Cost-Aware Duration Prediction for Software Upgrades in Datacenters

Yi Ding, Aijia Gao, Thibaud Ryden, Michal Sedlak, Essam Ewaisha, Igor Marnat, Henry Hoffmann

Collaboration with Meta Infra Data Center

MLSys Conference (Industry Track), Bellevue, WA, USA, 2026

PURDUE UNIVERSITY | Elmore Family School of Electrical and Computer Engineering

Software Upgrades Keep Datacenters Healthy

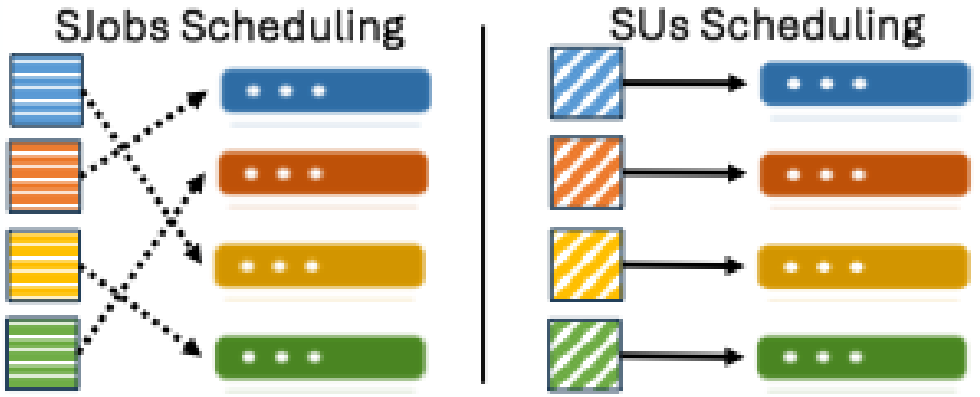
- Datacenters continuously upgrade server software, including:
  - OS and kernel updates
  - Firmware updates
- These software upgrades are necessary to:
  - Improve reliability
  - Fix bugs and vulnerabilities
  - Reduce unplanned downtime
- But at **hyperscale**, even routine software upgrades become a complex scheduling problem.

Software Upgrade (SU) Scheduling at Datacenter Scale

Scheduling SUs across hyperscale datacenters requires coordinating **millions of servers**.

However, SU scheduling differs from traditional service job (SJob) scheduling:

Traditional service job scheduling	Software upgrade scheduling
Jobs can be flexibly distributed	Upgrades are tied to specific servers
SLO focuses on job latency	SLO requires ≥95% of upgrades to finish within a fixed window



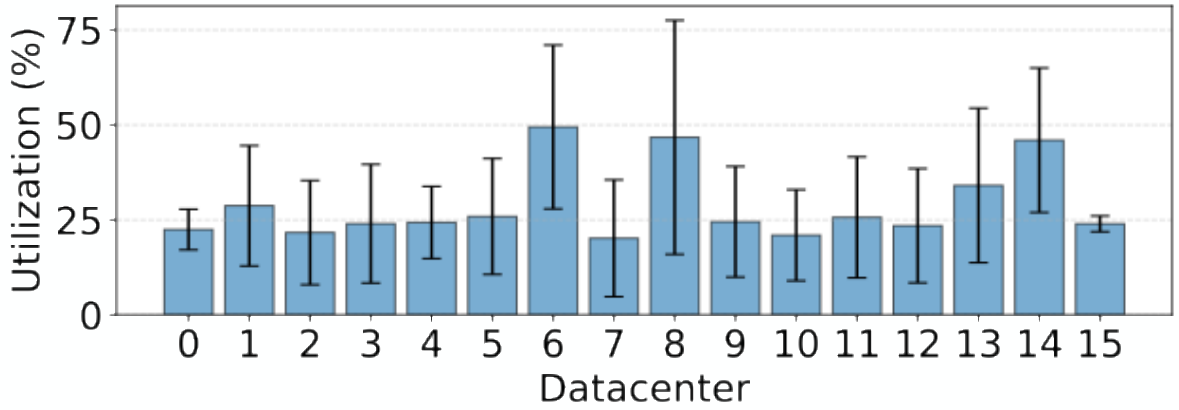
Meta's Existing Solution: Safe but Inefficient

Current approach: assume fixed worst-case upgrade durations.

Benefit: protects SLOs by conservatively limiting upgrades per server.

But: leaves substantial idle time in upgrade windows.

Observation: utilization averages only 20–40% across 16 datacenters over 5 months.



Upgrade window utilization remains low and highly variable.

Our solution: Acela

Acela improves upgrade scheduling efficiency while meeting SLOs.

Core idea:

Use **cost-aware duration prediction** instead of fixed worst-case estimates.

Three techniques:

- Quantile duration prediction

  Models asymmetric misprediction costs.
- SLO-aware model selection

  Selects models that balance prediction error and SLO compliance.
- Straggler-aware training

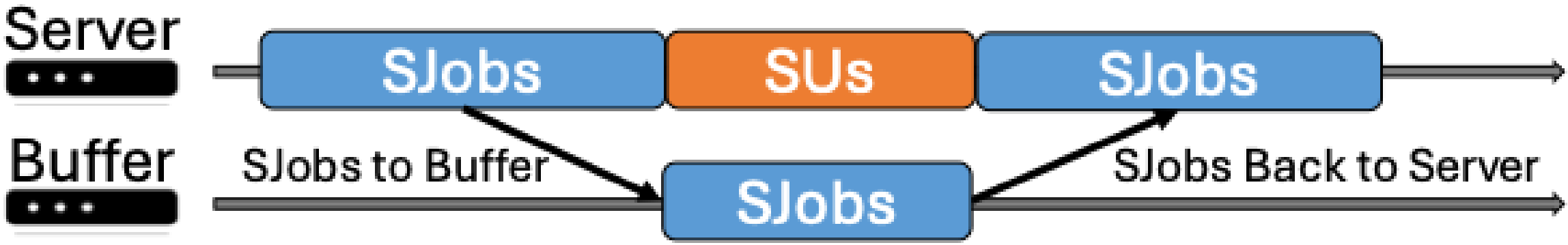
  Removes stragglers to reduce overprediction.

Takeaway:

Acela increases scheduling throughput by predicting ***realistic (not most accurate)*** upgrade durations, not worst-case durations.

A bit more background to understand Acela design…

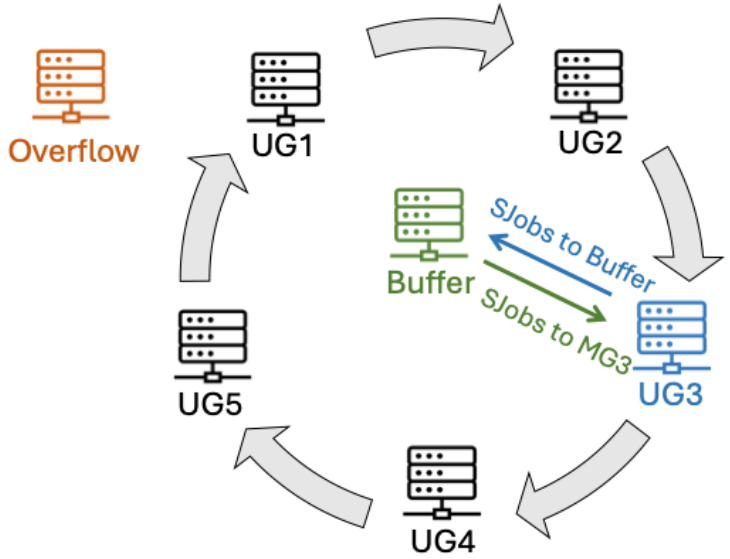
A Sever's Lifecycle



- SJobs (e.g., cloud services, search, video streaming) run during normal operation time.
- SUs and SJobs represent distinct phases of a server's lifecycle.

How Software Upgrades Rotate Across Server Groups

- **Upgrade unit:**
  A rack of servers forms an **upgrade group (UG)**.
- **Server roles:**
  Each UG temporarily serves as:
  - **Upgrade group:** servers being upgraded
    Buffer group: hosts migrated service jobs
    Overflow group: handles jobs from failed servers
- **Process:**
  Roles rotate cyclically across UGs to balance service load and upgrade progress.

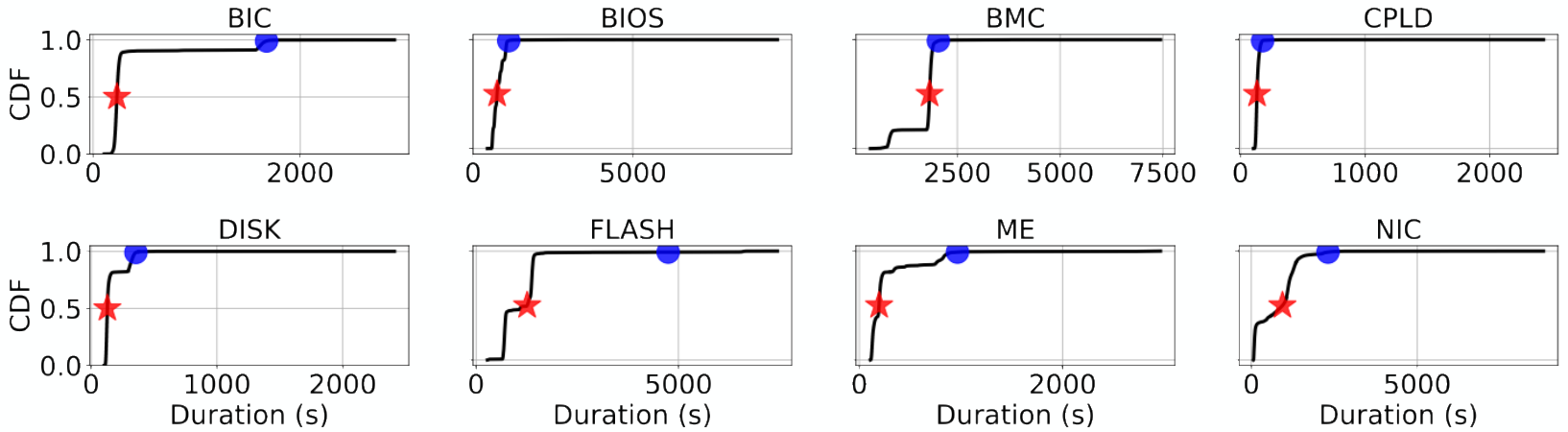


Circle-based scheduling upgrades one group at a time while keeping services running.

Why Duration Prediction Is Challenging

To improve scheduling efficiency, Acela must predict upgrade duration accurately and in an SLO-aware way.

Challenge	Why it matters
Diverse duration distributions	Upgrade types and hardware contexts produce long-tailed, heterogeneous durations
Mismatch with system objective	Low prediction error does not necessarily maximize throughput or protect SLOs



Different upgrade types exhibit different long-tail behaviors.

Mispredictions Have Asymmetric Scheduling Costs

Acela should slightly overpredict to protect SLOs, but avoid extreme overprediction that wastes upgrade windows.

	Mild error	Extreme error
Underprediction	Risks overload upgrade windows	Delays recovery; stalls upgrade cycle
Overprediction	Safer for SLO compliance	Lowers utilization; mimics worse-case scheduling

Use cost-aware prediction that penalizes underprediction and extreme overprediction differently.

Acela Design Principles

Acela predicts upgrade duration in a way that matches the scheduling objective, not just prediction accuracy.

Design principle	Why it matters	Acela technique
Account for asymmetric costs	Underprediction risks SLO violations more than mild overprediction	Cost-aware loss
Balance efficiency and SLOs	Conservative estimates waste upgrade windows	SLO-aware model selection
Limit extreme overprediction	Stragglers bias models toward overly long estimates	Straggler-aware training
Capture duration variability	Upgrade types have different duration distributions	Type-aware duration prediction

Acela is cost-aware, SLO-aware, and variability-aware.

Quantile Regression Handles Asymmetric Costs

Problem: Underprediction can violate SLOs, while mild overprediction is safer.

Key idea: Predict a high quantile, not the mean.

Quantile τ	Behavior	Scheduling effect
τ = 0.5	Median estimate	Balanced errors
τ > 0.5	Longer estimate	Fewer underpredictions
τ too high	Near worst-case estimate	Lower utilization

Acela uses τ as a scheduling parameter:
higher τ → safer SLOs, lower efficiency
lower τ → higher efficiency, higher SLO risk

Acela choice:

- **τ > 0.5** so predictions slightly overestimate duration
- Use Quantile Gradient Boosting Trees (QGBT) models to estimate conditional quantiles

Selecting the Best Quantile Model

Goal: choose the quantile parameter τ that balances efficiency and SLO compliance.

Problem

A higher quantile **τ** reduces underprediction, but too high a τ causes excessive overprediction.

Acela approach

Train multiple QGBT models with different **τ** values, then select the model with the best validation score.

Metric	Meaning
MAE	Prediction error
OPR	Overprediction rate
SLO	Target completion rate, e.g., 95%
α	Penalty for missing the SLO

Scoring intuition

Case	Selection goal
OPR ≥ SLO	SLO likely met ⟶ minimize MAE
OPR < SLO	SLO at risk ⟶ add penalty

The selected model is not simply the most accurate model; it is the model that best supports scheduling.

Reducing Straggler-Induced Prediction Bias

Problem

Training data includes rare stragglers with extremely long durations.

Impact:
Models overestimate normal upgrades and waste upgrade windows.

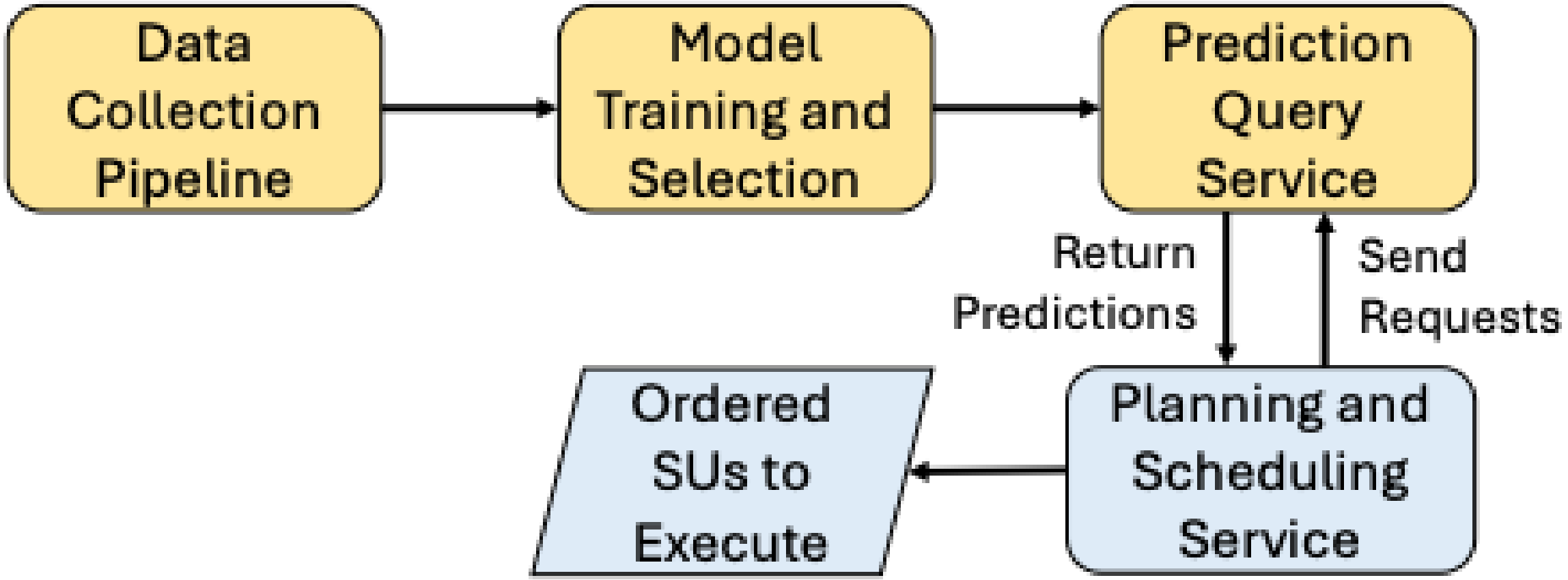
Acela approach

Create multiple training sets by removing extreme-duration upgrades:

Training set	What is removed
Original	No truncation
P99-truncated	Top 1% longest upgrades
P99.9-truncated	Top 0.1% longest upgrades

Result:
Less straggler bias → fewer extreme overpredictions → higher upgrade efficiency

Putting It All Together



Evaluation Methodology

Goal: Measure whether Acela improves scheduling efficiency while meeting the 95% completion SLO.

Dataset

- **4M+** upgrades for training
- **~1M** upgrades for evaluation
- **198** upgrade groups from production datacenters

Focus: firmware upgrades

- Long duration: **1–2 hours**
- High variance across servers and UGs
- Most difficult category for scheduling

Comparison

- **Acela:** duration prediction + SLO-aware model selection
- **Heuristic:** fixed worst-case duration estimates

Takeaway:
Acela is evaluated on real production upgrades under the same SLO constraints used by Meta's existing scheduler.

Main Production Results

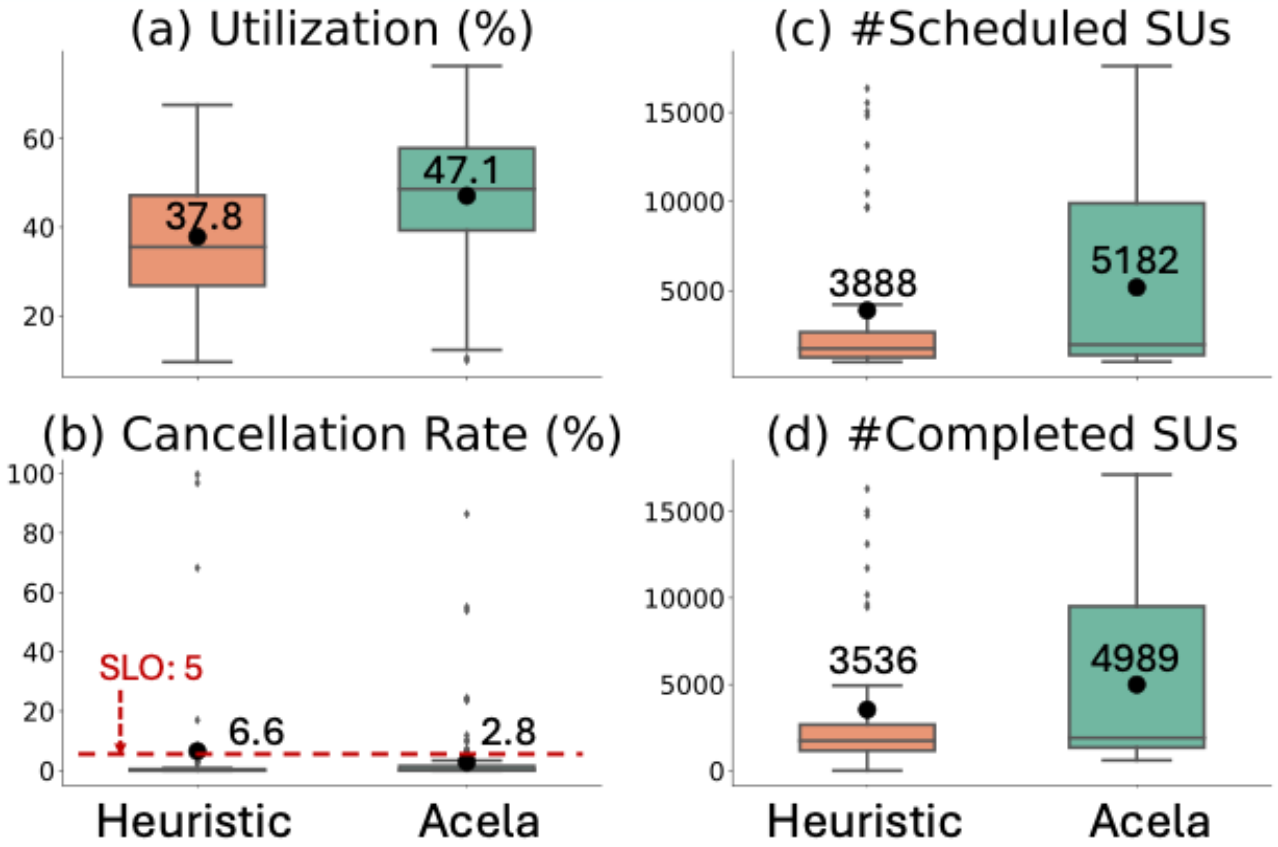
Metric	Heuristic	Acela	Improvement
Window utilization	37.8%	47.1%	1.25× higher
Scheduled SUs	3,888	5,182	+33%
Completed SUs	3,536	4,989	+41%
Cancellation rate	6.6%	2.8%	2.4× lower

SLO result

- Acela meets the 5% cancellation-rate SLO; Heuristic does not.

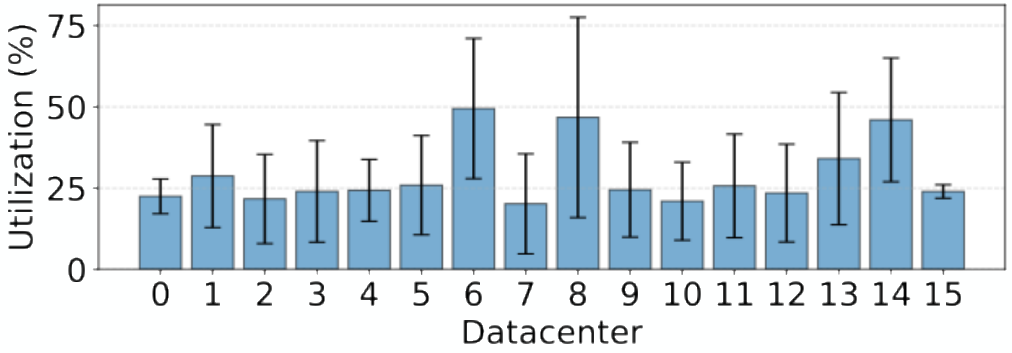
Why Acela improves

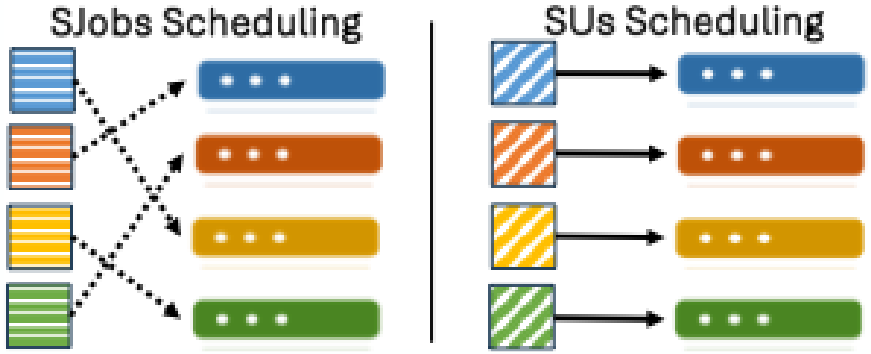
- Duration prediction lets Acela schedule more upgrades without overloading upgrade windows.

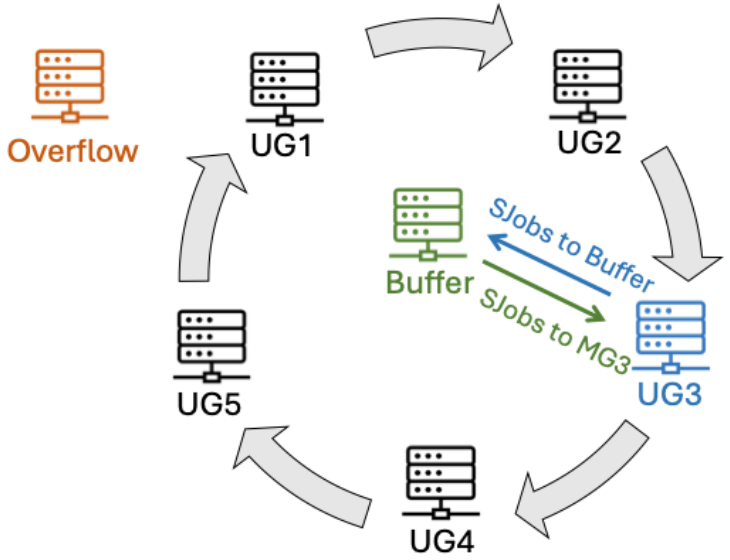


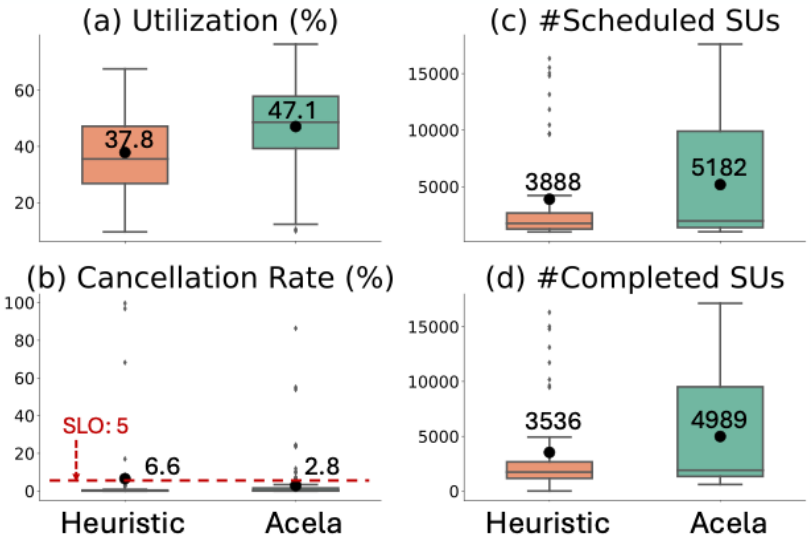
More results in the paper

Conclusion









Small improvements in duration prediction can unlock large gains in datacenter upgrade efficiency.

PURDUE UNIVERSITY | Elmore Family School of Electrical and Computer Engineering

Thank you! Questions?