APPENDIX: COST-AWARE DURATION PREDICTION FOR SOFTWARE UPGRADES IN DATACENTERS

Yi Ding [1] **Aijia Gao** [2] **Thibaud Ryden** [2] **Michal Sedlak** [2] **Essam Ewaisha** [2] **Igor Marnat** [2] **Henry Hoffmann** [3]

ABSTRACT

Software upgrades are critical to maintaining server reliability in datacenters. While job duration prediction and scheduling have been extensively studied, the unique challenges posed by software upgrades remain largely under-explored. This paper presents the first in-depth investigation into software upgrade scheduling at datacenter scale. We begin by characterizing various types of upgrades and then frame the scheduling task as a constrained optimization problem. To address this problem, we introduce Acela, a cost-aware duration prediction framework designed to improve upgrade scheduling efficiency and throughput while meeting service-level objectives (SLOs). Acela accounts for asymmetric misprediction costs, strategically selects the best predictive models, and mitigates straggler-induced overestimations. Evaluations on Meta's production datacenter systems demonstrate that Acela significantly increases efficiency of the existing upgrade scheduler by improving upgrade window utilization by 1.25×, increasing the number of scheduled and completed upgrades by 33% and 41%, and reducing cancellation rates by 2.4×. The code and data sets will be released after paper acceptance.

A FEATURE ANALYSIS

Features. Table 1 summarizes the 24 features used for upgrade duration prediction. We organize them into three categories: (1) hardware structural features, which capture platform-level resource capacity and architectural characteristics; (2) software state features, which describe the firmware type and version transition involved in the upgrade; and (3) workflow and reset features, which reflect procedural complexity and reset semantics. This categorization reflects the underlying factors that influence upgrade duration: platform-dependent flashing behavior, version-specific migration paths, and reset-induced downtime. Together, these features model both static hardware constraints and dynamic upgrade workflow characteristics, enabling accurate prediction of the highly variable firmware upgrade durations observed in practice.

Feature Importances. We obtain feature importance from the trained Gradient Boosting Tree models used for upgrade duration prediction. Specifically, we extract the built-in importance scores provided by the tree ensemble, which quantify each feature's contribution to reducing prediction error across all decision splits. These importance values therefore reflect how frequently and effectively a feature is used to partition the data when modeling upgrade duration.

Figures 1 to 8 visualize the feature importances for each firmware upgrade type. First, different firmware types exhibit distinct dominant features, indicating heterogeneous upgrade mechanisms. For example, mpn (manufacturer part number) is the top feature for BIOS and BMC upgrades, suggesting that component-specific revisions are influencing factors in these categories. In contrast, NIC upgrades are dominated by cpu_cores, while ME upgrades are primarily driven by serf_model_id. FLASH upgrades show a more balanced importance distribution across serf_model_id, logical_server_sub_type, and num_actions, indicating stronger interaction between platform identity and procedural complexity. These differences confirm that no single factor universally influences upgrade duration.

Second, version transition features appear among the top predictors across nearly all firmware types. In particular, kpp_current_version ranks within the top five for BIC, BIOS, DISK, FLASH, NIC, BMC, and ME. kpp_desired_version also appears in several firmware types (e.g., BIC, BIOS, BMC, DISK). This suggests that upgrade duration depends on the version-to-version transition rather than solely on hardware capacity. The current software state influences migration paths, compatibility checks, and post-upgrade validation behavior.

Third, platform identity features emerge as important across

[1]Elmore Family School of Electrical and Computer Engineering, Purdue University, West Lafayette, IN 47906, USA [2]Meta, 1 Meta Way, Menlo Park, CA 94025, USA. [3]Department of Computer Science, University of Chicago, Chicago, IL 60637, USA. Correspondence to: Yi Ding <yiding@purdue.edu>.

Proceedings of the 9^{th} MLSys Conference (Industry Track), Bellevue, WA, USA, 2026.

firmware types. Features such as serf_model_id, logical_server_sub_type, serf_model_make, and mpn appear in the top rankings. This indicates that hardware generation, deployment role, and vendor-specific characteristics affect upgrade behavior. Even when different firmware types emphasize different primary drivers, platform descriptors remain structurally influential. Together, these results demonstrate that accurate firmware upgrade duration prediction requires jointly modeling software state transitions and platform-specific characteristics, with firmware-dependent variation in their relative importance.

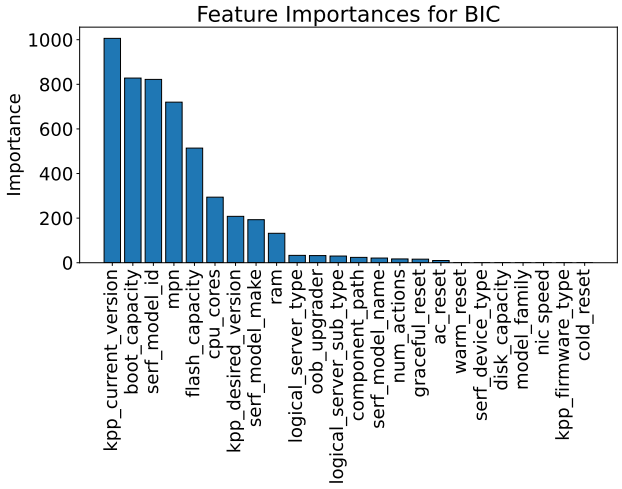

Figure 1. Feature importance for training on BIC upgrades.

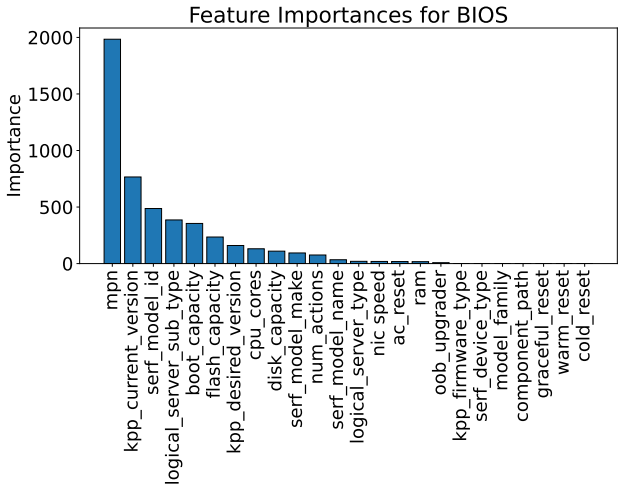

Figure 2. Feature importance for training on BIOS upgrades.

B Comparing to Other Learning Models

To understand the learning model choices of Acela, we compare it against seven representative baselines spanning both

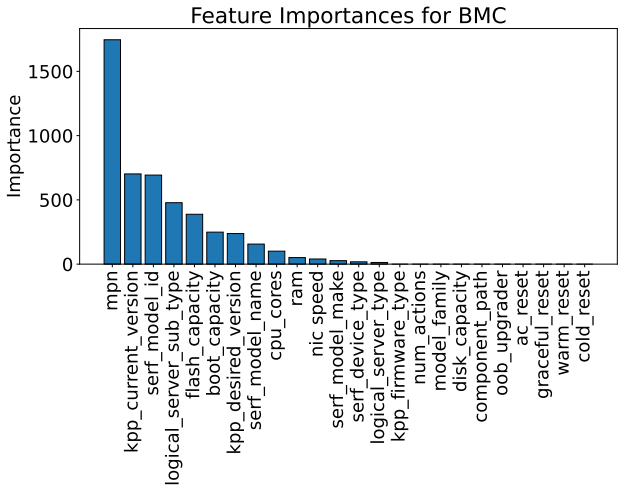

Figure 3. Feature importance for training on BMC upgrades.

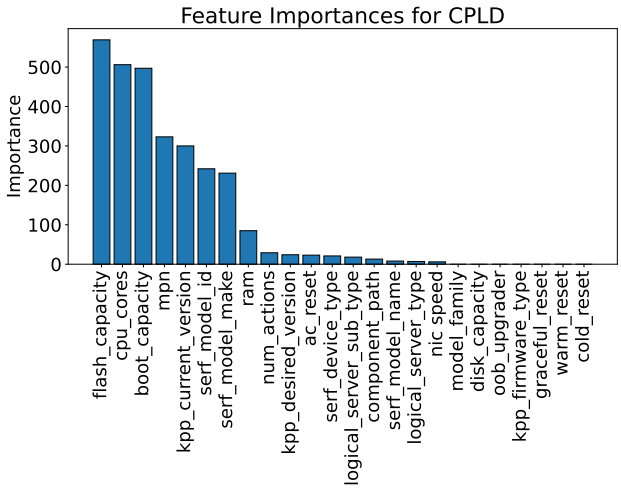

Figure 4. Feature importance for training on CPLD upgrades.

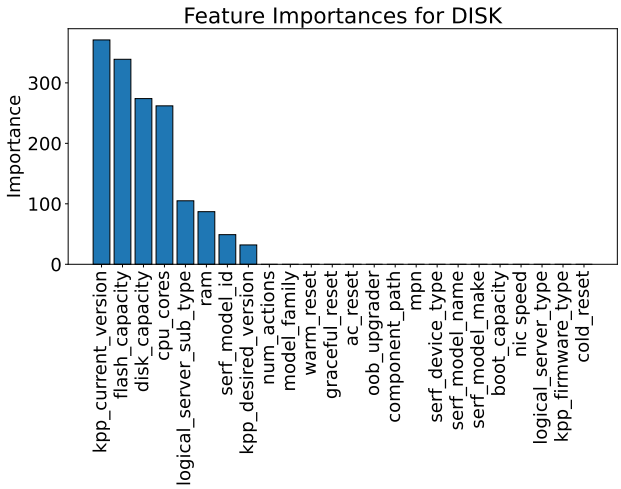

Figure 5. Feature importance for training on DISK upgrades.

Table 1. Features Used for Software/Firmware Upgrade Duration Prediction

Feature	Full Name	Category	Description / Relevance to Duration
Hardware Structural Features			
cpu_cores	Number of CPU Cores	Compute	Total logical/physical cores available. Higher parallelism may accelerate validation, decompression, or initialization phases.
ram	Installed Memory (RAM)	Memory	Total system memory. Insufficient memory may slow post-upgrade initialization or trigger swap overhead.
disk_capacity	Total Disk Capacity	Storage	Total installed storage capacity. Larger storage systems may require longer scanning, validation, or metadata checks.
flash_capacity	Flash Storage Capacity	Firmware Storage	Capacity of onboard flash used for firmware images. Larger flash components increase write and verification time.
boot_capacity	Boot Partition Capacity	Boot Device	Size of boot device/partition. Affects bootloader update and firmware flashing duration.
nic_speed	Network Interface Speed	Network	Maximum NIC link speed. Influences image transfer time and orchestration latency.
serf_model_id	Server Model Identifier	Platform	Internal hardware model identifier capturing platform generation and architectural constraints.
serf_model_make	Server Manufacturer	Vendor	Hardware vendor. Firmware tooling and flashing behavior vary across manufacturers.
serf_model_name	Server Model Name	Platform	Specific server product model. Strong determinant of firmware architecture and upgrade process.
serf_device_type	Device Type	Hardware Type	Physical device category. Influences reset strategy and orchestration workflow.
model_family	Model Family	Hardware Generation	Hardware generation grouping. Captures architectural differences across generations.
mpn	Manufacturer Part Number	Component	Exact component part number. Different revisions may exhibit different flashing durations.
component_path	Hardware Component Path	Topology	Logical/physical path to upgraded component. May impact access and validation latency.
Software State Features			
kpp_current_version	Current Firmware Version	Version State	Existing firmware/software version. Large version gaps may require migration steps or compatibility checks.
kpp_desired_version	Target Firmware Version	Version State	Intended post-upgrade version. Determines migration path and validation complexity.
kpp_firmware_type	Firmware Type Name	Component Type	Type of firmware (e.g., BIOS, NIC). Different firmware types have distinct flashing and reset behaviors.
logical_server_type	Logical Server Type	System Role	High-level server role (e.g., compute, storage). Different roles require different validation workflows.
logical_server_sub_type	Logical Server Sub-Type	System Role	More granular role classification (e.g., GPU node, metadata server). Captures specialization effects.
Workflow and Reset Features			
num_actions	Number of Upgrade Actions	Workflow Complexity	Total number of upgrade steps (flash, reboot, verify). Proxy for procedural complexity and duration.
oob_upgrader	Out-of-Band Upgrade Flag	Upgrade Mechanism	Indicates whether upgrade is performed via out-of-band controller (e.g., BMC). May reduce downtime but add orchestration overhead.
ac_reset	AC Power Reset Required	Reset Type	Indicates full AC power cycle requirement. Typically longest reset duration.
graceful_reset	Graceful Software Reset	Reset Type	Controlled OS-level restart. Usually shorter and more predictable.
warm_reset	Warm Reboot	Reset Type	Hardware reset without full power cycle.
cold_reset	Cold Reboot	Reset Type	Full reboot including hardware reinitialization without full AC cycle.

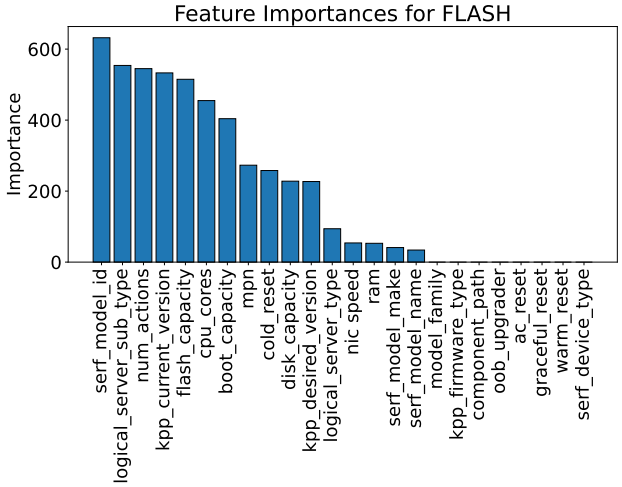

Figure 6. Feature importance for training FLASH upgrades.

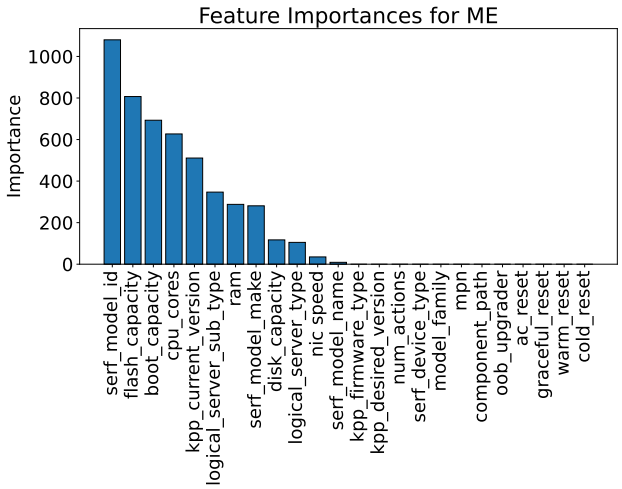

Figure 7. Feature importance for training on ME upgrades.

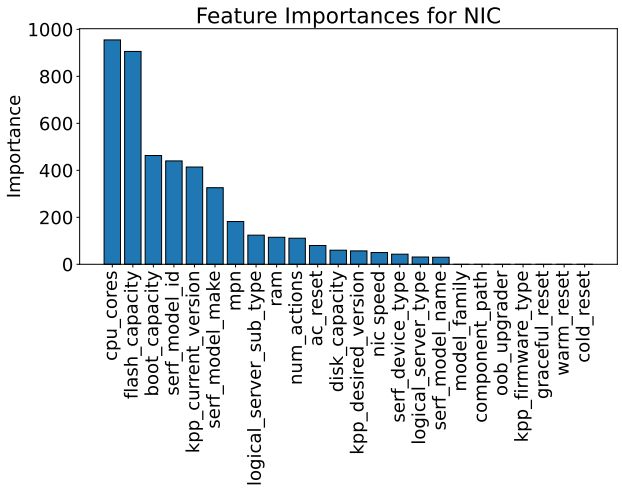

Figure 8. Feature importance for training on NIC upgrades.

learning-based and heuristic approaches. These baselines reflect common modeling strategies used for duration prediction, including symmetric-loss regression models and simple heuristics:

- **P95**: A heuristic baseline that predicts durations using the 95th percentile of historical data, reflecting conservative worst-case provisioning commonly used in production systems.
- **RF (Random Forest)**: An ensemble tree-based model that improves robustness through bagging and non-linear feature interactions.
- **ElasNet (Elastic Net)**: A linear model combining L1 and L2 regularization to balance sparsity and stability.
- **SVR (Support Vector Regression)**: A kernel-based model that captures non-linear relationships through margin-based optimization.
- **Ridge**: A linear regression model with L2 regularization to reduce variance.
- **LR (Linear Regression)**: A standard least-squares regression model that predicts the conditional mean.
- **Avg**: A heuristic baseline that predicts durations using the average per firmware type.

These baselines primarily optimize prediction accuracy under symmetric loss functions, which aim to minimize average error but do not account for the asymmetric operational consequences of prediction errors in upgrade scheduling.

To evaluate model performance, we consider two complementary metrics: Mean Absolute Error (MAE) and Overprediction Rate (OPR). MAE captures prediction accuracy, while OPR measures the fraction of predictions that overestimate actual durations and directly reflects the ability to meet the system's SLO (i.e., ensuring upgrades complete within the upgrade window). As discussed in the paper, these two objectives are inherently misaligned: minimizing MAE encourages predicting the conditional mean, while meeting the SLO requires intentional overprediction to avoid underestimation risks.

Figure 9 shows this MAE–OPR tradeoff across firmware types. Models optimized purely for accuracy (e.g., RF, SVR, Ridge, LR) achieve low MAE but consistently exhibit insufficient OPR, failing to satisfy the SLO requirement. This behavior stems from their reliance on symmetric loss functions, which treat under- and over-predictions equally and therefore center predictions around the conditional mean. As a result, they underpredict too frequently in a system where underprediction carries higher cost.

In contrast, heuristic approaches such as P95 and Avg achieve higher OPR by construction but suffer from large MAE due to their inability to capture fine-grained variability across upgrade types and system contexts. These methods resemble the existing worst-case scheduling strategy and

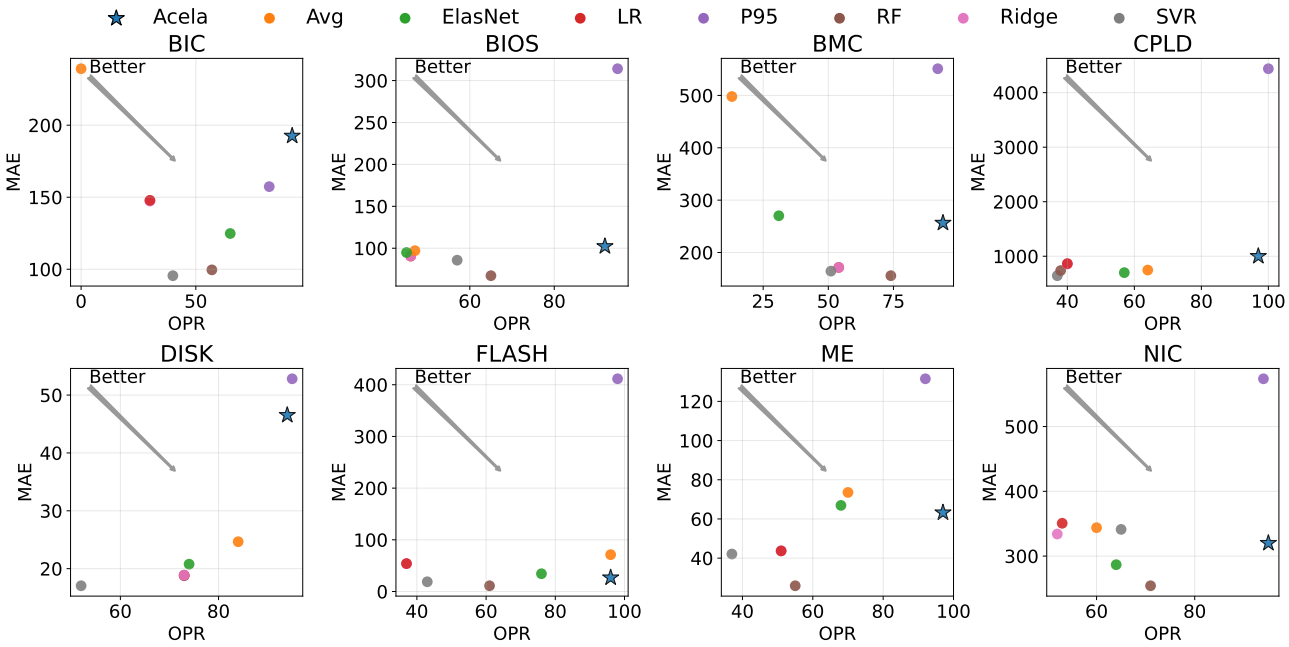

Figure 9. Tradeoffs between OPR and MAE across firmware types.

lead to inefficient resource utilization.

Acela achieves the best tradeoff between MAE and OPR. By explicitly modeling asymmetric misprediction costs through quantile regression and selecting models using a cost-aware scoring function, Acela shifts predictions toward slight overestimation while avoiding excessive conservatism. This enables Acela to achieve OPR close to the SLO target (95%) while maintaining competitive MAE.

Overall, these results highlight a key insight: prediction accuracy alone is insufficient for system optimization. Models that minimize MAE do not necessarily lead to better scheduling outcomes. Instead, Acela's cost-aware design enables it to align prediction behavior with system-level objectives, achieving a superior balance between accuracy and operational reliability across firmware types.