Latency-Aware Dynamic Server and Cooling Capacity Provisioner for Data Centers

Anuroop Desu[1], Udaya Puvvadi[1], Tyler Stachecki[1], Sagar Vishwakarma[1], Sadegh Khalili[2], Kanad Ghose[1] and Bahgat G Sammakia[2]
[1]Department of Computer Science, State University of New York, Binghamton, NY 13902
[2]Department of Mechanical Engineering, State University of New York, Binghamton, NY 13902
{adesu1, upuvvad1, tstache1, svishwa2, skhalil6, ghose, bahgat}@binghamton.edu

ABSTRACT

Data center operators generally overprovision IT and cooling capacities to address unexpected utilization increases that can violate service quality commitments. This results in energy wastage. To reduce this wastage, we introduce HCP (Holistic Capacity Provisioner), a service latency aware management system for dynamically provisioning the server and cooling capacity. Short-term load prediction is used to adjust the online server capacity to concentrate the workload onto the smallest possible set of online servers. Idling servers are completely turned off based on a separate long-term utilization predictor. HCP targets data centers that use chilled air cooling and varies the cooling provided commensurately, using adjustable aperture tiles and speed control of the blower fans in the air handler. An HCP prototype supporting a server heterogeneity is evaluated with real-world workload traces/requests and realizes up to 32% total energy savings while limiting the 99th-percentile and average latency increases to at most 6.67% and 3.24%, respectively, against a baseline system where all servers are kept online.

CCS CONCEPTS

• Computing Methodologies → Distributed Computing Methodologies.

KEYWORDS

Energy-performance optimization, data centers, resource management

ACM Reference Format:

Anuroop Desu, Udaya Puvvadi, Tyler Stachecki, Sagar Vishwakarma, Sadegh Khalili, Kanad Ghose, and Bhagat G. Sammakia. Latency-Aware Dynamic Online Server and Cooling Capacity Provisioning for Data Centers. In *Proceedings of SoCC '21, Seattle, WA, USA, November 1-4, 2021,* 15 pages. https://doi.org/10.1145/3472883.3487015

SoCC '21, November 1–4, 2021, Seattle, WA, USA

ACM ISBN 978-1-4503-8638-8/21/11…$15.00
https://doi.org/10.1145/3472883.3487015

1. Introduction

Data center operators generally keep spare IT capacity on hand to deal with any sudden growth in the offered workload that can otherwise result in the violation of service commitments. A report from Lawrence Berkley National Lab [38] and a study commissioned by the New York Times [20] indicates that many operators of online services keep all their servers on, irrespective of the instantaneous demand. Agility, achieved through overprovisioning, leads to poor energy utilization across the data center, which may be as low as 12% for data centers offering online services [6]. In [30], the most recent across-the-board study on data center utilization, on-premise data centers are reported to have utilizations in the 12% to 18% range. Low average server utilization is also reported in [38]. Lightly utilized or idling servers have low energy efficiency - a direct consequence of the relatively large idle power consumption. Even in the recent generation of servers, idle power is often over 20% of their peak power consumption. Factors contributing to high idle power consumption in servers include power supplies that are inefficient at low utilization levels, use of dual power supplies (for reliability), continuous power draw by components such as RAM, server fans, multiple storage drives and by many components whose energy usage is utilization independent.

1.1. Dynamic, Load-Dependent Server Capacity Adjustments

Dynamic provisioning of server capacity matches the number of deployed servers (hereafter called online servers) to the current requests. Using servers with low idle power is the obvious solution for avoiding efficiency losses at low utilization levels. This is an option that is available at newer cloud datacenters operated by large cloud service providers with custom-built servers. Alternatively, servers with ACPI S3 power state support are usable, as they can be suspended quickly into a quiescent state *and* can be revived from the quiescent rapidly. Unfortunately, both options are not viable for a significant number of data centers owned by enterprises and organizations that have to rely on widely available, stock servers without these

features. Many of such data centers have also not been migrated to the cloud, for a variety of valid reasons [17, 38]. These "private" data centers using stock servers also lack the extensive instrumentation, monitoring and scalable management facilities used in the mega data centers [38] for reducing the energy needs. Finally, private data centers account for about 60% of the energy used by all data centers even though they handle a smaller volume of requests [11]. *The solution presented here addresses the needs of such private enterprise datacenters.*

1.2. Goals and Scope of this Work

This paper presents **HCP**, an automated and dynamic **H**olistic **C**apacity **P**rovisioner that tracks and matches the deployed server and cooling capacity to the workload demands. This leads to improved energy savings and energy proportionality. HCP incorporates two modules, a **D**ynamic **C**apacity **P**rovisioner (**DCP**) for rightsizing the deployed online server capacity and **A**utomatic **C**ooling **P**rovisioner (**ACP**) that provides adequate cooling for the active servers. DCP and ACP both use a **S**hort-**T**erm utilization **P**redictor (**STP**) to activate servers and adjust the cooling provided to handle the instantaneous workload. STP is geared specifically to predict rapid fluctuations in the offered load, including rapid changes lasting for short durations. Rapid server deployment based on the prediction is supported using a pool of recently offlined servers and a small pool of standby, turned-on servers (*hot spares*). While the recently offlined servers and the hot spares absorb the sudden utilization growth, the increased utilization is gradually shifted onto the servers that are activated in the meantime to effectively hide the large turn-on time of servers. DCP turns off idling servers conservatively using a separate **L**ong-**T**erm **P**redictor (**LTP**) that ignores the impact of short-lived transients and focuses on the longer-term utilization trends.

DCP's goal is to realize IT energy savings compared to a baseline system where all servers are kept active, while maintaining the average tail latency of the requests as close as possible to that of a fully-provisioned baseline system. In addition to DCP, ACP also dynamically matches the cooling to the provisioned servers to save energy compared to a fully provisioned baseline system or a reactively controlled cooling system. A prototype implementation of HCP using 247 heterogeneous servers realized energy savings up to 32% while limiting the average and 99th-percentile tail latency increases to less than 3.24% and 6.67%, respectively, compared to the fully-provisioned baseline system.

2. Related Work and Comparison with HCP

In this section, representative related work is presented and contrasted to the two major components of HCP: DCP, the online server capacity provisioner and ACP, the automated cooling capacity provisioner, both of which rely on the prediction of the incoming requests.

2.1. Server Capacity Provisioning

In provisioning server capacity, DCP makes use of the actual system utilization to account for the service time variations that are inevitable even within a single job class and also to account for any unrelated background activity within the online servers. If these variations are not accounted for, service quality can be adversely affected. For example, Autoscale [19] considers the number of live connections to online servers as an estimate of the system load and experiences serious increases in the tail latencies of the requests served (over 200%, as reported in [19]). Similarly, Facebook's Autoscale [43], a different approach with the same name, relies on a typical diurnal load pattern and uses a simple linear extrapolation of the observed load to adjust the server capacity without offering hard service quality guarantees. PEGASUS improves energy proportionality with explicit CPU power state control, based on the actual request latencies to limit the tail latencies, without any server shutdown [26]. Idle servers contribute to power wastage and the impact on service latencies with complete server shutdowns (as supported in DCP) is unclear. Sleepscale [25] selects policies based on state transition latencies for sleep states to reduce power consumption. DCP does not rely on the existence of multiple sleep states and shuts off idle servers while still guaranteeing service quality.

Paragon [14] uses collaborative filtering to quickly recognize the similarities between an incoming workload and previously scheduled workload classes. It schedules the incoming requests to improve server utilization with reduced interference. Idle servers are put to sleep after a predetermined interval. The delay, however small, that Paragon's filtering technique has, makes it challenging to handle very short-duration requests. Incorrect job scheduling can result from rapid phase changes in the job, requiring rescheduling and further isolation steps [14]. Paragon claims to satisfy the latency requirements of 91% of the jobs. In contrast, DCP consciously provides the latency guarantees for short as well as longer duration jobs by monitoring online server utilization and predicting rapid utilization shifts.

Request replication or replaying is used in Google's hedging [16] to cap the very highest percentile latencies but this results in additional energy usage. DCP does not require such rescheduling, because it addresses tail latency increases very explicitly (Sec. 3). UBIS [22] uses resource needs and clustering techniques to improve the throughput and utilization, avoiding resource overprovisioning. In

[48], a linear programming model, along with DVFS is used to trade off network slack for computational capacity to reduce the energy consumed on latency sensitive jobs. DCP handles energy savings and latency guarantees directly and jointly using its predictors.

Other recent efforts focusing on minimizing power consumption and SLA violations in virtualized data centers are presented in [42, 47, 52]. Predicting the offered workload for data centers - the basis for proactive control is exemplified in [15, 48]. These predict the workload arrival pattern without looking at the system utilization. Most of these techniques look at the long-term data center workload patterns but are unable to deal with sudden utilization variations ("flash crowds") and previously unseen traffic patterns. DCP uses its short-term predictor to handle unseen traffic and flash crowds. Some techniques, such as [29], attempt to predict incoming request *rates* on a shorter time scale. DCP, in contrast, relies on the actual load imposed on the online servers for its predictions, accounting for the inevitable demand variations within a workload class from one request to another. Other server capacity management techniques are generally integrated into data center infrastructure management tools as operator-specified rule-based server onlining and offlining techniques [21, 36, 41]. Rule-based server capacity provisioning is only effective on predictable workload patterns (such as known hourly profiles during a day); these techniques fail to deal with unexpected workload patterns, such as flash crowds [22, 33, 44].

DCP is also unique among similar existing techniques in using two separate predictors, STP and LTP. STP is used to predict the short-term increases in the utilization caused by rapid, short-lived utilization variations and is used for making decisions related to increasing the online server capacity. DCP uses LTP for predicting the long-term load and taking online server capacity reduction decisions. The results from our full-system prototype implementation demonstrates DCP's success in providing tight service quality guarantees using STP and LTP.

2.2. Cooling Capacity Provisioning

Cooling management techniques have been studied extensively in the thermal community, largely without integration with the server management systems. Thermal awareness in managing a data center and its relation to workload/server management is explored in [13] to reduce the cooling costs. Explicit control of CRACs (computer room air conditioners) to reduce the cooling costs is presented in [2, 29] using temperature sensors within the server. Other examples of thermally-aware data center cooling management are seen in [29, 34, 35, 43]. Distributed sensors can monitor the temperature of IT equipment and adjust the CRAC supply air temperature and air flow rate [4]. Data center cooling microgrids are used in [49] to provision cooling. Workloads can be scheduled to a specific data center to take advantage of their environmental conditions and cooling efficiency [46]. Workload assignment using power and thermal models of the servers can result in cooling cost reduction [31]. For both [31] and [46], the impact on tail latencies and overall performance is not clear. A control framework for provisioning, transport, and distribution of cooling resources at both zonal and local levels using adjustable aperture tiles is presented in [50] without consideration of the IT workload or performance. An integrated management of data center involving performance, power, and cooling which uses VMs for shifting the workloads to avoid hot spots or for workload consolidation is presented in [12]. Rack inlet temperatures are used as an indicative for controlling CRAC supply air temperature and flow rates but a significant fraction of the requests experienced latencies higher than targeted.

ACP, the cooling capacity provisioner of HCP, takes a different approach compared to the above techniques. First, ACP relies on the short-term load prediction made by the STP and is thus a *pro-active* approach. *Existing techniques, in this regard, are reactive*, as they use temperature sensors and actuate controls for provisioning the cooling, rather than predicting any trend whatsoever. Second, ACP adjusts the supplied chilled air based on the sensed air pressure in the contained cold aisle and in the plenum, rather than controlling the supply indirectly (and reactively) using sensed temperatures. Finally, DCP, by limiting the core temperatures of the servers also avoids the formation of localized hot spots by temporarily limiting requests to a hot server, thus implementing objectives similar to those of [31] and [46], discussed above.

2.3. Other Differences with Existing Work

Two other unique attributes of HCP that stand out against existing work are now presented. The short-term load predictor, STP, is used by both DCP and ACP. STP relies on a regression-based machine learning technique and is unique in using the load trend on recently-onlined servers as *one* of the features used in predicting system utilization (Sec. 3.3). We are not aware of any other load predictor for similar applications that make use of this specific feature to predict the system utilization. Finally, DCP also stands out from most of the existing work in turning off servers to avoid energy wasted by idling servers and hides the long server turn-on times by: (a) quickly onlining recently offlined servers that are recently offlined but serving already-assigned requests, and, (b) using a small set of hot-spares. While servers from these two lists address utilization increases, the powered off servers are turned on in the background (Sec. 3.6). Overall, to the best of our

knowledge, HCP represents a unique service quality-aware dynamic provisioner for both server and cooling capacity management.

3. DCP: The Server Capacity Provisioner

Dynamic server capacity provisioning is used for matching the deployed online server capacity to the offered workload. This is accomplished by operating just the right number of online servers at high utilization levels to improve the overall energy efficiency and to reduce the power consumed by idle or poorly-utilized servers. HCP's approach to dynamic server capacity adjustments is proactive in nature and the online server capacity is dynamically adjusted by its DCP module. DCP performs the capacity adjustments based on *two* predictions derived from the periodically-reported utilizations from individual online servers. The first of the two predictions is provided by **STP**, a random forest based machine learning predictor that predicts the load growth a *few tens of seconds* into the future, particularly, accounting for very bursty load increases. STP makes a new prediction periodically at fixed scheduling intervals. STP's utilization prediction is extrapolated to determine if the online server capacity must be increased and how rapidly it has to be increased in order to avoid service quality degradations. STP also corrects for mispredictions very quickly by capturing the impact of rapid, short-lived utilization changes.

The second predictor used by DCP, **LTP**, is a far simpler predictor that uses a weighted moving average of the observed history of the actual system utilization to predict utilization changes a *few minutes* into the future. Capacity reductions are made only when the utilization is consistently predicted to be lowered for several consecutive scheduling intervals. It is to be noted that the long-term prediction cannot be derived from STP, as STP's prediction accuracy drops sharply when the duration of the prediction window increases beyond several tens of seconds.

3.1. Quantifying Server and System Utilization

DCP addresses service latency requirements by capping the utilization of the active servers at level U_{max} to guarantee that the service latencies of the 99th-percentile requests are not increased appreciably compared to the request latencies of a fully provisioned system where all servers are online. U_{max} is chosen through a calibration process to determine the maximum CPU utilization allowed on a server for given application without jeopardizing the service latency guarantees (Sec. 3.7). U_{max} is a function of the server model and the application. Thus, it is different for each pair of the server model and application running in the data center. Second, DCP eliminates the need to have fast wakeup servers by employing a small number of already powered-on servers.

DCP relies on the concept of **scaled system utilization**. For a single server, this metric is defined as $U_s \,/\, U_{max}$, where U_s is the conventionally reported server utilization. Servers in a typical data center are unlikely to be identical and thus have different U_{max} values when used to serve a given class of requests. To deal with such heterogeneity, servers are divided into **virtual groups**, consisting servers of identical type/configuration within each such group. Each virtual group has its own specific U_{max} and the average utilization of a virtual server group is defined as the average scaled utilization of all servers currently serving requests within the virtual group.

DCP distributes the incoming requests evenly across online servers within multiple permissible virtual groups. The **service capacity of a virtual group** (say, group *i*) is defined by the product of the maximum permitted utilization (U_{max}) of the server in the virtual group and the total number of active servers (say, N_i) in the group. Put in other words, the load balancer distributes a fraction $W_i = (U_{max_i} \times N_i) / \sum_i (U_{max_i} \times N_i)$ of the requests to virtual group *i*, where the summation is performed over all virtual groups. W_i represents the fraction of incoming requests that the group can serve at any time without exceeding its service capacity. Once the load balancer selects a group, the request is assigned to the online server of the group that has the least number of client connections. The **instantaneous system utilization**, U_{sys}, is thus the mean of the average scaled utilization of all groups serving online requests.

The resulting technique guarantees that the overall utilization of all turned-on servers is kept in the range $(U_{max} - \Delta)$ to U_{max} at all times, with Δ providing the necessary hysteresis for control. The value of Δ is chosen to provide enough hysteresis to avoid frequent server activations and deactivations.

3.2. Accommodating Server Heterogeneity

Server capacity provisioners need to consider the inevitable heterogeneity of servers in typical data centers with different classes, configurations and generations of servers. As indicated in Sec. 3.1, DCP uses the notion of virtual groups of servers, where members within each virtual group are identical in all respects. Requests directed to the data center can be directed to online servers within any group by the load balancer. Each virtual group also requires its own utilization predictor to determine when a server within a virtual group must be bought online or taken offline, based on the predicted utilization. The use of virtual groups also enables different load balancing strategies to be implemented, for example, at lower system utilization levels, the most energy-efficient virtual groups can be deployed to save power. In the specific system presented here, the load balancer did not implement any

policy that differentiated one server group from another and distribute the incoming requests across all existing virtual server groups uniformly.

3.3. STP: The Short-term Utilization Predictor

The goal of an utilization predictor, in general, is to use historical information of certain system parameters that have high correlation with the actual utilization to predict utilization at a future point in time. What makes utilization prediction challenging for data centers serving online requests is the highly transient nature of the requests. The request arrival pattern can be smooth, changing slowly or changing very rapidly; the transient utilization pattern over time can also be very bursty. The implication of all of this is that *simple* extrapolations of the historical values of the observable system parameters (that have high co-relationship to system utilization) cannot predict rapid changes in the actual utilization into the future. To limit the average and 99-th percentile request service latencies compared to the corresponding values for a fully provisioned system (where all servers are kept online), we need a predictor capable of predicting rapid load variations with very high accuracy. From a practical standpoint, this also means that one cannot expect to predict the utilization too much ahead into the future.

Identification of Features for Prediction: To determine system parameters that have a high degree of correlation with the system utilization, the data of easily measurable system parameters and the actual utilization were collected from a variety of synthetically-generated utilization traces. The utilization traces had randomly-varying request rates that generated requests whose characteristics were equally varied: single-threaded as well as multi-threaded requests were used, with the request service latencies varying over a wide range. The rate of utilization changes in these synthetic traces also varied over a wide range and included, but were not limited to, very sharp increases and decreases in the request arrival volume. Stepped increases in utilization (taking place instantaneously) were not used as network and server queues in a real system will limit the rate at which requests ultimately get served.

The measured (and logged) system parameters with the actual execution of the synthetic traces included the number of requests being served, total power consumption (measured in real-time using rack PDUs), highest and lowest utilization level of any online server, highest and average server core temperature and others. All these parameters are fairly intuitive and correlate to the total system utilization. The parameters values were collected at an interval of 500 milliseconds throughout the run.

Recently-Activated Servers as Indicators of Transient Load Changes: To capture highly transient and short-burst utilization spikes (utilization increases), another unique parameter was measured and logged – this is the *average* utilization *of the **R**ecently **A**ctivated **S**ervers* (**RAS**) within the past **six** measurement intervals. The reason for using this parameter is best understood from the way load balancing switches typically operate. Load balancing switches direct incoming request to servers based on the number of connections, usually weighted, such as weighted least connections. When a server is activated, it has the least number of actual connections/requests assigned to it by the load balancer. If the weighted least connection algorithm is used to direct incoming requests to the servers, the most recently-activated servers would get relatively more requests directed to them. Thus, one would expect the utilization of recently-activated servers to be indicative of (that is, correlate to) the short-term trends in the volume of workload directed to the data center.

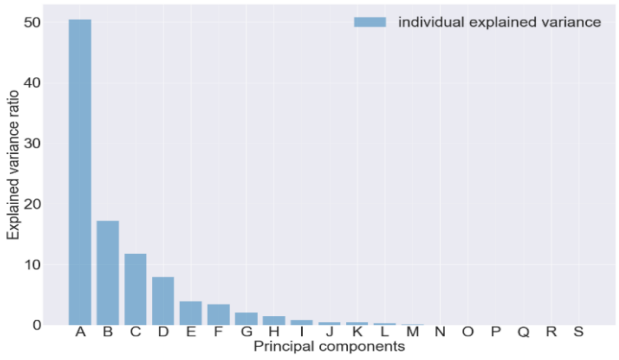


Figure 1: PCA on the training dataset

To reduce the number of input parameters for the utilization prediction, Principal Component Analysis (PCA) [1] was performed to determine the most dominant measurable system parameters that will have a high correlation to the actual system utilization. The result, shown in Figure 1, shows that the six dominant factors (labeled A through F) that affect the system utilization are:

A: Total utilization of online servers that are serving requests and can accept new requests (this excludes servers taken offline that may still be running requests already assigned to them).

B: Total number of requests being served.

C: The average utilization of recently-activated servers.

D: The total power consumption of the system.

E: The highest utilization level of any server.

F: The lowest utilization level of any server.

For the short-term utilization prediction, we used a machine learning approach to predict the system utilization based on the Random Forest regression model [8] with adaptive boosting using the Adaboost algorithm [18]. The

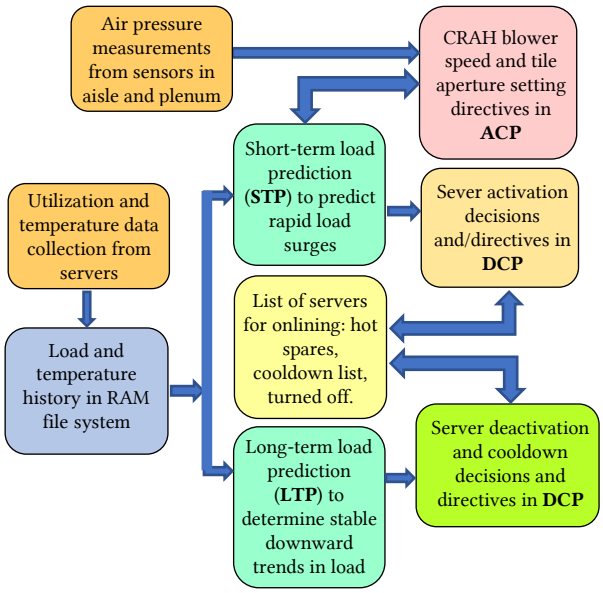


Figure 2: Use of utilization predictors in DCP and ACP

overall model was trained on the synthetically generated traces which resemble real-world workload traces, as noted earlier.

Training and Validation: The goal of DCP is to predict the overall system utilization *N* seconds ahead into the future based on the observed data of the six features in the last 30 seconds and uses the six features filtered with the PCA. During the training process, the previous 30 seconds of data for all the six features are mapped to the target label ("avg. system utilization") *N* seconds into the future. In short, the six feature values at time *T* are mapped to the label at time *T+N* seconds. For training, 80% of the entire data set was used and the remaining 20% was used for validation.

Random Forest Regression Model of STP: A Random Forest **R**egression **M**odel (**RFRM**) was used to predict the system utilization as it uses the ensemble learning method and combines predictions from multiple trees. Within the RFRM, each tree is created from a different sample of rows and at each node, a different sample of the features is selected for splitting. Each tree makes its own individual prediction, and all these predictions are then averaged to produce a single result (prediction). Hence, the prediction made by the RFRM is much more accurate than the prediction made by a single model. The machine learning model used by STP uses a random forest with 1000 trees in the forest, each with a maximum depth of six (corresponding to number of selected features from PCA).

Adaptive Boosting [18], which is an ensemble technique, was used to further improve the RFRM's prediction by training a sequence of base models. Each model compensates the weakness of its predecessors, and this allows any common errors produced by the decision trees in the RFRM to be removed. In Adaptive Boosting, higher points are assigned to the data which is incorrectly predicted by the previous model. For STP, a base model with 100 trees in the forest, similar to the RFRM was employed, with 30 stages of adaptive boosting and an (empirically-derived) learning rate of 0.02. As expected, with the features used, the overall prediction accuracy decreased as the temporal range of prediction (N) was increased.

3.4. LTP: The Long-term Workload Predictor

When the system load drops, instead of idling servers, DCP powers off servers to avoid the overall low utilization of the remaining online servers and to eliminate power wastage. However, these shutdowns are done conservatively as premature server shutdowns can jeopardize service times by not leaving enough server capacity when the load goes up again. DCP uses **LTP** to reach server shutdown decisions more conservatively, looking at more consistent and relatively stable utilization trends and shutting off servers only when the utilization is consistently predicted to be lower over several consecutive predictions made at the end periodic scheduling intervals.

For powering servers down, DCP estimates the minimum number of servers (N_{min}) needed in each virtual group to maintain an overall system utilization at U_{max} and to deal with the load projected for C scheduling cycles into the future based on short term load prediction. To estimate the minimum servers needed for each virtual group, DCP first estimates the exponential moving average of the virtual group's load (U_{Load}). Once U_{Load} is estimated using U_{grp} (utilization of the virtual group) , N_{min} is determined based on the number of active servers (AS) currently serving requests:

$$U_{Load}\ (t) = \alpha * U_{grp} + (1 - \alpha) * U_{Load}\ (t\text{-}1)$$

$$N_{min}\ (t) = (U_{Load} * AS) / U_{max}$$

The use of the exponential moving average effectively filters out the impact of sudden changes in the workload trends. DCP maintains the estimated minimum server count at any time to avoid performance loss in case of a decreasing load trend reverse. Figure 2 depicts how the STP and LTP are incorporated into DCP and ACP.

3.5. Wear Leveling and Temperature Limiting

To reduce the wear and tear on the servers due to power cycling, DCP employs a timeout strategy, where recently offlined servers that have completed their assigned requests need to spend at least 180 seconds in the *free_list* before going through another power on/power off cycle. An exception to this timeout period is made only when there is

an immediate need for the servers that cannot be satisfied with deployments from other server lists (Sec. 3.6).

3.6. Dynamic Server Management

The dynamic server capacity management technique of DCP, summarized in Figure 3, is as follows.

A statistics collection daemon (**SCD**) runs in each turned-on server (at the user level) and collects the utilization, frequency, temperature, other critical information and sends it to a collector submodule within DCP. The collector submodule aggregates the data from all the powered-on servers in the data center and stores them in a RAM-fs (RAM file system). DCP gathers the required information from the collector submodule at the end of regularly-spaced scheduling intervals (of 500 milliseconds), computes the instantaneous system utilization and uses STP to predict the workload change expected. DCP then estimates the right number of servers needed for processing the predicted workload and adjusts the number of online servers dynamically at an interval of half a second to match the expected workload.

As mentioned in Sec. 3.1 and 3.2, servers are maintained in different virtual groups based on the server model and configuration. Within each virtual group, servers are dynamically placed in five unique lists based on the server state. The first list contains already turned-on servers that are online and serving requests (**active list**). The second is the list of servers that are recently offlined when the workload level dropped and are still serving previously assigned requests (**free list**). The third is the group of servers that are turning on to serve the expected increase in the workload (**waking_up list**). The fourth list contains group of servers that are powered off (**turned_off list**). The final list is the group of servers that are always kept on in reserve, to deal with rapid increases in the workload and not serving any requests unless they are activated (**hotspare list**).

On an increasing load trend, the prediction made from STP is linearly extrapolated to determine the time, t_{max}, when the utilization of the deployed servers is going to exceed ($U_{max} - \Delta$). If the difference, C, between the current time and t_{max} is higher than the time it takes to turn on a server (T_{ON}, expressed in terms of the number of scheduling intervals), a situation called "slow load growth", in which case the function **slow_load_growth( )** is invoked to turn on additional servers in the background to deal with the expected load increase. If C is less than the time it takes to turn on servers, the situation is called "fast load growth" and the function **fast_load_growth( )** is invoked. If the servers in the *free list* and *waking_up list* cannot serve the expected workload, additional servers are deployed from *hotspare list* in both the cases. If the workload is predicted

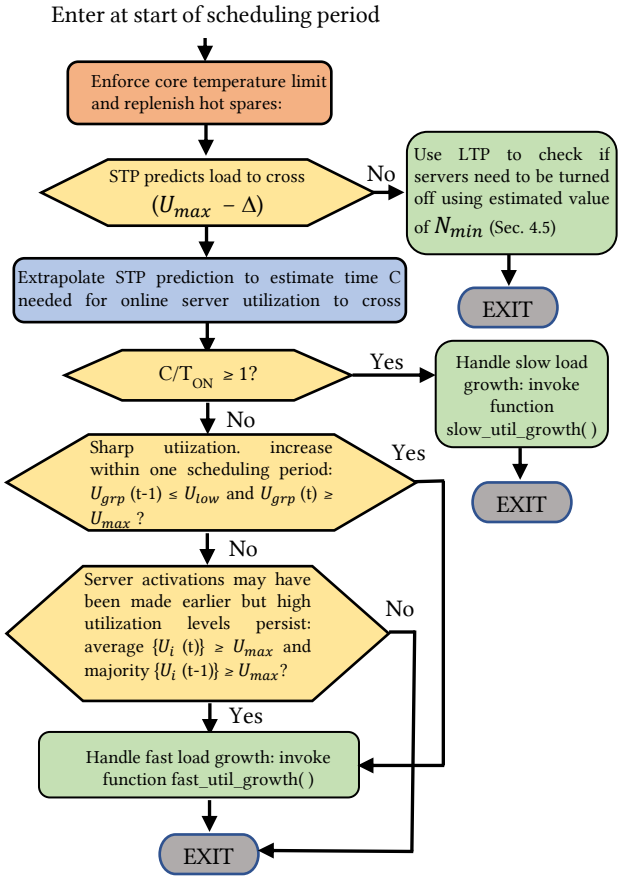


Figure 3: Flowchart for DCP

to increase even after using all the servers from *free, hotspare and waking_up lists*, then servers from the *turned_off list* are powered on. When servers from the *hotspare* list are deployed, they are replaced, when possible, with servers from the *turned_off list.* The function **fast_load_growth( )** also looks at the extent to which the fast load growth is taking place (based on the evaluated ratio of C/T_{ON}) and uses a preset table (**activation table**) to look up the number of servers that need to be activated simultaneously (called a **bundle**) for handling the load growth. For lower values of C/T_{ON}, more servers need to be activated compared to a situation when C/T_{ON} is closer to unity.

The process for turning off servers, follows a more conservative approach and uses LTP to look at utilization average over longer intervals in the past to eliminate the impact of utilization spikes or sharp utilization drops that occurred for a short duration. Servers can be powered down, when the system utilization is expected not to cross U_{max} by turning off a group of servers. This scenario is called "shrinking utilization". A minimum count of powered-on servers, N_{min}, is always maintained to accommodate utilization increases below the utilization level of U_{max}.

3.7. System Tuning

System tuning is essential to guarantee stringent service latencies and is typically undertaken by all data center operators – DCP is no exception to this. The parameter U_{max} is chosen to limit the utilization of servers while guaranteeing the service latencies. This threshold should be as high as possible, but not high enough to jeopardize service latencies. U_{max} is chosen by running exemplar instances of the application, as is typically done by data center operators as part of configuration tuning to guarantee average and tail request latencies. The parameters K and α, used, respectively, by STP and LTP are chosen by the operator based on how tight average and tail latency guarantees have to be. With a lower value of K, the predicted utilization only accentuates the impact of recent bursty utilization, so K should be moderately high. Choosing K beyond a specific value does not help as recent bursts in utilization are de-emphasized in the prediction. Similar comments apply to the choice of α. The exact values are again determined in the tuning process. Finally, the contents of the activation table, including the number of entries and the value for each entry determines how aggressively DCP handles fast utilization growths. These are also determined in the system tuning process.

Again, we note that the system tuning effort is not exclusive to DCP and instances of such parameters and the need to tune them appear in all similar techniques [14, 15, 20, 37], many often buried in parameters used in machine learning artifacts and elsewhere. In virtualized systems, similar tuning efforts are also needed to determine the amount of RAM and logical CPUs to be allocated for maintaining service quality. Thus, DCP is not unique in any way for relying on the system tuning used in data centers that guarantee service quality.

4. ACP: Dynamic Cooling Provisioning

We focus on traditional air-cooled systems and assume that the data center is organized as alternating cold and hot aisles, with two adjacent rows of front-facing servers making up a cold aisle. Chilled air is supplied to the servers by a Computer Room Air Handler (CRAH) via an underfloor plenum and perforated floor tiles in the cold aisles. The chilled air passing through the servers picks up the heat generated in the server, comes into the hot aisle at the rear of the servers and is drawn back into the CRAH and re-cooled before it re-enters the supply plenum. Further, the cold aisle is *contained*, that is fully enclosed, to prevent hot air from recirculating back into the servers from the hot aisles before being cooled by the CRAH and avoid any consequential inefficiencies related to cooling. As is typical of many mid-sized air-cooled data centers, we assume that the CRAH uses chilled water from a physical facility and no provision exists for adjusting the supplied chilled water temperature. Of course, further efficiencies are to be gained by having a dedicated chiller that enables the chilled water supply to be adjustable leading to further energy efficiency improvements, as discussed in [24]. This is beyond the scope of our work, as our experimental facility lacks this feature.

The solution (Automatic Cooling Provisioner, ACP) also relies on the use of adjustable apertures floor tiles (**AAT**s), whose aperture (opening) can be adjusted remotely to control the airflow rate from the plenum into the contained cold aisles. Pressure sensors located within the cold aisle, outside the aisle and in the plenum are used to monitor the aisle's positive air pressure, **APAP** (which is obtained by subtracting the pressure space outside the contained aisle from the pressure inside the cold aisle). Similarly, we monitor the positive plenum air pressure, **PPAP**, into the cold aisle which is obtained by subtracting the cold aisle pressure from the plenum pressure. The aisle and plenum contain three pressure sensors each that are located at their ends and their middle. The air pressure in the space outside the aisle is averaged from two sensors for every aisle.

The two control knobs for adjusting the provisioned cooling in our solution are: 1) the opening in the AATs and 2) the speed of the blower fan (**VFD speed**, for Variable Frequency Drive speed) within the CRAH. Unnecessary high positive air pressure results in wasting electrical power in the CRAH blower, which is primarily responsible for maintaining the air pressure within the aisle. High positive air pressure also results in cold air escaping into the hot aisle via openings within the inactive servers. Keeping the positive air pressure too low can fail to provide adequate cooling and/or permit hot air to recirculate back into the cold aisle through the openings in the powered off servers. It is thus imperative to operate with just enough positive air pressure to improve the overall efficiency of the cooling system. AATs help in improving the energy efficiency further by opening apertures in front of the racks with active servers, so that most of the supplied air is drawn in by fans running in the active servers, with little escaping through the openings in the turned off servers.

The basic tenet for the control system to adjust the amount of cooling based on utilization prediction is as follows:

- Maintain sufficient capacity of chilled air supply in the plenum to permit additional cooling to be quickly provided by opening the tile aperture, well before the CRAH can react to the increase in demand when servers are onlined on a sudden workload increase. Simultaneously, the CRAH's VFD speed has to be increased in the background to maintain the air supply in the plenum. The PPAP is indicative of the amount of chilled air available in the plenum.

- At all times, a sufficient APAP must be maintained in the cold aisle to prevent hot air exiting the back of the servers to recirculate into the cold aisle via turned-off servers whose fans are also turned off. The APAP is a function of the PPAP and the tile openings.

ACP is ***pro-active*** in adjusting the required cooling based on the prediction made by STP (Fig. 2) for servers in the aisle and the sensed aisle and plenum pressures. Based on these, ACP determines the settings of the adjustable aperture tiles and the CRAH VFD speed using a DNN regression model. Although, a simpler neural network model could have sufficed for the current version of ACP, a proactive forward-looking design using a DNN was used to enable future capabilities for addressing physical configuration changes, transient hot spots, containment leaks and fan failures.

The DNN-regression model used by ACP has 5 layers with 3 hidden layers each using the Relu activation function and the Adam optimizer. The model outputs the necessary aperture settings for the AATs and the VFD speed to meet the two requirements above.

As the offered workload increases, DCP powers on a bundle of servers which changes the aisle and plenum pressures. As a result, if the pressure in the aisle or plenum falls below the required pressure levels as more fans are drawing in the cold air to cool different components of the server, the cooling provided is adjusted by either increasing AATs openness or by increasing CRAH blower fan speed using VFD. Similarly, when DCP powers off idle servers, aisle pressure (APAP) increases as the same level of previously provisioned cooling is not needed. Therefore, ACP decreases the CRAH blower fan speed and/or the AATs aperture to reduce the chilled air supply into the plenum.

To train the DNN, a reactive approach was used to increase the workload in steps and at each step determine the AAT opening and the minimum VFD speed needed to meet the cooling needs to maintain the APAP and PPAP. We emphasize the need to maintain the minimum necessary VFD speed as the power consumption of the CRAH's VFD increases with the VFD speed albeit in a non-linear fashion.

ACP improves its efficacy by relying on a systematic order used by DCP for activating and deactivating the servers. DCP uses a systematic activation and deactivation pattern for aisles, racks within an aisle and servers within a rack. Aisles may be activated one after another or together (to spread out the heat utilization across the aisles). Within an aisle, server activations take place in a rack that has already got some active servers. A turned off rack is deployed when all servers within the active racks are deployed. Server activations within a rack takes place in a consistent vertical direction – either from the bottom of the rack to the top or from the top to the bottom. Server turnoffs or idling takes place in the opposite order. *An idle rack, when activated, toggles between these two activation orders to wear level servers from thermal stresses caused by activation and deactivation.* Finally, servers in racks facing each other within an aisle are activated as symmetrically as possible across the aisle to present symmetric cooling needs across the opposing racks. The systematic server activation and deactivation permits the amount of cooling needed to be estimated in advance as a function of the number and type of active servers based on their utilization. ACP is trained using the data generated by the reactive approach which consists of 62,247 rows of dataset split into 80% for training and 20% for validation. The loss and mean squared error of the trained model are 0.07 and 0.009, respectively.

ACP is currently limited to data centers that use chilled air cooling and requires the use of variable aperture tile and pressure sensors. It can be extended to systems that use direct-contact water cooling via coldplates that replace traditional CPU heatsinks and still rely on chilled air cooling for other server components.

5. Experimental Assessment

For the assessments of HCP, we used 247 heterogeneous severs which are classified into 4 virtual groups: 72 identical Dell PowerEdge R530, 57 identical Dell PowerEdge R520, 74 HPE ProLiant DL360 G9 and 45 HPE ProLiant DL380G8 servers distributed across 14 racks within a single contained, cold aisle. Servers run the Ubuntu 16.04.2 LTS and Linux Kernel 4.4.0-130. Other servers are used to carry out various infrastructure related work in our experimental setup and are connected to an independent power delivery chain, to isolate them from the services performed on the heterogeneous server pool. All of HCP runs on a single low-end server.

A F5 Networks BIG-IP 5000s LTM programmable load-balancer switch is used for load balancing at the front end. Its active target server list is adjusted dynamically by DCP to consolidate the incoming work onto the fewest number of online servers. The networking equipment consists of 2 Cisco Nexus 5672, 6 Cisco Nexus 3064 and 10 Cisco Nexus 2248 TP switches; their power consumption is not considered in this study, as a major portion of these switches also support other aisles/parts of the facility that contain the servers used here. Power consumption data of the servers is collected from a pair of Server Tech 24V2C415A1 PDUs within each rack via polling through a per-PDU SNMP interface.

A Leibert CRAH (Model CW114DC1A2B667) provides the chilled air via an underfloor plenum to the IT equipment in

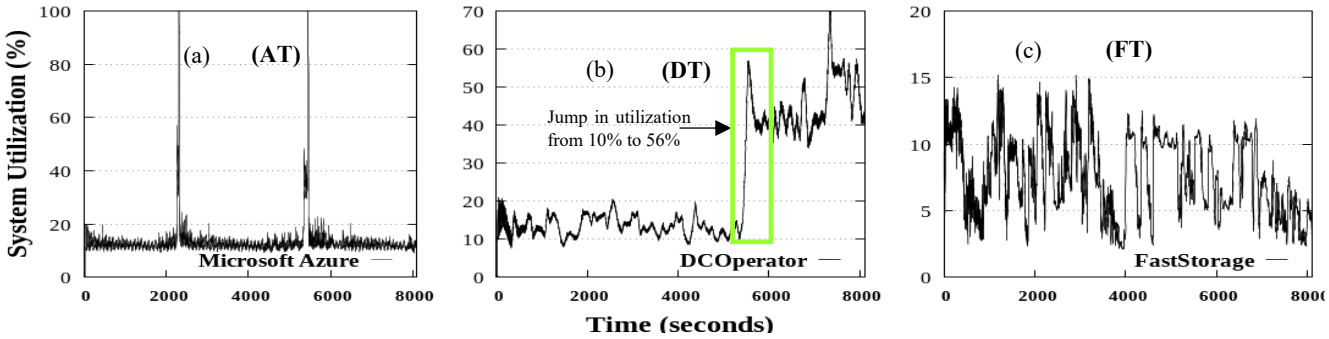


Figure 4: Load Profile of three real-world traces used for evaluation

the aisle. The adjustable aperture tiles are located in the floor area between opposing rows of servers within the cold aisle. The CRAH status and its blower fan speed are monitored through Siemens Insight system which allows changes to CRAH VFD. Multiple zone pressure sensors [3] are used and read via the facilities of a DAQ [23].

5.1. Tunable Parameters

Although the parameters mentioned in Section 3.6 can be tuned to every specific application, we used a fixed set of values across all our assessments (except for U_{max}, which is tuned for each application and each server type) and these are: **K=9, α=0.45, N=10** and Δ = **10% of** U_{max}. No significant gains were realized by fine tuning these parameters to individual applications. Thus, deploying HCP requires only a one-time calibration for determining the U_{max} value of individual pairs of server type and an application. The bundle size (Sec. 3.7) for turning on servers on utilization increases was fixed at 4 (instead of varying the bundle size based on the slope of the load growth). Finally, the number of hot spares in each virtual group was set at 1/8th of the total number of servers in the virtual group. The values of these last two parameters are empirical and determine how aggressively HCP must deal with very sharp load spikes. A higher value of Δ enables early detection of the load approaching U_{max} and permits DCP to turn on servers beforehand. Using higher Δ and bundle size results in reduced impact on the 99th-percentile request latency but this reduces the energy savings.

5.2. Utilization Patterns and Requests

Three real-world ***utilization patterns*** (traces) are used for evaluating HCP and their temporal profiles are shown in Figure 4. The pattern shown in Fig. 4(a) comes from Microsoft Azure's cloud computing platform [41] and includes different metrics collected from roughly 2.6 M VMs. A detailed analysis of this utilization trace, hereafter referred to as Azure trace (**AT**), appears in [14]. The second utilization pattern, Fig. 4(b), was obtained from a data center operator (who prefers to remain anonymous) and is hereafter referred to as **DT**. DT represents data from an online transaction system. The dataset from 1,750 VMs of Bitbrains [7] forms the final utilization trace, fastStorage (**FT**) [38]. The data of the Azure and fastStorage traces, which were collected at intervals of 300 seconds, is not suitable for the direct use in our evaluation. This is because our solution is designed to handle workload level variations at finer sub-second granularity. Because of this, in our evaluations, *we scale these two patterns temporally to place the original data points at the end* intervals of 0.3 seconds to match our evaluation requirements. This allows for the requests generated to change at higher frequency and does not keep the utilization of the system constant for each 300 second interval, as the original data implied. These traces all have bursty regions that enable the assessment of HCP's guarantee for limiting the 99th -percentile tail latency.

The actual requests serviced using these utilization patterns have average service times (as experienced by clients) that range from very short (around 120 milliSeconds) to relatively long (around 11 Seconds) and are summarized in Table 1, which shows the approximate average service time on a fully-provisioned system across the three traces. These requests are considered to be representative of typical online requests of different classes. To generate requests based on the utilization patterns of Fig. 4, a harness tool was developed for generating a stream of request following the utilization patterns and to permit each sequence of requests to be replayed consistently for an apples-to-apples comparison of different variants of capacity provisioners. The harness tool issues and

Benchmark Name/Abbreviation	Description/Average Service Time
SPECjvm Crypto.rsa/***Rsa***	Encrypts/decrypts input dataset/~0.74 secs.
SPECjvm Serial/***Serial***	Producer-consumer processing using sockets/~5 secs.
SPECjvm XmlV/***Xmlv***	Compression and validation of the XML.xsd files/~11 secs.
Memcached/***Mem***	Query processing request for wiki database with thousands of concurrent requests/~120 milliseconds

Table 1: Description of benchmark requests used

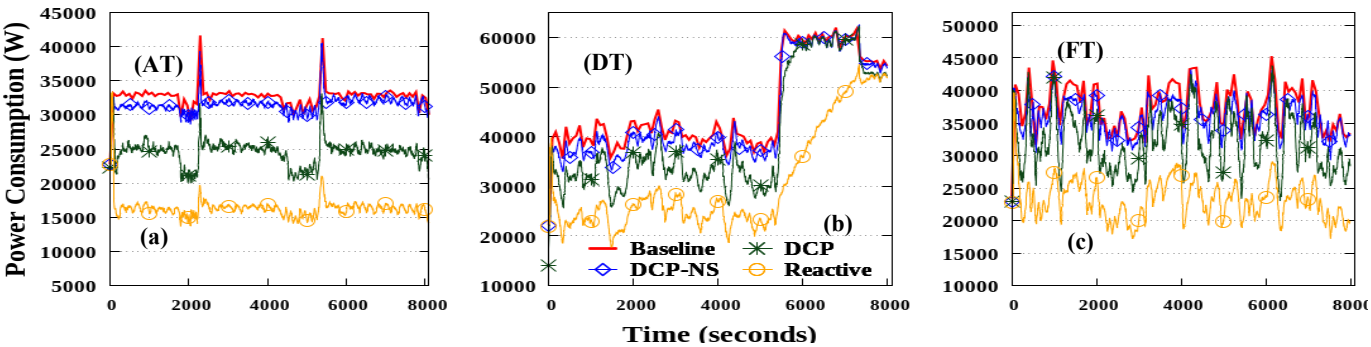


Figure 5: Power consumption for the utilization traces for Baseline, Reactive, DCP-NS and DCP with *Serial* requests

measures the latency of each HTTP GET request that triggers the execution of the actual requests from a client to the servers - from the point just prior to the opening of the socket for sending the request to the closure of the socket following the delivery of the response.

5.3. Evaluated Provisioner

Experimental comparisons comparing other techniques were not possible, as some of their critical implementation details are unavailable, a fact that is true for most complex systems of this nature. Instead, two DCP variants are used in the assessments. One is DCP, as presented earlier in Sec. 3, where idling offline servers are powered off to save energy and the other is, **DCP-NS (No Shutdown)**, where idle offline servers are *not* powered down. DCP-NS has the advantage that offline idling servers can be quickly activated to handle utilization increases. However, by idling unused servers, DCP-NS misses a significant opportunity to enhance the energy savings. Two other provisioners were also compared. One is **Baseline**, representative of the prevalent practice of keeping all servers on and where requests are dispensed among all servers based on least connections. The second of the other provisioner used against DCP is called ***Reactive***, where no utilization prediction is used, and servers are powered on or off in response to the current observed system utilization. *Reactive* is implemented by disabling the predictors in DCP.

5.4. Power Reductions and Energy Savings

The variations in the total power consumption for all variants is presented in Fig. 5 for the *Serial* requests. As expected, Baseline has the highest power consumption and is closely followed by DCP-NS since it does not power off unused idle servers. Even though all host servers are configured to run in "powersave" mode, DCP-NS has a power consumption pattern very close to Baseline demonstrating that idling at a lower DVFS setting is not enough to deliver considerable energy savings, as idle power consumption of a server is a significant fraction of its peak power consumption.

Fig. 5 shows that across all three utilization traces, during low utilization periods, DCP's power consumption is less than that of Baseline and DCP-NS. During high utilization periods, since DCP activates all servers, the power consumption matches that of the others. For the Azure trace (AT) the two power consumption peaks in Fig. 5(a) coincide with the utilization peaks seen in Fig. 4(a). During the period when the abrupt workload spikes occur, DCP is able to adjust the online server capacity in advance so as not to impact the 99th-percentile tail latencies, as will be seen later. When the short-term workload predictor senses the onset of the spike and predicts an increase in the utilization level, DCP mostly activates servers from the *free list* for AT. Sometimes, DCP also powers on additional servers to handle the load growth, leading to an increased power consumption. Power consumption trends for other request classes are similar (see also Fig. 6(a)).

The DT trace evaluates if DCP can handle a sudden and sustained transition from low utilization to high utilization. This is seen in Fig. 5 (b), where during the time when system utilization increases from ~10% to ~56% (as highlighted in Fig. 4 (b)), DCP rapidly turns on all available servers, and even make use of hot spares, to handle the sustained growth predictions made by STP over several consecutive scheduling intervals. For DT, DCP's power consumption approaches that of the Baseline at high utilization levels.

The FT (fastStorage) trace evaluates DCP's ability to handle rapid short-lived workload fluctuations which are not necessarily abrupt in nature. Fig. 5 (c) shows that DCP is able to match the server capacity to the utilization changes, shown separately in Fig. 4(c). Reactive has the lowest power consumption of all the variants because it powers on and off the servers based on the current utilization.

The energy consumption of the servers for all four applications for each of the three utilization traces is shown in Fig. 6(a). Of all the three DCP variants, DCP-NS has the least average energy savings of 4.14%, 2.96% and 3.66% for AT, DT and FT traces, respectively. Hence, relying solely

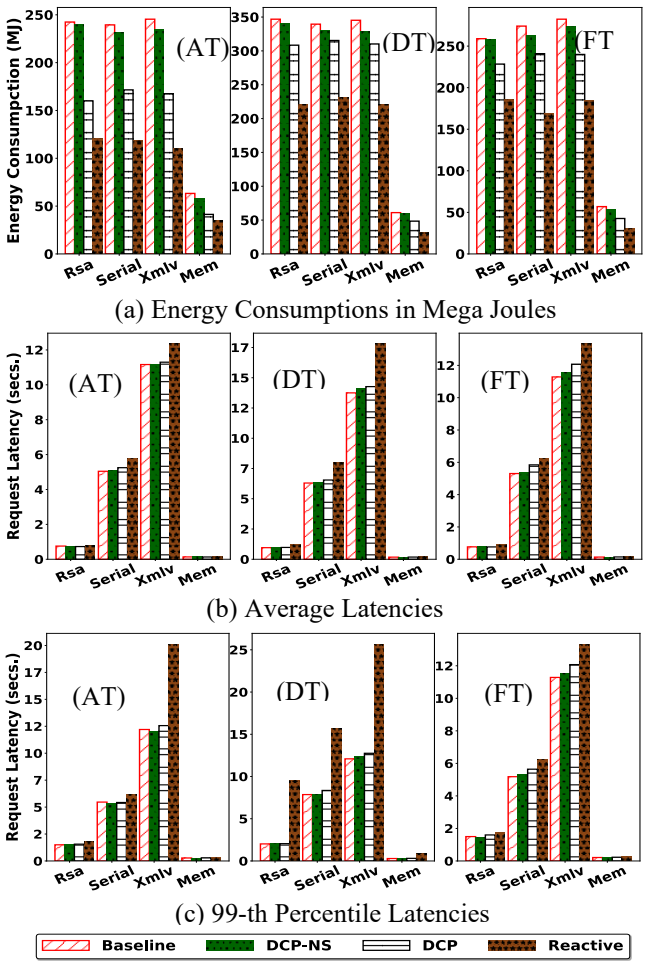


Figure 6: Energy savings, average and 99th percentile

on DVFS to adjust the CPU core frequency during idling period is not enough for reducing the energy expended.

From the power consumption levels of DCP shown in Fig. 5, we can establish that load traces dominated by the low utilization periods will have the highest energy savings with DCP. This is confirmed in Fig. 6(a), where DCP's average energy savings is the highest for the AT trace among the three utilization traces since AT's average utilization of 11.24% is less than that of DT (38.15%) and FT (17.39%). DCP's average energy savings across all applications for each trace are 32.18%, 12.2%, 15.98% for the AT, DT and FT traces, respectively, and these are roughly inversely proportional to the average utilization of the load traces. The average energy savings for Reactive are 50.8%, 38.12%, 36.4%. These higher energy savings come at the cost of service quality to the end user (higher latencies) as seen in Figs. 6(b) and 6(c). DCP consciously tries to avoid such service latency increases.

5.5. Impact on Service Latencies

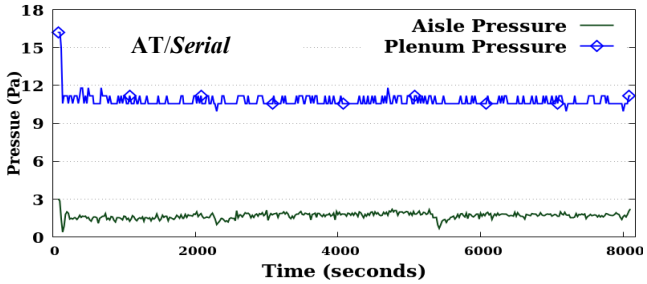
Even though DCP powers off idle servers for significant energy savings, its use of short-term and long-term utilization predictors limit the impact on the request latencies effectively. The mean and 99th-percentile request latencies for the four request classes for each of the three traces are shown in Figures 6 (b) and 6 (c). DCP-NS has negligible impact on request latency when compared to Baseline, since DCP-NS sacrifices energy savings for performance by keeping all the servers online and idling. The highest increase in the mean and 99th-percentile request latencies across the AT, DT and FT for DCP-NS are: (a) 0.08% (mean) and 0.07% (99-th %) for *Memcached*; (b) 0.6% and 0.5% for *Rsa*; (c) 0.9% and 0.27% for *Serial*, and (d) 1.7% and 1.78% for *Xmlv*. For latency-critical applications, DCP-NS can be used to realize a small energy savings (of up to 4.14%) with negligible impact on the request latencies.

In contrast, even though DCP powers off idle servers for significant energy savings, DCP's impact on the request latency is very limited. The ***highest increase*** in the mean and 99th-percentile request latencies, respectively, across the three utilization traces are: (a) 2.6% and 5.3% for *Memcached*; (b) 1.5% and 4.65% for *Rsa*; (c) 3.24% and 5.02% for *Serial*, and (d) 2.02% and 6.67% for *Xmlv*. Thus, for a load trace like AT, DCP can deliver significant IT energy savings of 29.3% with very little impact on mean request latency and less than 6.67% increase for the 99-th percentile request latency compared to a fully provisioned baseline system. Note that the 6.67% increase is the **worst-case increase across ALL individual requests** within the observed periods and does represent the average, which is considerably lower (about 4.1%). This demonstrates that DCP can handle any class of service whose request latency ranges from milliseconds to tens of seconds with acceptably low increases in service latencies compared to the Baseline while realizing significant energy savings.

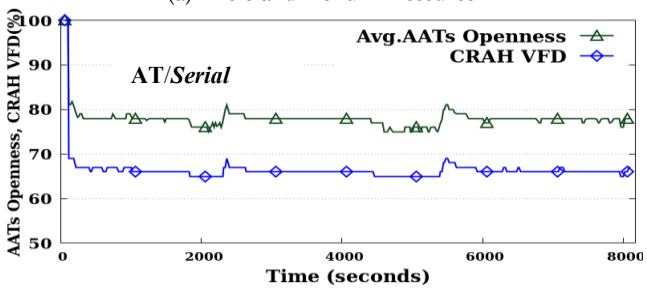
Reactive's impact on the mean and 99-th percentile request latencies, respectively, is very high compared to the Baseline: (a) 16.8% and 81.97% for *Memcached*; (b) 17.5% and 136.81% for *Rsa*; (c) 19.28% and 44.26% for *Serial*, and (d) 19.36% and 65.07% for *Xmlv*. Reactive's request latencies suffer the most, violating the service commitments severely because utilization of active servers has to often exceed U_{max} before Reactive can deploy additional servers.

5.6. Savings on Cooling

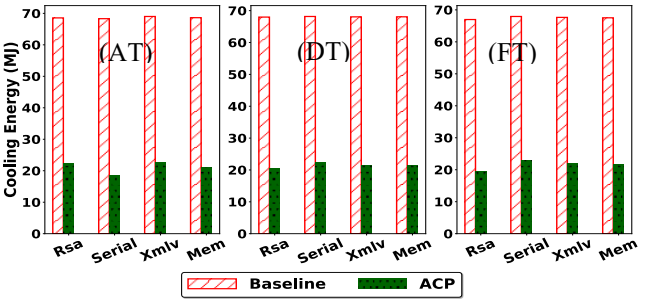
We now show how ACP, the cooling capacity provisioner, proactively matches the cooling to the IT utilization. ACP is compared against a baseline system that uses setpoint control to maintain the cold aisle temperature. With ACP, the lower and upper aisle pressure thresholds used in for evaluation are 1 and 1.6 Pa (Pa = Pascals), and plenum thresholds are 8.5 Pa and 12 Pa. Fig. 7(a) shows that ACP maintains the average cold aisle and plenum pressures within these thresholds without either over provisioning or



(a) Aisle and Plenum Pressures



(b) Avg. Zone Openings Across All AATs



(c) Total CRAH Energy Savings

Figure 7: Pressures, AATs, CRAH and Cooling Savings

under provisioning for the AT. ACP maintains an average cold aisle pressure of 1.29 Pa while Baseline's cold aisle pressure averages at 3.11 Pa. ACP's average plenum pressure is 9.8 Pa against Baseline's 16.36 Pa. Figure 7(b) shows the average openness of all the adjustable aperture tiles AATs and CRAH VFD speed for the AT. When the system utilization spikes for a brief period (Fig. 4(a)), ACP uses the predicted increase in utilization to adjust the AATs and CRAH VFD to match the cooling to the IT capacity.

The energy consumption of the CRAH for Baseline and ACP-P is seen in Fig. 7(c). The average energy consumption of CRAH at full blower fan speed is 68.04 MJ (Mega Joules) and ACP is able to reduce this to 21.15 MJ, which represents a savings of 68.92%. For AT, DT and FT traces, the CRAH energy savings are 69.19%, 68.45% and 69.05%, respectively. For a data center operating 50 CRAH units and charged at 10 cents per KWh, then the average annual operational cost savings for the CRAHs is thus $281,758. Even if the Baseline CRAH blower fan speed is set at 75%, the energy consumption during an 8000-second long workload trace is 40.32 MJ and the energy savings against this new Baseline is 47.55%, which represents a savings of $115,508 in costs across 50 CRAHs.

5.7. Overall Energy Savings

Finally, we summarize the overall energy savings due to HCP stemming from dynamic IT and cooling capacity provisioning. Averaged across all four request classes within the three request utilization traces, HCP realizes:

- IT energy savings of 32.18% (AT), 12.2% (DT) and 15.98% (FT).
- Cooling energy savings of 69.19% (AT), 68.45% (DT), and 69.05% (FT).
- Total IT and cooling energy savings of 41.18% (AT), 17.54% (DT) and 22.2% (FT).

These savings are realized with very little increase in the average and 99-th percentile request latencies against a fully provisioned system. The observed cooling energy savings are exaggerated as the CRAH used in our studies can accommodate twice as many servers as used here. With a correspondingly scaled utilization, the cooling energy savings will be reduced to half of what is reported above (about 35%) as the CRAH air flow rate is almost linearly proportional to the blower power consumption. ACP also scales up with the number of aisles and servers using the server deployment/deactivation order of Sec.4.

6. Conclusions and Discussions

HCP dynamically provisions server and cooling capacities for chilled air-cooled datacenters to realize significant energy savings, with tight average and 99-th percentile guarantees against a fully provisioned baseline system. HCP also accommodates background and batch jobs, as it accounts for these by using the system utilization to drive request scheduling; results were excluded for brevity. The tight resource provisioning in HCP requires failures to be handled. Failure mitigation is beyond the scope of this paper, but some solutions are outlined below. For aisles with independent CRAHs and plenum partitioning, requests that would have targeted a failed aisles are redirected to other aisles with spare capacity. For CRAH failures with a shared plenum, a higher plenum pressure providing reserve cooling is used to allow for backup CRAHs to kick-in. Server failures with critical services require requests replication, as done in solution providing hard 24/7 service guarantees. Conscious tracking of server health can avoid such service losses.

ACKNOWLEDGEMENTS AND NOTES

This work is supported in part by NSF awards 1738793, 1134867, 1040666, a SUNY 2020 award and the Center for Energy-Smart Electronic Systems at SUNY-Binghamton and its industry members. HCP concepts are protected through issued and pending patents.

REFERENCES

[1] Anshul Jindal. Dimensionality Reduction using PCA on Multivariate Timeseries Data. https://medium.com/@ansjin/dimensionality-reduction-using-pca-on-multivariate-timeseries-data-b5cc07238dc4

[2] Ayan Banerjee, Tridib Mukherjee, Georgios Varsamopoulos, and Sandeep K. S. Gupta. Cooling-aware and thermal-aware workload placement for green hpc data centers. In Green Computing Conference, 2010 International (Aug 2010), pp. 245–256.

[3] Bapi Zone Pressure Touch Differential Pressure Transmitters: https://www.bapihvac.com/product/zone-pressure-touch-zpt-differential-pressure-transmitter-field-selected-range-and-output/

[4] Cullen E. Bash, Chandrakant D. Patel and Ratnest K. Sharma. "Dynamic thermal management of air cooled data centers," Thermal and Thermomechanical Proceedings 10th Intersociety Conference on Phenomena in Electronics Systems, 2006. ITHERM 2006., San Diego, CA, 2006, pp. 8 pp.-452, doi: 10.1109/ITHERM.2006.1645377.

[5] Antonio Barbalace, Robert Lyerly, Christopher Jelesnianski, Anthony Carno, Ho-Ren Chuang, Vincent Legout, and Binoy Ravindran. 2017. Breaking the Boundaries in Heterogeneous-ISA Datacenters. SIGARCH Comput. Archit. News 45, 1 (March 2017), 645–659. DOI:https://doi.org/10.1145/3093337.3037738

[6] Alex Benik, and Battery Ventures. The sorry state of server utilization and the impending post-hypervisor era, November 2013. [Online; posted 30-November-2013].

[7] Bitbrains. http://www.bitbrains.nl/solvinity-en.

[8] L.Breiman.Randomforests.MachineLearning, 45(1):5–32, 2001.18

[9] Shane Case, Furat Afram, Erdem Aktas, and Kanad Ghose. Energyaware load direction for servers: A feasibility study. In Proc. PDP 2012, pp. 359–367.

[10] Sara Casolari, and Michele Colajanni. Short-term prediction models for server management in internet-based contexts. Decis. Support Syst. 48, 1 (dec 2009), 212–223.

[11] Julie Chao. Data Centers Continue Proliferate Energy Use Plateaus. https://newscenter.lbl.gov/2016/06/27/data-centers-continue-proliferate-energy-use-plateaus/

[12] Yuan Chen et al., "Integrated management of application performance, power and cooling in data centers." 2010 IEEE Network Operations and Management Symposium - NOMS 2010, Osaka, 2010, pp. 615-622, doi: 10.1109/NOMS.2010.5488433.

[13] Miyuru Dayarathna, Yonggang Wen, and Rui Fan. Data center energy consumption modeling: A survey. In *IEEE Commununications Surveys and Tutorials.* , vol. 18, no. 1, pp. 732– 794, First Quart. 2015.

[14] Christina Delimitrou, and Christos Kozyrakis. Paragon: QoS-aware scheduling for heterogeneous datacenters. Proc. ASPLOS 2013, pp. 77-88.

[15] Sheng Di, Derrick Kondo, and Walfredo Cirne. Host load prediction in a google compute cloud with a bayesian model. In High Performance Computing, Networking, Storage and Analysis (SC), 2012 International Conference for (Nov 2012), pp. 1–11.

[16] Jeffrey Dean, Luiz Andre Barroso. The tail at Scale. https://cacm.acm.org/magazines/2013/2/160173-the-tail-at-scale/fulltext

[17] Katie Fehrenbacher. Data Centers Are No Longer The Energy Hogs They Once Were. https://fortune.com/2016/06/27/data-center-energy-report/

[18] Yoav Freund, Robert E. Schapire. A Short Introduction to Boosting. In Journal of Japanese Society for Artificial Intelligence, 14(5):771-780, September, 1999.

[19] Anshul Gandhi, Mor Harchol-Balter, Ram Raghunathan, and Michael Alan Kozuch. Autoscale: Dynamic, robust capacity management for multi-tier data centers. ACM Trans. Comput. Syst. 30, 4 (Nov. 2012), 14:1–14:26.

[20] James Glanz. Power, pollution and the internet. New York Times, September 22, 2012.

[21] IBM Tivoli:

https://www.ibm.com/support/knowledgecenter/en/SSRULV_8.6.0/com.ibm.tivoli.itws.doc_8.6/eqqg1mst82.htm

[22] Karthik Kambatla, Vamsee Yarlagadda, Inigo Goiri, and Ananth Grama. UBIS: Utilization-aware cluster scheduling. In Proc. IPDPS 2018, pp. 358-267.

[23] Keysight Technologies' 34980A: https://www.keysight.com/en/pc-1000003024%3Aepsg%3Apgr/34980a-multifunction-switch-measure-mainframe-and-modules?cc=US&lc=eng

[24] Sadegh Khalili, Ghazal Mohsenian, Anuroop Desu, Kanad Ghose, & Bhagat Sammakia. (2019). Airflow Management Using Active Air Dampers in Presence of a Dynamic Workload in Data Centers. *SEMI-THERM 2019*.

[25] Yanpie Liu, Stark C. Draper, and Nam Sung Kim. Sleepscale: Runtime joint speed scaling and sleep states management for power efficient data centers. SIGARCH Comput. Archit. News 42, 3 (June 2014), 313–324.

[26] David Lo, Liqun Cheng, Rama Govindaraju, Luiz Andre Barroso, and Christos Kozyrakis. Towards energy proportionality for largescale latency-critical workloads. SIGARCH Comput. Archit. News 42, 3 (June 2014), 301–312.

[27] Microsoft Azure worklaod traces: https://github.com/Azure/AzurePublicDataset

[28] Justin Moore, Jeff Chase, Parthasarathy Ranganathan, and Ratnesh Sharma. Making scheduling "cool": Temperature-aware workload placement in data centers. In Proceedings of the Annual Conference on USENIX Annual Technical Conference (Berkeley, CA, USA, 2005), ATEC '05, USENIX Association, pp. 5–5.

[29] Koyel Mukherjee, Samir Khuller, and Amol Deshpande. Saving on cooling: The thermal scheduling problem. SIGMETRICS Perform. Eval. Rev. 40, 1 (June 2012), 397–398.

[30] Natural Resources Defense Council (NRDC), Data Center Efficiency Assessment, Issue Paper IP:14-08-A, June 2014.

[31] Seyed M. M. Nejad, Ghada Badawy and Douglas G. Down. "EAWA: Energy-Aware Workload Assignment in Data Centers," 2018 International Conference on High Performance Computing &

Simulation (HPCS), Orleans, 2018, pp. 260-267, doi: 10.1109/HPCS.2018.00053.

[32] Eduardo Pinheiro, Ricardo Bianchini, Enrique V. Carrera, and Taliver Heath. Compilers and operating systems for low power. Kluwer Academic Publishers, Norwell, MA, USA, 2003, ch. Dynamic Cluster Reconfiguration for Power and Performance, pp. 75–93.

[33] John J. Prevost, KranthiManoj Nagothu, Brain Kelley, and Mo Jamshidi. Prediction of cloud data center networks loads using stochastic and neural models. In Proc. System of Systems Engineering (SoSE), 2011, pp. 276–281.

[34] Deepak Rajan, and Philip S. Yu. Temperature-aware scheduling: When is system-throttling good enough? In Web-Age Information Management, 2008. WAIM '08. The Ninth International Conference on (July 2008), pp. 397–404.

[35] Luiz Ramos, and Ricarod Bianchini. C-oracle: Predictive thermal management for data centers. In High Performance Computer Architecture, 2008. HPCA 2008. IEEE 14th International Symposium on (Feb 2008), pp. 111–122.

[36] Raritan Inc., The New Truth in Data Center Management, white paper, 2016, available at: https://www.raritan.com/assets/ram/resources/white_papers/raritan-wp-New_Truth_in_DCIM.pdf

[37] Mohammad Shahrad, Rodrigo Fonseca, Íñigo Goiri, Gohar Chaudhry,Paul Batum, Jason Cooke, Eduardo Laureano, Colby Tresness, MarkRussinovich, and Ricardo Bianchini. 2020. Serverless in the Wild:Characterizing and Optimizing the Serverless Workload at a LargeCloud Provider.

[38] Arman Shehabi, Sarah Smith, Dale Sartor, Richard Brown, Magnus Herrlin, Jonathan Koomey, Eric Masanet, Nathaniel Horner, Ines Azevedo, and William Lintner. "United States Data Center Energy Usage Report", Lawrence Berkeley National Lab Report No. LBNL-1005775, June 2016.

[39] Siqi Shen, Vincent Van Beek, and Alexandru Iosup. Statistical characterization ofbusiness-critical workloads hosted in cloud datacenters. In *Proc.of IEEE/ACM International Symposium on Cluster, Cloud andGrid Computing (CCGrid)*, pages 465 – 474, Shenzhen, China,May 2015.

[40] SPEC.org.,SPECjvm2008, https://www.spec.org/jvm2008.

[41] Sunbird Software Inc., DCIM Product Suite information available at: https://www.sunbirddcim.com/dcim-products

[42] Anurina Tarafdar, Mukta Debnath, Sunirmal Khatua, Rajib K. Das. Energy and quality of service-aware virtual machine consolidation in a cloud data center. J. Supercomput. (2020). https://doi.org/10.1007/s11227-020-03203-3

[43] Lizhe Wang, Samee U. Khan, and Jai Dayal. Thermal aware workload placement with task-temperature profiles in a data center. The Journal of Supercomputing 61, 3 (2012), 780–803.

[44] Qiang Wu. Making facebooks software infrastructure more energy efficient with autoscale, August 2014. [Online; posted 8-August-2014].

[45] 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. *SIGARCH Comput. Archit. News* 44, 3 (June 2016), 469–480. DOI: https://doi.org/10.1145/3007787.3001187

[46] Hong Xu, Chen Feng and Baochun Li. "Temperature Aware Workload Managementin Geo-Distributed Data Centers," in IEEE Transactions on Parallel and Distributed Systems, vol. 26, no. 6, pp. 1743-1753, 1 June 2015, doi: https://doi.org/10.1109/TPDS.2014.2325836.

[47] Rahul Yadav, Weizhe Zhang, Omprakash Kaiwartya, Prabhat Ranjan Sing, Ibrahim A. Elgendy and Yu-chu Tian, "Adaptive Energy-Aware Algorithms for Minimizing Energy Consumption and SLA Violation in Cloud Computing," in IEEE Access, vol. 6, pp. 55923-55936, 2018, doi: https://doi.org/10.1109/ACCESS.2018.2872750.

[48] Lian Zhou, Chih-Hsun Chou, Laxmi N. Bhuyan, K. K. Ramakrishnan, and Daniel Wong. Joint Server and Network Energy Saving in Data Centers for Latency-Sensitive Applications. In Proc. IPDPS 2018, pp. 700-709.

[49] Rongliang Zhou, Zhikui Wang, Alan McReynolds, Cullen E. Bash, Thomas W. Christian and Rocky Shih. "Optimization and control of cooling microgrids for data centers," 13th InterSociety Conference on Thermal and Thermomechanical Phenomena in Electronic Systems, San Diego, CA, 2012, pp. 338-343, doi: https://doi.org/10.1109/ITHERM.2012.6231449.

[50] Rongliang Zhou, Zhikui Wang, Cullen E. Bash, Alan McReynolds, Christopher Hoover, Rocky Shih, Niru Kumari, Ratnesh K. Sharma. "A holistic and optimal approach for data center cooling management." Proceedings of the 2011 American Control Conference, San Francisco, CA, 2011, pp. 1346-1351, doi: https://doi.org/10.1109/ACC.2011.5991575.

[51] Tian, X., "Cooling fan reliability: failure criteria, accelerated life testing, modeling and qualification", In Proc.. Annual Reliability and Maintainability Symp. 2006, pp. 380-384.

[52] Zhou Zhou, Jemal Abawajy, Morshed Chowdhury, Zhigang Hu, Keqin Li, Hongbing Cheng, Abdulhammed A. Alelaiwi, Fangmin Li. Minimizing SLA violation and power consumption in Cloud datacenters using adaptive energy-aware algorithms. Future Gener. Comput. Syst. 86,836–850.