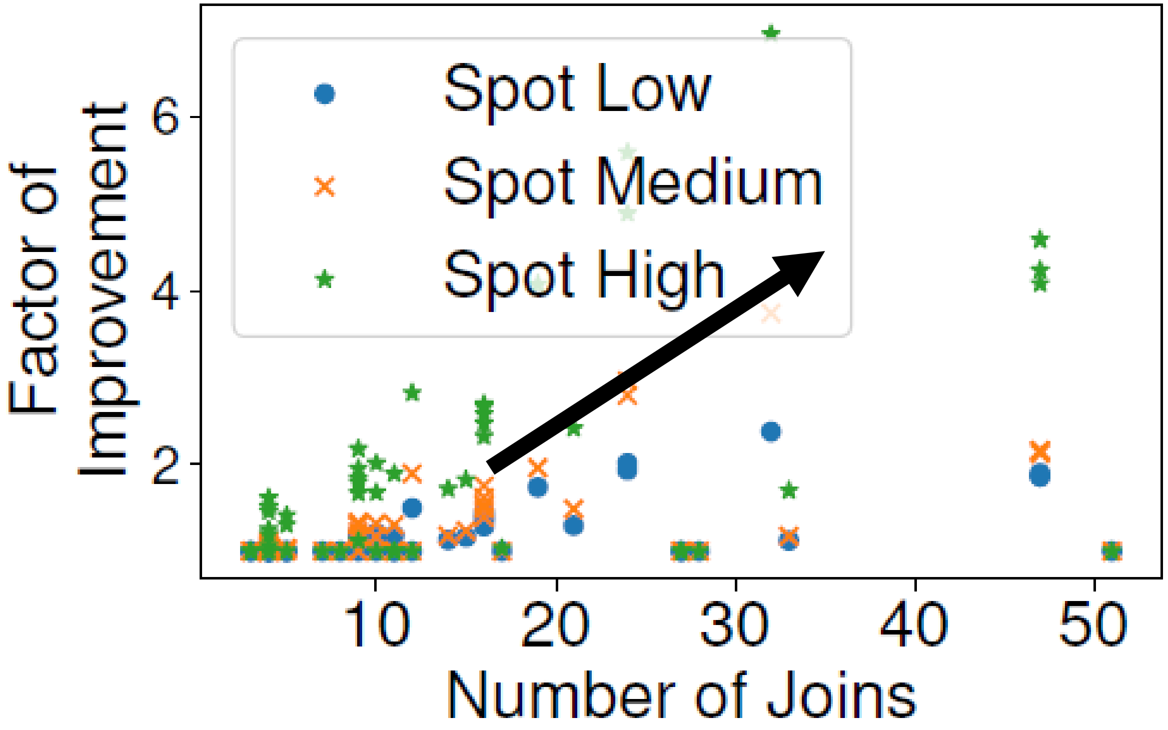
QOOP Evaluation – Micro-benchmark

- Factor of Improvement = Running Time with Clarinet / Running Time with QOOP
- Gains increase with increasing resource volatility
- ~10% jobs > 4x gains
- ~35% queries see no improvements –
  - low complexity queries
  - low duration queries



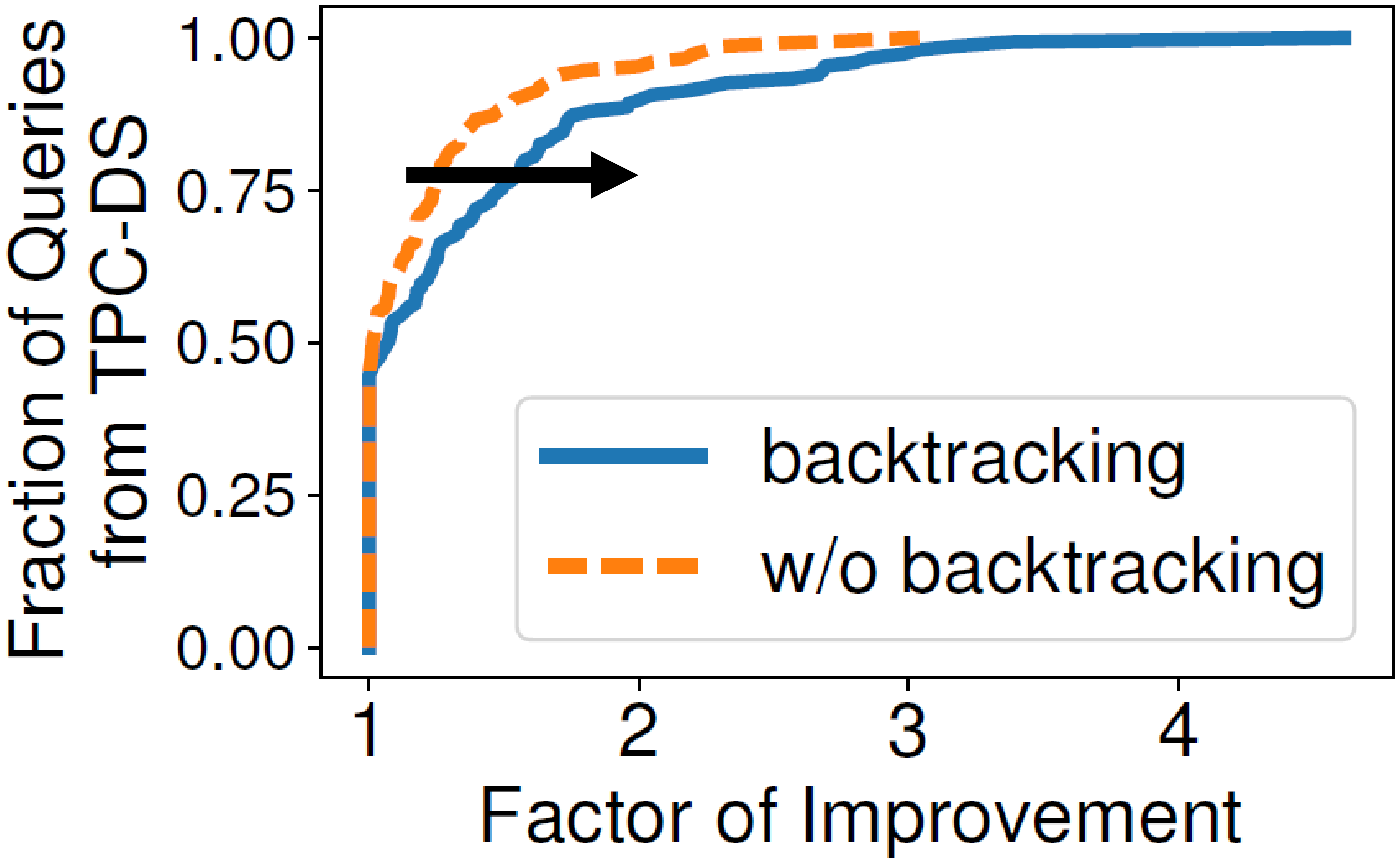
QOOP Evaluation – Micro-benchmark

- Increasing complexity i.e. number of joins => higher gains
- More alternative QEP's => higher likelihood to find a better QEP switch



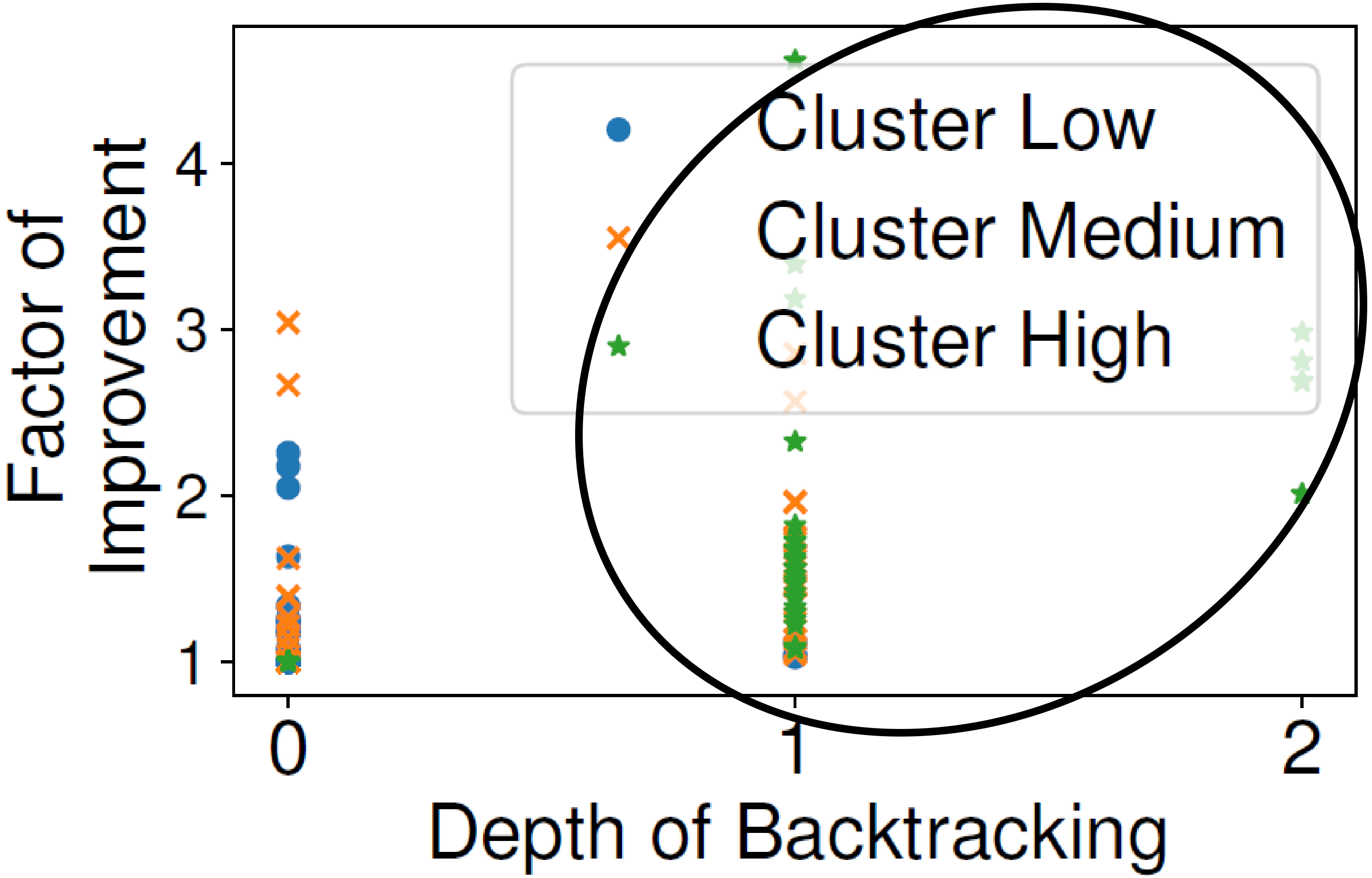
QOOP Evaluation – Micro-benchmark

- Backtracking is beneficial



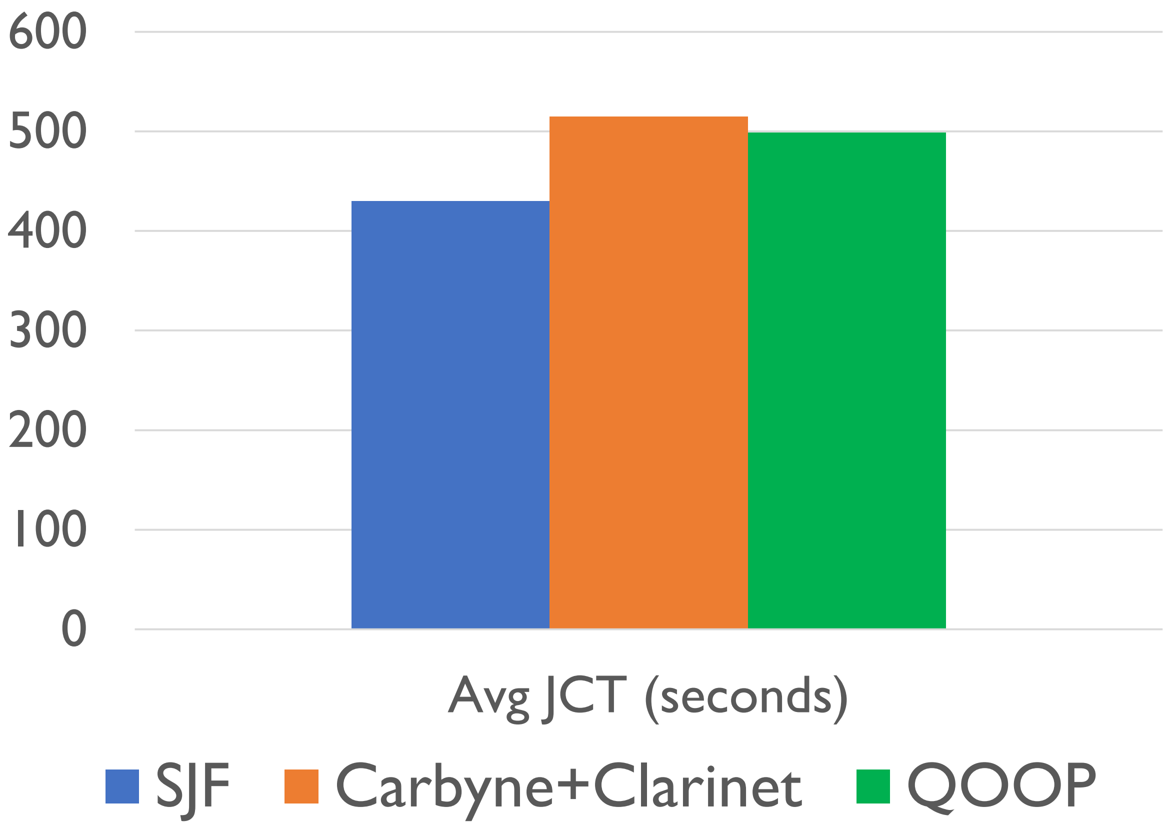
QOOP Evaluation – Micro-benchmark

- Backtracking is beneficial
- 5.7% of all QEP switches involve backtracking
  - pre-dominantly due to high resource volatility
  - at-most 2 stages deep



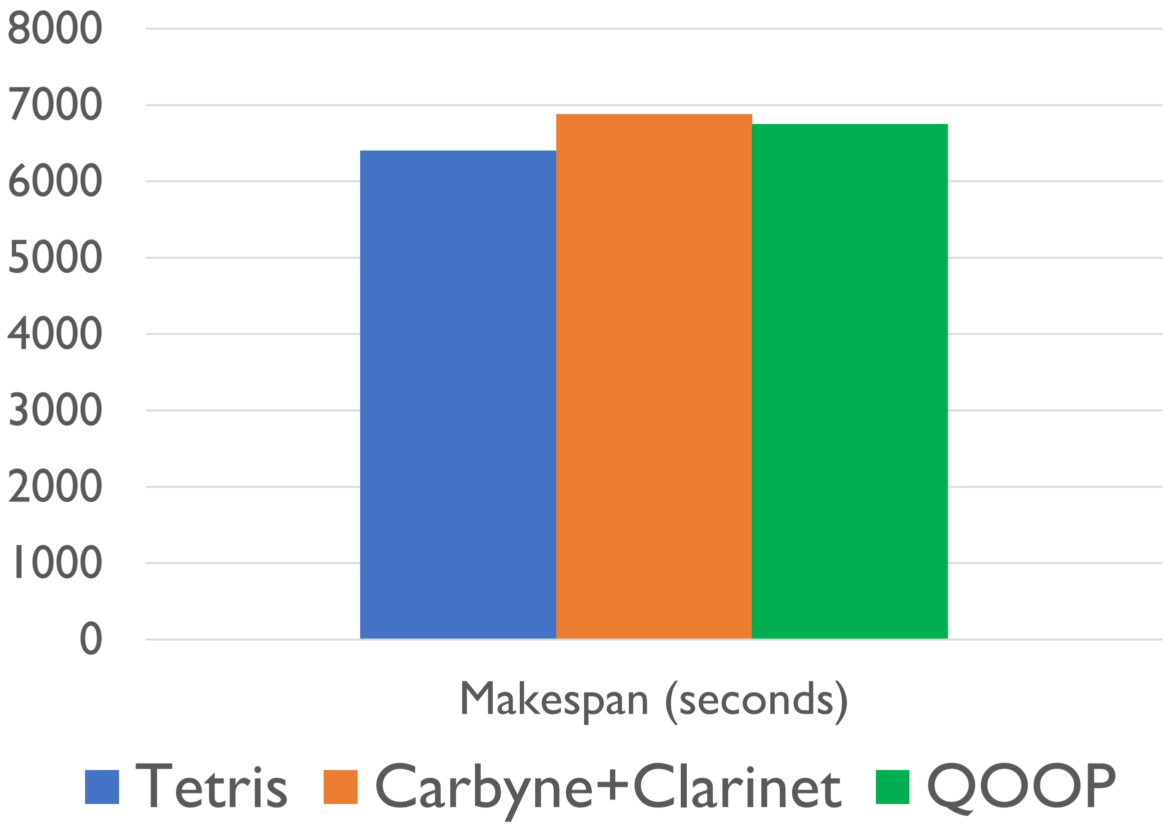
QOOP Evaluation – Macro-benchmark

- Job Performance
- Carbyne (OSDI'16) + Clarinet (OSDI'16) – two complex solutions put together
- Closest to ideal baseline SJF – even with a simple max-min fair scheduler



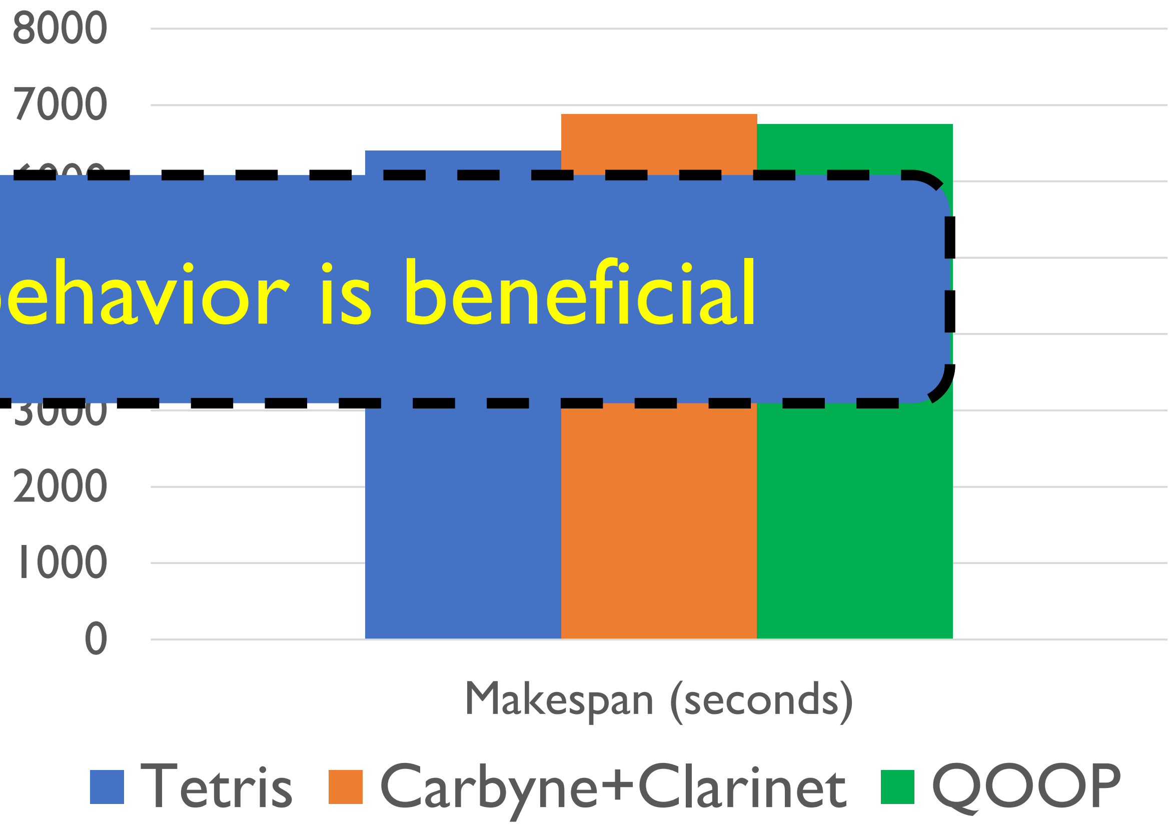
QOOP Evaluation – Macro-benchmark

- Cluster Efficiency
- Carbyne (OSDI'16) + Clarinet (OSDI'16) – two complex solutions put together
- Closest to ideal baseline Tetris – even with a simple max-min fair scheduler



QOOP Evaluation – Macro-benchmark

- Cluster Efficiency
- Carbyne (OSDI'16) +
  Clar
  com
- Closest to ideal baseline Tetris – even with a simple max-min fair scheduler



Each job's greedy behavior is beneficial

QOOP Summary

- Resource volatilities exist in practice
- QOOP is suited for distributed data analytics under resource volatilities
  - Simple scheduler choice + feedback
  - Dynamic QEP switching at the Query Planner

Thank you!
Poster #40
Questions?

Backup Slide – Prevalence of Small Clusters

#Machine	% Users
1 - 99	75%
100-1000	21%
1000+	4%

Reference: Mesosphere Survey, 2016.