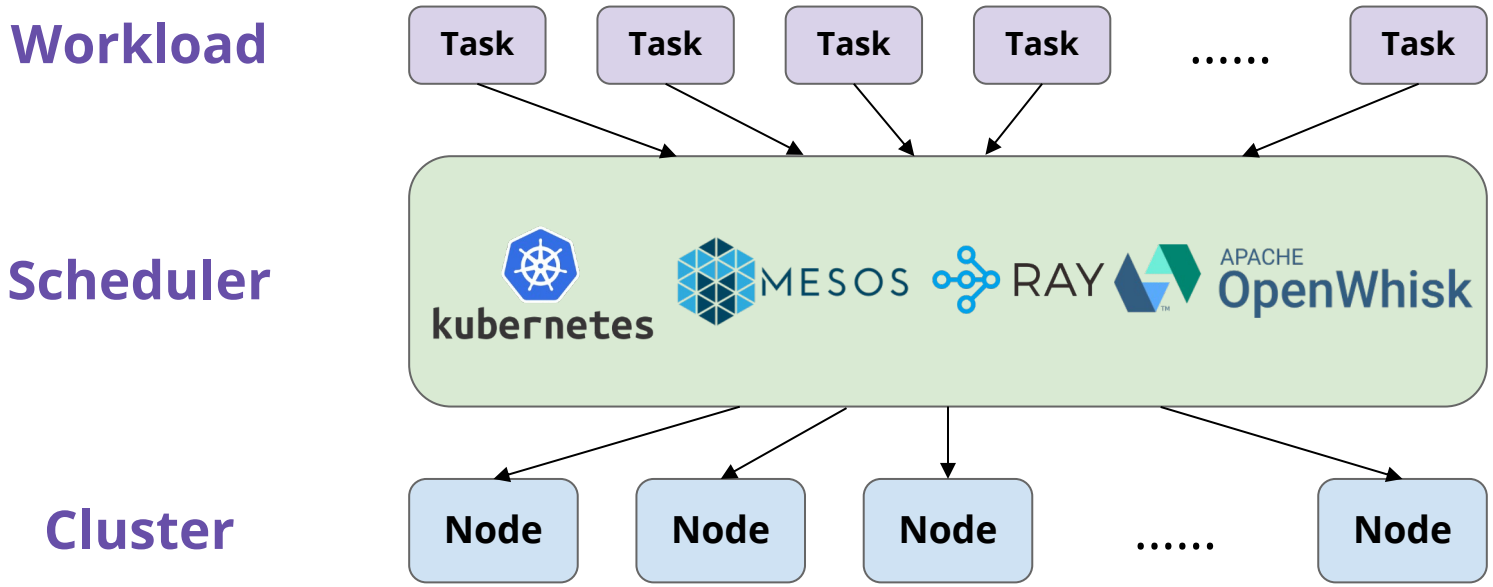
Programmable and Adaptive Scheduling for Distributed Systems

Yuyao Wang, Xiangfeng Zhu, Ratul Mahajan, Stephanie Wang

PAUL G. ALLEN SCHOOL
OF COMPUTER SCIENCE & ENGINEERING

Task Scheduling is Ubiquitous



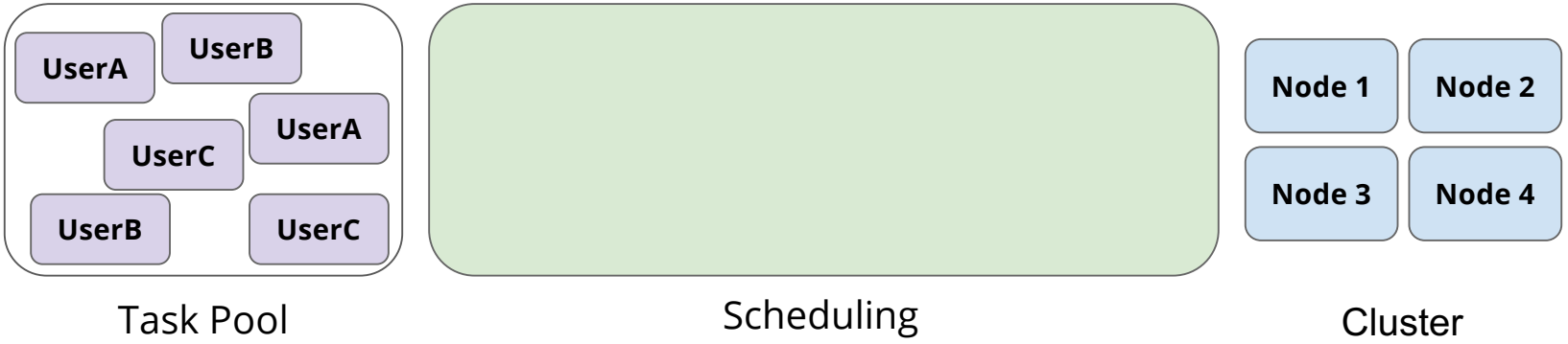
Problems of Today's Scheduling

Scheduling policies are **hard-coded** in frameworks.

1. Limited expressiveness
   - Diverse policies cannot be specified.
2. Poor application performance
   - Fixed implementation cannot adapt to workload changes.

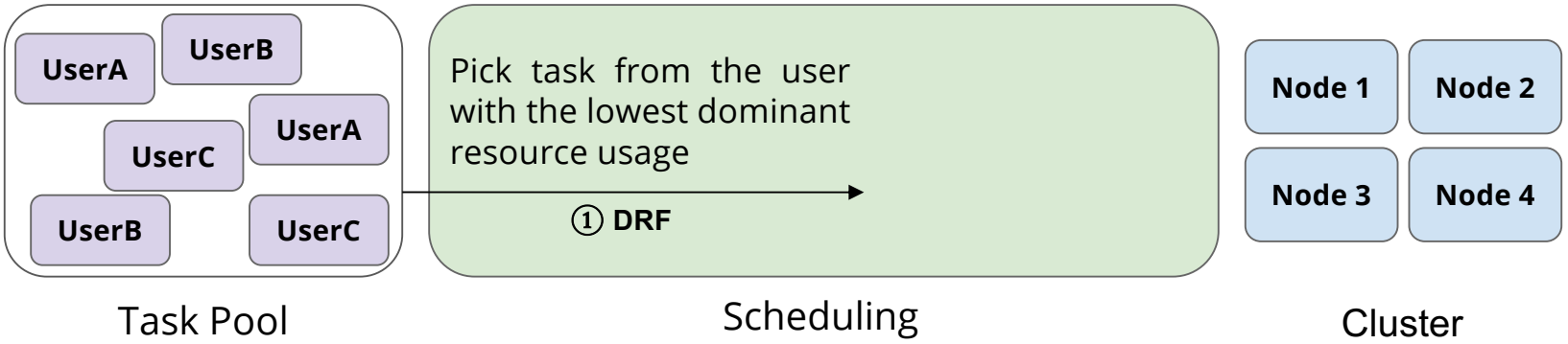
Example: DRF-LB

Dominant Resource Fairness (DRF) + Load Balancing (LB)



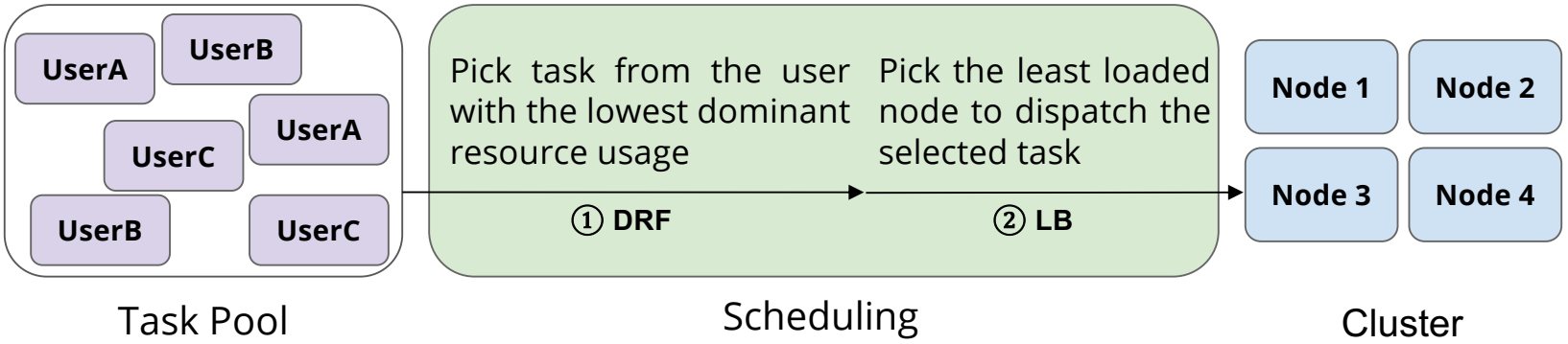
Example: DRF-LB

Dominant Resource Fairness (DRF) + Load Balancing (LB)



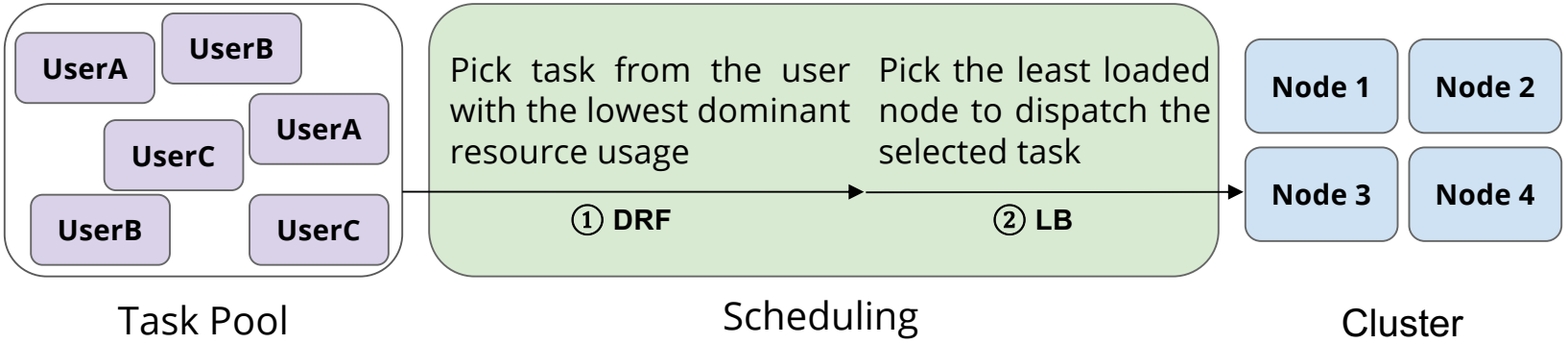
Example: DRF-LB

Dominant Resource Fairness (DRF) + Load Balancing (LB)



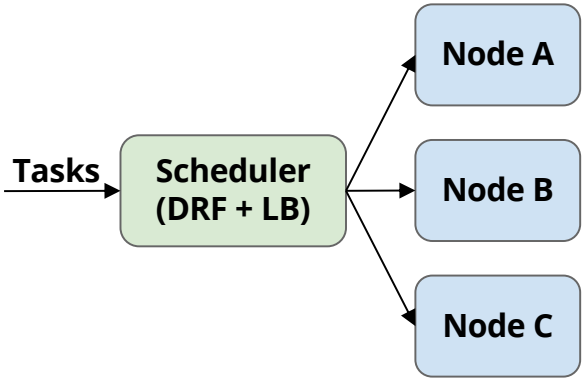
Example: DRF-LB

Dominant Resource Fairness (DRF) + Load Balancing (LB)

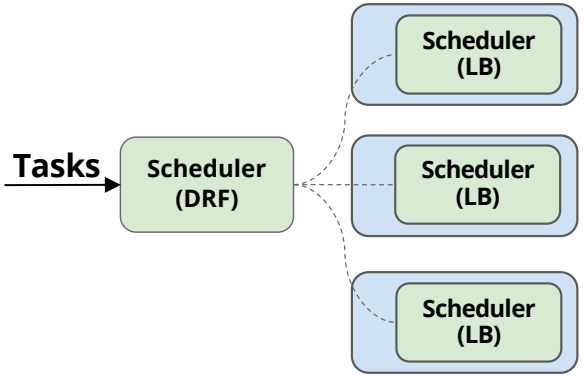


- Limited expressiveness: DRF-LB cannot be specified.
- Poor application performance: the best implementation choice is *workload-* and *environment-dependent*.

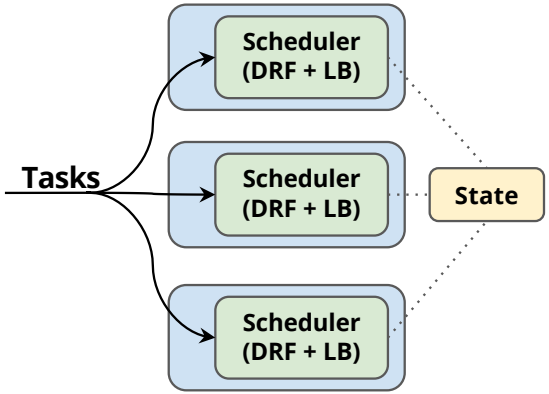
Three Possible Implementations of DRF-LB



Centralized

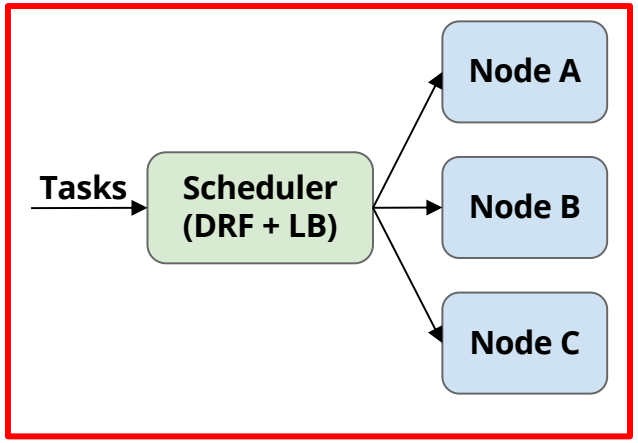


Hybrid

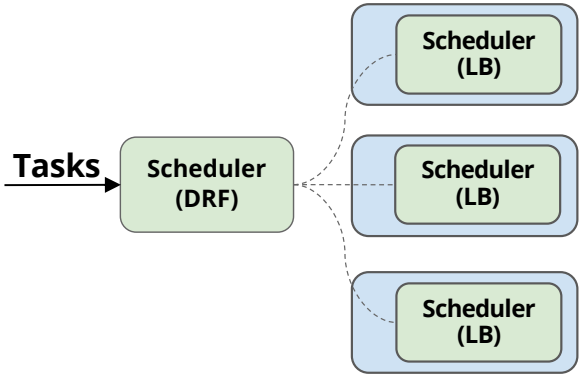


Decentralized

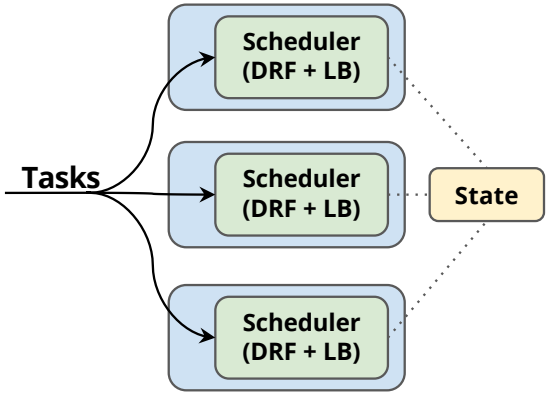
Three Possible Implementations of DRF-LB



Centralized

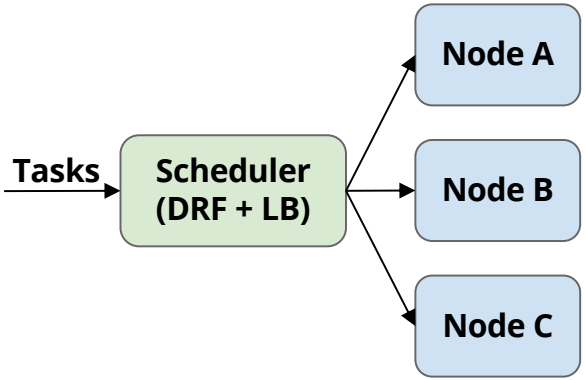


Hybrid

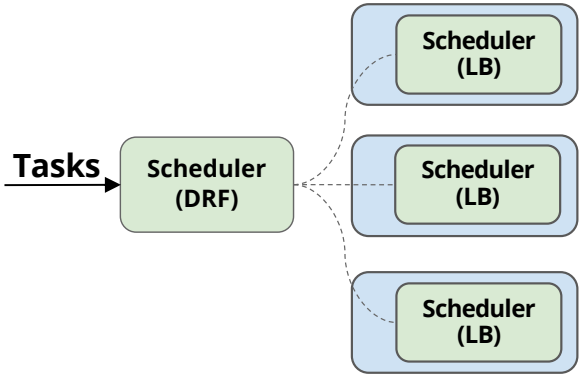


Decentralized

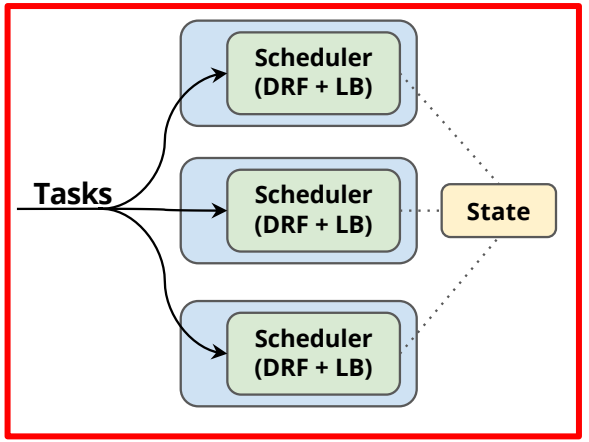
Three Possible Implementations of DRF-LB



Centralized

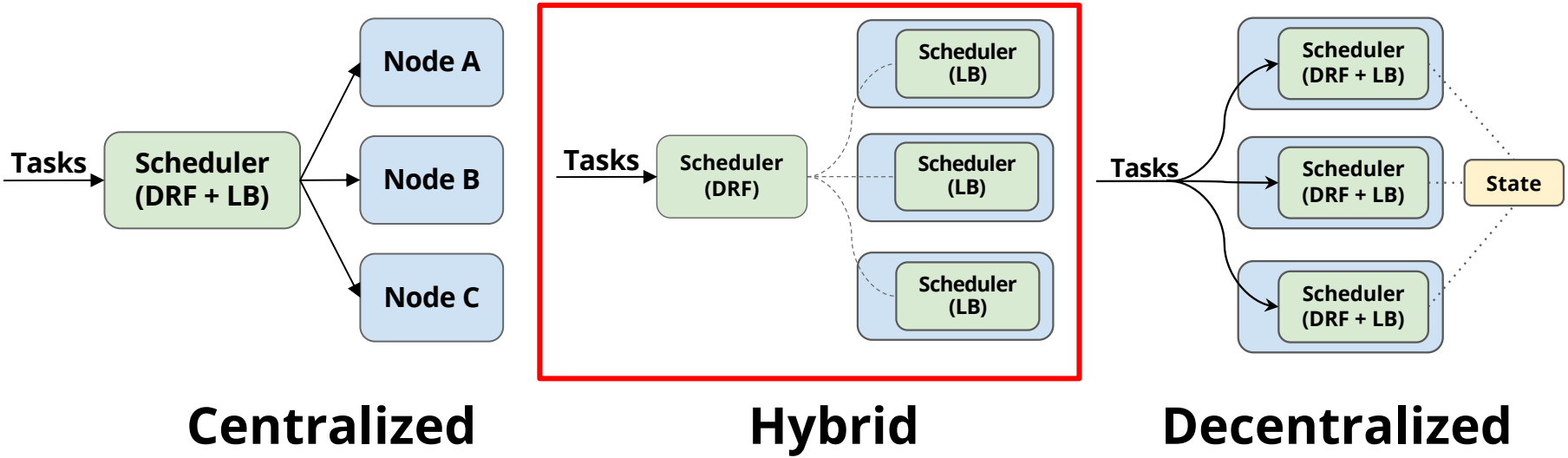


Hybrid



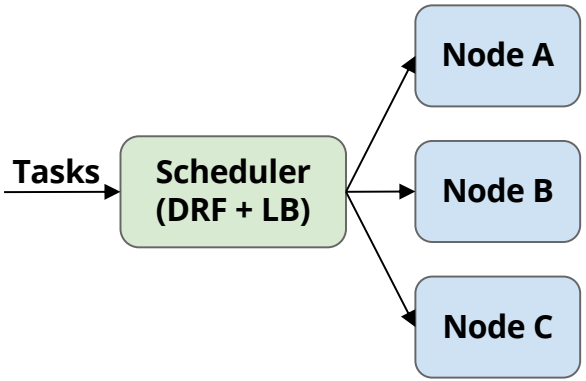
Decentralized

Three Possible Implementations of DRF-LB

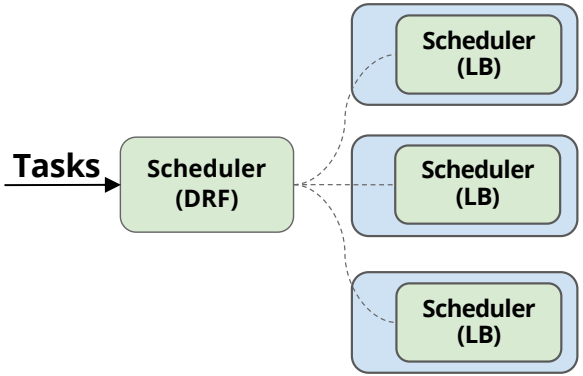


Centralized **Hybrid** **Decentralized**

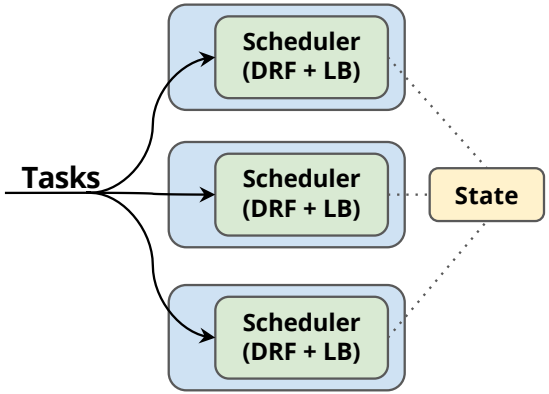
Three Possible Implementations of DRF-LB



Centralized



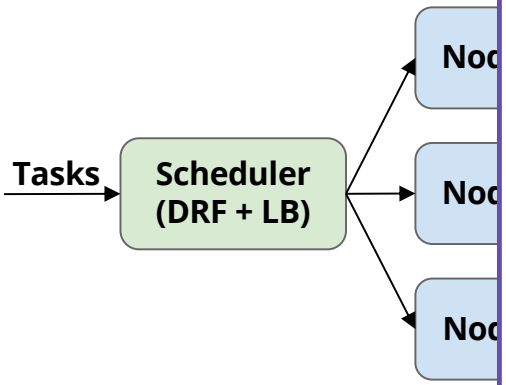
Hybrid



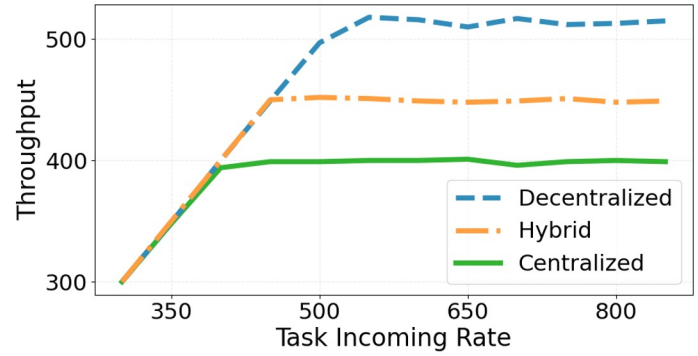
Decentralized

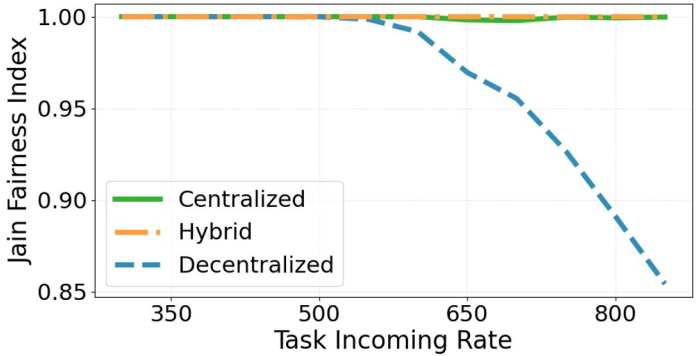
	Centralized	Hybrid	Decentralized
Throughput (decisions/sec)	📦	📦📦	📦📦📦
Fairness	✅	✅	❌

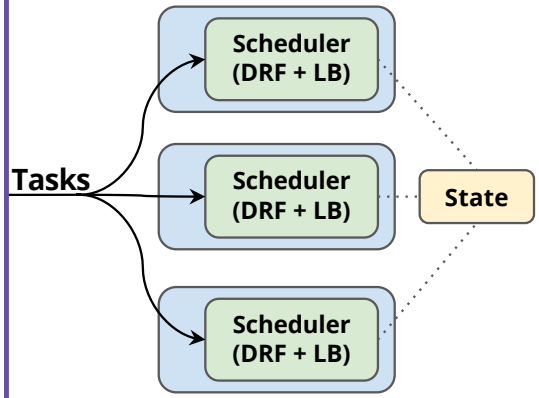
Three Possible Implementations of DRF-LB



Centralize





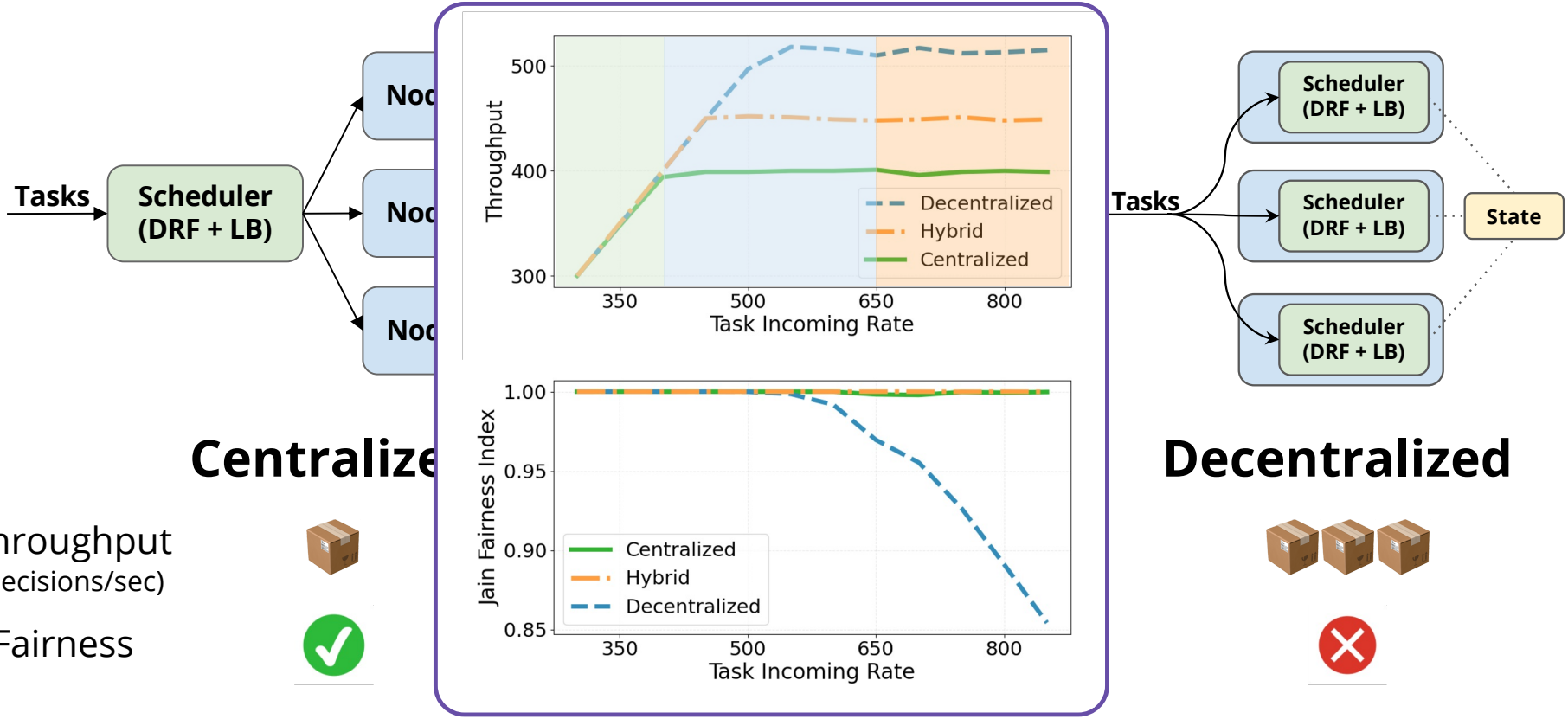


Decentralized

Throughput
(decisions/sec)

Fairness

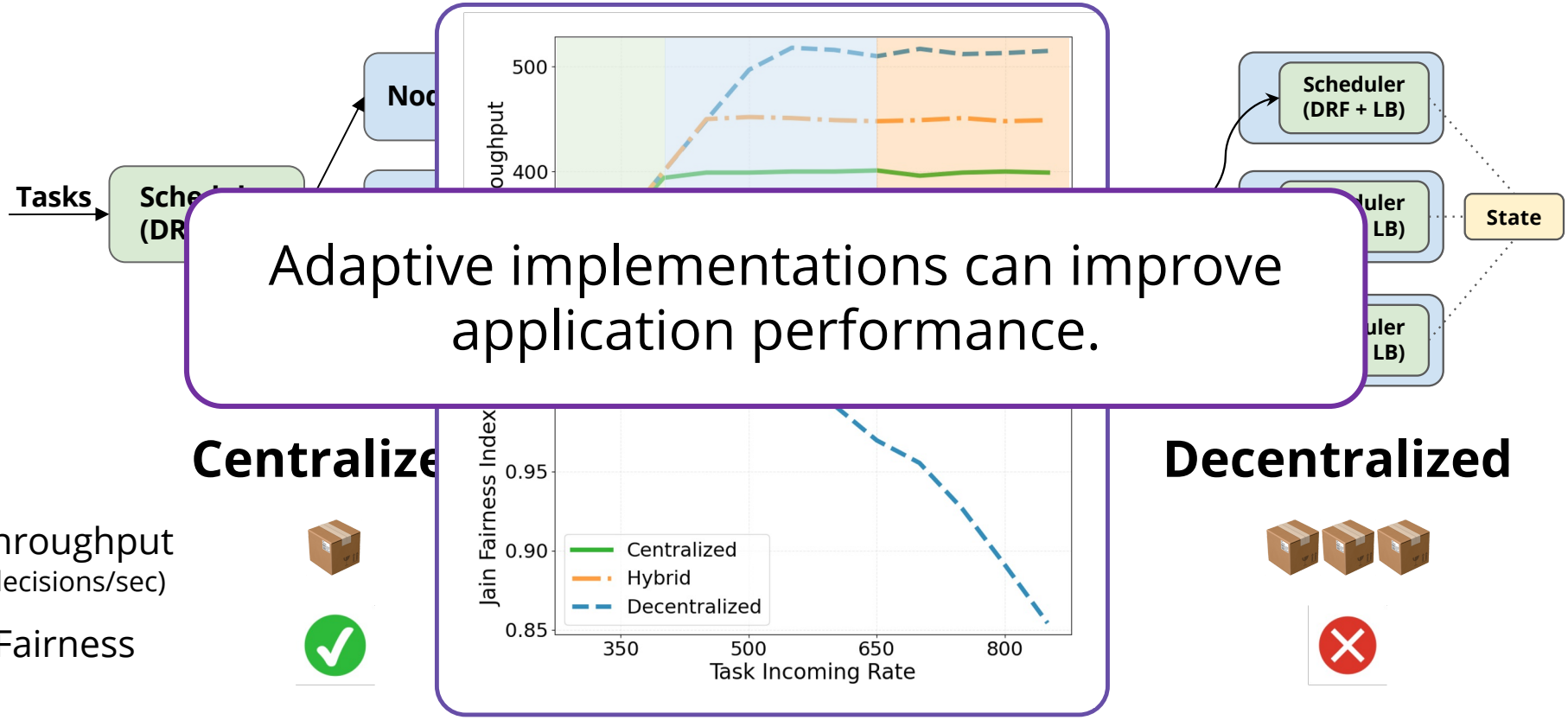
Three Possible Implementations of DRF-LB



Throughput (decisions/sec)

Fairness

Three Possible Implementations of DRF-LB



Towards Programmable and Adaptive Scheduling

Problem recap of hard-coded scheduling:

1. Limited expressiveness
2. Poor application performance

Towards Programmable and Adaptive Scheduling

Problem recap of hard-coded scheduling:

1. Limited expressiveness
2. Poor application performance

Policy specification and implementation should be **decoupled**.

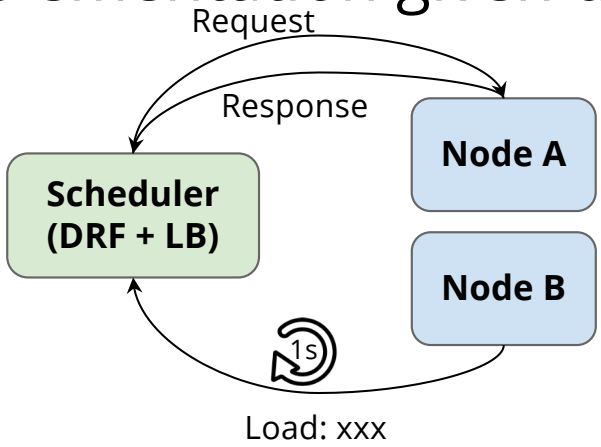
1. High-level programmability
2. Adaptive execution
    - Dynamic adjustment of implementation strategies

Key Challenges

- Programming abstractions?
- Intelligent adaptation: how to choose the best implementation given diverse application metrics?

Key Challenges

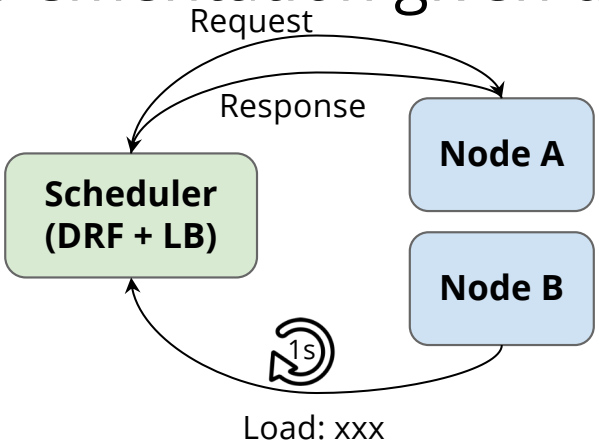
- Programming abstractions?
- Intelligent adaptation: how to choose the best implementation given diverse application metrics?



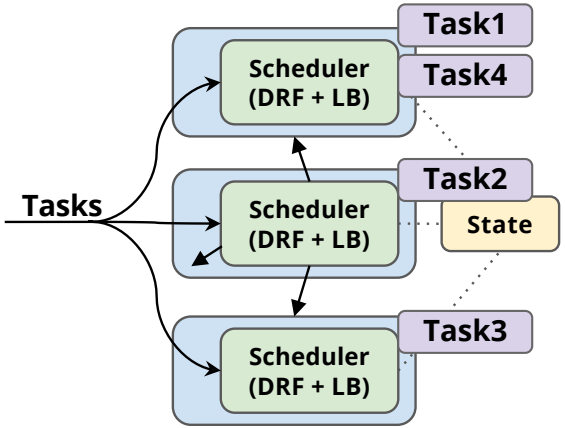
Centralized schedulers cannot avoid stale information.

Key Challenges

- Programming abstractions?
- Intelligent adaptation: how to choose the best implementation given diverse application metrics?

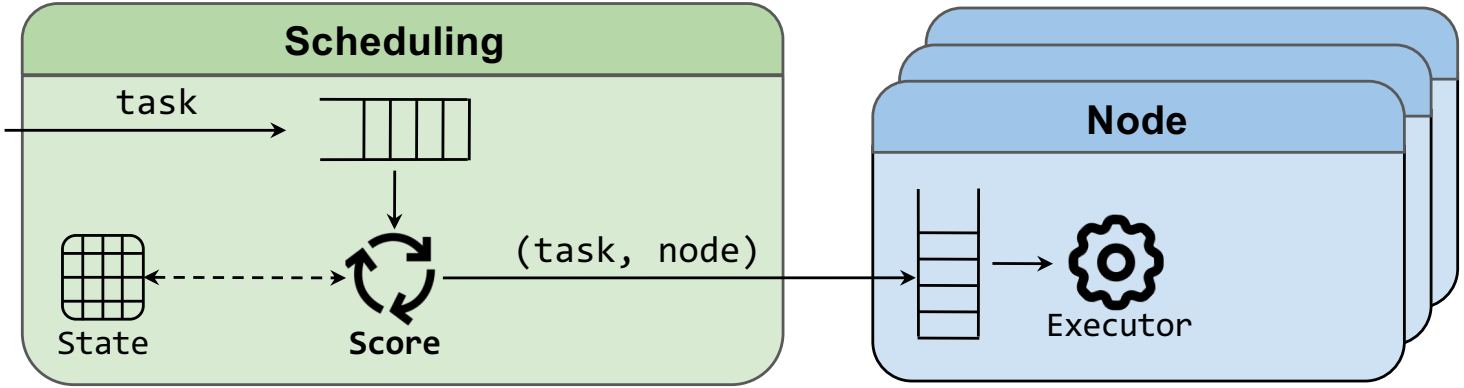


Centralized schedulers cannot avoid stale information.

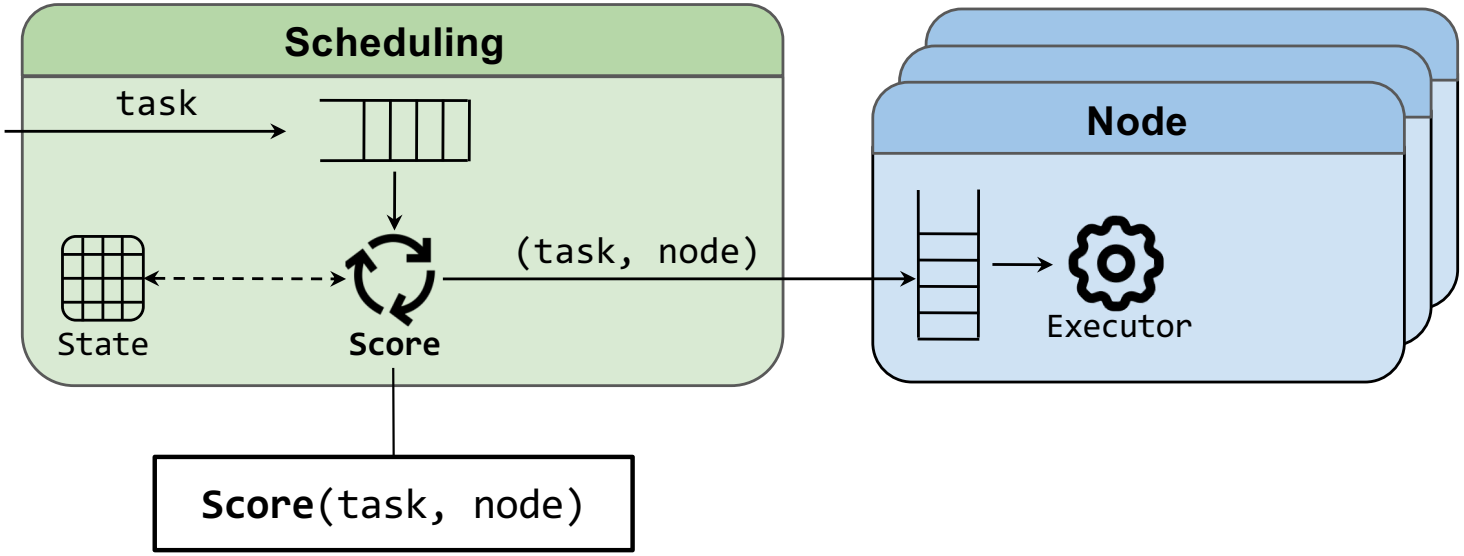


Decentralized schedulers suffer from partial views.

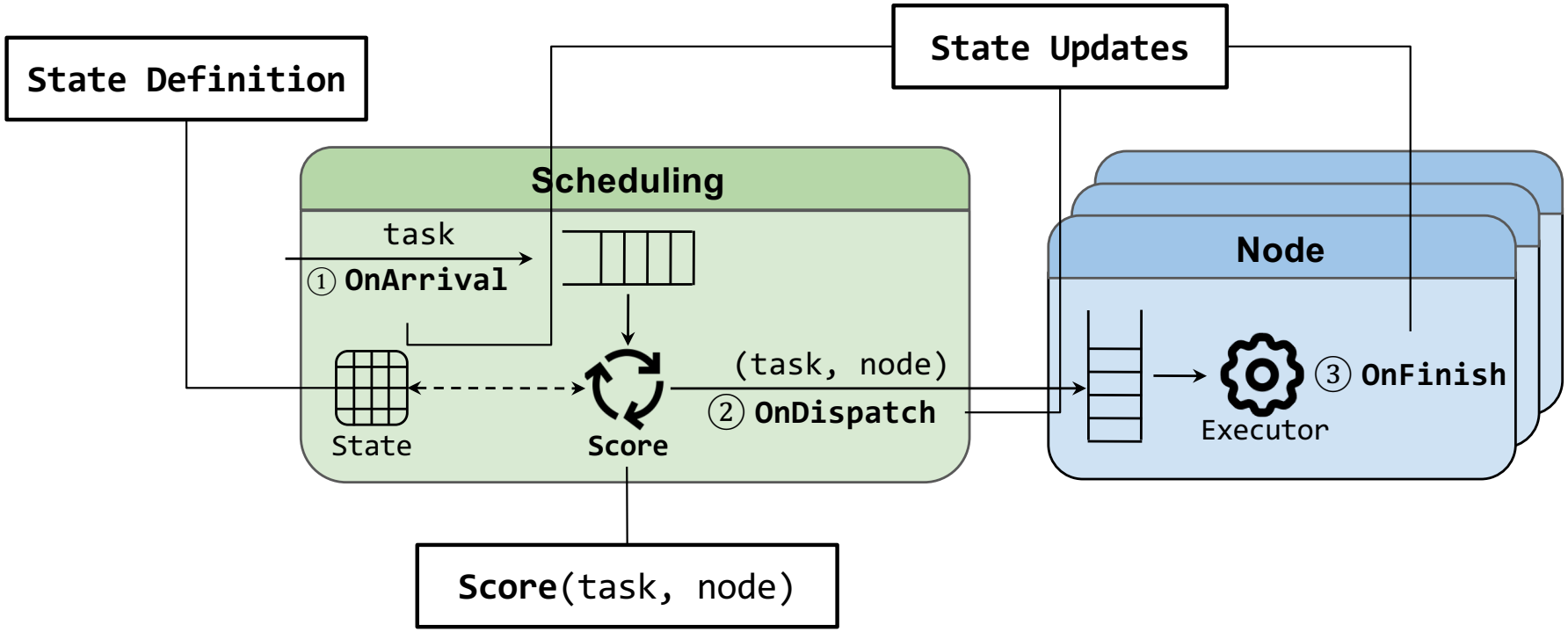
Programming Abstractions



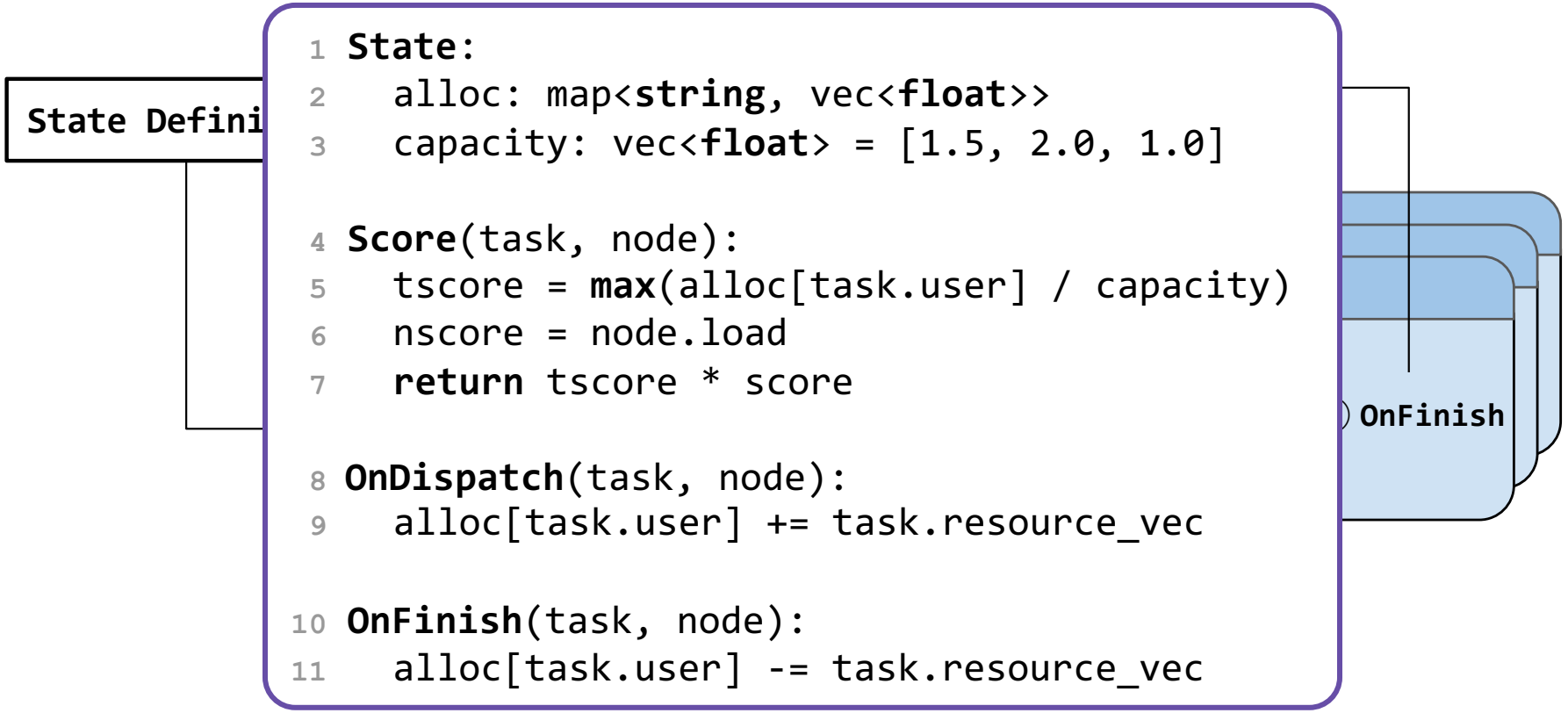
Programming Abstractions



Programming Abstractions



Programming Abstractions



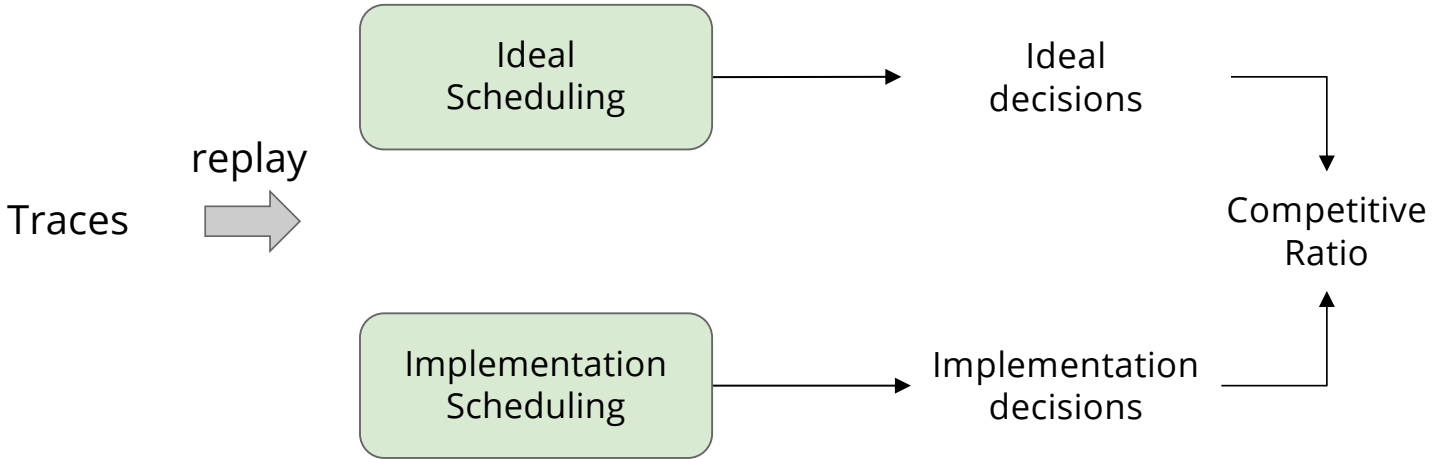

```
1  State:
2    alloc: map<string, vec<float>>
3    capacity: vec<float> = [1.5, 2.0, 1.0]

4  Score(task, node):
5    tscore = max(alloc[task.user] / capacity)
6    nscore = node.load
7    return tscore * score

8  OnDispatch(task, node):
9    alloc[task.user] += task.resource_vec

10 OnFinish(task, node):
11   alloc[task.user] -= task.resource_vec
```

Intelligent Adaptation

- How to quantify semantics deviations from policy specification?
- Competitive ratio on scores = $\frac{\texttt{Score}(\text{impl decisions})}{\texttt{Score}(\text{ideal decisions})}$

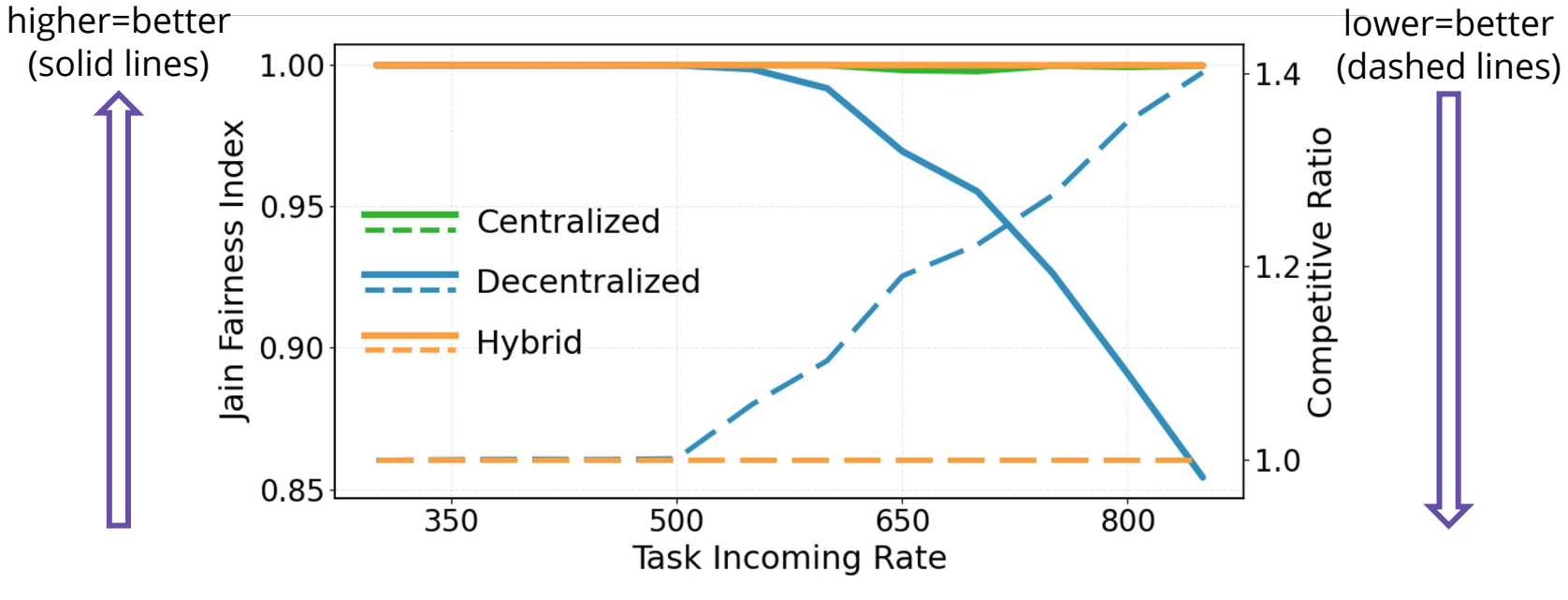
Intelligent Adaptation

- How to quantify semantics deviations from policy specification?
- Competitive ratio on scores = $\frac{\texttt{Score}(\text{impl decisions})}{\texttt{Score}(\text{ideal decisions})}$



Intelligent Adaptation

- Competitive ratio changes ⇔ JFI changes



Conclusion

- Hard-coded scheduling is problematic:
  - Limited expressiveness
  - Poor application performance

- We argue for programmable and adaptive scheduling:
  - A DSL based on scores
  - Intelligent adaptation via competitive ratio analysis