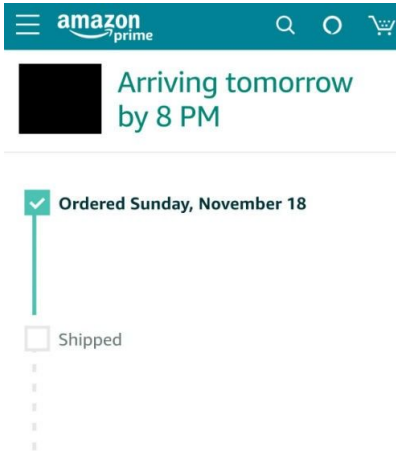
When will my ML Job finish?

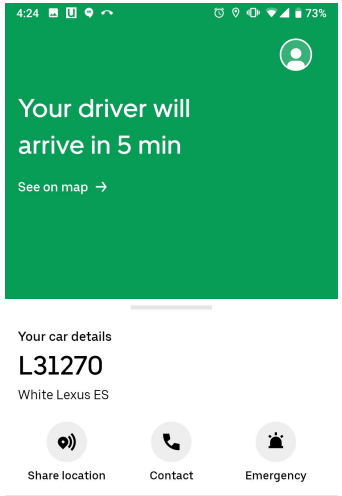
Toward providing Completion Time Estimates through Predictability-Centric Scheduling

Abdullah Bin Faisal,
Noah Martin, Hafiz Mohsin Bashir, Swaminathan Lamelas, Fahad R. Dogar

Predictability in Real-World Systems

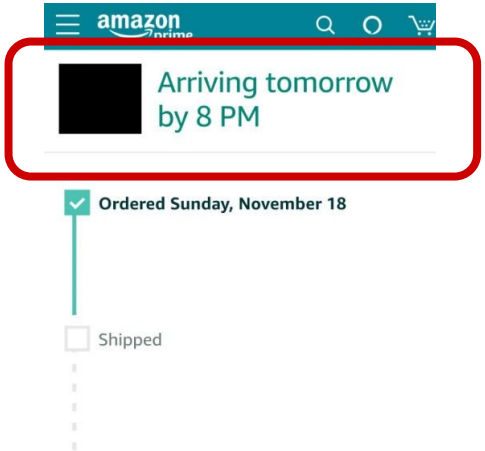
- Most real-world systems provide users with a time estimate
  - Improves service quality

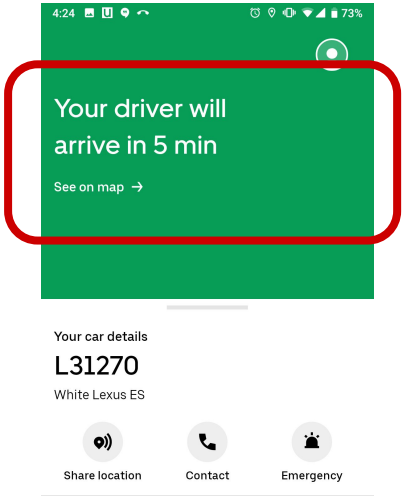




Predictability in Real-World Systems

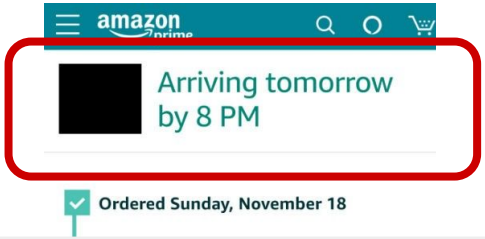
- Most real-world systems provide users with a time estimate
  - Improves service quality

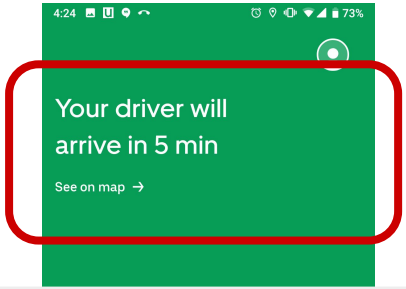




Predictability in Real-World Systems

- Most real-world systems provide users with a time estimate
  - Improves service quality

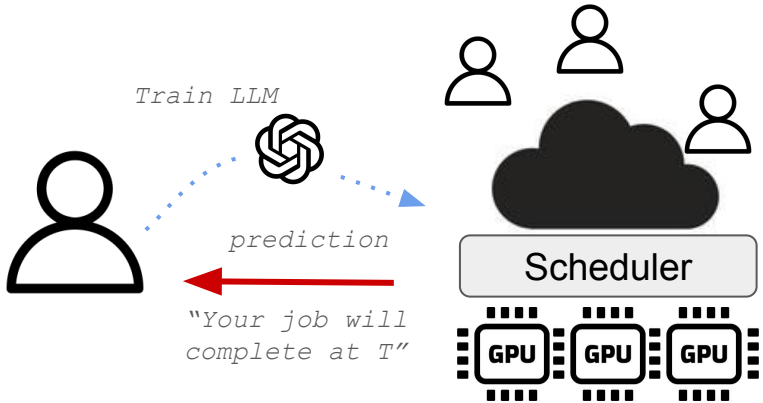




Can the Cloud do the same?

Can the Cloud provide predictions?

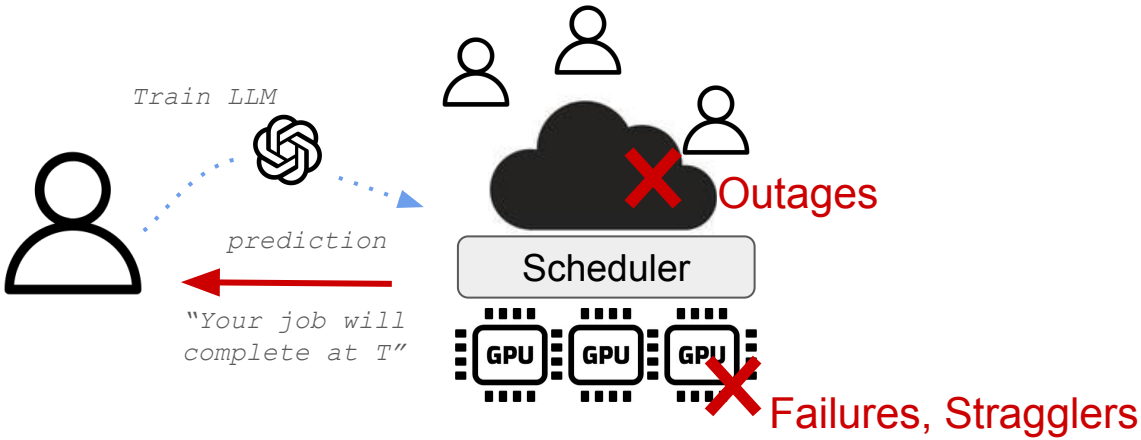
- Multi-tenant GPU clusters
  - Growing use-case
  - Highly loaded
  - Predictions can improve experience; if they are reliable



"By the end of 2024, we're aiming to [...] feature compute power equivalent to nearly 600,000 H100s." - Meta[1]

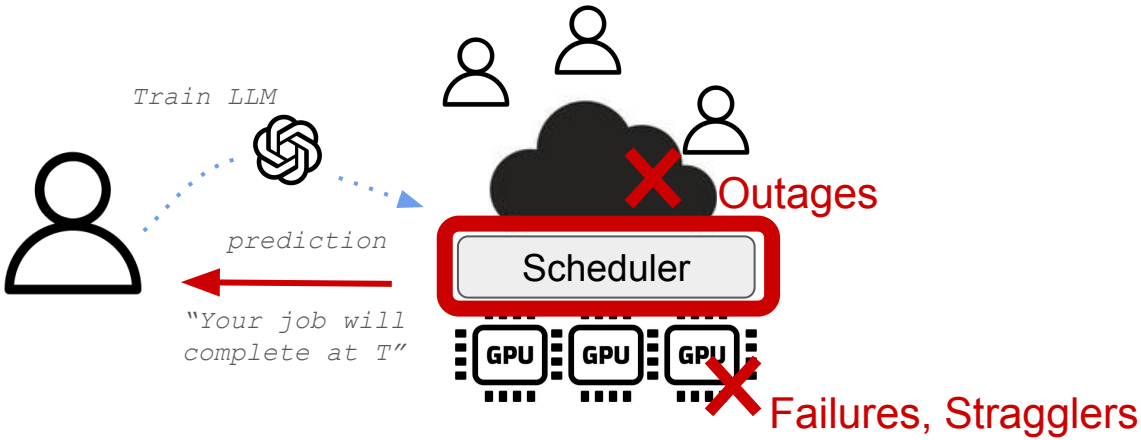
What impacts the reliability of predictions?

- Several factors can make predictions unreliable



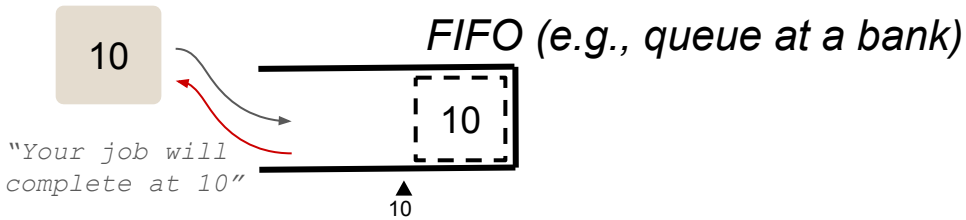
What impacts the reliability of predictions?

- Several factors can make predictions unreliable
  - Resource allocation policy (scheduler) is an important contributor
  - Preemptive *vs.* non-preemptive



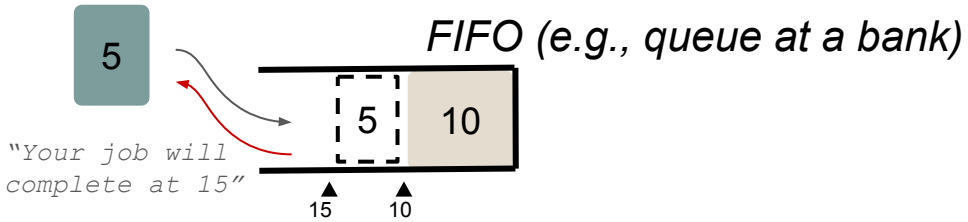
Role of scheduling in predictability

- Most real-world systems are non-preemptive (FIFO-like)
  - Future jobs (e.g., customers) do not impact existing ones



Role of scheduling in predictability

- Most real-world systems are non-preemptive (FIFO-like)
  - Future jobs (e.g., customers) do not impact existing ones

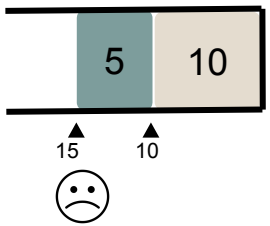


Role of scheduling in predictability

- Most real-world systems are non-preemptive (FIFO-like)
  - Future jobs (e.g., customers) do not impact existing ones

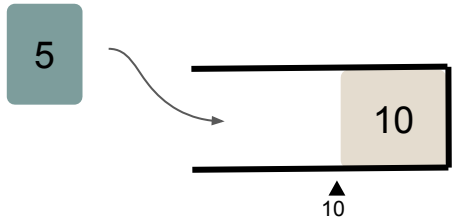
- Non-preemptive systems suffer from practical issues
  - E.g., Head of line blocking

FIFO (e.g., queue at a bank)



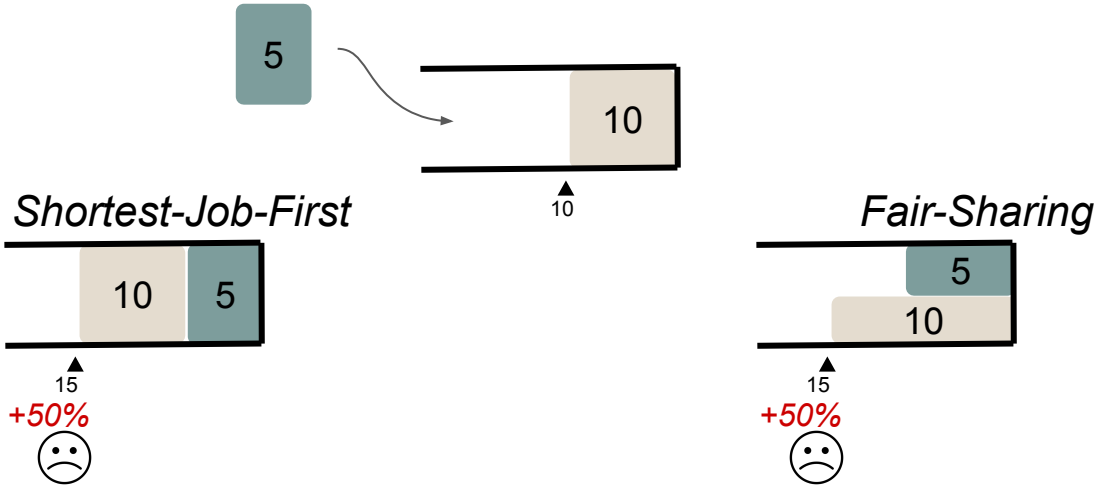
Role of scheduling in predictability

- Cloud systems employ preemption
  - Typically in an unbounded fashion
  - E.g., Prioritize shorter jobs (minimize JCTs) or share resources across jobs (fairness)
  - Causes unpredictability (prediction error)

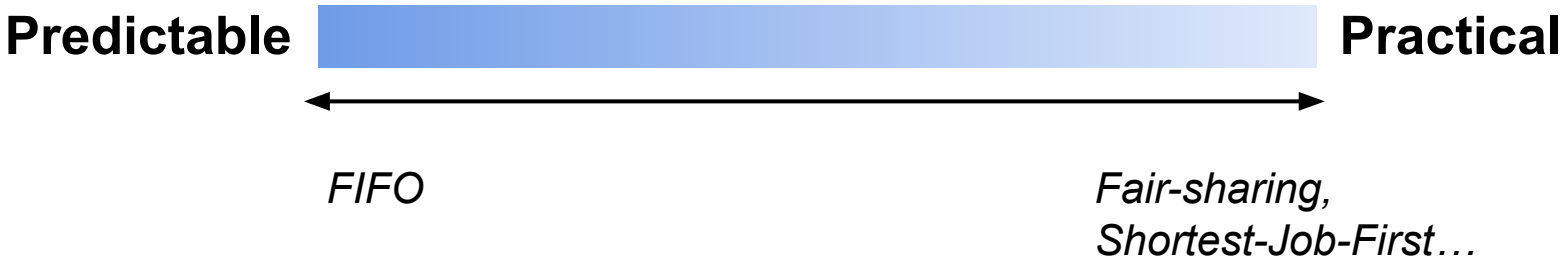


Role of scheduling in predictability

- Cloud systems employ preemption
  - Typically in an unbounded fashion
  - E.g., Prioritize shorter jobs (minimize JCTs) or share resources across jobs (fairness)
  - Causes unpredictability (prediction error)



Trade-off between predictability and practicality



Trade-off between predictability and practicality

FIFO

Fair-sharing,
Shortest-Job-First…

10x increase in avg JCTs compared to performance based schemes!

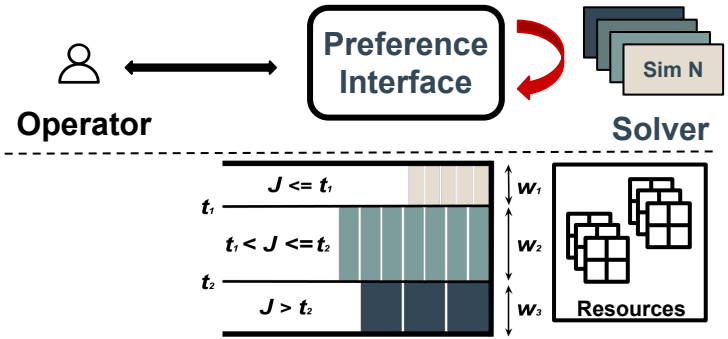
Greater than 100% average prediction error

Can we offer predictability while balancing other practical objectives?

Predictability-**C**entric **S**cheduling (PCS)

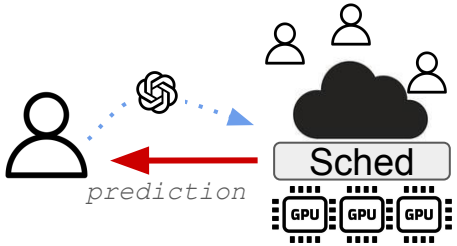
PCS in a single slide

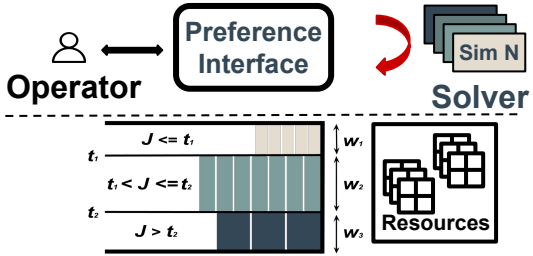
- Weighted-Fair-Queueing (WFQ) ➡ Bounded and Controlled Preemption
  - Different configurations can achieve different trade-offs
- Simulation based search ➡ Practical way to realize different trade-offs
  - Heuristics narrow the search space
- High level interface ➡ Simplify navigating trade-offs
  - Reduce trade-offs to Pareto-optimal choices



Outline

- Motivation - supporting PCS for ML workloads
  - Why provide predictions for ML workloads?
  - Feasibility and Challenges
- Design
  - Use of Weighted-Fair-Queues (WFQ)
  - Simulation based search strategy
  - High level interface to navigate trade-offs
- Evaluation
  - Benefits and feasibility of PCS





Why provide predictions for ML workloads?

- Evidence of user frustration in shared GPU clusters

> “[...] we do find users frustrated [...] The frustration frequently reaches the point where groups attempt or succeed at buying their own hardware” [Themis, NSDI’20]

Why provide predictions for ML workloads?

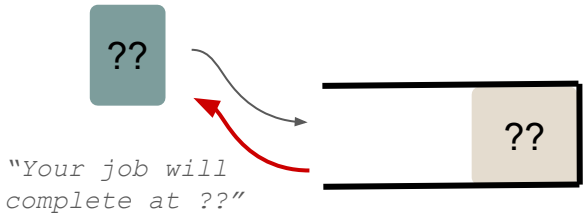
- Evidence of user frustration in shared GPU clusters

> "[...] we do find users frustrated [...] The frustration frequently reaches the point where groups attempt or succeed at buying their own hardware" [Themis, NSDI'20]

- GPU cluster users are already making predictions [Chronus, SoCC'21]
  - User generated predictions can be off by more than 100%

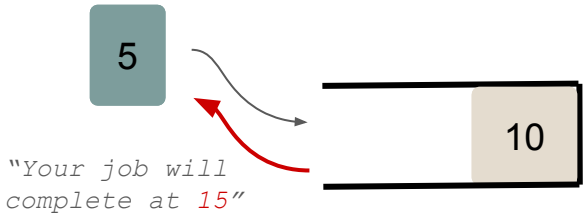
> "[...] completion time of a DLT job varies under different scheduling algorithms [...] Hence, it is infeasible to accurately estimate [...]" [Chronus, SoCC'21]

Feasibility and Challenges - Workload characteristics



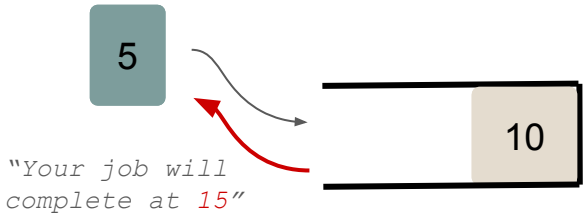
Feasibility and Challenges - Workload characteristics

- An ML job's demand function is known or can be estimated
  - Demand function: allocated resources ➡ execution time (#epochs / thrpt)
  - Necessary to compute a reasonable prediction
  - Has been leveraged by prior systems (e.g., Themis, NSDI'20)



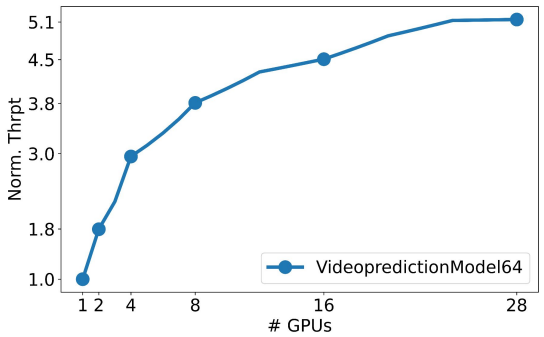
Feasibility and Challenges - Workload characteristics

- An ML job's demand function is known or can be estimated
  - Demand function: allocated resources ➡ execution time (#epochs / thrpt)
  - Necessary to compute a reasonable prediction
  - Has been leveraged by prior systems (e.g., Themis, NSDI'20)
- Accuracy of prediction still impacted by scheduler choice!



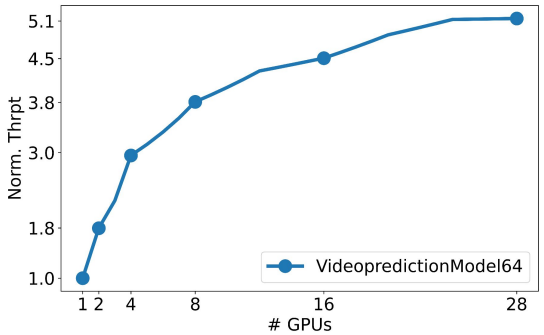
Feasibility and Challenges - Workload characteristics

- An ML job's demand function is known or can be estimated
- Demand functions can be complex
  - ML jobs exhibit sub-linear throughput scaling



Feasibility and Challenges - Workload characteristics

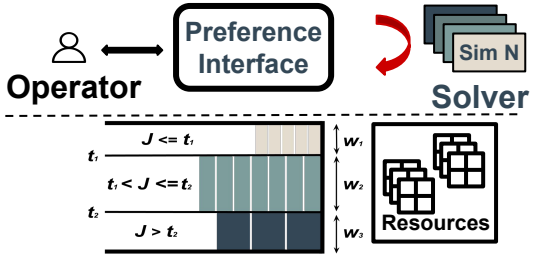
- An ML job's demand function is known or can be estimated
- Demand functions can be complex
  - ML jobs exhibit sub-linear throughput scaling
- Unclear how to allocate GPUs to such jobs!



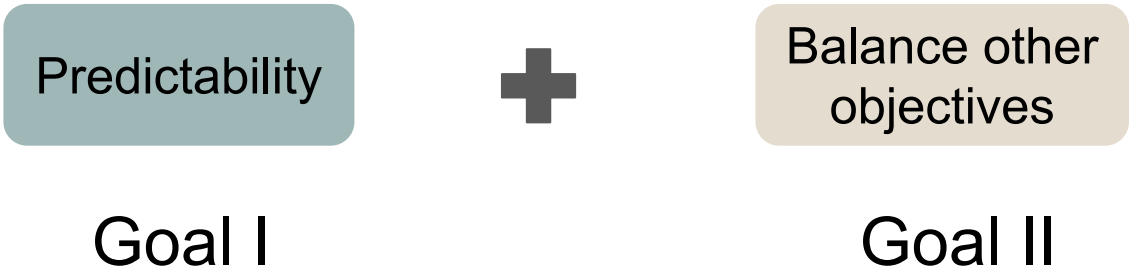
Outline

- Motivation - supporting PCS for ML workloads
  - Why provide predictions for ML workloads?
  - Feasibility and Challenges

- Design
  - Use of Weighted-Fair-Queues (WFQ)
  - Simulation based search strategy
  - High level interface to navigate trade-offs
- Evaluation
  - Benefits and feasibility of PCS

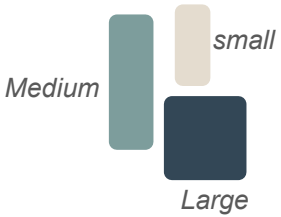


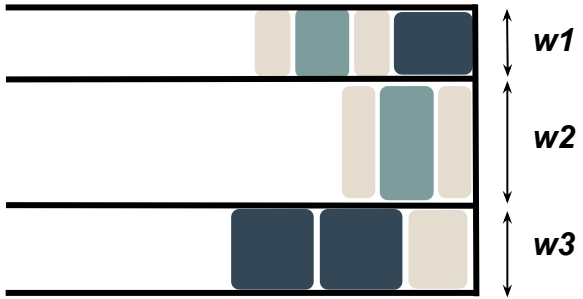
Design goals



Background on WFQ

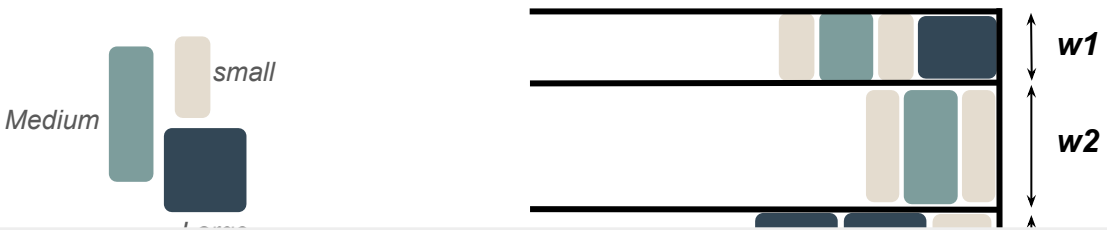
- Different jobs mapped to different queues
- Within a queue, jobs are processed in FIFO order
- Each queue gets a guaranteed resource share (weights)
- # of queues, weights, job mapping criterion etc. are configurable





Background on WFQ

- Different jobs mapped to different queues
- Within a queue, jobs are processed in FIFO order
- Each queue gets a guaranteed resource share (weights)
- # of queues, weights, job mapping criterion etc. are configurable

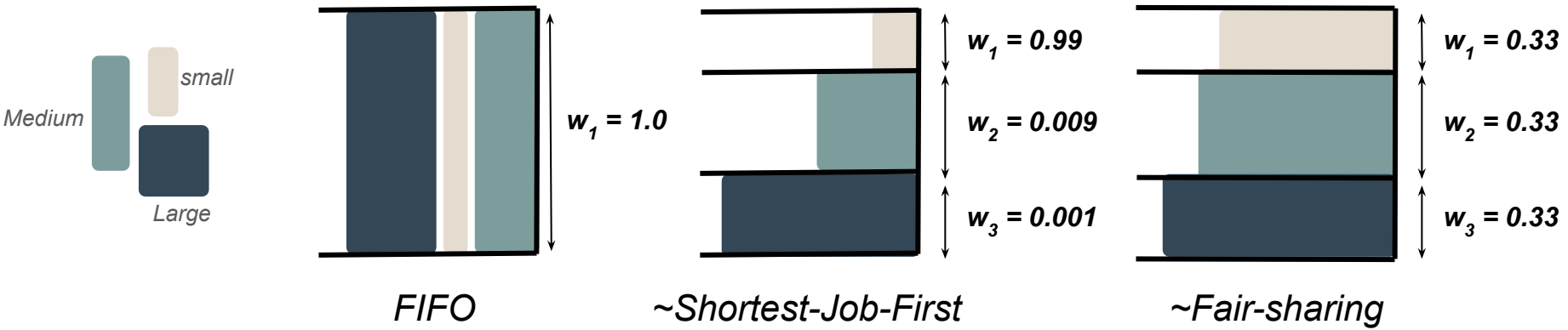


Guaranteed resource share + FIFO ordering within queue bounds prediction error

WFQ provides the necessary baseline flexibility

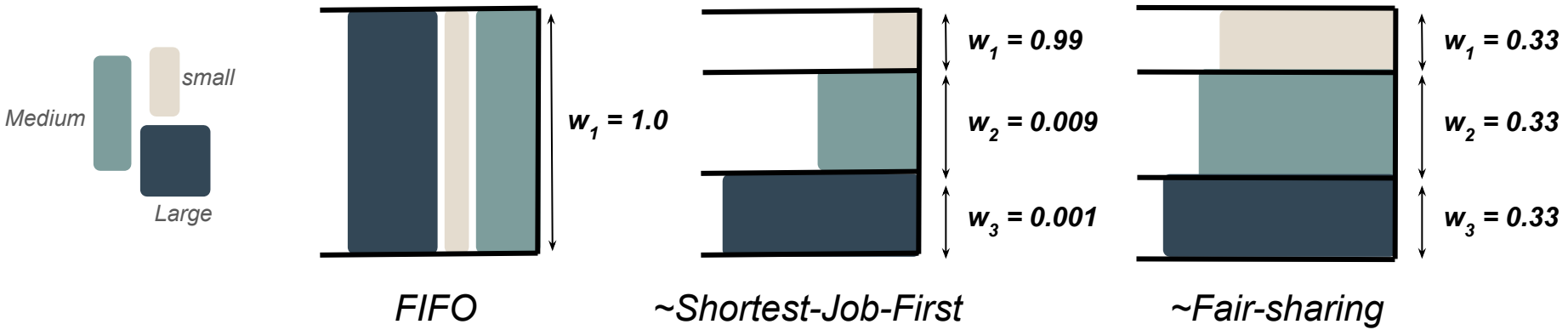
Spanning the scheduler space with WFQs

- WFQ parameters can be intelligently configured
  - Allows to achieve different trade-offs



Spanning the scheduler space with WFQs

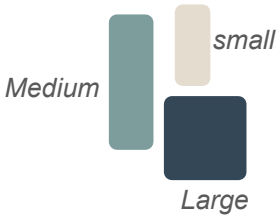
- WFQ parameters can be intelligently configured
  - Allows to achieve different trade-offs

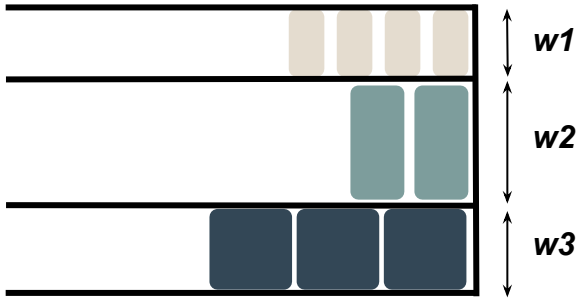


WFQ can approximate extreme and *potentially* intermediate points

WFQ optimization I: Avoiding HOL blocking

- Similar sized jobs are mapped to the same queue (avoids HOL blocking)
  - Appropriate number of queues and thresholds





WFQ optimization II: Handling complex demand functions

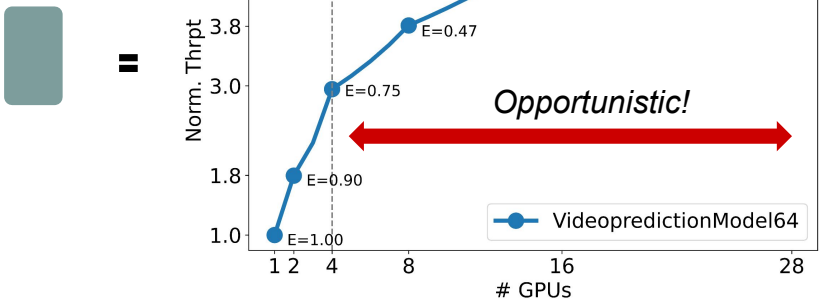
- SJF optimal if job's thrpt scales linearly w.r.t. resources [AFS, NSDI'21]
  - Aggressively prioritizing efficient jobs ➡ unpredictability

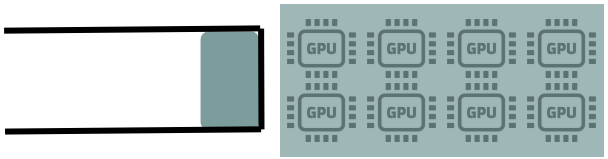
Cap a job's maximum possible allocation

- Compute a job's efficiency: $E(n) = thrpt(n)/n$
- Cap maximum allocation to k s.t. $E(k) \geq E_{min}$
- E_{min} is another configurable parameter
- Higher E_{min} ➡ higher chances of GPUs being preempted

Example

- $E_{min} = 0.75$, 8 GPU system

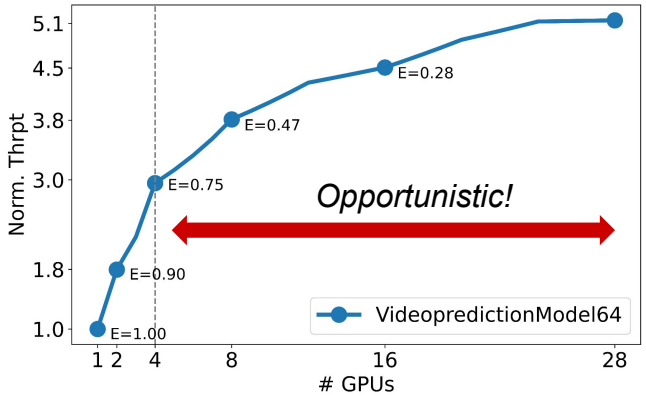


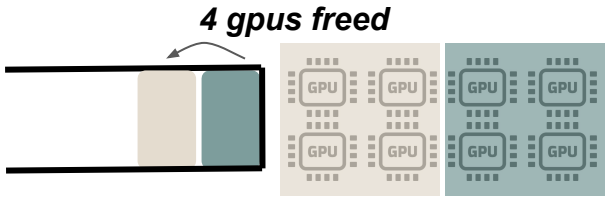


Example

- $E_{min} = 0.75$, 8 GPU system

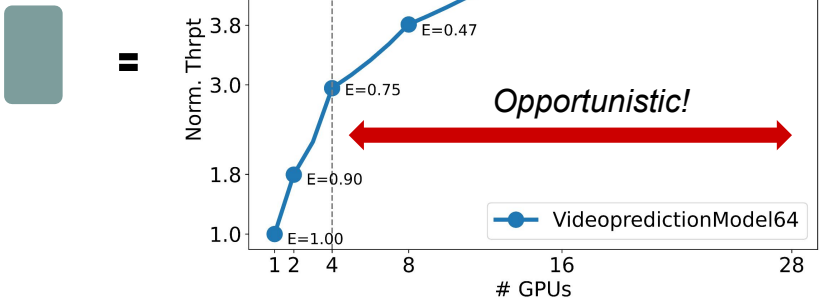
=

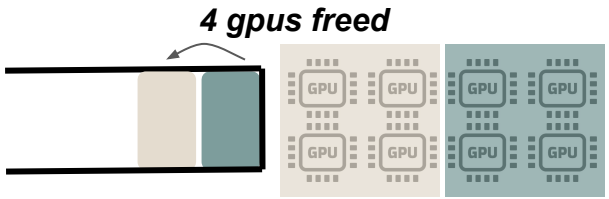




Example

- $E_{min} = 0.75$, 8 GPU system



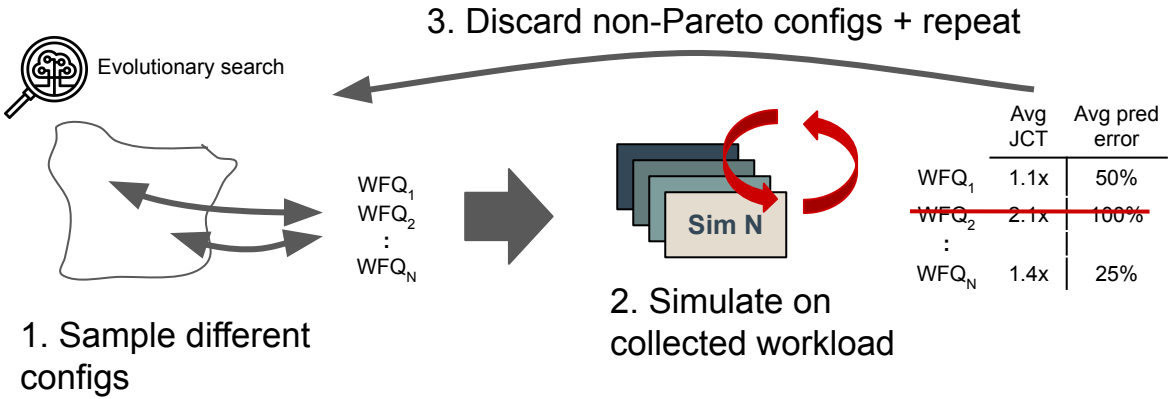


Conservative strategy trades off predictability for additional performance

Searching for WFQ configurations

- # queues, weights, E_{min} etc. jointly influence trade-offs
  - Hard to reason about the combined impact

Empirically determine impact of different parameters



Narrowing the search space

- Search space is combinatorial + large
  - ~ *O*(# queues * thresholds * weights * E_{min})

Increase the likelihood of a random sample being Pareto-optimal

- Intelligently parameterize configurations

Example: Use exponentially shrinking weights

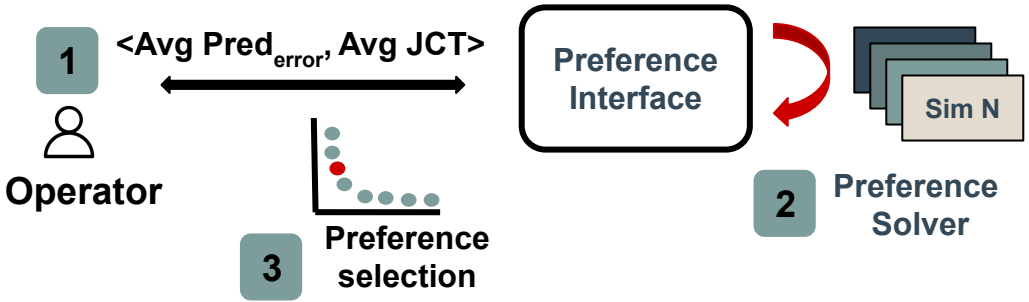
- Assigning exponentially lower weights to longer jobs improves performance
- Formally: $w_k \propto \exp(-k * W)$
- Larger W ➡ more aggressive shrinking; W=0 ➡ fair division
- Other heuristics discussed in the paper!

Simplify navigating trade-offs

- Exact trade-offs are unknown a priori
  - Trade-off space is workload dependent
  - Can't say: "give me 10% error but okay to take a 1.5x performance hit"

Operators specify higher level objectives; PCS provides Pareto-optimal choices

- Supported Objectives: Prediction error, JCTs, unfairness
- Supported Measures: Avg + specific percentiles



Outline

- Motivation - supporting PCS for ML workloads
  - Why provide predictions for ML workloads?
  - Feasibility and Challenges
- Design
  - Use of Weighted-Fair-Queues (WFQ)
  - Simulation based search strategy
  - High level interface to navigate trade-offs
- Evaluation
  - Benefits and feasibility of PCS

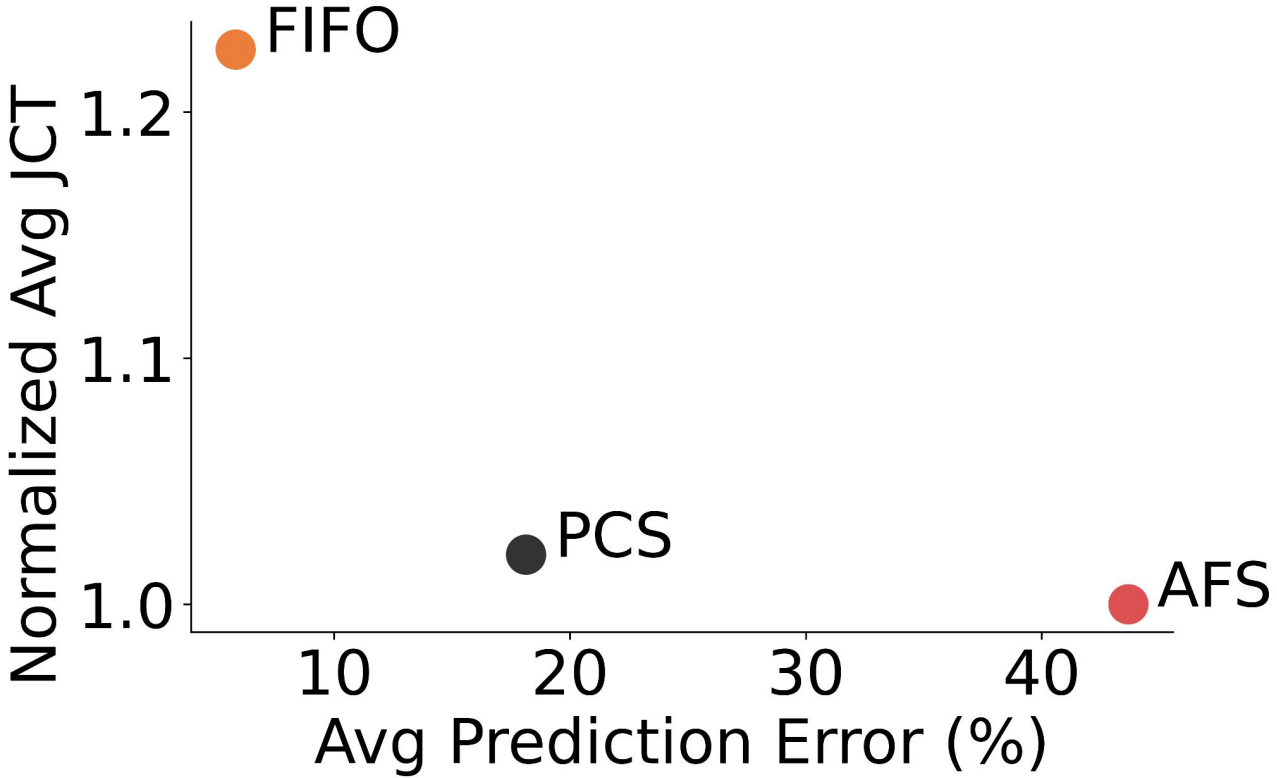
Evaluation - Key questions

- How does PCS perform for realistic workloads?
- Can the simulation-based search find the Pareto-frontier?
- How effective are the heuristics in improving search efficiency?

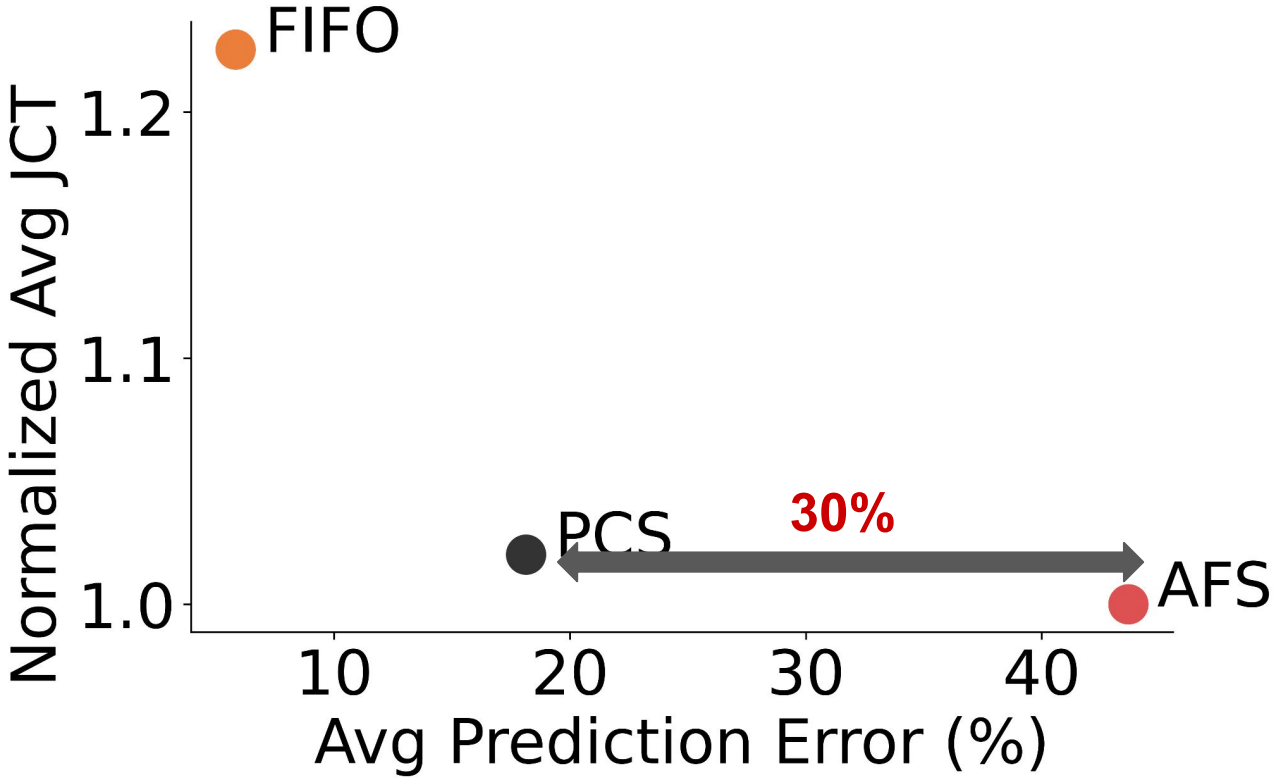
How does PCS fare under realistic settings?

- Testbed setup
  - 16 GPU cluster (NVIDIA P100s)
- Workload
  - AutoML jobs - each job spawns 1-20 DNN trials
  - Poisson job arrivals at 80% load
- Comparison points
  - Performance baseline: AFS (NSDI'21); efficiency + size based scheduling
  - Predictability baseline: FIFO
- Metrics
  - JCTs
  - Prediction error = (JCT - JCTpred) / JCTpred

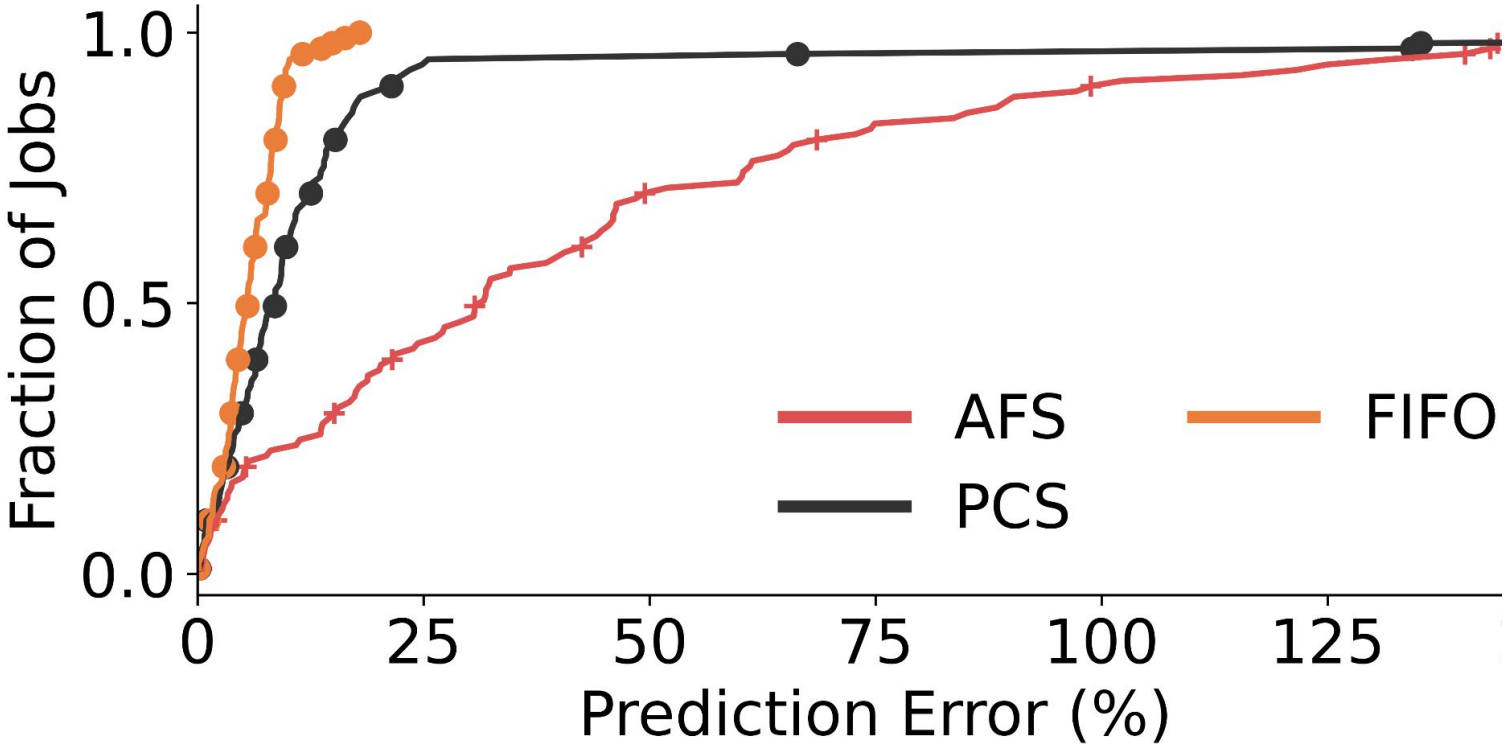
Achieving intermediate trade-offs



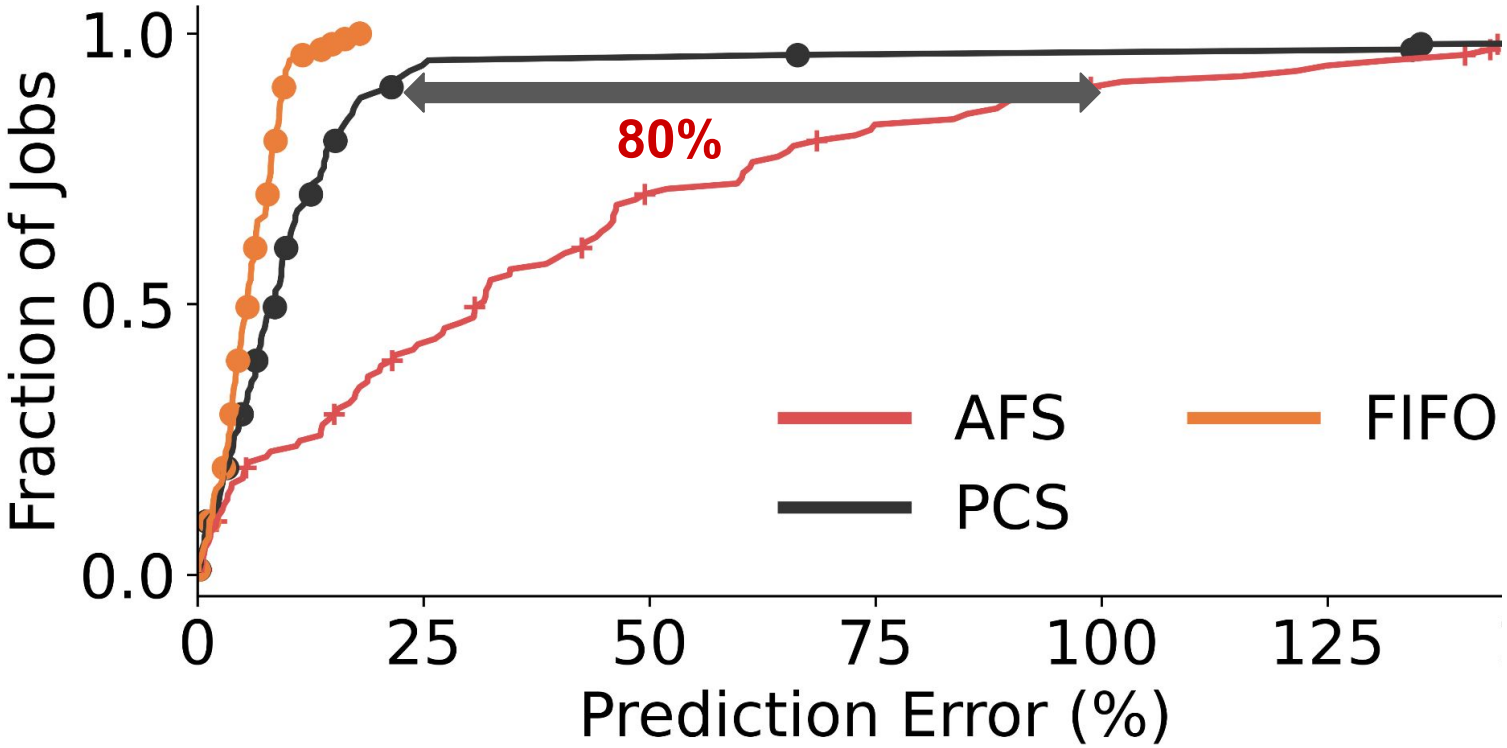
Achieving intermediate trade-offs



Providing reliable predictions with PCS

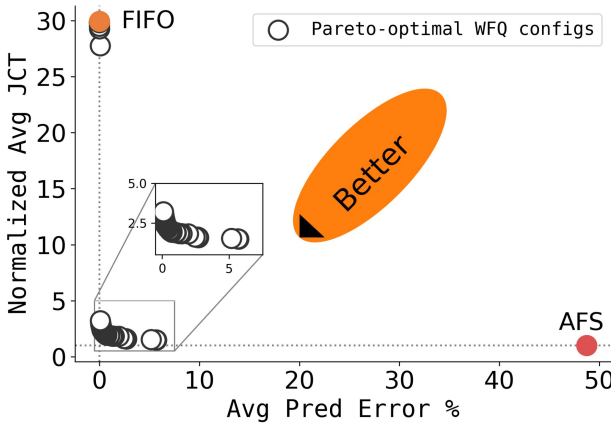


Providing reliable predictions with PCS

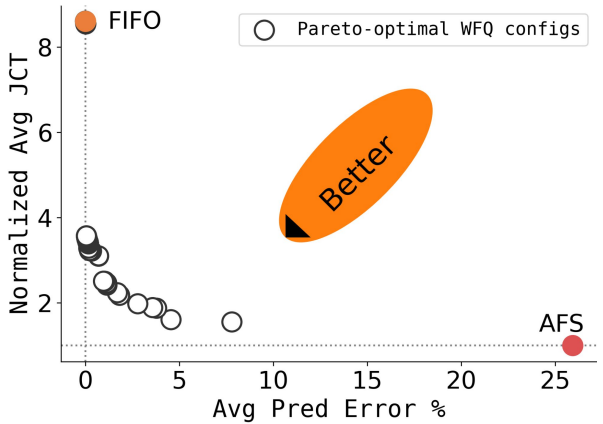


Can the simulation-based search find the Pareto-frontier?

- Run search on two realistic traces (Philly)



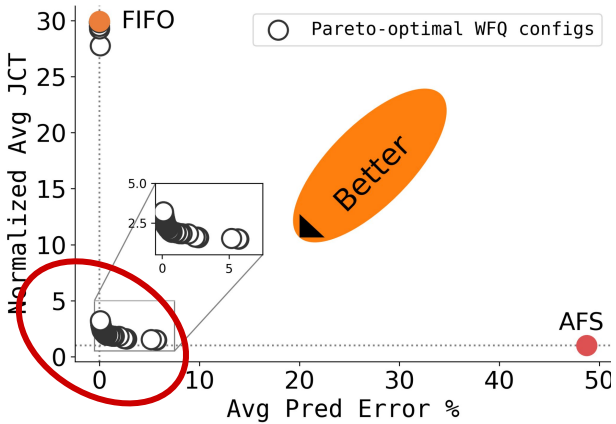
Trace 1



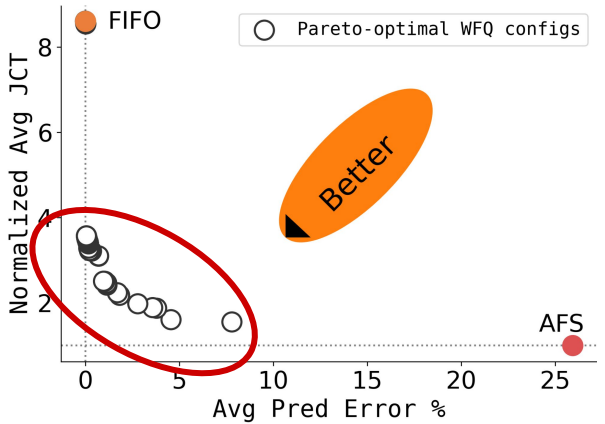
Trace 2

Can the simulation-based search find the Pareto-frontier?

- Run search on two realistic traces (Philly)



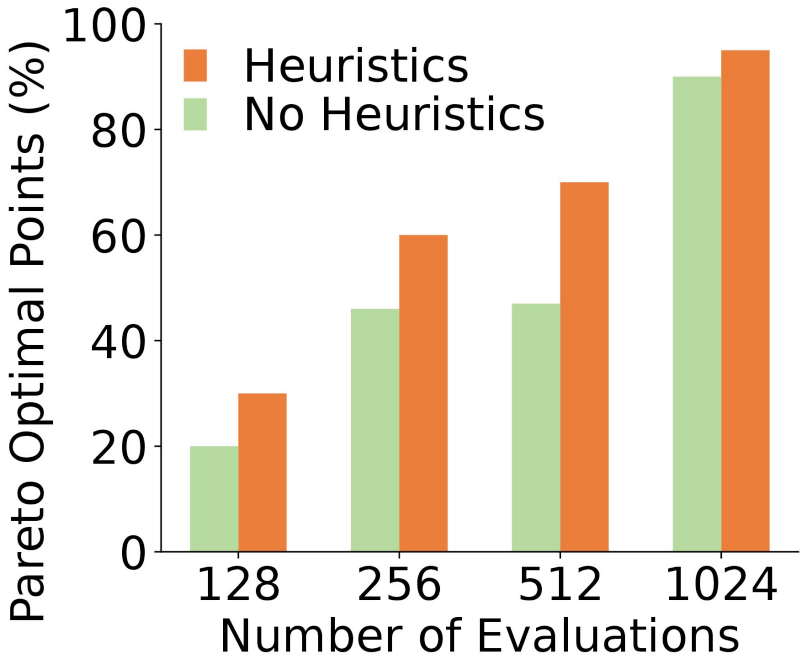
Trace 1



Trace 2

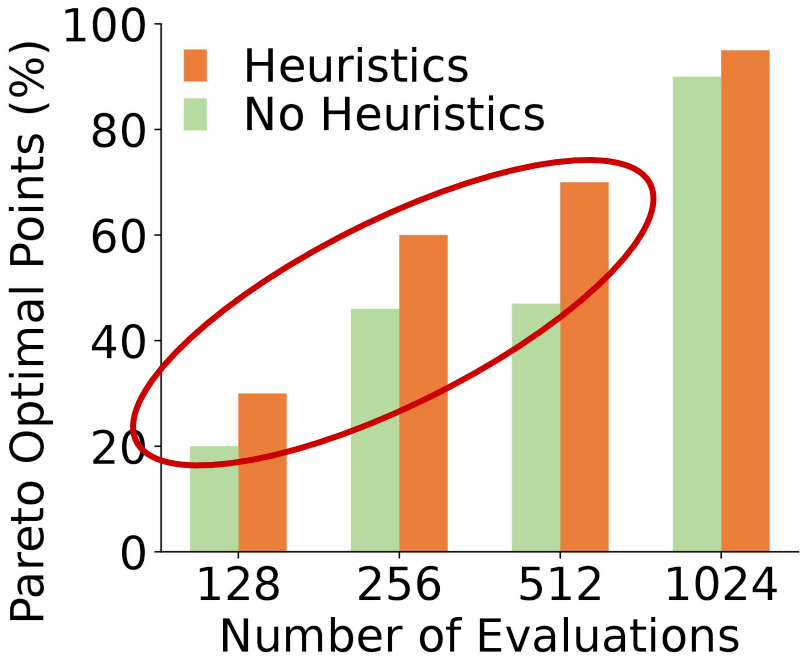
How effective are heuristics in improving search efficiency?

- Track percentage of Pareto-frontier discovery w/ & w/o heuristics w.r.t. budget



How effective are heuristics in improving search efficiency?

- Track percentage of Pareto-frontier discovery w/ & w/o heuristics w.r.t. budget



Summary

- Providing predictions improves user experience
  - Predictability vs. Practicality trade-off -> existing schedulers lie on extremes
- WFQ to bound prediction errors and offer flexibility
- Simulation-based search to achieve Pareto-optimal WFQ configurations
- High-level interface to simplify navigating the trade-off space
- PCS achieves lower prediction errors while being competitive with other schedulers

References:

1. source: https://engineering.fb.com/2024/03/12/data-center-engineering/building-metas-genai-infrastructure/

Thank you!