Journal of Computer Science and Electrical Engineering
Print ISSN: 2663-1938
Online ISSN: 2663-1946
DOI: https://doi.org/10.61784/jcsee4155

PRISM: PREDICTION-GUIDED RISK-AWARE DEEP REINFORCEMENT LEARNING FOR ENERGY-AWARE DYNAMIC RESOURCE SCHEDULING IN DISTRIBUTED CLOUD DATA CENTERS

EnMiao Feng[1*], PengFei Li[1], TianJun Mo[1], GaiKe Wang[2]
[1]*Duke University, Durham, NC, USA.*
[2]*New York University, New York, NY, USA.*
[*]*Corresponding Author: EnMiao Feng*

Abstract: Geo-distributed cloud data centers spend a large share of their electricity on lightly loaded servers, while aggressive consolidation causes overloads, service-level agreement (SLA) violations and migration storms. This paper presents PRISM, a prediction-guided, risk-aware deep reinforcement learning (DRL) framework for energy-aware dynamic resource scheduling in distributed cloud data centers. PRISM couples three components. A lightweight multi-horizon quantile forecaster (MQF) predicts the mean, median and 90th percentile of the CPU demand of every virtual machine (VM), including newly arrived VMs without history. A risk-aware host model aggregates these quantiles into a chance-constrained capacity test that drives overload detection and placement feasibility at a tunable risk level. A residual Q-network, whose action value is an analytic energy-cost prior plus a learned correction, scores a compact set of candidate hosts for arrival dispatch, overload relief and consolidation across all data centers and is trained with Double DQN. Trace-driven experiments with 11,746 real PlanetLab VM traces on three heterogeneous data centers with SPECpower server models, time-of-use tariffs and different PUE values show that, on unseen test days and relative to the strongest baseline, PRISM reduces energy consumption by 12.5%, electricity cost by 5.2%, carbon emissions by 14.1%, the SLA violation time per active host by 44.6%, the energy–SLA-violation product by 90.1% and the number of migrations by 84.6%, while requiring 16.3 ms of computation per five-minute scheduling interval.
Keywords: Cloud computing; Energy efficiency; Virtual machine consolidation; Deep reinforcement learning; Quantile forecasting; Geo-distributed data centers

1 INTRODUCTION

Data centers account for around 1% of the world's electricity consumption [1], and much of this energy is spent on servers that run at low utilization. Because server power is far from proportional to load [2], an idle server still draws most of its peak power: the HP ProLiant servers considered in this paper consume 69–74% of their peak power when idle. Dynamic virtual machine (VM) consolidation reduces this waste by packing VMs onto fewer hosts through live migration and switching the released hosts to a low-power state [3-6]. In geographically distributed clouds the problem acquires another dimension: electricity prices, power usage effectiveness (PUE) and carbon intensity differ between sites and vary over the day, so the location of a workload matters as much as the density of its packing [7-9].

Three difficulties make energy-aware dynamic scheduling hard in practice. First, VM demand is volatile and heavy-tailed. Threshold- and regression-based consolidation heuristics react to the most recent utilization and therefore trigger thousands of migrations per day [4], each of which degrades the migrated application. Prediction-based schemes anticipate the load [10-12], but they rely on point forecasts that convey no information about the risk of a burst. Second, deep reinforcement learning (DRL) has been applied to placement and consolidation [13-18], yet learning a policy over hundreds of hosts from scratch is sample-inefficient, and early policies can be considerably worse than simple heuristics, which an operator cannot accept. Third, consolidation within a data center and load distribution across data centers are usually designed separately, which forgoes the benefit of pooling delay-tolerant workloads on the most efficient and cheapest servers of the whole system.

This paper addresses these difficulties with PRISM (Prediction-guided RIsk-aware Scheduling for Multi-data-center clouds). Its main contributions are as follows.

1) A multi-horizon quantile forecaster (MQF) that predicts the mean, median and 90th percentile of the CPU demand of each VM for the next three intervals, handles cold-start VMs through masked inputs, and attains the lowest error among five predictors on unseen days with well-calibrated upper quantiles.

2) A risk-aware host model that aggregates per-VM quantiles into a host-level chance constraint. The same constraint governs overload detection and placement feasibility, and its risk level can be changed at run time without retraining.

3) A residual Q-network that adds a learned correction to an analytic energy-cost prior and scores a bounded set of candidate hosts in all data centers. Because the residual is initialized to zero, the untrained agent coincides with a strong expert policy, and Double DQN training only has to learn improvements. One network serves arrival dispatch, overload relief and consolidation.

© By the Author(s) 2022, under the CC BY-NC-ND license (http://creativecommons.org/licenses/by-nc-nd/4.0).

4) A trace-driven evaluation with 11,746 real PlanetLab VM traces on three heterogeneous data centers against seven baselines, including a reactive DRL scheduler, complemented by ablation and sensitivity studies.
The remainder of the paper is organized as follows. Section 2 reviews related work, Section 3 presents the system model, Section 4 describes PRISM, Section 5 reports the experimental results, and Section 6 concludes the paper.

2 RELATED WORK

2.1 Heuristic and Prediction-Based Consolidation

Beloglazov *et al.* introduced the power-aware best-fit decreasing (PABFD) placement and the minimum migration time (MMT) selection policy [5], and Beloglazov and Buyya proposed adaptive overload detectors based on the median absolute deviation (MAD), the interquartile range (IQR) and local regression (LR) [4], together with the SLATAH, PDM and ESV metrics that have become the de facto standard of the CloudSim PlanetLab benchmark [19-20]. pMapper considered power and migration cost jointly [6]. Farahnakian *et al.* used regression to predict the future utilization of VMs and hosts and thereby reduced migrations and SLA violations [10]. ARIMA models [11], neural networks trained by differential evolution [12], the production-scale predictors of Resource Central and the ML-centric forecasting framework of Khan *et al.* further improved prediction accuracy [21-22]. These approaches use point forecasts, whereas PRISM predicts quantiles and converts them into an explicit, tunable overload risk [23].

2.2 Reinforcement Learning for Cloud Resource Management

Farahnakian *et al.* used tabular Q-learning to decide when hosts should switch between active and sleep modes [24]. DeepRM demonstrated that DRL can learn multi-resource packing policies [13], and Liu *et al.* combined a DRL-based global tier with an LSTM-based local power manager [14]. DRL-Cloud minimized the energy cost of cloud providers under realistic electricity prices with deep Q-learning [15], Wang *et al.* extracted QoS features with stacked denoising autoencoders before DRL-based scheduling [16], Shaw *et al.* compared Q-learning and SARSA with reward shaping for consolidation [18], and Zeng *et al.* proposed an adaptive DRL-based consolidation framework [17]. These agents act on the currently observed state and are mostly trained from scratch. PRISM differs in three respects: its candidate features include forecast-derived risk; its Q-function embeds an analytic cost prior so that learning starts from expert-level behavior, a goal shared with learning from demonstrations [25]; and a single agent built on deep Q-learning takes decisions across several data centers [26-27].

2.3 Energy Cost in Geo-Distributed Data Centers

Qureshi *et al.* showed that routing requests towards locations with cheaper electricity can save millions of dollars per year [7]. Rao *et al.* minimized the total electricity cost of distributed Internet data centers across multiple electricity markets under delay guarantees [8], and Liu *et al.* studied geographical load balancing with dynamic pricing to encourage the use of renewable energy [9]. These works operate on request routing and server counts, whereas PRISM jointly selects the data center and the host of each flexible VM and couples this choice with consolidation inside each data center.

3 SYSTEM MODEL AND PROBLEM FORMULATION

3.1 Geo-Distributed Cloud

A provider operates a set D of data centers. Data center $d \in D$ contains a set H_d of heterogeneous hosts; host h has CPU capacity C_h (MIPS), memory capacity M_h and a SPECpower power curve. Time is divided into scheduling intervals of Δt = 300 s. VM v is allocated c_v MIPS and m_v MB of memory and has CPU utilization $0 \le u_v(t) \le 1$, so that its demand in interval t is $c_v u_v(t)$. VMs arrive and depart over time. Latency-sensitive VMs must run in their home data center, whereas geo-flexible (delay-tolerant) VMs may be dispatched to any data center at arrival; live migration takes place only within a data center. At the beginning of interval t, the scheduler knows the utilization only up to $t-1$, so every decision is taken under uncertainty about the demand that will actually be served.

3.2 Power, Energy and Cost

The power of host h at utilization u is obtained, as in CloudSim [19], by linear interpolation of its eleven SPECpower measurements $P_{h,0}, \ldots, P_{h,10}$ taken at 0%, 10%, …, 100% load:

$$P_h(u) = P_{h,k} + \left(P_{h,k+1} - P_{h,k}\right)(10u - k) \quad (1)$$

where $k = \lfloor 10u \rfloor$; a sleeping host consumes no power. With a time-varying $\mathrm{PUE}_d(t)$ and tariff $\pi_d(t)$, the facility-level electricity cost is

$$Cost = \sum_t \sum_{d \in D} PUE_d(t)\pi_d(t) \sum_{h \in H_d} P_h(u_h(t))\Delta t \quad (2)$$

Carbon emissions are computed in the same way with the carbon intensity of data center d in place of $\pi_d(t)$.

3.3 QoS and Migration Model

A host is overloaded in interval t when the aggregate demand of its VMs reaches its capacity, in which case part of the demand is not served. Following [4], QoS is measured by the SLA violation time per active host (SLATAH) and the performance degradation due to migrations (PDM):

$$SLATAH = \frac{1}{N}\sum_{i=1}^{N}\frac{T_{s,i}}{T_{a,i}} \tag{3}$$

$$PDM = \frac{1}{M}\sum_{j=1}^{M}\frac{C_{d,j}}{C_{r,j}} \tag{4}$$

where $T_{s,i}$ is the time during which host i is overloaded, $T_{a,i}$ its active time, N the number of hosts, $C_{d,j}$ the degradation of VM j caused by its migrations (10% of its demand during each migration), $C_{r,j}$ its total requested capacity and M the number of VMs. The two metrics are combined into

$$SLAV = SLATAH \cdot PDM, \quad ESV = E \cdot SLAV \tag{5}$$

where E is the IT energy consumption. A migration of VM v lasts $m_v/(B/2)$, with B the host network bandwidth [4]. We additionally report the fraction of the requested CPU demand that is not served.

3.4 Problem Formulation

Let $x_{v,h}(t) \in \{0, 1\}$ indicate that VM v runs on host h in interval t. The scheduler seeks

$$min \sum_t \left[Cost(t) + \lambda_S N_{ovl}(t) + \lambda_M N_{mig}(t)\right] \tag{6}$$

subject to the memory constraints $\Sigma_v x_{v,h}(t) m_v \le M_h$, the assignment constraints $\Sigma_h x_{v,h}(t) = 1$ and the home-data-center constraint of latency-sensitive VMs, where $N_{ovl}(t)$ is the number of overloaded hosts, $N_{mig}(t)$ the number of migrations and λ_S, λ_M their prices. The problem is an online, stochastic, multi-dimensional bin-packing problem with switching costs; it is NP-hard even with perfect knowledge of demand, and the demand is revealed only after the decisions have been taken. PRISM therefore combines learned forecasts, an explicit risk model and a learned decision policy.

4 THE PRISM FRAMEWORK

4.1 Overview

Figure 1 shows the architecture of PRISM. At the start of every interval, PRISM (i) forecasts the demand of all running and arriving VMs, (ii) computes a risk-adjusted load bound for every host, (iii) detects hosts at risk of overload and selects the VMs to be moved with MMT [5], and (iv) consolidates under-loaded hosts by trying to evacuate them in increasing order of predicted load. Every placement decision, i.e., dispatching an arriving VM, relocating a VM from a host at risk, or evacuating an under-loaded host, is delegated to the same residual Q-network, which scores a compact set of feasible candidate hosts. Algorithm 1 summarizes the procedure.

Algorithm 1. PRISM scheduling at the start of interval t

Input: utilization history up to t − 1, tariffs, PUE, risk level ε
1: Forecast $\hat{\mu}v(k)$, $\hat{q}0.5,v(k)$, $\hat{q}0.9,v(k)$, k = 1, 2, 3, for running and arriving VMs (MQF)
2: Compute the risk-adjusted host loads $\hat{U}h(k)$ with (9)
3: for each arriving VM v, largest first do
4: build candidates A in the allowed data centers; place v on arg max$a \in$ A Q(s, a)
5: for each active host h with $\hat{U}h(1) > Ch$ do
6: remove VMs by MMT until (10) holds; relocate each within its data center via arg max Q
7: for active hosts h in increasing order of predicted load do
8: if all VMs of h have a feasible active target then migrate them via arg max Q and put h to sleep
9: Serve interval t; compute rewards (11); store transitions and update θ (training only)

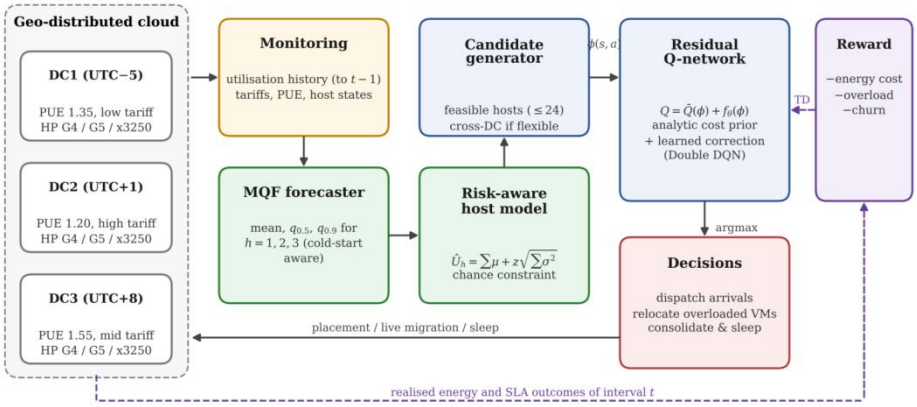


Figure 1 Architecture of PRISM

Note: Quantile forecasts and the risk-aware host model feed a candidate generator; a residual Q-network scores the candidates for dispatch, relocation and consolidation in all data centers and is trained on the realized energy and SLA outcomes.

4.2 Multi-Horizon Quantile Forecaster

For every VM and horizon $k \in \{1, 2, 3\}$ (5, 10 and 15 min), MQF predicts the mean $\hat{\mu}_v(k)$, the median $\hat{q}_{0.5,v}(k)$ and the 90th percentile $\hat{q}_{0.9,v}(k)$ of the utilization. Its input comprises the last 12 utilization samples (one hour); the mean, standard deviation, maximum and minimum of the last 36 samples; the fraction of available history; and a sine–cosine encoding of the time of day. Recently arrived VMs have short histories, so missing samples are masked and replaced by the mean of the available ones, or by the population prior when no sample exists, while the history-fraction feature tells the network how much the inputs can be trusted. A single multilayer perceptron with layers 19–128–128–9 and ReLU activations is trained with the loss

$$L = \sum_{k=1}^{3} \left[(\hat{\mu}(k) - y(k))^2 + \sum_{\tau} \rho_\tau (y(k) - \hat{q}_\tau(k)) \right] \tag{7}$$

where $y(k)$ is the observed utilization, $\tau \in \{0.5, 0.9\}$, and $\rho_\tau(r) = \max(\tau r, (\tau - 1)r)$ is the pinball loss of quantile regression [23]. During training, the history of 25% of the samples is randomly truncated to emulate newly arrived VMs.

4.3 Risk-Aware Host Model

PRISM converts the spread between the 90th percentile and the median into a standard deviation of the demand,

$$\hat{\sigma}_v(k) = \frac{c_v(\hat{q}_{0.9,v}(k) - \hat{q}_{0.5,v}(k))}{z_{0.9}} \tag{8}$$

where z_q denotes the q-quantile of the standard normal distribution. Treating the demands of co-located VMs as independent and invoking the central limit theorem, the risk-adjusted load of host h is

$$\hat{U}_h(k) = \sum_{v \in V_h} c_v \hat{\mu}_v(k) + z_{1-\varepsilon} \sqrt{\sum_{v \in V_h} \hat{\sigma}_v^2(k)} \tag{9}$$

where V_h is the set of VMs on h, and the chance constraint $\Pr\{D_h(t + k) > C_h\} \le \varepsilon$ on the host demand D_h is approximated by

$$\hat{U}_h(k) \le C_h \tag{10}$$

A host is declared at risk of overload when (10) is violated for $k = 1$, and VMs are removed from it by MMT until (10) holds. A host is a feasible target for a VM when its memory suffices and (10) holds for all $k \in \{1, 2, 3\}$ after adding the VM, which prevents placements that become infeasible a few minutes later. Aggregating variances rather than quantiles exploits statistical multiplexing: the safety margin of a host with n similar VMs grows with $\sqrt{n}$ rather than with n. Because the risk level enters only through $z_{1-\varepsilon}$, the operator can change it at deployment time without retraining (Section 5.5).

4.4 Decision Process

Each placement is a decision step of a Markov decision process. A step is described by its candidate set $A = \{a_1, \ldots, a_K\}$ with $K \le 24$, and the action selects one candidate. Candidate a is represented by a 24-dimensional feature vector $\varphi(s, a)$ that contains the observed and predicted utilization of the host before and after the placement; the risk ratio $\hat{U}_h/C_h$ for $k = 1$ and its maximum over k; the overload probability $1 - \Phi((C_h - \mu)/\sigma)$; the memory occupancy and number of VMs after the placement; the marginal power and the marginal cost at the current and at the 6-hour average tariff; a sleep indicator; the peak efficiency (MIPS/W) and capacity of the host; the current price, 24-hour average cost rate and PUE of its data center; the fraction of active hosts in that data center; the predicted mean and spread of the VM; a one-hot code of the decision type (arrival, overload relief or consolidation); and the local time of day. After interval t has been served, the decision receives the reward

$$r = -\left[\hat{c} + (\kappa - 1) w \hat{c}_{idle} + \beta_S o(1 + 10\delta) + \beta_C g\right] \tag{11}$$

Here $\hat{c}$ is the realized marginal cost of the chosen host in interval t (facility energy multiplied by the tariff and normalized by the cost of a 135 W host over one interval at the highest tariff and PUE); $w \in \{0, 1\}$ indicates that the decision woke a sleeping host and $\hat{c}_{\text{idle}}$ is the idle cost of that host at the 6-hour average tariff, so that waking a host is charged an activation commitment of κ intervals; $o \in \{0, 1\}$ indicates that the chosen host was overloaded and δ is its unserved demand relative to its capacity; and $g \in \{0, 1\}$ indicates that the VM had to be migrated again in the next interval (churn).

4.5 Residual Q-Network with an Analytic Prior

Learning a placement policy from scratch is slow and, during early episodes, can be much worse than a heuristic. PRISM therefore decomposes the action value into an analytic prior and a learned residual,

$$Q(s, a) = \bar{Q}(\varphi) + f_\theta(\varphi) \tag{12}$$

$$\bar{Q}(\varphi) = -\hat{c}_6(\varphi)[1 + (\kappa - 1) w(\varphi)] - \eta(1 - \rho(\varphi)) \tag{13}$$

where $\hat{c}_6$ is the predicted marginal cost at the 6-hour average tariff, $w(\varphi)$ is the sleep indicator, $\rho(\varphi)$ is the post-placement risk ratio and $\eta = 10^{-3}$ is a tie-breaker that favors tight packing. The prior alone is a forward-looking, cost-aware variant of PABFD. The residual f_θ is a 24–64–64–1 ReLU network whose output layer is initialized to zero, so that the untrained agent reproduces the ordering of the prior exactly; learning only has to discover where the prior should be corrected, typically when the true overload risk, the likelihood of churn or a cross-data-center trade-off differs from the analytic cost. The network is trained with Double DQN on transitions stored in an experience-replay buffer [26-27], using the target

$$y = r + \gamma Q_\theta\left(s', \operatorname{argmax}_{a'} Q_\theta(s', a')\right) \tag{14}$$

where s' is the next decision and θ^- are the parameters of a softly updated target network, together with a Huber loss. Exploration selects a random action among the four highest-scoring candidates with probability p_x, which avoids harmful random choices such as waking an inefficient host in the most expensive data center. The checkpoint with the lowest validation objective

$$J = Cost + w_S N_{ovl} + w_M N_{mig} \tag{15}$$

is retained, where N_{ovl} is the number of overloaded host-intervals, N_{mig} the number of migrations, and w_S = \$0.0103 and w_M = \$0.0017 are the monetary equivalents of β_S and β_C.

4.6 Candidate Generation and Geo-Distributed Dispatch

For an arriving VM, the allowed data centers are all data centers if the VM is geo-flexible and its home data center otherwise; relocation and consolidation are restricted to the data center of the source host. Among the feasible active hosts, PRISM keeps the eight with the tightest post-placement fit and the eight with the lowest forward-looking marginal cost, and it adds one representative of every (data center, host type) pair that has sleeping hosts, except during consolidation, which must not wake hosts. The candidate set thus contains at most 24 hosts regardless of the size of the cloud. Feature construction is vectorized over all hosts in $O(|H|)$ time, and the network evaluates the candidates in $O(KFn)$ operations, where $F = 24$ and $n = 64$ is the hidden width.

5 EXPERIMENTAL EVALUATION

5.1 Experimental Setup

We use the PlanetLab CPU-utilization traces collected by the CoMon monitoring system and distributed with CloudSim [19-20], the standard benchmark for dynamic VM consolidation since [4]. They comprise 11,746 VM-day traces from ten days in March and April 2011 (898–1,516 VMs per day) at a 5-min resolution, with a mean utilization of 9.3–12.4%. The first five days (March 3 to March 25) are used to train the forecaster and the agents; the last five days (April 3 to April 20) are used only for testing. To obtain dynamic scenarios, half of the VMs of a day are long-running services present from midnight, while the others arrive during the day with a diurnal intensity in the local time of their home region and live for 1–12 h, consuming their real trace values during their lifetime; 40% of all VMs are geo-flexible. Each VM is assigned one of the four EC2-like types of the CloudSim setup (2500, 2000, 1000 and 500 MIPS with 870, 1740, 1740 and 613 MB) [4], and a scenario contains between 511 and 1,000 concurrently running VMs. Each test day is simulated with two scenario seeds, which yields ten test scenarios.

Table 1 lists the three server types, whose SPECpower curves are taken from CloudSim [19]; memory is provisioned such that CPU is the bottleneck resource. Table 2 describes the three data centers, located in UTC−5, UTC+1 and UTC+8, with 185 hosts in total. Each data center has a time-of-use tariff (off-peak 22:00–07:00, on-peak 11:00–17:00 local time, mid-peak otherwise), a PUE with a ±0.05 diurnal cooling component peaking at 15:00 local time, and a representative grid carbon intensity. These site parameters are synthetic but follow typical regional structures. Hosts are connected by 1 Gbit/s links, half of whose bandwidth is available to migrations.

Table 1 Server Types (SPECpower Models from CloudSim)

Server	CPU (cores × MIPS)	RAM (GB)	Pidle / Pmax (W)
HP ProLiant ML110 G4 (Xeon 3040)	2 × 1860	32	86 / 117
HP ProLiant ML110 G5 (Xeon 3075)	2 × 2660	48	93.7 / 135
IBM x3250 (Xeon X3470)	4 × 2933	96	41.6 / 113

Table 2 Data Center Configuration

Site	UTC	G4 / G5 / x3250	PUE	Tariff off / mid / on ($/kWh)	CO_2 (kg/kWh)
DC1 (America)	−5	30 / 30 / 4	1.35	0.055 / 0.085 / 0.130	0.38
DC2 (Europe)	+1	20 / 30 / 8	1.20	0.090 / 0.130 / 0.190	0.30
DC3 (Asia)	+8	35 / 25 / 3	1.55	0.070 / 0.100 / 0.150	0.49

PRISM is compared with seven methods. NM places arriving VMs with PABFD and never migrates. THR-MMT, IQR-MMT, MAD-MMT and LR-MMT are the static-threshold (0.8), IQR (1.5), MAD (2.5) and local-regression (1.2) detectors of [4] with MMT selection, PABFD placement and home-data-center dispatch [5]. LR-MMT-CD is our cost-aware extension of LR-MMT in which flexible VMs are placed by PABFD over all data centers using the tariff- and PUE-weighted power increase. DRL-R is a reactive DRL scheduler that combines THR-MMT detection with a Double DQN placement agent trained on 13 features of the observed (not predicted) utilization with an energy-only reward; it may dispatch flexible VMs to any data center and represents the class of reactive DRL placement methods [16-17]. The consolidation procedure examines at most 75 hosts per interval for all methods.

All methods run in the same discrete-time simulator, implemented in Python and NumPy, which follows the power, migration and SLA models of CloudSim [4, 19]. MQF is trained for eight epochs with Adam on 900,000 samples from the training days. PRISM uses $\varepsilon = 5\%$ ($z_{1-\varepsilon} = 1.645$), $\beta_S = 3$, $\beta_C = 0.5$, $\kappa = 6$, $\gamma = 0.3$, a learning rate of 5×10^{-4}, mini-batches of 64, a replay buffer of 1.2×10^5 transitions, a target update rate of 0.01 and one gradient step every four decisions; p_x decays from 0.10 to 0.02. The replay buffer is seeded with one episode of the prior policy, after which the agent is

trained for 16 episodes of one simulated day each, and model selection uses two training-day scenarios with unseen seeds. Training takes 35 s for MQF and about 8 min for the agent on a single CPU core. DRL-R is trained with the same procedure.

5.2 Forecasting Accuracy

Table 3 compares MQF with persistence, a 12-sample moving average (MA-12), the local-regression (Loess) predictor of [4] and a ridge-regularized autoregressive model trained on the same features (Ridge-AR), using all VMs of the five unseen test days. The mean output of MQF attains the lowest RMSE at every horizon and its median output the lowest MAE, 0.0477 at 5 min, which is 25% below persistence and 21% below Loess. The 90th-percentile output is well calibrated, with an empirical coverage of 0.873, 0.898 and 0.881 at the three horizons. The point-accuracy gain over the linear autoregressive model is modest; the main value of MQF lies in its calibrated quantiles and its treatment of cold-start VMs, as the scheduling results in Section 5.4 confirm.

Table 3 Forecasting Error on the Unseen Test Days

Predictor	MAE_1	MAE_2	MAE_3	$RMSE_1$	$RMSE_2$	$RMSE_3$
Persistence	0.0635	0.0648	0.0646	0.1202	0.1219	0.1213
MA-12	0.0511	0.0512	0.0512	0.0902	0.0907	0.0905
LR (Loess)	0.0603	0.0650	0.0694	0.1076	0.1157	0.1233
Ridge-AR	0.0499	0.0502	0.0500	0.0888	0.0893	0.0890
MQF (mean)	0.0497	0.0499	0.0495	**0.0886**	**0.0891**	**0.0889**
MQF (q50)	**0.0477**	**0.0480**	**0.0480**	0.0910	0.0918	0.0923

Note: Subscripts 1–3 denote horizons of 5, 10 and 15 min; MQF (q50) is the median output of MQF.

5.3 Overall Comparison

Table 4 Results on the Ten Test Scenarios (Mean ± Standard Deviation; Best Value per Column in Bold)

Method	Energy (kWh)	Cost ($)	CO_2 (kg)	SLATAH (%)	PDM ($\times10^{-4}$)	SLAV ($\times10^{-6}$)	ESV ($\times10^{-4}$)	Migrations	Unmet (%)	Active hosts
NM	49.95±13.86	7.32±1.97	29.72	10.41	—	—	—	**0**	1.890	21.1
THR-MMT	53.21±18.06	7.88±2.53	31.74	5.47	4.50	25.02	14.07	23583	0.575	22.2
IQR-MMT	52.57±17.17	7.81±2.41	31.43	5.22	4.59	24.26	13.68	24429	0.636	21.9
MAD-MMT	50.45±16.58	7.49±2.33	30.05	6.97	4.52	32.01	17.32	24071	0.815	20.8
LR-MMT	50.08±17.23	7.42±2.43	29.72	7.42	2.50	18.81	9.67	13364	0.664	20.7
LR-MMT-CD	45.72±14.88	6.26±2.02	23.90	6.54	2.28	15.11	7.30	12020	0.597	19.1
DRL-R	48.79±16.31	7.11±2.24	27.60	5.13	4.22	21.80	11.29	22193	0.506	20.6
PRISM	**39.98±13.49**	**5.94±1.99**	**20.55**	**3.55**	**0.45**	**1.60**	**0.67**	1812	**0.272**	**16.6**

Note: NM performs no migration, so its PDM, SLAV and ESV are undefined (—).

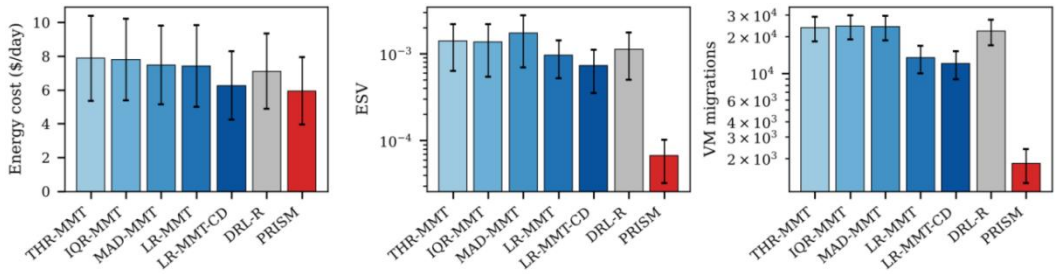


Figure 2 Electricity Cost, ESV and Number of VM Migrations on the Ten Test Scenarios (Mean ± Standard Deviation; Logarithmic Scale for ESV and Migrations)

Table 4 reports the averages over the ten test scenarios, and Figure 2 visualizes three key metrics. The adaptive heuristics trade energy against QoS, but all of them migrate VMs on a massive scale: THR-, IQR- and MAD-MMT perform 23,583–24,429 migrations per day and LR-MMT 13,364, because they react to the most recent utilization and repeatedly relocate VMs with fluctuating load. Cost-aware dispatch (LR-MMT-CD) lowers energy by 8.7% and cost by 15.6% relative to LR-MMT, which confirms the value of geo-flexible workloads, but it leaves SLA violations and migrations almost unchanged. DRL-R achieves a lower SLATAH than LR-MMT but performs 22,193 migrations, as it inherits the threshold detector and cannot anticipate the load.

PRISM attains the best value of every metric except the number of migrations, for which only NM, which never migrates, is lower. In a paired comparison with LR-MMT-CD over the ten scenarios, PRISM reduces energy by 12.5% (from 4.8% to 22.4%), carbon emissions by 14.1%, SLATAH by 44.6%, ESV by 90.1%, migrations by 84.6% and unserved demand by 52.4%, each in all ten scenarios; the electricity cost decreases by 5.2% on average and in eight of the ten scenarios. Relative to LR-MMT, energy and cost are both reduced by about 20% and ESV by 92.4%. Figure 3 illustrates the mechanism on one test day: PRISM keeps fewer hosts active throughout the day (16.6 on average over all scenarios, versus 19.1 for LR-MMT-CD), and its power curve stays below those of the baselines even at the daily peak. The chance

constraint allows hosts to be packed to a quantified risk rather than to a fixed threshold, and the learned residual steers VMs away from hosts on which they would cause churn.

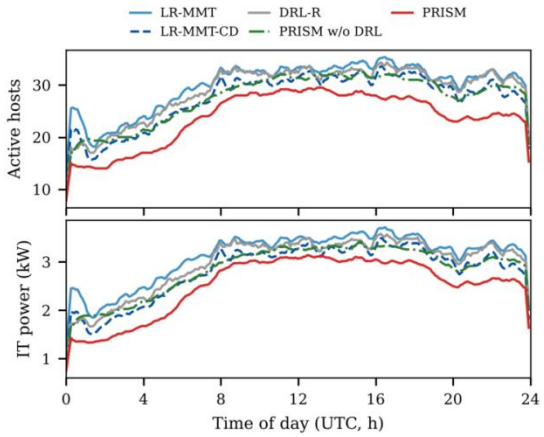


Figure 3 Active Hosts and IT Power over the Test Day 2011-04-03 (Seed 1, 30-min Moving Average)

5.4 Ablation Study

Table 5 removes one component at a time on the same scenarios. The variants without MQF and without uncertainty quantification (UQ) are retrained with the same procedure, the variant without DRL uses the prior (13) alone, and the variant without geo-dispatch deploys the trained agent with home-data-center dispatch. Removing the learned residual (w/o DRL) increases energy by 21.6% and ESV by 81%, which shows that the analytic prior is a strong but clearly improvable policy. Replacing MQF by persistence forecasts with a constant spread of 0.11 of the VM capacity (the persistence RMSE on the training days) multiplies migrations by 11.1 and ESV by 8.7, because a constant spread overstates the risk of stable VMs and understates that of bursty ones. Removing UQ (mean forecasts only) yields the lowest energy and cost, but SLATAH rises to 23.9% and the unserved demand to 2.6%, about ten times the values of PRISM, since packing to the expected demand ignores bursts. Restricting arrivals to their home data center (w/o Geo) increases cost by 21.5% and energy by 21.3%, as flexible VMs can no longer be pooled on the most efficient and cheapest servers.

Table 5 Ablation Study (Mean over the Ten Test Scenarios)

Variant	Energy (kWh)	Cost ($)	SLATAH (%)	ESV ($\times 10^{-4}$)	Mig.	Unmet (%)	J
PRISM (full)	39.98	5.94	3.55	**0.67**	**1812**	**0.272**	**10.86**
w/o DRL	48.63	6.53	3.93	1.21	2341	0.365	13.02
w/o MQF	48.66	7.39	**3.20**	5.82	20134	0.326	43.62
w/o UQ	**34.19**	**5.05**	23.87	6.71	2509	2.646	20.55
w/o Geo	48.51	7.22	3.70	1.15	2320	0.327	13.46

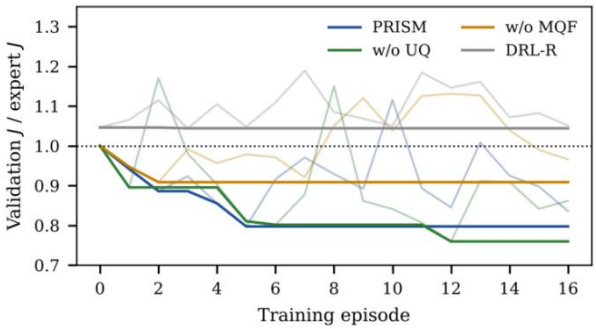


Figure 4 Validation Objective during Training, Normalized by That of Each Agent's Starting Policy (Faint: per Episode; Solid: Best So Far)

Figure 4 shows the validation objective (15) during training, normalized by that of the starting policy. Owing to the zero-initialized residual, the PRISM variants start exactly at their prior, and the retained checkpoint is never worse than the prior on the validation scenarios. PRISM improves on its prior by 20.3% within five episodes, and the variants without UQ and without MQF improve by 24.0% and 9.2%, respectively, whereas DRL-R, whose features carry neither forecasts nor risk information, does not improve on its starting point.

5.5 Sensitivity to the Risk Level

Figure 5(a) compares the target risk ε with the realized frequency of overloaded host-intervals when $z_{1-\varepsilon}$ is changed at deployment time. The realized frequency follows the target closely: 6.6%, 3.4% and 2.0% for ε = 10%, 5% and 2%, and 1.16% and 0.84% for ε = 1% and 0.5%; the slight excess at the two strictest levels stems from correlated bursts that the independence assumption in (9) does not capture. Figure 5(b) shows the resulting energy–QoS trade-off. At every risk level, PRISM uses 13.7–19.1% less energy than its prior at a similar or lower SLATAH, and every baseline is dominated by at least one PRISM operating point. For instance, with ε = 1%, PRISM attains a SLATAH of 0.86% with 43.4 kWh per day, whereas LR-MMT-CD consumes 46.0 kWh and still exhibits a SLATAH of 6.9%.

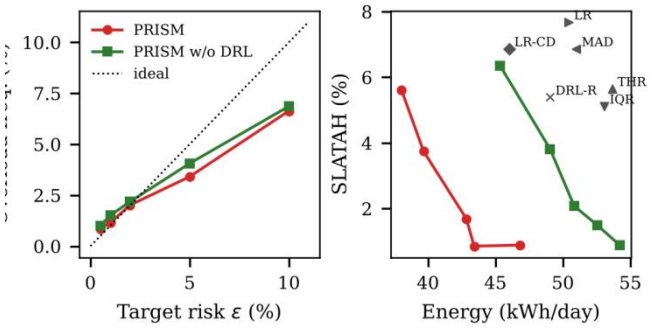


Figure 5 (a) Realized Overload Frequency versus Target Risk Level ε. (b) Energy–SLATAH Trade-Off of PRISM and of Its Prior for ε ∈ {10, 5, 2, 1, 0.5}% Compared with the Baselines (Test Days, Seed 1)

5.6 Effect of Geo-Flexibility

Figure 6 varies the share of geo-flexible VMs from 0% to 80% while the agent, trained at 40%, is left unchanged. Without flexible VMs, PRISM and LR-MMT-CD consume a similar amount of energy (49.7 vs. 50.4 kWh) at a similar cost ($7.41 vs. $7.46), but PRISM halves SLATAH (3.8% vs. 7.7%). As the share of flexible VMs grows, PRISM converts the additional freedom into larger savings: at 80% its cost is 22.1% below the value without flexibility (19.1% for LR-MMT-CD), and its carbon emissions fall from 29.6 to 18.6 kg per day, against 22.8 kg for LR-MMT-CD. Most of the benefit is already obtained when 40% of the VMs are flexible.

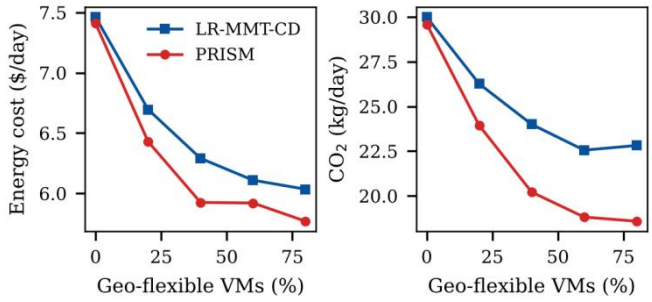


Figure 6 Electricity Cost and CO_2 Emissions versus the Share of Geo-Flexible VMs (Test Days, Seed 1)

5.7 Computational Overhead

PRISM requires 16.3 ms of computation per 5-min interval for the entire three-data-center system (185 hosts and up to 1,000 VMs), including forecasting, on a single CPU core, compared with 10.6 ms for LR-MMT, 34.3 ms for IQR-MMT and 55.0 ms for MAD-MMT. The overhead is negligible relative to the scheduling interval and to the duration of a live migration, and the bounded candidate set makes the cost of evaluating the network independent of the number of hosts.

6 CONCLUSION

This paper presented PRISM, a prediction-guided, risk-aware DRL framework for energy-aware dynamic resource scheduling in distributed cloud data centers. Calibrated quantile forecasts are turned into a host-level chance constraint, and a residual Q-network built on an analytic energy-cost prior takes dispatch, relocation and consolidation decisions across data centers. On real PlanetLab traces, PRISM reduced energy consumption by 12.5%, ESV by 90.1% and migrations by 84.6% relative to the strongest baseline, and its risk level can be tuned at deployment time to move along a dominant energy–QoS frontier. Future work will extend PRISM to memory and network resources, model correlated demand among co-located VMs, incorporate on-site renewable generation and evaluate the framework on further production traces.

COMPETING INTERESTS

The authors have no relevant financial or non-financial interests to disclose.

REFERENCES

[1] Masanet E, Shehabi A, Lei N, et al. Recalibrating global data center energy-use estimates. Science, 2020, 367(6481): 984-986.

[2] Barroso L A, Hölzle U. The case for energy-proportional computing. Computer, 2007, 40(12): 33-37.

[3] Clark C, Fraser K, Hand S, et al. Live migration of virtual machines. In: Proceedings of the 2nd USENIX Symposium on Networked Systems Design and Implementation (NSDI), 2005: 273-286.

[4] Beloglazov A, Buyya R. Optimal online deterministic algorithms and adaptive heuristics for energy and performance efficient dynamic consolidation of virtual machines in cloud data centers. Concurrency and Computation: Practice and Experience, 2012, 24(13): 1397-1420.

[5] Beloglazov A, Abawajy J, Buyya R. Energy-aware resource allocation heuristics for efficient management of data centers for cloud computing. Future Generation Computer Systems, 2012, 28(5): 755-768.

[6] Verma A, Ahuja P, Neogi A. pMapper: Power and migration cost aware application placement in virtualized systems. In: Proceedings of ACM/IFIP/USENIX Middleware, LNCS 5346, 2008: 243-264.

[7] Qureshi A, Weber R, Balakrishnan H, et al. Cutting the electric bill for internet-scale systems. In: Proceedings of ACM SIGCOMM, 2009: 123-134.

[8] Rao L, Liu X, Xie L, et al. Minimizing electricity cost: Optimization of distributed internet data centers in a multi-electricity-market environment. In: Proceedings of IEEE INFOCOM, 2010: 1145-1153.

[9] Liu Z, Lin M, Wierman A, et al. Greening geographical load balancing. In: Proceedings of ACM SIGMETRICS, 2011: 233-244.

[10] Farahnakian F, Pahikkala T, Liljeberg P, et al. Energy-aware VM consolidation in cloud data centers using utilization prediction model. IEEE Transactions on Cloud Computing, 2019, 7(2): 524-536.

[11] Calheiros R N, Masoumi E, Ranjan R, et al. Workload prediction using ARIMA model and its impact on cloud applications' QoS. IEEE Transactions on Cloud Computing, 2015, 3(4): 449-458.

[12] Kumar J, Singh A K. Workload prediction in cloud using artificial neural network and adaptive differential evolution. Future Generation Computer Systems, 2018, 81: 41-52.

[13] Mao H, Alizadeh M, Menache I, et al. Resource management with deep reinforcement learning. In: Proceedings of the 15th ACM Workshop on Hot Topics in Networks (HotNets), 2016: 50-56.

[14] Liu N, Li Z, Xu J, et al. A hierarchical framework of cloud resource allocation and power management using deep reinforcement learning. In: Proceedings of the IEEE 37th International Conference on Distributed Computing Systems (ICDCS), 2017: 372-382.

[15] Cheng M, Li J, Nazarian S. DRL-cloud: Deep reinforcement learning-based resource provisioning and task scheduling for cloud service providers. In: Proceedings of the 23rd Asia and South Pacific Design Automation Conference (ASP-DAC), 2018: 129-134.

[16] Wang B, Liu F, Lin W. Energy-efficient VM scheduling based on deep reinforcement learning. Future Generation Computer Systems, 2021, 125: 616-628.

[17] Zeng J, Ding D, Kang K, et al. Adaptive DRL-based virtual machine consolidation in energy-efficient cloud data center. IEEE Transactions on Parallel and Distributed Systems, 2022, 33(11): 2991-3002.

[18] Shaw R, Howley E, Barrett E. Applying reinforcement learning towards automating energy efficient virtual machine consolidation in cloud data centers. Information Systems, 2022, 107: Art. 101722.

[19] Calheiros R N, Ranjan R, Beloglazov A, et al. CloudSim: A toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms. Software: Practice and Experience, 2011, 41(1): 23-50.

[20] Park K, Pai V S. CoMon: A mostly-scalable monitoring system for PlanetLab. ACM SIGOPS Operating Systems Review, 2006, 40(1): 65-74.

[21] Cortez E, Bonde A, Muzio A, et al. Resource Central: Understanding and predicting workloads for improved resource management in large cloud platforms. In: Proceedings of the 26th ACM Symposium on Operating Systems Principles (SOSP), 2017: 153-167.

[22] Khan T, Tian W, Ilager S, et al. Workload forecasting and energy state estimation in cloud data centres: ML-centric approach. Future Generation Computer Systems, 2022, 128: 320-332.

[23] Koenker R, Bassett G. Regression quantiles. Econometrica, 1978, 46(1): 33-50.

[24] Farahnakian F, Liljeberg P, Plosila J. Energy-efficient virtual machines consolidation in cloud data centers using reinforcement learning. In: Proceedings of the 22nd Euromicro International Conference on Parallel, Distributed, and Network-Based Processing (PDP), 2014: 500-507.

[25] Hester T, Vecerik M, Pietquin O, et al. Deep Q-learning from demonstrations. In: Proceedings of the 32nd AAAI Conference on Artificial Intelligence (AAAI), 2018: 3223-3230.

[26] Mnih V, Kavukcuoglu K, Silver D, et al. Human-level control through deep reinforcement learning. Nature, 2015, 518(7540): 529-533.

[27] van Hasselt H, Guez A, Silver D. Deep reinforcement learning with double Q-learning. In: Proceedings of the 30th AAAI Conference on Artificial Intelligence (AAAI), 2016: 2094-2100.