Intelligent Overclocking for Improved Cloud Efficiency

Aditya Soni
Microsoft
India
t-asoni@microsoft.com

Mayukh Das
Microsoft
India
mayukhdas@microsoft.com

Pulkit Misra
Microsoft
USA
pumisra@microsoft.com

Chetan Bansal
Microsoft
USA
chetanb@microsoft.com

ABSTRACT

Overclocking computing nodes in datacenters is emerging as an area of great interest since it presents an opportunity to temporarily boost performance and handle transient demand surges more cost-effectively for cloud providers. However, indiscriminate overclocking increases risks like power capping events that negate performance gains, accelerated component wear, and power consumption spikes. This work proposes an intelligent overclocking orchestration framework that employs a holistic system continuously monitoring workload characteristics, resource utilization, and power telemetry across the infrastructure. It leverages advanced modeling techniques to anticipate demand and optimize overclocking decisions based on performance benefits and mitigation of associated risks through adaptive policies. Preliminary evaluations on production cloud workload traces demonstrate the efficacy of the intelligent overclocking system in reducing power capping incidents by up to 100 times compared to naive approaches. Furthermore, the work outlines ongoing research directions, including the application of reinforcement learning techniques to derive globally optimal overclocking policies while incorporating fairness and reliability objectives, and seamless extensibility to emerging accelerator technologies.

KEYWORDS

Resource scheduling and optimization, predictive capacity management, resource allocation and packing

ACM Reference Format:
Aditya Soni, Mayukh Das, Pulkit Misra, and Chetan Bansal. 2024. Intelligent Overclocking for Improved Cloud Efficiency. In *Proceedings of 5th International Workshop on Cloud Intelligence / AIOps (AIOps '24). Co-located with ASPLOS '24 (AIOps '24 workshop @ ASPLOS).* ACM, New York, NY, USA, 6 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

AIOps '24 workshop @ ASPLOS, April 27, 2024, San Diego, CA

ACM ISBN 978-x-xxxx-xxxx-x/YY/MM
https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 INTRODUCTION

Cloud services allocate sufficient resources to meet their peak performance demands [8, 11, 24], such as maintaining tail latency within predetermined Service-Level Objectives (SLOs) [7]. Although this approach provides the best performance, it incurs high operating costs as resources are always provisioned for peak, even during periods of low activity. While dynamic resource allocation techniques like autoscaling [3, 19] and serverless computing [2, 18] can reduce costs, they may increase application latency (e.g., due to long VM creation times [1]) or are unsuitable for stateful services [13].

On the other hand, technological advancements have enabled overclocking i.e., operating components (e.g., CPU) beyond typical voltage and power limits [14]. Overclocking boosts workload performance and allows cost-effective handling of transient demand surges. Prior work shows that overclocking can significantly reduce (e.g., by 54%) tail latency for cloud workloads compared to conventional autoscaling, while also lowering operating costs [14].

However, overclocking has several challenges that cloud providers need to carefully manage. First, indiscriminate use of overclocking can increase power draw and trigger frequent power capping events that require reducing component frequency (e.g., via Intel RAPL [6]) for controlling power, thereby negating any performance gains. Second, overclocking can reduce component lifetime by accelerating wearout and, thus, cannot be performed indefinitely. The limited amount of overclocking needs to be used carefully. For example, overclocking the CPU of a memory-bound workload or during non-peak periods does not provide much benefit. Finally, workload SLOs need to be protected when overclocking is unavailable. For example, a workload might have under-provisioned due to reliance on overclocking, but it would miss its SLOs under peak load if its VMs cannot be overclocked due to lack of power. Fairness is also an important criterion in constrained scenarios and policies that determine which workload can overclock need to avoid starvation. Therefore, overclocking introduces complexity in the management and orchestration of cloud infrastructure, necessitating specialized monitoring, control, and failover mechanisms.

We approach the problem of intelligent overclocking in the cloud by designing a holistic system that continuously monitors workload characteristics, resource utilization, power/thermal profiles, and performance metrics across the infrastructure. We employ advanced modeling and prediction techniques to estimate potential performance gains from overclocking diverse workloads. Our approach incorporates multi-objective optimization algorithms to judiciously

allow overclocking by balancing the performance benefits against specified constraints like power draw, component reliability, and fairness. Our system makes dynamic overclocking decisions in real-time based on the current cluster state, anticipated demand, and specified policies for performance targets and overclocking limits. Under constraints (e.g., power), it creates an overclocking schedule that maximizes a pre-defined cluster-level utility, which can be defined as a function of workload characteristics, such as how much a workload benefits from overclocking and its deployment size (# of cores). This utility is used to create a schedule for which workloads can be overclocked in any time period. For any workload(s) that cannot be overclocked, the system protects the workload's SLOs through predictive autoscaling or resource reallocation.

This paper presents our ongoing work on designing such an intelligent overclocking framework that can adapt to a spectrum of constraints while optimizing for performance objectives. At this stage, we focus on managing power while overclocking, where workloads specify time periods for when they need overclocking. We investigate methods that take decisions with local optimization objectives by defining utility functions for workloads, enabling a greedy decision maker to prioritize overclocking requests judiciously. Extensive evaluations on real-world cluster traces demonstrate the efficacy of the intelligent overclocking system in reducing power capping incidents by up to 100 times compared to naive approaches. The results also highlight the trade-offs between different overclocking protocols and their impact on autoscaling strategies. We are currently exploring approaches that employ a longer decision horizon to learn global optimization policies. One such technique involves offline reinforcement learning (RL), which has the advantage of learning a policy for the cluster from the trajectory of any arbitrary overclocking policy acting on it. Deep RL takes away the complexity of defining fine-grained utility functions and can learn to estimate the utility from a set of features. Thus, RL provides an end-to-end solution for learning policies, and we aim to explore this in our future work.

2 RELATED WORK

Extensive prior research has explored computational sprinting techniques like turbo-boosting CPU frequencies for short periods of time [9, 20–22], while using ideas from game theory [9], formal control [21], and performance modeling [20] to manage sprinting. However, these works primarily focus on single-server setups, assume transparent knowledge of applications, or overlook multi-tenancy considerations on servers or racks and, thus, do not holistically address the overclocking challenges in cloud environments.

Another line of research has focused on datacenter power management [10, 17, 23, 26], proposing oversubscription techniques that leverage statistical properties of concurrent power usage across servers to improve datacenter power utilization and reduce costs. While complementary, these works influence the non-overclocked baseline and factor in power demand from turbo-boosting to meet performance Service-Level Agreements (SLAs). However, naively adding overclocking to the existing power utilization increases the probability of power capping events, necessitating a more intelligent approach to leverage unutilized power while ensuring workload SLOs when overclocking is unavailable.

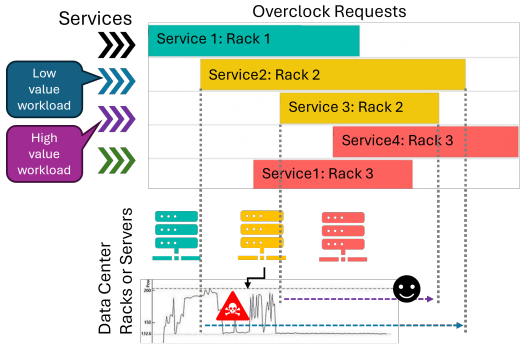

Figure 1: Complex overclock scheduling problem

Furthermore, researchers have explored workload intelligence to optimize performance, energy consumption, and cost [4, 5, 15, 27, 28]. Techniques like machine learning models [27], application-specific feature analysis [4, 15, 28], and resource telemetry gathering [5] have been employed to allocate resources efficiently, predict optimal frequencies, and provide insights for resource management. While valuable, these approaches do not directly address the unique challenges introduced by overclocking in cloud environments, such as managing reliability impacts, power constraints, and workload SLO protection when overclocking is unavailable.

3 SYSTEM OVERVIEW

Overclocking can lead to high capacity efficiency while ensuring minimum power capping only with intelligent orchestration and scheduling policies. As highlighted in prior work [14], cores cannot be overclocked beyond a vendor recommended duration owing to hardware degradation. More importantly, aggressive overclocking at any arbitrary point in time is not feasible since extra power draw due to the overclocked cores may cross the power budget (power capping) of a server or server rack. This will trigger corrective actions by the data center leading to performance impact to critical workloads. Conservative overclocking policy on the other hand will possibly prevent such power capping incidents, but will also prevent us from getting maximum possible benefit.

3.1 Problem Setting

Intelligent overclocking policy design is a complex multi-objective scheduling/optimization problem with various constraints. The complexity of the problem comes from 3 dimensions - (1) It is a stochastic dynamic process, and the load and power usage patterns change over time dependent on both the workload's demand patterns as well as VM allocation on various servers and racks. So overclocking policy of whether to overclock at some time point is subject to uncertain estimates of available power in future, (2) Since it is practically impossible to overclock for all possible resources and services it is paramount that the policy optimized some benefit or value. (3) As illustrated in Figure 1, services have different overclocking requirements, wherein these requests differ based on the start time and duration of overclocking, and how their VMs are distributed across different physical resources such as servers and racks which have their own power budgets, thus requiring the

policy to anticipate future requests - how they would be distributed across time and physical resources - and take them into consideration when making a decision. Orthogonally, we need to factor in the value of the workload to determine the most valuable action among the set of possible actions that does not violate the constraints. Thus, this is akin to a complex combinatorial scheduling problem.

We define a utility/benefit of overclocking as the potential savings when increased demand is handled without overclocking. Consider a particular vCore of a VM is overclocked. The optimization problem is defined as

$$\arg\max_{\mathbf{O}} U \quad \text{s.t.} \sum_{r \in \mathcal{R}} PowerCapping_r = 0 \tag{1}$$

where U is the utility gained from overclocking, $\mathbf{O}$ stands for the set of possible overclocking actions, $PowerCapping_r$ represents a power capping event on a rack r and $\mathcal{R}$ is the set of all racks.

Utility can be formulated as a composite function comprising multiple service characteristics and metrics. For instance, quantifying the number of cores utilized by a workload can serve as an indicator of its dominance or resource footprint. Additionally, the utility function may encapsulate a measure of priority based on the client or workload type, such as attributing higher utility to high-availability user-facing services. The selection and combination of relevant metrics to construct the utility function itself presents an area for further investigation and optimization.

This equation takes into consideration the constraint of being within the power availability limit. Assessing the power availability for the optimization process requires computing a stochastic estimate of the power consumption in the future. This involves predicting changes in power consumption with time due to workloads' power draw patterns and also accounting for the excess power usage due to overclocking requests granted earlier.

In the event that overclocking fails, there needs to be an alternate strategy for workload owners to protect their SLOs. One possible solution is to perform a lookahead while making decisions to anticipate the future state of the cluster, which would give workload owners a heads-up if they need to provision more resources.

This makes this a complex optimization problem which does not have a closed form solution. Hence, we propose a multi-component overclocking orchestration system.

3.2 Solution Architecture Overview

An intelligent overclocking system should be designed as a holistic platform that continuously monitors workload characteristics, resource utilization, power/thermal profiles, and performance metrics across the infrastructure. Leveraging this telemetry, it should employ advanced modeling and prediction techniques to accurately anticipate demand and estimate potential performance gains from overclocking diverse workloads. The system must then incorporate optimization algorithms to maximize the long-term benefit gained from overclocking over an infinite horizon, by maximizing said utility under any constraints present at multiple granularities in the environment. Figure 2 shows the architecture of such a system.

The power estimation module feeds on the racks' telemetry history, such as their power consumption patterns, the workloads they

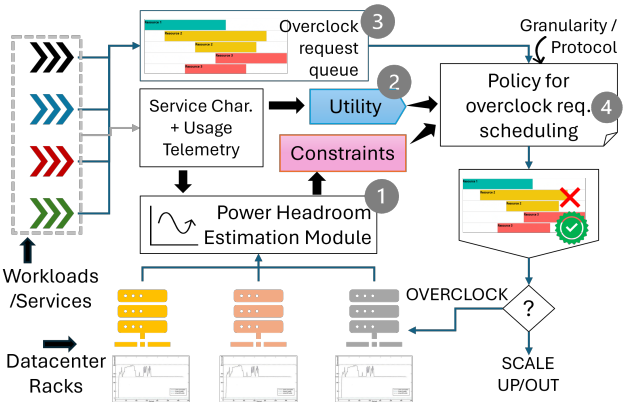

Figure 2: Overclocking orchestration system architecture

are hosting, and the workload usage telemetry. It can then estimate the power using a time-series forecasting or sequential decision making-based approach. The utility function operates on the services' telemetry, that can include the workloads' own telemetry from the clients, or meta-telemetry which indicates the CPU usage, the number of cores, VMs and other service characteristics. The policy for request scheduling is chosen from a library of policies, which then suggests actions for the incoming overclocking request queue based on the specified constraints and utilities.

4 METHODOLOGY

Building upon the high-level design principles outlined in the system overview, our methodology focuses on the key components and algorithms that underpin the intelligent overclocking framework. Central to our approach is a dynamic request handling mechanism that continuously processes incoming overclocking requests instead of assuming a priori knowledge of services' peak load periods. The system computes the utility for each request as a function of the workload's dominance, quantified by the number of cores hosting the workload across all racks in the infrastructure.

Power Estimation: Power estimation can be done through various techniques. For our environment, we employ a regression-based forecasting algorithm that is based on the linear regression model of the Darts library [12]. However, our system is flexible to incorporate alternative forecasters, such as Facebook's Prophet model, which can handle non-stationary environments. Furthermore, we are exploring sequential decision-making based methods that can provide power estimates by considering the environment holistically.

Partial Overclocking: Besides implementing different power estimating methods, we also implement different overclocking protocols. Initially, we make the assumption that workload owners would autoscale the entire service if even one rack lacks the power headroom to overclock some of its VMs. Subsequently, we explore partial overclocking, wherein we assume workload owners would modify their autoscaling policies to only scale up/out VMs on racks where overclocking is not possible, and would overclock their VMs on racks where it is feasible.

Overclocking Policy Library: Our system incorporates a library of overclocking policies to cater to the diverse requirements and constraints that cloud providers may encounter. Our current policy set aims to approximate the globally optimal solution through locally optimal solutions obtained via greedy strategies that prioritize granting requests with the highest utility, subject to power constraints.

Baseline Power Estimator: As a baseline, we implement a power estimator that does not forecast future power consumption when making overclocking decisions. When evaluating a request, it calculates the potential power increase in the involved racks and grants the request if sufficient power headroom is available in all the racks at that instant. This estimator does not account for the overclocking duration or potential changes in rack power consumption during this period.

Forecasting-based Power Estimators: In contrast, forecasting-based estimators provide more conservative estimates of the power headroom by predicting the respective rack's power consumption for a certain duration from the intended overclocking start time. The power headroom of a rack is taken as the minimum power headroom in the forecast duration on that rack. We further adapt these estimator-based policies for scenarios where overclocking decisions must be made before the request's intended start time, allowing workloads additional time to scale up/out and protect their Service-Level Objectives (SLOs) when overclocking is unavailable.

The forecast length is judiciously chosen as a high percentile value of the observed intervals between the start times of ongoing overclocking requests and occurrences of power capping events. This approach maximizes the likelihood of identifying requests that may lead to power capping events while minimizing forecasting errors and overhead.

Reinforcement Learning for Global Optimization: We are currently exploring reinforcement learning (RL) based methods to learn globally optimal policies for the overclocking environment. Implicit Q-Learning [16], an offline RL algorithm, can learn a policy from trajectories collected by arbitrary policies. This presents an end-to-end optimal policy learning approach wherein we run arbitrary overclocking policies on a cluster and use the collected trajectories to learn a globally optimal solution.

5 EVALUATION

To comprehensively evaluate the proposed intelligent overclocking orchestration system, we developed a discrete event simulator that interacts with real-world cluster traces. The simulator incorporates a power model that, given the CPU utilization and core frequency, estimates the power impact of overclocking on nodes equipped with Intel or AMD processors. Overclocking requests from various services are marked in the trace files at their observed peak load hours, representing realistic scenarios.

5.1 Experimental Setup

We use production traces from multiple datacenters exhibiting low, medium, and high power consumption patterns for evaluation. These datacenters are composed of hundreds of racks and a few thousand servers with either Intel or AMD CPUs. Workloads running in these datacenters are composed of multiple VMs that can be located on the same or different servers and racks. The traces include rack and server power, and VM-level CPU utilization. All data is collected for 6 weeks (April 10th - May 12th, 2023), at a 5-minute granularity. Stojkovic et al. [25] provides a more extensive outline and analysis of such traces and their characterizations.

In brief however, while clusters with moderate power consumption are our major focus, since they are expected to fully leverage the overclocking capabilities, we also analyze extreme cases (high and low power consumption) to ensure a comprehensive assessment across different operating regimes.

As a baseline, we implement a naive policy that grants all overclocking requests without estimating future power consumption. This approach serves as a worst-case reference, highlighting the potential consequences of overclocking without intelligent orchestration, such as an increased number of power capping events.

Our analysis showed that the 90th percentile interval between an overclocking request start and a subsequent power capping event was approximately 60 minutes. Consequently, we set the forecast time horizon to 60 minutes for the power estimation component. The power forecaster had an average RMSE of 97 kW across the traces, where the rack power to be forecasted was around 10,000 - 13,000 kW.

As a follow up, to account for the time required by workload owners to scale out their services in case of overclocking unavailability, we also set a lookahead time of 30 minutes for the corresponding evaluations. This essentially means that overclocking decisions are taken while keeping a lookahead buffer time, instead of looking at requests right when they are supposed to begin.

5.2 Evaluation Metrics

Policies are primarily compared against each other based on the following metrics:

- *Requests Granted.* The number of overclocking requests successfully approved by the policy, further categorized into full and partial grants.
- *Cumulative Cores Granted.* The total number of CPU cores approved for overclocking.
- *Cumulative Cores Overclocked.* The number of CPU cores that were effectively overclocked disruption due to power capping events.
- *Unique Cores Granted.* The count of distinct CPU cores granted overclocking across all requests, indicating the policy's fairness in resource allocation.
- *Number of Power Capping Events.* The instances where total power consumption exceeded the available budget, necessitating corrective actions.

5.3 Partial Overclocking

Our evaluation explores the impact of partial overclocking, a protocol that allows services to overclock their virtual machines (VMs) on racks with sufficient power headroom while scaling out on other racks. This approach significantly outperforms previous policies in terms of the number of cores overclocked, as services are no longer bottlenecked by a marginal number of VMs running on high-power racks. However, partial overclocking requires workload owners to

Overclocking Policy	Full Requests Granted			Partial Requests Granted			Cum. Cores Granted			Cum. Cores Overclocked			Power Cap Events		
	Low	*Med*	*High*	*Low*	*Med*	*High*	*Low*	*Med*	*High*	*Low*	*Med*	*High*	*Low*	*Med*	*High*
Grant All	100.0	100.0	100.0	-	-	-	100.0	100.0	100.0	100.0	66.7	29.8	0.0	37.0x	105.2x
No Forecast	100.0	93.2	70.0	-	-	-	100.0	32.9	31.5	100.0	32.8	31.3	0.0	1.0x	1.0x
Forecast	100.0	94.2	69.7	-	-	-	100.0	32.4	28.1	100.0	32.3	28.0	0.0	0.0x	1.0x
Lookahead	100.0	95.2	59.5	-	-	-	100.0	38.2	25.9	100.0	37.9	25.6	0.0	2.0x	1.2x
Partial	100.0	91.3	55.3	0.0	8.7	34.2	100.0	72.9	63.0	100.0	71.0	59.1	0.0	9.0x	6.5x
IQL	98.0	11.1	8.6	2.0	33.9	57.8	99.9	42.5	27.0	99.9	30.2	21.6	0.0	14.0x	11.0x

Table 1: System evaluation results with different overclocking policies on clusters with low, medium and high power consumption. Grant All is a naive baseline. Lookahead evaluations have a forecast of the same duration as in the Forecast method. Partial also builds upon Forecast for its underlying power estimation. IQL is the implicit Q-learning implementation. Number of power capping events are normalized within the cluster.

adapt their autoscaling policies accordingly, as traditional strategies relying solely on overall service metrics may not be effective.

The results for the medium power cluster in Table 1 highlight a scenario where a large service makes substantial overclocking requests that cannot be concurrently granted across all the racks it is deployed on. The partial overclocking protocol effectively addresses this issue. Nevertheless, partial overclocking also leads to a slight increase in the number of power capping events. This is because the errors in the power forecaster become more pronounced when overclocking decisions are made based on individual rack forecasts, as opposed to the previous method that considered forecasts across all racks simultaneously.

5.4 Reinforcement Learning Approach

To further improve the overclocking decision-making process, we explore reinforcement learning (RL) techniques to derive policies that can model the environment's dynamics without relying on a forecasting algorithm. Specifically, we employ Implicit Q-Learning (IQL) [16], an offline RL algorithm, for our initial experiments. IQL learns a sequential decision-making model from cluster trajectories collected by arbitrary overclocking policies.

Our implementation extends the partial overclocking approach by not prioritizing complete request grants before partial grants, as greedy partial overclocking does. We do not define a utility function here since we implement a deep RL approach that has the potential to learn a refined utility function from the cluster state based off the high-level feedback it receives during training.

While IQL shows results comparable to other approaches on low-power clusters, it struggles to learn that there is sufficient power headroom to grant all requests, indicating the need for improved power estimation. Furthermore, IQL performs worse on medium and high-power clusters compared to other policies, possibly due to the more complex environment dynamics present in these clusters and the naive feedback it receives during training, which requires further refinement.

Regardless, the reported results indicate that even a naive RL-based implementation, without any modifications made to the observations to handle the heterogeneous duration of the requests, and that is learning on sparse feedback from the environment that could be augmented with more information, performs reasonably well. Making these adjustments to allow the agent to better observe the environment would lead to much better performing policies. More importantly, such an approach would be the more principled formulation for a stochastic, possibly non-stationary, time-sensitive overclocking request scheduling problem such as ours.

5.5 Fairness Analysis

Fairness of an intelligent decision-making system can be viewed in various ways, such as analysing the starvation rates of workload requests and their importance. To measure the fairness of our policies, we track the number of unique cores granted overclocking, identifying if any workloads are being starved. Table 2 shows the percentage of unique cores requesting overclocking that were granted a request. Even with a partial overclocking approach, we observe that some cores are being starved, indicating the need to accommodate fairness as an additional constraint in our policies. As an immediate next step, we will work on incorporating other fairness metrics in our evaluation.

Overclocking Policy	Unique Cores Granted (%)		
	Low	*Med*	*High*
Grant All	100	100	100
No Forecast	100	32.3	31.5
Forecast	100	33.9	28.1
Lookahead	100	36.8	25.9
Partial	100	77.5	63.0
IQL	99.95	49.4	27.0

Table 2: Evaluation of policy fairness, as a measure of the unique cores granted overclocking. Grant All is a naive baseline. Lookahead evaluations have a forecast of the same duration as in the Forecast method. Partial also builds upon Forecast for its underlying power estimation. IQL is the implicit Q-learning implementation.

5.6 Summary

Our evaluation demonstrates the efficacy of the proposed intelligent overclocking orchestration system in maximizing the benefits of overclocking while adhering to power constraints. The exploration of different policies and protocols lays the foundation for further optimizations and enhancements, as discussed in the future work section.

6 FUTURE WORK

For future work, we plan to explore reinforcement learning techniques for deriving overclocking policies. Reinforcement learning methods have the capability to learn globally optimal policies by maximizing utility over an infinite horizon. They offer the advantage of an end-to-end approach, learning from trajectories collected under arbitrary overclocking policies, and deep RL methods can circumvent the complexity of explicitly defining utility functions by estimating utility directly from workload telemetry data.

A key focus will be incorporating fairness and reliability as essential objectives in the overclocking decision-making process. Fairness constraints are crucial to prevent scenarios where smaller workloads are starved of overclocking opportunities due to prioritization based solely on resource dominance. Reliability considerations, on the other hand, will enable the system to handle scenarios where overclocking fails or is unavailable. In such cases, the system should incorporate alternative strategies, such as predictive autoscaling, to protect workload Service-Level Objectives (SLOs). Robust failover mechanisms are imperative for ensuring performance guarantees under all circumstances.

Furthermore, we intend to design our overclocking framework generic and extensible, capable of adapting to other overclockable components beyond CPUs, such as GPUs.

REFERENCES

[1] Samiha Islam Abrita, Moumita Sarker, Faheem Abrar, and Muhammad Abdullah Adnan. 2019. Benchmarking VM Startup Time in the Cloud. In *Benchmarking, Measuring, and Optimizing*, Chen Zheng and Jianfeng Zhan (Eds.).

[2] Amazon AWS. [n. d.]. AWS Lambda. https://aws.amazon.com/lambda/.

[3] Amazon AWS. 2023. AWS Auto Scaling. https://aws.amazon.com/autoscaling/.

[4] Shuang Chen, Angela Jin, Christina Delimitrou, and José F. Martínez. 2022. ReTail: Opting for Learning Simplicity to Enable QoS-Aware Power Management in the Cloud. In *HPCA*.

[5] Eli Cortez, Anand Bonde, Alexandre Muzio, Mark Russinovich, Marcus Fontoura, and Ricardo Bianchini. 2017. Resource Central: Understanding and Predicting Workloads for Improved Resource Management in Large Cloud Platforms. In *SOSP*.

[6] Howard David, Eugene Gorbatov, Ulf R. Hanebutte, Rahul Khanna, and Christian Le. 2010. RAPL: Memory power estimation and capping. In *ISLPED*.

[7] Jeffrey Dean and Luiz André Barroso. 2013. The Tail at Scale. *Commun. ACM* 56 (2013), 74–80.

[8] Christina Delimitrou and Christos Kozyrakis. 2014. Quasar: Resource-Efficient and QoS-Aware Cluster Management. In *ASPLOS*.

[9] Songchun Fan, Seyed Majid Zahedi, and Benjamin C. Lee. 2016. The Computational Sprinting Game. In *ASPLOS*.

[10] Sriram Govindan, Jeonghwan Choi, Bhuvan Urgaonkar, Anand Sivasubramaniam, and Andrea Baldini. 2009. Statistical profiling-based techniques for effective power provisioning in data centers. *Proc. EuroSys Conference (EuroSys '09)*.

[11] Jing Guo, Zihao Chang, Sa Wang, Haiyang Ding, Yihui Feng, Liang Mao, and Yungang Bao. 2019. Who Limits the Resource Efficiency of My Datacenter: An Analysis of Alibaba Datacenter Traces. In *IWQoS*.

[12] Julien Herzen, Francesco LÃ¤ssig, Samuele Giuliano Piazzetta, Thomas Neuer, LÃ©o Tafti, Guillaume Raille, Tomas Van Pottelbergh, Marek Pasieka, Andrzej Skrodzki, Nicolas Huguenin, Maxime Dumonal, Jan KoÅ›cisz, Dennis Bader, FrÃ©dÃ©rick Gusset, Mounir Benheddi, Camila Williamson, Michal Kosinski, Matej Petrik, and GaÃ«l Grosch. 2022. Darts: User-Friendly Modern Machine Learning for Time Series. *Journal of Machine Learning Research* 23, 124 (2022), 1–6. http://jmlr.org/papers/v23/21-1177.html

[13] IBM Cloud. [n. d.]. "Scaling stateful and stateless services". https://www.ibm.com/docs/en/cloud-app-management/2019.3.0?topic=sizing-scaling-stateless-stateful-services.

[14] Majid Jalili, Ioannis Manousakis, Inigo Goiri, Pulkit A. Misra, Ashish Raniwala, Husam Alissa, Bharath Ramakrishnan, Phillip Tuma, Christian Belady, Marcus Fontoura, and Ricardo Bianchini. 2021. Cost-Efficient Overclocking in Immersion-Cooled Datacenters. In *ISCA*.

[15] Harshad Kasture, Davide B. Bartolini, Nathan Beckmann, and Daniel Sanchez. 2015. Rubik: Fast analytical power management for latency-critical systems. In *MICRO*.

[16] Ilya Kostrikov, Ashvin Nair, and Sergey Levine. 2021. Offline Reinforcement Learning with Implicit Q-Learning. arXiv:2110.06169 [cs.LG]

[17] 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. In *USENIX ATC*.

[18] Microsoft Azure. [n. d.]. Microsoft Azure Functions. https://azure.microsoft.com/en-gb/services/functions/.

[19] Microsoft Azure. 2023. Overview of autoscale in Azure. https://learn.microsoft.com/en-us/azure/azure-monitor/autoscale/autoscale-overview.

[20] Nathaniel Morris, Christopher Stewart, Lydia Chen, Robert Birke, and Jaimie Kelley. 2018. Model-Driven Computational Sprinting. In *EuroSys*.

[21] Raghavendra Pradyumna Pothukuchi, Joseph L. Greathouse, Karthik Rao, Christopher Erb, Leonardo Piga, Petros G. Voulgaris, and Josep Torrellas. 2019. Tangram: Integrated Control of Heterogeneous Computers. In *MICRO*.

[22] Arun Raghavan, Yixin Luo, Anuj Chandawalla, Marios Papaefthymiou, Kevin P. Pipe, Thomas F. Wenisch, and Milo M. K. Martin. 2012. Computational Sprinting. In *HPCA*.

[23] P. Ranganathan, P. Leech, D. Irwin, and J. Chase. 2006. Ensemble-level Power Management for Dense Blade Servers. In *ISCA*.

[24] Charles Reiss, Alexey Tumanov, Gregory R. Ganger, Randy H. Katz, and Michael A. Kozuch. 2012. Heterogeneity and Dynamicity of Clouds at Scale: Google Trace Analysis. In *SoCC*.

[25] Jovan Stojkovic, Pulkit Misra, Íñigo Goiri, Sam Whitlock, Esha Choukse, Mayukh Das, Chetan Bansal, Jason Lee, Zoey Sun, Haoran Qiu, Reed Zimmermann, Savyasachi Samal, Brijesh Warrier, Ashish Raniwala, and Ricardo Bianchini. 2024. SmartOClock: Workload- and Risk-Aware Overclocking in the Cloud. In *ISCA*. https://www.microsoft.com/en-us/research/publication/smartoclock-workload-and-risk-aware-overclocking-in-the-cloud/

[26] Qiang Wu, Qingyuan Deng, Lakshmi Ganesh, Chang-Hong Hsu, Yun Jin, Sanjeev Kumar, Bin Li, Justin Meza, and Yee Jiun Song. 2016. Dynamo: Facebook's Data Center-Wide Power Management System. In *ISCA*.

[27] Yanqi Zhang, Weizhe Hua, Zhuangzhuang Zhou, G. Edward Suh, and Christina Delimitrou. 2021. Sinan: ML-Based and QoS-Aware Resource Management for Cloud Microservices. In *ASPLOS*.

[28] Liang Zhou, Laxmi N. Bhuyan, and K. K. Ramakrishnan. 2020. Gemini: Learning to Manage CPU Power for Latency-Critical Search Engines. In *MICRO*.