CAPES: Causal Analysis of Power Effect under Score-based Scheduling

Jiali Xing
University of Pennsylvania
Philadelphia, USA
xjiali@seas.upenn.edu

William Meng
University of Pennsylvania
Philadelphia, USA
willmeng@seas.upenn.edu

Ziqi Meng
University of Pennsylvania
Philadelphia, USA
zqmeng@seas.upenn.edu

Liangcheng Yu
Microsoft Research
Redmond, USA
liangchengyu@microsoft.com

Vincent Liu
University of Pennsylvania
Philadelphia, USA
liuv@seas.upenn.edu

Benjamin Lee
University of Pennsylvania
Philadelphia, USA
leebcc@seas.upenn.edu

Abstract

Accurate Virtual Machine (VM) power estimation is critical for safe oversubscription and power-aware scheduling, but score-based placement policies distort historical observations and bias template-based predictive models. We present the CAPES framework, an offline inference layer that estimates the *causal* Physical Machine (PM) power increase from launching new VMs under rule-based schedulers. CAPES exploits the score cutoff induced in each burst and applies regression discontinuity to compare near-cutoff PMs with similar pre-state but different placement outcomes. Our experiments show that CAPES reduces underestimation and improves interval coverage versus template-based ML baseline while achieving similar mean absolute error.

Keywords

Virtual Machine, Power, Public Cloud, Causal Analysis.

1 Introduction

Datacenter management requires answering *what-if* questions: what would happen if we changed policies? How much more power would a server consume with versus without a new VM? What would the rack temperature be with versus without a new job? Unlike mere correlations, causal analysis predicts counterfactual outcomes—what *would* happen under different placement decisions. Estimates for such questions support capacity planning, admission control, and service-level guarantees. Yet obtaining unbiased estimates is difficult in production systems where selection biases inherent in schedulers can obscure true causal relationships. Power estimation illustrates this challenge.

Power has emerged as a binding constraint as computation consolidates in hyperscale facilities and deploys on high-density GPUs [19]. Datacenter operators often oversubscribe facility power to maximize utilization without tripping circuit breakers [13, 16, 18]. Estimating power used by particular software and hardware combinations is therefore essential for safe power management.

Historical scheduling decisions could inform power estimates when placing a particular workload on a candidate server. Yet such observational data is biased by the scheduler, which tends to favor particular workload-server assignments according to its policy and rules. Machine learning with this data will produce biased, rule-driven correlations between workload types, server features, and power outcomes rather than the causal effect of placing a workload on a server.

Bias is a particular issue when estimating workload power on a server rarely selected under the scheduler's policy. Machine learning would extrapolate from a skewed slice of the power distribution. Even worse, scheduling rules might evolve as operators tune for efficiency or new system configurations. Each rule change induces a distribution shift and breaks learned correlations even when underlying causal relationships are unchanged.

Yet, the same scheduler that creates selection bias also creates opportunity for causal identification. For each workload, modern schedulers score candidate servers based on their hardware resources and operating conditions. The scheduler prioritizes servers with the lowest scores and results in an implicit cutoff: servers below a score cutoff receive the workload and servers above do not. Such cutoffs exist in Google's Borg [22, 23], Azure's Portean [7], and emerging LLM serving systems such as Serverless LLM and Llumnix [6, 17, 20].

The scheduling cutoff creates a discontinuity that we exploit for causal analysis. Near the cutoff, two servers have nearly identical scores and thus nearly identical hardware resources and operating conditions. But they receive different workload assignments due to the scheduler's server selection rule. This discontinuity approximates a randomized experiment as workload assignment changes sharply at the score cutoff while server conditions change smoothly. Workload arrivals, which can be bursty, further facilitate

natural comparisons around the cutoff. When many workload placement decisions occur within short windows, each burst of decisions creates its own cutoff and generates repeated natural experiments from the normal operation of a production system.

We present the CAPES, a new framework that exploits scheduler cutoffs and estimates causal effect from a production system. CAPES ingests workload placement logs and power telemetry. It then identifies windows with bursts of scheduling decisions that reveal implicit cutoffs. Finally, it uses Regression Discontinuity Design (RDD) and causal forests to estimate marginal power used by a virtual machine [2, 5, 10, 14, 21]. We evaluate CAPES on Azure VM workloads under Protean-style scheduling [7]. Compared to a machine learning baseline, CAPES achieves comparable accuracy while mitigating systematic bias (under-estimation) when predicting virtual machine power on physical machines rarely selected by scheduler's preference rules.

2 Causal Analysis of Power Effect

This section presents the causal analysis framework underlying CAPES. We first show how score-based scheduling creates natural experiments (§2.1). We then introduce RDD as the method to exploit these experiments (§2.2) and formalize the identification strategy for power effects (§2.3). We extend RDD with causal forests to estimate heterogeneous power effects across diverse server conditions (§2.3). Finally, we describe the CAPES framework architecture (§2.4).

2.1 Score-Based Scheduling Creates Natural Experiments

The gold standard for causal analysis is the randomized controlled trial (RCT): randomly assign workloads (the *treatment*) to servers, then measure the difference in server power. However, production public clouds use optimized scheduling policies instead of random placement, and A/B tests that deliberately vary VM assignments would violate service-level agreements. We observe that modern datacenter schedulers create natural experiments that approximate randomized assignment.

Datacenter schedulers assign workloads to servers using rule-based scoring. For each virtual machine (VM) that seeks assignment to a physical machine (PM) or server, the scheduler evaluates the PM against a set of preference rules that encode constraints and objectives. These considerations might account for hardware resources and power capacity, software versions and server configurations, as well as proximity to data. The scheduler computes a scalar score for each candidate based on the extent to which various rules and preferences are satisfied by the PM. Lower scores indicate a better fit. The scheduler selects from the best candidates, typically the PM with the lowest score, to receive the workload.

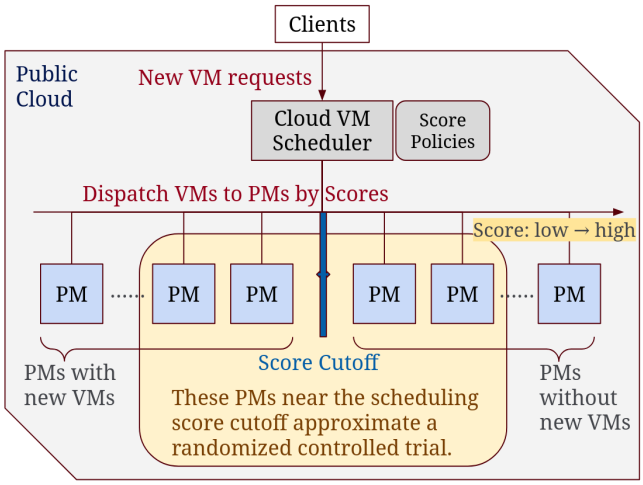

Figure 1: Score-based scheduling induces a cutoff separating selected from unselected servers. Near-cutoff servers have similar pre-placement states but different placement outcomes, enabling quasi-experimental comparison.

This selection procedure induces an implicit cutoff for each scheduling decision. The score of the best server defines a threshold. Servers scoring below this threshold receive the workload while servers scoring above do not. Crucially, the cutoff is not fixed and set by the datacenter operator. Rather, the cutoff emerges from dynamic interactions between VM demand, available system capacity, and the scoring rules employed by the scheduler. The cutoff also varies depending on the scheduling window and mix of VM types within that window.

Figure 1 illustrates the cutoff in scheduling scores. Clients submit VM requests to the scheduler, which scores PMs and then dispatches workloads to the top-ranked servers. The cutoff partitions the pool of candidate PMs into groups of selected and unselected servers. Near this cutoff, servers exhibit nearly identical pre-placement system conditions, yet receive different VM placement decisions.

Production VM traces exhibit several characteristics that further support this natural experiment formulation [7, 13].

- **VM Types.** VM requests are organized into discrete types and placement decisions are made per type. Although datacenters support hundreds of VM types, the top three types account for more than 50% of arrivals.
- **Bursty Arrivals.** Arrivals are highly bursty. Within short ten-minute windows and for dominant VM types, the peak arrival rate is 6–22× higher than the average[1].

[1]Less frequent VM types exhibit even higher burstiness (10–50×).

Table 1: Mapping score-based scheduling to causal analysis.

RDD Concept	Scheduler Analog	Description
Unit i	Physical machine	Entity receiving treatment
Features X_i	PM features	PM state and configuration
Running variable r_i	Scheduler score	Determines selection; continuous
Cutoff c_j	Selection threshold	Implicit; varies per window
Treatment T_i	VM placement count	Binary or multi-valued
Outcome Y_i	PM power	P_{95} post-placement power

This burstiness produces repeated, comparable placement decisions.

- **VM Type Isolation.** Within short ten-minute windows, 95-100% of PMs that receive a particular VM type host only that type during the window. The concurrent placement of multiple, heterogeneous VM types is rare.
- **Short VM Lifetimes.** Most VMs run for less than an hour and power effects manifest quickly after placement.

These characteristics, combined with the score-based cutoff, create natural experiments: servers near the cutoff have similar conditions but different placement outcomes. This structure enables causal analysis via RDD, which we introduce next.

2.2 RDD: Causal Analysis at the Cutoff

RDD is a quasi-experimental method for causal analysis when a treatment decision is determined by whether a continuous variable crosses some threshold [2, 5, 10, 14, 21]. Under specific conditions and assumptions, RDD allows us to estimate the causal effect of that treatment from purely observational data, avoiding the costs of randomized controlled trials (*e.g.*, A/B tests).

The underlying intuition is best illustrated by a canonical example. Suppose we wish to estimate the earnings effect of attending a selective university [8]. Naively comparing graduates from selective versus non-selective universities confounds the effects of education and selection. Admitted students typically exhibit higher ability or socioeconomic status that would lead to higher earnings regardless of the university they attend. However, if admission depends on a quantitative cutoff (e.g., SAT exam score greater than 1200), applicants that report scores immediately above and below the cutoff (e.g., 1200 versus 1199) are statistically indistinguishable in their latent ability such that we can view their one-point difference as noise. Near the cutoff, admission is effectively random. Any discontinuity in later earnings between these groups of students reveals the causal effect of university attendance.

Formally, let r denote the running variable, c denote the cutoff, and $D = \mathrm{I}[r \geq c]$ denote the treatment decision. For our example, r corresponds to the exam score, c the admission threshold, and D the admission decision. RDD identifies the causal effect by comparing outcomes just above and below the cutoff where treatment decision changes sharply but other features of the applicant change smoothly.

The RDD method makes two assumptions. First, expected potential outcomes are continuous at the cutoff. Units just above and below the threshold would have similar outcomes in the absence of treatment. With this assumption, any discontinuity or difference in observed outcomes must therefore be caused by the treatment itself. Second, units cannot precisely control or manipulate their running variable to influence their position across the cutoff. If applicants could manipulate their exam scores to land at exactly 1200, the comparison of individuals across this cutoff would be confounded by their unobserved characteristics.

The datacenter scheduler naturally satisfies both assumptions. For continuity, pre-treatment PM features (hardware type, available cores, memory, and CPU utilization) vary smoothly at the cutoff with no significant discontinuities.[2] For non-manipulation, scores are computed deterministically by the scheduler based on system state.[3]

Table 1 shows how the VM scheduling system maps directly to the causal analysis framework. We model each PM as a unit and focus on short time windows with high VM arrivals for a given VM type. The treatment is the count of newly placed VMs on that type of a PM within the windows. The PM's power increase after VM placement is viewed as the causal effect of that placement. Production traces provide strong evidence that this formulation is both representative and suitable for causal analysis. We formalize this identification strategy next.

2.3 Formalizing Power Effect Identification

Consider a VM type j and a short scheduling window. For each candidate PM i, the scheduler computes a scalar score $r = S(PM_i, j)$. The scheduler selects servers below an implicit cutoff c_j. Let $D_i = \mathrm{I}[r_i < c_j]$ indicate this selection. Let $T_i \in \{0, 1, 2, \ldots\}$ denote the number of VMs of type j that were placed on PM i in this scheduling window.

Potential outcomes. Let $Y_i(t)$ be the power outcome if PM i were to be assigned t additional VMs of type j. The causal effect of launching one VM on PM i depends on the difference in potential outcomes:

$$\tau_i = Y_i(1) - Y_i(0)$$

[2] RDD does not require globally homogeneous PMs—only that PM features vary continuously at the cutoff. The continuity of important features is confirmed statistically via *robust local polynomial regressions* [1] ($p > 0.05$).
[3] The McCrary density test [3, 15] confirms no discontinuity in score density at the cutoff ($p > 0.05$).

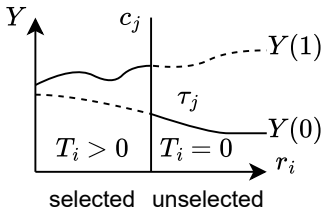

Figure 2: Illustration of rule-based scoring and Regression Discontinuity.

Note that only one of these potential outcomes is observed per PM. Once the scheduler makes a scheduling decision $T_i = t$, we observe $Y_i(t)$ and the counterfactual $Y_i(t')$, $t' \neq t$ is unobservable. This is the core challenge in causal identification.

Figure 2 illustrates the relationship between scheduler scores and power. The x-axis shows scores with a vertical line to indicate cutoff c_j. Servers to the left of this cutoff ($r_i \leq c_j$) are selected and receive a VM for computation and we observe $Y_i(t)$. Those to the right ($r_i > c_j$) are not and we observe $Y_i(0)$. The vertical gap between $Y_i(1)$ and $Y_i(0)$ represents the treatment effect τ_i that we wish to estimate.

Regression Discontinuity Design. The RDD framework estimates τ_i by exploiting the discontinuity at the scheduling cutoff. Assume expected potential outcomes are continuous at the threshold. This assumption states that servers just above and below the cutoff would have similar power outcomes under the same treatment decision.

$$\lim_{r \downarrow c_j} \mathbb{E}[Y_i(t) \mid r_i = r] = \lim_{r \uparrow c_j} \mathbb{E}[Y_i(t) \mid r_i = r] \quad \text{for } t \in \{0, 1\}$$

RDD defines the outcome discontinuity as follows. Because the cutoff creates only a discontinuity in treatment decision and not discontinuities in server conditions, any discontinuity in observed power outcomes Y is driven solely by the discontinuity in treatment T.

$$\Delta Y \triangleq \lim_{r \downarrow c_j} \mathbb{E}[Y_i \mid r_i = r] - \lim_{r \uparrow c_j} \mathbb{E}[Y_i \mid r_i = r]$$

The causal effect at the cutoff is determined by normalizing the jump in power outcomes ΔY by the jump in treatment decisions ΔT.

$$\tau_j^{\text{RDD}} \triangleq \frac{\Delta Y}{\Delta T}$$

For binary treatment, where $\Delta T = 1$, this causal effect simplifies to the following.

$$\tau_j^{\text{RDD}} = \Delta Y = \mathbb{E}[Y_i(1) - Y_i(0) \mid r_i = c_j]$$

However, RDD yields only an average effect at the cutoff. We next extend it to estimate server-specific effects.

2.4 Causal Forests for Heterogeneous Effects

Practically, we wish to understand heterogeneous power effects. The same VM type j may increase power by varying degrees depending on server configurations, resource utilization, colocated computation, and hardware architecture. The RDD estimand τ_j^{RDD} summarizes an average power effect at the cutoff but for power-related decision making, we would prefer predictions tailored to a server's specific operating conditions.

We define the conditional average treatment effect (CATE) as follows. X_i captures PM features such as background processor utilization, memory pressure, hardware configurations, or colocated computation. CAPES estimates $\tau_j(X)$ using causal forests, a non-parametric method for estimating heterogeneous treatment effects. Causal forests extend random forests to causal analysis, using tree splitting criteria that maximize heterogeneity in treatment across sub-groups.

$$\tau_j(X) \triangleq \mathbb{E}\big[Y_i(1) - Y_i(0) \mid X_i = X\big] \tag{1}$$

$$\hat{\tau}_j(X) = \frac{1}{|\mathcal{L}(X)|} \sum_{i \in \mathcal{L}(X)} \hat{\tau}_i \tag{2}$$

CAPES trains a causal forest using only the data for PMs within a band ϵ around the observed cut-off. The bandwidth ϵ can be set using data-driven optimal selectors [1, 9] or chosen as a fixed percentage. It pools data for all PMs near the cutoff across multiple scheduling windows. The pooling allows CAPES to accumulate enough data for causal forest training. The forest then learns how effects τ_j vary with server features X. Estimated value $\hat{\tau}_j(X_i)$ is a server-specific power effect based on a sub-group of similar servers in the causal forest $\mathcal{L}(X)$. These power estimates can be integrated into the existing cloud schedulers. We next describe how CAPES implements this pipeline.

2.5 CAPES Framework and Implementation

Illustrated in Figure 3, CAPES is an offline inference layer that augments a production scheduler. It targets schedulers that (i) compute scores for candidate servers per workload type, (ii) select the best candidate servers based on scores, (iii) induce score cutoffs when scheduling bursts of workloads. Given historical placement logs and server power telemetry, CAPES executes in four stages.

(1) Log ingestion. For each VM type, CAPES extracts the set of candidate PMs considered for each placement decision, their calculated scores, and the placement outcomes. The logs indicate which PMs received VMs and how many.

(2) Scheduling window. CAPES identifies a scheduling window with bursts of VM requests. For each VM type and window, it determines a cutoff score based on the best score

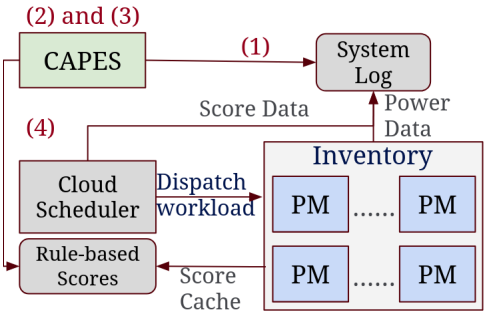

Figure 3: CAPES framework design overview. PM refers to servers.

among PMs that received at least one new VM. CAPES collects PM features before the scheduling window and it collects PM power telemetry after the scheduling window for servers within the ϵ band around the cutoff.

(3) Causal forest. CAPES fits a causal forest on pooled PM data around the scheduling cutoff to estimate heterogeneous treatment effects $\tau_j(X)$. The framework supports multi-valued treatments. Specifically, when multiple VMs of the same type are placed on a single PM within a single scheduling window, the treatment variable T_i takes a value greater than one.

(4) Effect estimation. CAPES estimates the power effect $\hat{\tau}_j(X_i)$ and confidence intervals. These estimates can be shared with datacenter operators for capacity planning or integrated directly into the scheduler's scoring rules.

All inference runs offline or asynchronously. CAPES consumes historical logs and power telemetry after system execution to produce a causal forest. Today's schedulers already cache scores r_i for each server and workload type. The causal effect estimate $\hat{\tau}_j(X_i)$ can be similarly cached and incorporated into the scoring function. At runtime, the scheduler performs its routine lightweight score lookup with no additional overhead. The design and implementation requires no scheduler modifications, no active experimentation, and no additional instrumentation beyond existing scheduling logs and power telemetry.

3 Experimental Methodology

We evaluate CAPES against template-based ML baselines on estimating the PM-level power increase from new VM launches.

Trace replay. We replay the Azure VM trace [7] on a synthetic cluster of 121,510 PMs (112-core, 252 GB PMs) using Protean-style scoring (*BestFit + PreferNonEmptyMachines*) [7]. For VM type #42 (most popular), we extract the top busiest 10-minute windows per day over 5 days, logging PM features, VM placements, scores, and cutoffs.

Ground-truth power measurement. We run controlled experiments on CloudLab (dual AMD EPYC 7543, 256 GB, Ubuntu 20.04). Each VM executes `stress-ng` at its trace-derived 95th-percentile CPU utilization [7]. We measure IPMI power *with and without* new VMs; the difference is ground-truth τ_i. This controlled experiment provides ground-truth labels for evaluation only and is invisible to both ML baselines and CAPES during training.

Models. CAPES trains Causal Forest on PMs within ±5% of cutoffs using aggregated PM features (CPU/memory utilization percentiles, VM subscription/type entropy, pre-burst state). ML baseline trains on full replay history and the observed PM power. The ML baseline implements Azure's PBPO [13]: (1) a two-stage RandomForest classifies each VM into four CPU utilization buckets based on VM configuration and subscription history; (2) a linear regression predicts PM CPU delta from bucket predictions of new VMs; (3) a cubic polynomial maps predicted CPU delta to power delta.

4 Evaluation

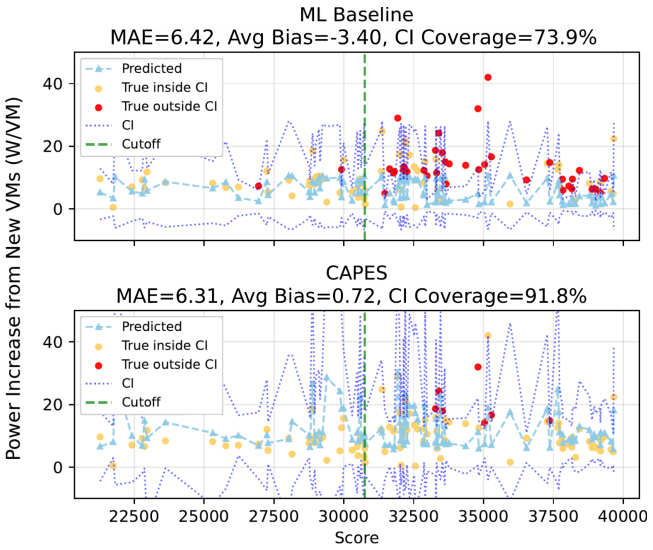

Figure 4: Predicted (blue triangles) vs. ground-truth (yellow/red circles) marginal power increase per VM for ML baseline (top) and CAPES (bottom). Purple dashed lines: 95% confidence intervals. Circles: ground-truth power increases from controlled experiments, yellow if within the confidence interval, red otherwise. Green vertical dashed line: score cutoff.

Figure 4 plots, for each treated PM, the ground-truth marginal power increase per VM and the corresponding predictions from ML baseline (top) and CAPES (bottom). The

x-axis is the scheduler score, and the vertical dashed line marks the empirical cutoff. For each method, we also plot its predicted uncertainty interval (purple dashed) and mark interval misses as red points.

PMs with lower scores than the cutoff will receive new VMs, so for each server i the statistical models observe $Y_i(T_i),\ T_i > 0$ on the left side of cutoff line. The remaining PMs are not selected and models observe $Y_i(0)$ on them. This is same as the illustration in Figure 2 but Figure 4 plots the τ_i instead of Y_i on y-axis.

Results. Overall, the ML baseline attains Mean Absolute Error (MAE) =6.42 W/VM with average bias=−3.40 W/VM and interval coverage=73.9%. CAPES attains MAE=6.31 W/VM with bias=0.72 W/VM and coverage=91.8%.

Breaking down by treatment groups reveals a stark difference. The ML baseline performs well on treated PMs (n=159), achieving MAE=4.95 W/VM and bias=−0.23 W/VM. However, it performs poorly on control PMs (n=285), with MAE=7.47 W/VM and bias=−5.28 W/VM. In contrast, CAPES maintains consistent performance across both groups. On control PMs, bias is only −0.08 W/VM. Visually, the ML baseline exhibits more interval misses (red circles).

Implication. The ML baseline exhibits systematic underestimation of VM-induced power increase, with error concentrated on control PMs where bias reaches −5.28 W/VM—nearly half the average marginal effect of 11 W/VM. In contrast, CAPES maintains low bias and higher interval coverage, suggesting better-calibrated uncertainty and improved generalization beyond the rule-selected region.

The template-based ML baseline is trained on historical placements shaped by scheduler rules, so treated observations concentrate in the selected region ($r_i \le c_j$). When predicting for PMs outside this region, the baseline extrapolates from a biased slice and systematically underestimates power increase—consistent with the large negative bias on control PMs in Figure 4. In contrast, CAPES ties estimation to the score cutoff and uses near-cutoff samples as derived in Section 2.3. This local comparison reduces rule-tied selection effects and yields lower-bias estimates and better interval coverage.

5 Discussion and Related Work

Performance-counter based VM power estimation. Joulemeter and follow-up work estimate VM power from OS-level resource signals and (when available) hardware counters [11, 12]. Unlike statistical approaches, counter-based methods do not scale well in public clouds due to the overhead of collecting fine-grained microarchitectural data across thousands of heterogeneous servers. Moreover, they cannot predict power for VMs that have not yet arrived, as they require runtime measurement. CAPES instead targets the *marginal* power increase from new VM launches and identifies it from near-cutoff comparisons under score-based scheduling.

Power capping and oversubscription. Datacenter power delivery is traditionally provisioned based on conservative "nameplate" peak power, leaving significant capacity stranded [13]. Production systems employ oversubscription to reclaim this capacity by exploiting statistical variance in multi-tenant workloads [13, 18]. Most solutions rely on reactive power capping (*e.g.*, Intel RAPL) to prevent circuit breaker trips [13, 18], but hardware-level throttling creates "noisy neighbor" effects by impacting all VMs on a socket [11, 13]. Operators thus set conservative budgets where capping is a rare emergency rather than a proactive signal [13].

RDD and causal forests. CAPES builds on two econometric advancements. Cattaneo et al. formalize normalizing and pooling RDD estimates across multiple cutoffs [4], allowing CAPES to aggregate many burst windows. CAPES employs Causal Forests [24] for heterogeneous treatment effects, whose "honesty" property ensures pointwise consistency and rigorous confidence intervals for server-specific predictions.

Non-bursty arrivals. CAPES targets bursty arrival patterns where multiple placements per window create a well-defined score cutoff for RDD. For VMs with sparse, uniform arrivals, RDD degenerates to matched-pair comparisons with reduced statistical power.

6 Conclusion and Future Work

Score-based schedulers complicate power modeling by introducing selection bias into historical data. CAPES demonstrates that this scoring mechanism can be repurposed for causal inference: scheduling bursts induce implicit cutoffs, enabling an RDD-style quasi-experiment to identify the marginal power impact of VM launches. In our trace-driven evaluation, CAPES yields lower bias and better calibration than template-based ML baselines, particularly on server states rarely selected by current policies.

For future work, we plan to conduct more comprehensive power evaluations using diverse and realistic workloads beyond `stress-ng`, to test model generalization. In addition, we aim to compare CAPES against a broader range of ML baselines to rigorously benchmark its robustness against distribution shifts in production environments.

Acknowledgments

We thank Zijun Wang (HEC Paris) for econometric feedback, and Arush Mehrotra and Rohit Chawla for data collection. This work is supported by the National Science Foundation via the Expeditions in Computing program (2326606).

References

[1] Sebastian Calonico, Matias D. Cattaneo, and Rocio Titiunik. 2014. Robust Nonparametric Confidence Intervals for Regression-Discontinuity Designs. *Econometrica* 82, 6 (2014), 2295–2326. https://doi.org/10.3982/ECTA11757

[2] Matias D. Cattaneo, Nicolás Idrobo, and Rocío Titiunik. 2019. A Practical Introduction to Regression Discontinuity Designs: Foundations. *Elements in Quantitative and Computational Methods for the Social Sciences* (Nov. 2019). https://doi.org/10.1017/9781108684606 ISBN: 9781108684606 9781108710206.

[3] Matias D. Cattaneo, Michael Jansson, and Xinwei Ma. 2020. Simple Local Polynomial Density Estimators. *J. Amer. Statist. Assoc.* 115, 531 (July 2020), 1449–1455. https://doi.org/10.1080/01621459.2019.1635480 _eprint: https://doi.org/10.1080/01621459.2019.1635480.

[4] Matias D. Cattaneo, Luke Keele, Rocío Titiunik, and Gonzalo Vazquez-Bare. 2016. Interpreting Regression Discontinuity Designs with Multiple Cutoffs. *The Journal of Politics* 78, 4 (Oct. 2016), 1229–1248. https://doi.org/10.1086/686802

[5] Thomas D. Cook. 2008. "Waiting for Life to Arrive": A history of the regression-discontinuity design in Psychology, Statistics and Economics. *Journal of Econometrics* 142, 2 (Feb. 2008), 636–654. https://doi.org/10.1016/j.jeconom.2007.05.002

[6] Yao Fu, Leyang Xue, Yeqi Huang, Andrei-Octavian Brabete, Dmitrii Ustiugov, Yuvraj Patel, and Luo Mai. 2024. ServerlessLLM: Low-Latency Serverless Inference for Large Language Models. 135–153. https://www.usenix.org/conference/osdi24/presentation/fu

[7] Ori Hadary, Luke Marshall, Ishai Menache, Abhisek Pan, Esaias E. Greeff, David Dion, Star Dorminey, Shailesh Joshi, Yang Chen, Mark Russinovich, and Thomas Moscibroda. 2020. Protean: VM Allocation Service at Scale. 845–861. https://www.usenix.org/conference/osdi20/presentation/hadary

[8] Mark Hoekstra. 2009. The Effect of Attending the Flagship State University on Earnings: A Discontinuity-Based Approach. *The Review of Economics and Statistics* 91, 4 (Nov. 2009), 717–724. https://doi.org/10.1162/rest.91.4.717

[9] Guido Imbens and Karthik Kalyanaraman. 2012. Optimal Bandwidth Choice for the Regression Discontinuity Estimator. *The Review of Economic Studies* 79, 3 (July 2012), 933–959. https://doi.org/10.1093/restud/rdr043

[10] Guido W. Imbens and Thomas Lemieux. 2008. Regression discontinuity designs: A guide to practice. *Journal of Econometrics* 142, 2 (Feb. 2008), 615–635. https://doi.org/10.1016/j.jeconom.2007.05.001

[11] Aman Kansal, Feng Zhao, Jie Liu, Nupur Kothari, and Arka A. Bhattacharya. 2010. Virtual machine power metering and provisioning. In *Proceedings of the 1st ACM symposium on Cloud computing (SoCC '10)*. Association for Computing Machinery, New York, NY, USA, 39–50. https://doi.org/10.1145/1807128.1807136

[12] Bhavani Krishnan, Hrishikesh Amur, Ada Gavrilovska, and Karsten Schwan. 2011. VM power metering: feasibility and challenges. *SIGMETRICS Perform. Eval. Rev.* 38, 3 (Jan. 2011), 56–60. https://doi.org/10.1145/1925019.1925031

[13] Alok Gautam Kumbhare, Reza Azimi, Ioannis Manousakis, Anand Bonde, Felipe Frujeri, Nithish Mahalingam, Pulkit A. Misra, Seyyed Ahmad Javadi, Bianca Schroeder, Marcus Fontoura, and Ricardo Bianchini. 2021. {Prediction-Based} Power Oversubscription in Cloud Platforms. 473–487. https://www.usenix.org/conference/atc21/presentation/kumbhare

[14] David S. Lee and Thomas Lemieux. 2010. Regression Discontinuity Designs in Economics. *Journal of Economic Literature* 48, 2 (June 2010), 281–355. https://doi.org/10.1257/jel.48.2.281

[15] Justin McCrary. 2008. Manipulation of the running variable in the regression discontinuity design: A density test. *Journal of Econometrics* 142, 2 (Feb. 2008), 698–714. https://doi.org/10.1016/j.jeconom.2007.05.005

[16] Pratyush Patel, Esha Choukse, Chaojie Zhang, Íñigo Goiri, Brijesh Warrier, Nithish Mahalingam, and Ricardo Bianchini. 2024. Characterizing Power Management Opportunities for LLMs in the Cloud. In *Proceedings of the 29th ACM International Conference on Architectural Support for Programming Languages and Operating Systems, Volume 3 (ASPLOS '24, Vol. 3)*. Association for Computing Machinery, New York, NY, USA, 207–222. https://doi.org/10.1145/3620666.3651329

[17] You Peng, Youhe Jiang, Wenqi Jiang, Chen Wang, and Binhang Yuan. 2025. HEXGEN-FLOW: Optimizing LLM Inference Request Scheduling for Agentic Text-to-SQL. https://doi.org/10.48550/arXiv.2505.05286 arXiv:2505.05286 [cs].

[18] Varun Sakalkar, Vasileios Kontorinis, David Landhuis, Shaohong Li, Darren De Ronde, Thomas Blooming, Anand Ramesh, James Kennedy, Christopher Malone, Jimmy Clidaras, and Parthasarathy Ranganathan. 2020. Data Center Power Oversubscription with a Medium Voltage Power Plane and Priority-Aware Capping. In *Proceedings of the Twenty-Fifth International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS '20)*. Association for Computing Machinery, New York, NY, USA, 497–511. https://doi.org/10.1145/3373376.3378533

[19] Jovan Stojkovic, Chaojie Zhang, Íñigo Goiri, Esha Choukse, Haoran Qiu, Rodrigo Fonseca, Josep Torrellas, and Ricardo Bianchini. 2025. TAPAS: Thermal- and Power-Aware Scheduling for LLM Inference in Cloud Platforms. In *Proceedings of the 30th ACM International Conference on Architectural Support for Programming Languages and Operating Systems, Volume 2*. Association for Computing Machinery, New York, NY, USA, 1266–1281. https://doi.org/10.1145/3676641.3716025

[20] Biao Sun, Ziming Huang, Hanyu Zhao, Wencong Xiao, Xinyi Zhang, Yong Li, and Wei Lin. 2024. Llumnix: Dynamic Scheduling for Large Language Model Serving. 173–191. https://www.usenix.org/conference/osdi24/presentation/sun-biao

[21] Donald L. Thistlethwaite and Donald T. Campbell. 1960. Regression-discontinuity analysis: An alternative to the ex post facto experiment. *Journal of Educational Psychology* 51, 6 (1960), 309–317. https://doi.org/10.1037/h0044319 Place: US.

[22] Muhammad Tirmazi, Adam Barker, Nan Deng, Md E. Haque, Zhijing Gene Qin, Steven Hand, Mor Harchol-Balter, and John Wilkes. 2020. Borg: the next generation. In *Proceedings of the Fifteenth European Conference on Computer Systems (EuroSys '20)*. Association for Computing Machinery, New York, NY, USA, 1–14. https://doi.org/10.1145/3342195.3387517

[23] Abhishek Verma, Luis Pedrosa, Madhukar Korupolu, David Oppenheimer, Eric Tune, and John Wilkes. 2015. Large-scale cluster management at Google with Borg. In *Proceedings of the Tenth European Conference on Computer Systems (EuroSys '15)*. Association for Computing Machinery, New York, NY, USA, 1–17. https://doi.org/10.1145/2741948.2741964

[24] Stefan Wager and Susan Athey. 2018. Estimation and Inference of Heterogeneous Treatment Effects using Random Forests. *J. Amer. Statist. Assoc.* 113, 523 (July 2018), 1228–1242. https://doi.org/10.1080/01621459.2017.1319839