On Demand Elastic Capacity Planning for Service Auto-Scaling

Pavel Chuprikov*†
*Steklov Mathematical Institute at St. Petersburg, Russia
†IMDEA Networks Insitute, Madrid, Spain

Sergey Nikolenko‡§
‡National Research University Higher School of Economics, St. Petersburg, Russia
§Steklov Mathematical Institute at St. Petersburg, Russia

Kirill Kogan
IMDEA Networks Institute, Madrid, Spain

***Abstract*—Cloud computing allows on demand elastic service scaling. The capability of a service to predict resource requirements for the next operational period defines how well it will exploit the elasticity of cloud computing in order to reduce operational costs. In this work, we consider a capacity planning process for service scale-out as an online pricing model. In particular, we study the impact of buffering service requests on revenues in various settings with allocation and maintenance costs. In addition, we analyze the incurred latency implied by buffering service requests. We believe that our insights will allow to significantly simplify predictions and mitigate the unknowns of future demands on resources.**

I. Introduction and Motivation

One of the biggest advantages of cloud service platforms such as Amazon EC2, Microsoft Azure, and IBM Smart Cloud Enterprise is the elastic usage of resources that can be translated into greater flexibility. This elasticity allows to reduce operational costs of implemented services. However, to exploit the elasticity of resource allocation, capacity planning should be as precise as possible: for instance, in scale-in/out of implemented services any significant difference between required and planned resources leads either to extra operational costs in case of over-provisioning or to missed revenues for rejected service requests in case of under-provisioning.

Since different types of resources (memory, bandwidth, processing power etc.) need to be allocated with different resolutions, we consider virtual *resource units*. To make operational settings more real, we assume that every resource unit has *maintenance* and *allocation costs*, there is an upper bound on the number of simultaneously allocated resources, every *service request* requires one virtual resource unit, and service requests are independent. We do not presume any specific distribution or pattern in the arriving stream; arrivals can be adversarial. In this environment, resource capacity planning for the next operational period becomes very complicated and interesting problem. Unlike other objectives such as average response time, latency, etc., where average case analysis is preferable, worst case guarantees become extremely important in economic settings: average case guarantees may need a long run of operation to be actually realized, and negative revenues may occur in intermediate points of operation, a loss that often cannot be afforded by service providers.

Our Contributions: In this work, we explore the effect of delaying service requests on revenues in various settings. There is a fundamental tradeoff between the ability to delay service requests in time (giving a better opportunity for capacity planning) and potentially missed opportunities to service requests if resource units available at a given time are limited. Delaying service requests may introduce additional latency, and a resource management policy can heavily depend on relations between allocation and maintenance costs. We study these effects in detail and propose efficient online policies with worst case performance guarantees. In our analytic study, we use *competitive analysis* that compares the performance of an online policy with an optimal clairvoyant offline algorithm [1]. We also supplement our theoretical results with a comprehensive simulation study.

This paper is organized as follows. In Section II, we summarize related work, covering three different existing approaches to auto-scaling. Section III introduces two discrete time models, with and without service requests deadlines, and the desired objectives. Section IV shows the simplest policies that do not have a buffer (or, equivalently, deal with urgent jobs that cannot be delayed). In Section V, we provide a comprehensive theoretical study of policies with buffers, showing a general lower bound and introducing two new online policies. Section VI shows results on the latency (both maximal and average) of the proposed online policies. In Section VII we consider an updated model, where latency constraints are embedded on the model level. The proposed policies are evaluated in Section VIII. We conclude with Section IX.

II. Related Work

There is a long prior art that considers buffer management policies in various settings [2], [3]. For a single queue architecture, throughput maximization is considered in [4], [5], [6],

https://doi.org/10.1109/INFOCOM.2016.7524616

[7], [8]. More complex buffering architectures are considered in [9], [10], [11], [12], [13], [14]. A recent survey [15] classifies auto-scaling techniques into several major categories: static policies, threshold-based polices, reinforcement learning, control theory, and time-series analysis. This classification can be viewed in terms of three major research lines. The first line deals with reactive mechanisms that do not try to anticipate future needs. A decision to increase or decrease capacity is based on the arrival patterns during last time slots (or values of other important variables). There is a long line of research that deals with benchmarking frameworks that assist in the evaluation of resources in cloud systems for service auto-scaling, either micro-architecturally [16], [17], [18] or at the system level [19], [20], [21], [22], [23]. Second, Amazon AWS AutoScaling and other cloud service brokers such as RightScale and Enstrautus implement rule-based auto-scaling. These mechanisms are applicable when service providers understand their resource demands. In this case service providers are responsible for defining reasonable scaling rules, and auto-scaling without explicit user intervention is hard. Naturally, there have been works that attempt to predict future resource requirements based on historical data of resource usage; usually, some machine learning model is trained on historical data and used to predict future loads, making auto-scaling decisions based on its predictions; to predict workload, researchers have used both nonparametric techniques [24], [25] and supervised learning methods such as neural networks and linear regression [26]. The works [27], [28], [29] attempt to predict patterns of service requests. Finally, one can take a hybrid path; for instance, the work [30] proposed a hybrid scaling technique where scaling up is done based on reactive rules while a supervised learning is used to scale down. [28] presented an online resource demand prediction model that achieves adaptive resource allocation. Cloud computing economics are considered in [31], [32], [33]. To represent leasing, the *parking permit problem* was introduced in [34], with [35] presenting the leasing model more generally for many infrastructure problems such as *facility location* and *set cover*. To our best knowledge, no related work has considered the economic effect of delaying service requests.

III. Model Description

We assume that time is slotted, and deploying a service for every request requires one unit of allocated resource per time slot. Note that a unit of virtual resource is defined by the implemented service (e.g., a service request has specific bandwidth requirements). Each resource unit has *allocation cost* α (including deallocation costs) and *maintenance cost* β per time slot. Each request has an intrinsic value gained by the service if the service request is serviced by an available allocated resource unit. In most cases, the behavior of an allocation process can be formulated as a pricing model, when at the end of every time slot a capacity planning process allocates resources for the next time slot to maximize revenue, given α and β costs.

Formally speaking, on every time slot t there arrives a sequence $\mathcal{J}_t$ of service requests. Each request j has an intrinsic positive value v_j, and we scale values so that $\min_j v_j = 1$. We assume that each request requires one virtual *resource unit*; virtual units can represent processing cores, virtual machines, or storage resources. Each time slot consists of three phases: (1) *admission*: new requests arrive, and the allocation policy decides whether to drop or admit each request; (2) *processing*: admitted requests are assigned to resource units for processing (at most one to each resource unit); (3) *prediction*: the allocation policy determines the *processing capacity* (number of resource units) allocated for the next time slot.

Changing capacity for time slot t costs c_t; we define c_t as a function of *additionally* allocated resource units and assume that deallocation cost is also amortized in c_t, so it is always free to decrease capacity for the next time slot. In most cases, we assume $c_t = \alpha n_t$, where n_t is the number of additional resource units. A_t is the set of requests admitted at time slot t. The goal is to maximize total revenue $\sum_t (\sum_{j \in A_t} v_j - \beta \cdot N_t - c_t)$, where N_t is the number of allocated resource units at time slot t. We call this model *Bufferless Heterogeneous values* (BLH) and consider it in Section IV. Fig. 1 shows sample timeslots of an allocation policy that predicts as many resource units as requests arrived on this timeslot; dropped requests are shaded in gradient. On the first timeslot, all requests are dropped since there are no resource units. On the second, two highest valued requests are admitted by an allocation policy; then processing capacity is overestimated, and two extra resource units are allocated for the last timeslot. Note that the revenue we just defined can be negative, so it is important to clarify the notion of α-competitiveness. Specifically, we say that an online algorithm A is α-competitive, if there is some universal constant c such that for any sequence of requests σ, A's revenue $\mathrm{A}(\sigma)$ and OPT's revenue $\mathrm{OPT}(\sigma)$ satisfy $\mathrm{A}(\sigma) + c \geq \mathrm{OPT}(\sigma)/\alpha$.

In the second model, we assume that a resource allocation system has a possibility to buffer at most δB requests, $\delta > 0$. We will see that these properties allow for efficient allocation policies and at the same time let us simplify predictions significantly. We call this model *Buffered with Heterogeneous values* (BH), with the corresponding uniform version *Buffered with Uniform values* (BU), and consider it in Section V.

Finally, in the last model we assume that each incoming request i in addition to value has a deadline $d(i)$ ($d(i) \geq 1$). A request i that arrived at timeslot t will be automatically dropped at the end of timeslot $t + d(i)$. In this model the latency constraints are embedded and not a property of specific algorithms as before. We call this model *Bounded Delay* (BD) and consider it in Section VII.

IV. Bufferless Allocation

In the *bufferless* model BLH, it is impossible to postpone servicing, so the maximal number of serviced requests equals the number of resource units allocated at the end of the previous time slot. The next theorem demonstrates that in the BLH model, online algorithms are non-competitive.

Theorem 1. *If $\alpha < 1 - \beta$, then in the* BLH *model any deterministic online allocation policy is non-competitive.*

Proof. Consider an arbitrary online deterministic algorithm A. Assume that A allocates m resource units at the first time slot. Then km service requests arrive at the first time slot, and A's revenue is at most $m(1-\alpha-\beta)$, while OPT's is $km(1-\alpha-\beta)$ if it allocates km resource units for the first time slot, and the sequence can be repeated. Thus, the competitive ratio is $\geq k$ for an arbitrary k. □

To avoid the rather trivial obstacle of Theorem 1, we could try to impose a natural constraint: suppose that the maximal number of resource units that can be allocated at a given time slot is bounded by B, which is also an input to the algorithm. We call the resulting model *Bounded Bufferless with Heterogeneous Values* (BBLH). Unfortunately, even under the BBLH constraint a similar adversarial sequence still works.

Theorem 2. *If $\alpha < 1 - \beta$, then in the* BBLH *model any deterministic online allocation policy is non-competitive.*

Proof. Consider an arbitrary online deterministic algorithm A. Let $k = \sqrt{B\beta}$, i.e., $B = k^2/\beta$. On the first time slot, if A allocates $\geq k/\beta$ resource units and no requests arrive, A spends $\geq \beta k/\beta = k$, while OPT allocates nothing and pays zero. Otherwise, if A allocates less than k/β resource units and there arrive B requests, its revenue is at most $k/\beta(1 - \alpha - \beta)$, while if OPT allocates B resource units, its revenue is $B(1-\alpha-\beta)$, and the competitive ratio is at least k, which is arbitrary large. □

Note that the maximal number of allocated resource units is expected to be high because higher number of resource units means, in general, more requests serviced and larger revenue. Taking B as an input to the algorithm, we are able to provide guarantees independent of this large number.

V. Buffered Allocation

A. General lower bound

In the previous section, we have shown that without a buffer we get high competitive ratios even with limited number of allocated resource units. Unfortunately, the BU model still does not allow us to build optimal online algorithms.

Theorem 3. *If $\alpha < 1-\beta$, then in the* BU *model every online algorithm is at least $1 + \frac{1}{\delta}\left(1 - \frac{\alpha}{1-\beta}\right)$-competitive.*

Proof. First, note that no competitive deterministic online algorithm A allocates resource units if no requests have arrived, or at least does so finitely many times, because otherwise it will process the empty input sequence with an arbitrarily low (negative) objective function. Let t_0 be the first time slot when A allocates no resource units, with no requests arriving before t_0. Then consider the following sequence: at time slot t_0 there arrive $(1 + \delta)B$ unit-valued requests. OPT predicts B machines at time slot $t_0 - 1$ and thus immediately runs B requests at time slot t_0; the δB requests are stored in the buffer to service them on the next time slot. Now OPT's revenue is at least $B(1 + \delta)(1 - \beta) - B\alpha$. On the other hand, A is able to service at most δB requests, and its revenue is no more than $\delta B(1 - \beta)$. Thus, the resulting competitive ratio is $\frac{B(1+\delta)(1-\beta)-B\alpha}{\delta B(1-\beta)} = 1 + \frac{1}{\delta}\left(1 - \frac{\alpha}{1-\beta}\right)$, as needed. □

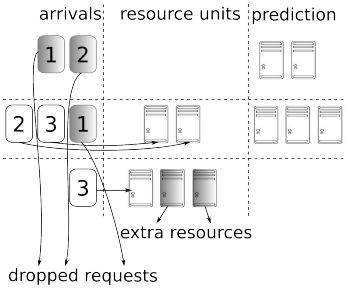


Fig. 1. Sample operation of a bufferless policy that predicts as many resources as there have been requests on the last time slot.

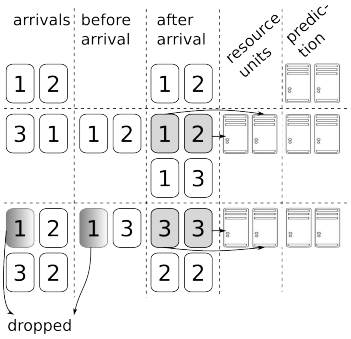


Fig. 2. Sample operation of the NRAP algorithm with buffer size 4.

B. Next round allocation policy

Now we present the first of the two allocation policies that we study in this work in detail. This policy, NRAP, simply predicts as many resource units for the next time slot as there are service requests currently residing in the buffer.

Definition V.1. *The* Next Round Allocation Policy *(*NRAP*):*

- *accepts incoming requests are as long as there is room in the buffer;*
- *pushes out the lowest valued request in the buffer by a higher valued arrival;*
- *predicts k resource units for the next time slot if there are k requests in the buffer by the end of a time slot;*
- *services either requests from the previous timeslot or those that pushed them out.*

Fig. 2 shows sample NRAP operation over three time slots with buffer size 4; gradient shading shows dropped service requests; gray, requests that will be processed on the current time slot. Note that while NRAP by definition does not spend extra resources, it can drop many incoming requests and spend a lot on (de)allocation. On the other hand, NRAP is simple, has low memory requirements, and, as the next theorem shows, has constant competitive ratio for some values of β and α.

Theorem 4. *If $\alpha < 1-\beta$, then* NRAP *is at most $2\left(1+\frac{\alpha}{1-\alpha-\beta}\right)$-competitive.*

Proof. For any incoming sequence, we map every request serviced by OPT to a request serviced by NRAP in such a way that the value of the image is no less than the value of its preimages, and at most two requests accepted by OPT are mapped to every request serviced by NRAP. OPT never pushes requests out, so "accepted by OPT" is equivalent to "serviced by OPT". To see that this mapping will imply the necessary bound, note that for each request of value v accepted (and later serviced) by OPT, OPT gets revenue at most $v-\beta$. By the mapping, there also exists a request with value $v' \geq v$ serviced by NRAP with revenue at least $v'-\beta-\alpha \geq v-\beta-\alpha$. Half of this revenue corresponds to j. Hence, for each request we have the ratio of $\frac{v-\beta}{2(v-\beta-\alpha)} \leq \left(1+\frac{\alpha}{1-\beta-\alpha}\right)/2$.

Next we show the mapping itself. It preserves the following invariant: the preimage of each request in the buffer at the beginning of a timeslot has size one. Consider the buffer state after arrival. Due to push-out, the ith highest valued request in NRAP's buffer has value no less than the ith highest valued request admitted by OPT, and we map these requests one to one. Some requests in NRAP's buffer that had another request mapped to them before arrival could be pushed out; in this case, we map their preimages to those requests that pushed them out. Requests with preimage of size two are exactly those that will be processed in the processing phase. □

This is an impressive result, given the simplicity of the policy, but, unfortunately, this upper bound can be arbitrarily high in case $\alpha \approx 1$ and $\beta \approx 0$, because NRAP has to pay allocation cost for each processed request and thus cannot profit from it.

Theorem 5. *Assume that $\alpha < 1-\beta$. Then in the BU model* NRAP *is at least $\left(2+\frac{\alpha}{1-\alpha-\beta}\right)$-competitive.*

Proof. Consider the following sequence: on the first and second time slots, there arrive B requests. NRAP drops the second burst of B requests, with total revenue $B(1-\beta-\alpha)$. OPT could predict B resource units for both timeslots with total revenue $2B(1-\beta)-B\alpha$, getting the bound. □

Although NRAP is nearly optimal when the arrival pattern is uniform and has acceptable competitiveness when allocation cost is low, the policy still has a major drawback: it always allocates a resource unit to service a single request and never reuses computational resources. This makes NRAP lose its competitiveness guarantees when allocation cost is high.

C. Amortizing Allocation Policy

Here we propose a different strategy that tries to avoid NRAP's limitations by amortizing allocation costs across several requests, performing well for high allocation costs. We denote by $\mathrm{PreAssigned}(v)$ the subset of requests in the buffer preassigned to an allocated resource unit v.

Definition V.2. *Upon arrival of k requests, the* Amortizing Allocation Policy *ρ-AAP with parameter $\rho > 1$:*

- *accepts incoming requests as long as the remaining buffer capacity plus the number of allocated resource units is more than $(B-\lceil\frac{\rho\alpha}{1-\beta}\rceil)/(\lceil\frac{\rho\alpha}{1-\beta}\rceil+1)-\lceil\frac{\rho\alpha}{1-\beta}\rceil$ (we assume it to be a positive integer for simplicity); only requests accepted on the current time slot can be pushed out.*
- *while the total value of non-preassigned requests residing in the buffer (those that do not belong to any $\mathrm{PreAssigned}(v)$) is at least $\lceil\frac{\rho\alpha}{1-\beta}\rceil$, repeats:*
  - *predict an additional resource unit v' to be needed for the next time slot;*
  - *move as few as possible non-preassigned requests with total value at least $\lceil\frac{\rho\alpha}{1-\beta}\rceil$ to $\mathrm{PreAssigned}(v')$;*
- *then predicts for each resource unit v such that $\mathrm{PreAssigned}(v) \neq \emptyset$ ρ-AAP that it will need v for the next time slot; every allocated resource unit with $\mathrm{PreAssigned}(v) = \emptyset$ is deallocated.*

Figure 3 shows the sample operation of the ρ-AAP algorithm for buffer size 7 with $\lceil\frac{\rho\alpha}{1-\beta}\rceil = 2$. In all columns except the first, each row represents either an allocated resource unit v (shown with its index in the last two columns) together with its corresponding $\mathrm{PreAssigned}(v)$ (second and third columns) or a set of non-preassigned requests. In the first timeslot, three requests arrive, two of them get a resource unit, and the last one remains non-preassigned (since the AAP threshold is 2 in this example). On the last timeslot, resource unit 1 has processed its queue entirely and is deallocated. The next theorem provides an upper bound for the ρ-AAP competitiveness; note how α no longer appears in the denominator.

Theorem 6. *For $\alpha \geq (1-\beta)$, ρ-AAP with an increased buffer of size $B+\lceil\frac{\rho\alpha}{1-\beta}\rceil$ has competitive ratio at most $\frac{\rho}{\rho-1}\left(\left\lceil\frac{\rho\alpha}{1-\beta}\right\rceil+1\right)$ against the optimal algorithm* OPT *with a buffer of size B.*

Proof. We denote $c = \lceil\frac{\rho\alpha}{1-\beta}\rceil$. Note that requests are preassigned to a resource unit in batches, with total value of a batch at least c. We get at least $c-\beta n-\alpha$ from servicing those requests, where n the number of requests in a batch. Note that $n \leq c$, so revenue per value unit is at least $\frac{\rho-1}{\rho}(1-\beta)$.

Next, assume that the number of preassigned requests is m. Then the servicing capacity is at least m/c, and there may be admitted at least $B\frac{\lceil\frac{\rho\alpha}{1-\beta}\rceil}{\lceil\frac{\rho\alpha}{1-\beta}\rceil+1} - m\left(1-\frac{1}{c}\right)$ requests. Since $m \leq B\lceil\frac{\rho\alpha}{1-\beta}\rceil/(\lceil\frac{\rho\alpha}{1-\beta}\rceil+1)$ (by the first step of the algorithm), we have a lower bound on this value as at least $B/(\lceil\frac{\rho\alpha}{1-\beta}\rceil+1)$. This means that ρ-AAP takes at least $1/(\lceil\frac{\rho\alpha}{1-\beta}\rceil+1)$ fraction of total value admitted by OPT due to push-outs. Suppose now that OPT accepts and services L requests in total. Obviously, its revenue is at most $L(1-\beta)$. Now we can bound from below the number of requests serviced by δ-AAP as $L/(\lceil\frac{\rho\alpha}{1-\beta}\rceil+1)$, and the above remark gives us ρ-AAP's revenue as at least $\frac{L(\rho-1)(1-\beta)}{\rho(\lceil\frac{\rho\alpha}{1-\beta}\rceil+1)}$. This results in an upper bound on the competitive ratio as $\frac{\rho}{\rho-1}(\lceil\frac{\rho\alpha}{1-\beta}\rceil+1)$. Since there may be some "tail"

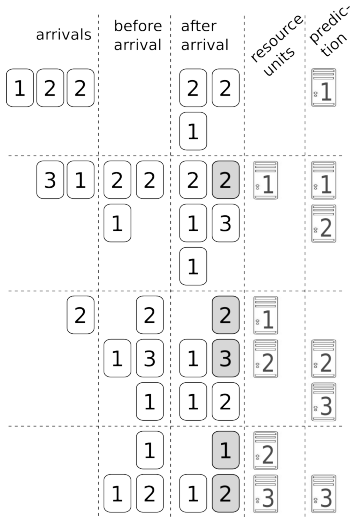


Fig. 3. Sample operation of the ρ-AAP algorithm.

requests never serviced by ρ-AAP (less than c of them), we have non-strict competitiveness. □

We finish this section with a matching lower bound for the ρ-AAP (up to a factor of $\rho/(\rho-1)$).

Theorem 7. *In the* BU *model the ρ-AAP algorithm with an increased buffer of size $B + \lceil \frac{\rho\alpha}{1-\beta} \rceil$ is at least $(\lceil \frac{\rho\alpha}{1-\beta} \rceil + 1)$-competitive against* OPT *with a buffer of size B.*

Proof. Consider a long sequence of B requests per time slot, which OPT services with B resource units. Since ρ-AAP never predicts more than $B/(\lceil \frac{\rho\alpha}{1-\beta} \rceil + 1)$ resource units, the competitive ratio follows. □

VI. Latency considerations

A. Worst-case analysis

For a given service request i accepted at time slot t_a and processed at time slot t_p, its *latency* is $\mathrm{lat}(i) = t_p - t_a$. The latency of an algorithm A is defined as $\mathrm{lat}(A) = \sup_{\mathcal{J}} \max_i \mathrm{lat}(i)$, where $\mathcal{J}$ is an input sequence and i is any request from $\mathcal{J}$ serviced by A. If no requests have arrived we set $\max_i \mathrm{lat}(i) = 0$. The goal is to minimize $\mathrm{lat}(A)$ while keeping the competitive ratio low. First, note that NRAP is very good when it comes to latency: every accepted request is immediately processed or pushed out in the next time slot.

Theorem 8. *In the* BH *model* $\mathrm{lat}(\mathrm{NRAP}) = 1$.

Unfortunately, since ρ-AAP waits to process packets, its latency becomes arbitrarily high in the general BH model, so we make an additional assumption that at least one request arrives on each time slot in the sequence from first to last; previous theorems obviously hold in this case. Under this restriction, we modify ρ-AAP to minimize maximal latency.

Definition VI.1. *The algorithm ρ-AAPm (modified ρ-AAP) behaves exactly like ρ-AAPm, but if no requests arrive at a given time slot then ρ-AAPm immediately drops all non-preassigned packets from its buffer.*

This modification lets us compute the latency of ρ-AAP.

Theorem 9. *If at least one request arrives on each time slot,* $\mathrm{lat}(\rho\text{-}AAPm) = \left\lceil \frac{\rho\alpha}{1-\beta} \right\rceil$.

Proof. Assume that before a request arrives there are at least k non-preassigned requests. Then we need to wait at most $\lceil \frac{\rho\alpha}{1-\beta} \rceil - k - 1$ time slots before enough requests are collected and a resource unit is allocated (or the request sequence ends). Next, due to FIFO order, this request waits for $k+1$ time slots until it is finally processed. Adding up these delays, we get the bound. It is easy to check that if exactly one request arrives per time slot then the bound is realized. □

B. Average-case analysis

While it is important to have worst-case guarantees for processing latency, the average case is also important, providing better guarantees for most requests while perhaps tolerating high latency for a small fraction. For an algorithm A, we define the average latency as $\mathrm{lat}_{\mathrm{avg}}(\mathrm{A}) = \sup_{\mathcal{J}} \frac{\sum_{i \in \mathcal{J}} \mathrm{lat}(i)}{|\mathcal{J}|}$, where $\mathcal{J}$ is an incoming sequence of requests.

NRAP is still straightforward to analyze, and we can also bound the average latency for ρ-AAPm.

Theorem 10. *In the* BH *model,* $\mathrm{lat}_{\mathrm{avg}}(\mathrm{NRAP}) = 1$.

Theorem 11. *If at least one request arrives on each time slot,* $\mathrm{lat}_{\mathrm{avg}}(\rho\text{-}AAPm) \geq \frac{1}{2}\lceil \frac{\rho\alpha}{(1-\beta)} \rceil + \frac{1}{2}$.

Proof. Consider $\lceil \frac{\rho\alpha}{1-\beta} \rceil$ service requests assigned to the same resource unit. If a request j is the ith to be processed among them, then $\mathrm{lat}(j) \geq i$. Summing over i and dividing by $\lceil \frac{\rho\alpha}{1-\beta} \rceil$, we get average latency $\frac{1+2+\ldots+\lceil \frac{\rho\alpha}{1-\beta} \rceil}{\lceil \frac{\rho\alpha}{1-\beta} \rceil} = \frac{1}{2}\left\lceil \frac{\rho\alpha}{1-\beta} \right\rceil + \frac{1}{2}$. It remains to note that the average over a set is no less than the minimum average among a partition of this set. □

VII. Bounded-delay allocation

In the previous section, incoming requests had only a value characteristic, and the incurred latency was a property of specific algorithms. In this case we were able to estimate upper bounds for maximum and average delays that a request may experience. Here, we consider the Bounded Delay model, where the latency constraints are embedded on the model level: packets have both value and allowed delay.

A. General lower bound

Our first result shows that this model is very different from BH and BU: it is now impossible to find an online algorithm with good competitiveness if $\alpha + \beta > 1$ (in the BH model ρ-AAP had an upper bound on competitiveness independent of B).

Theorem 12. *If $\alpha + \beta > 1$, then any online algorithm is non-competitive in BD model. The statement still holds even if all values and delays are equal to one.*

Proof. Assume that we have some c-competitive online algorithm A. Since A is competitive then there is some lower bound l (possibly negative) on its revenue. We will build an adversarial sequence of arrivals in such a way, that A's revenue is non-positive, while OPT's revenue can be arbitrary high.

Now, if A has no resource units allocated for a timeslot then there arrives a single request i with $d(i) = 1$ and $v(i) = 1$, otherwise no requests arrive. First, note that every resource unit allocated by A processes at most one request (the one that arrived on the timeslot immediately before its allocation). Since $1 - \alpha - \beta < 0$, it means that every time A allocates a resource unit it pays, and since A's revenue is bounded from below, A allocates only a finite number of resources units. Hence, after some timeslot T the sequence has a request arriving on each time slot. Requests arriving after T may be processed by OPT with a single resource unit, getting an arbitrarily high revenue (allocation cost α becomes negligible). Since A's revenue is non-positive and OPT's may be arbitrarily high, A is in fact non-competitive. □

Having proved the theorem, in the following we assume $\alpha + \beta < 1$. Note that the lower bound implies that there is no way to get more than a constant factor advantage of processing multiple requests by a single request unit.

B. Unbounded resources

Again, we begin with a default setting with no upper bound on the number of allocated resources units (BD model). Again, we are able to show a general lower bound.

Theorem 13. *If $\alpha+\beta < 1$, then in the* BD *model every online algorithm is at least $\left(1 + \frac{\alpha}{2(1-\alpha-\beta)}\right)$-competitive.*

Proof. Consider an arbitrary competitive online algorithm A; again, if no packets arrive then there is some timeslot t for which A has not allocated any resource units. Then the adversarial sequence has no requests until t and then on timeslot t two requests with unit delay and value arrive. A clearly is unable to gain more than $2(1 - \beta - \alpha)$, since after timeslot $t+1$ all requests will be dropped, and at time t A has no resource units. OPT, in turn, can predict a single resource unit on timeslot $(t - 1)$ and process both requests on this resource unit, getting revenue $(2 - 2\beta - \alpha)$, as needed. □

On the positive side, NRAP can be adapted for the BD settings in such a way that its competitiveness is at most a constant factor away from the global lower bound. To achieve this, we assume that NRAP: (1) predicts the number of allocated resource units for the next timeslot equal to the number of requests arrived on this timeslot, and (2) processes exactly those requests that arrived on the previous timeslot.

Theorem 14. *If $\alpha + \beta < 1$, then in the* BD *model* NRAP *is at most $(1 + \frac{\alpha}{1-\beta-\alpha})$-competitive.*

Proof. Every arriving request will be processed by NRAP on the next time slot with a revenue at least $(1-\beta-\alpha)$. If the total number of arrived requests is L, then NRAP's total revenue is no less than $L(1 - \beta - \alpha)$, while OPT's revenue is clearly no more than $L(1 - \beta)$. □

C. Bounded resources

We now bound the number of simultaneously allocated resource units by B and call it the *Limited Bounded Delay* (LBD) model. Interestingly, in this case competitiveness becomes provably worse. The problem is that with limited resources available for allocation over a given time period, if an algorithm fails to predict the needed resource units at the start of the period, it cannot compensate by allocating more in the future; the optimal algorithm always predicts correctly.

Theorem 15. *If $\alpha+\beta < 1$, then in the* LBD *model any online algorithm is at least $(2 + \frac{\alpha}{1-\beta-\alpha})$-competitive.*

Proof. Again, for a competitive algorithm Athere exists a time slot t when it allocates no resources. At time t, there arrive $2B$ requests with unit delay and value. Again, similar to Theorem 13, A is unable to process more than B requests on the next time slot (the maximal number of resource units is B, so A's total revenue is at most $B(1-\beta-\alpha)$. OPT predicts B resource units for timeslot t and processes $2B$ requests by the end of timeslot $t+1$ with revenue $2B(1 - \beta - \alpha/2)$. □

NRAP's performance remains close to the absolute bound.

Theorem 16. *If $\alpha + \beta < 1$, then in the* LBD *model* NRAP *is at most $3(1 + \frac{\alpha+\beta}{1-\beta-\alpha})$-competitive.*

Proof. Consider an arbitrary sequnce of requests σ. Let O be the set of requests processed by OPT; N, by NRAP. To prove the bound we map requests from $O \setminus N$ to requests in N in such a way that the preimage of every request $i \in N$ is of size at most 2 and the value of each request in the preimage is at most $v(i)$. If we can construct this mapping, the claim follows since $\sum_{i\in O\setminus N} v(i) \le 2\sum_{i\in N} v(i)$, so $\sum_{i\in O} v(i) \le 3\sum_{i\in N} v(i)$. Because OPT pays at least β for each processed request and NRAP pays at most $\beta + \alpha$, the competitive ratio is at most

$$\frac{3\sigma - \beta|O|}{\sigma - (\alpha+\beta)|I|} \le 3 + \frac{\beta(3|I| - |O|) + 3|I|\alpha}{\sigma - (\alpha+\beta)|I|} \le 3 + \frac{3(\beta+\alpha)}{1-\alpha-\beta}$$

since $\sigma \ge |I|$, where $\sigma = \sum_{i\in I} v(i)$.

For the mapping, we subdivide $O \setminus N$ into two disjoint subsets: D with requests served on the time slot when their deadline expires and all other requests $O \setminus D$. We will map each subset injectively to N. Assume that a request $i \in O \setminus D$ is processed by OPT on timeslot t; then it remains alive on timeslot $t + 1$ because t is not i's deadline. We map i to any request i' processed by NRAP on timeslot $t + 1$ that has no requests mapped to it; it is possible because at least as many resource units are allocated for timeslot $t+1$ as requests from $O \setminus D$ processed on timeslot t. Moreover, $v(i') \ge v(i)$ because otherwise i would be processed instead of i'. Next, assume that a request $j \in D$ is processed by OPT on timeslot t. Since j's deadline is t, it was alive at time $t-1$, and following the same logic we map it to a request processed by NRAP on timeslot t. Since mappings from both D and $O \setminus D$ are injective, in their combination every preimage is of size at most 2. □

Theorem 17. *If $\alpha + \beta < 1$, then in the* LBD *model* NRAP *is at least* 3*-competitive.*

Proof. As always, we present an adversarial sequence. Fix $V > 1$; on timeslot $t = 0$, there arrive $2B$ requests with delay 1 and value $(V - 1)$ and B requests with delay 2 and value V. It is easy to see that NRAP processes B requests with value V (on timeslot $t = 1$), while OPT is able to process all $3B$ requests, getting competitive ratio $3 - 1/V$ for arbitrary V. □

VIII. Simulations

A. Experimental Setting

To validate our theoretical results, we have conducted an extensive simulation study. It would be valuable to test the policies on real life traces, but unfortunately we have no access to real life datasets for resource demands. In addition, interesting interrelations between delaying service requests and allocation/maintenance costs are difficult to cover. Therefore, we have decided to test our algorithms on synthetic traces in a series of four experiments; two of them study how the objective function (total transmitted value less the cost of resource unit maintenance and allocation) depends on buffer size B and resource allocation cost α in the BH model, the third considers average latency among processed service requests, and the fourth adds delays in the BD model.

Note that the actual optimal algorithm in our model has prohibitive computational cost even if we assume that the optimal algorithm is clairvoyant: for $\alpha > 0$ it is a hard optimization problem to find the optimal sequence of (de)allocations. Therefore, as OPT we simply use the maximal objective function that can be generated, i.e., a clairvoyant algorithm that exactly predicts the need for capacity at the next turn and, moreover, gets to increase its capacity for free. Apart from OPT, NRAP, and three different versions of ρ-AAP for $\rho = 1.5$, 2.0, and 2.5, we have also added for comparison three other algorithms: Const5 simply always allocates 5 resource units, while Avg50 and Median50 use as their prediction the average and median value of buffer occupancy over the last 50 time slots (a value chosen by experimental cross-validation).

To generate traffic, we have, again lacking real life information regarding job distribution, followed the ideas of network traffic simulation for simulating incoming jobs. Network traffic has a long-tail distribution due to its bursty nature; its has long-range dependencies [36] that are not always perfectly captured by Poisson models [37]. Mathematical models for simulating long-tail traffic include Markov-modulated Poisson processes [38], [39] and fractional Brownian motion [40]; here, we use the Poisson Pareto burst process (PPBP), a recent successful traffic model [41], [42]. In PPBP, traffic is modeled with multiple overlapping bursts whose lengths conform to a Pareto (long-tail) distribution. We use PPBP to model the stream of incoming jobs, drawing each job's value from a Zipf (discrete inverse polynomial) distribution. We expect Avg50 and Median50 to perform well on average since the incoming stream distribution does not change during simulations, and they can learn the true mean quickly.

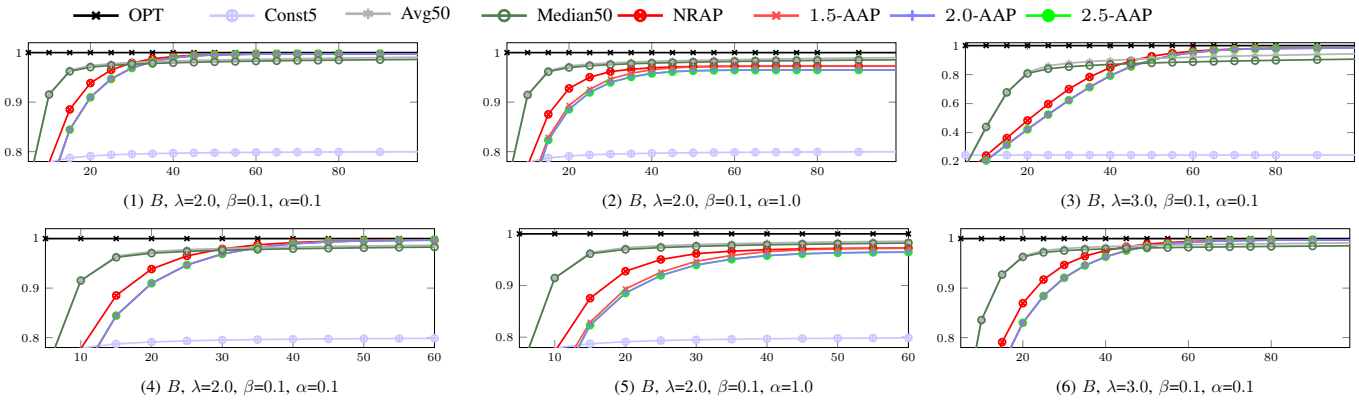
B. Objective function in the BH model

Figure 4 shows the results of our experiments for variable B; in the figure, the top row (Fig. 4, (1)–(3)) deals with the case of uniform values, and the bottom row (Fig. 4, (4)–(6)) adds values into consideration. The figures show that in our experiments, the proposed algorithms, NRAP and ρ-AAP, start by losing to the benchmark algorithms Avg50 and Median50 for small values of B but then overcome them as B grows, approaching 1 much faster than the algorithms whose predictions are based on average values. This effect is clearly visible for both uniform and variable values (Fig. 4, (1) and (4)), and it becomes even more pronounced as the input stream intensity increases, increasing congestion (Fig. 4, (3) and (6)). However, for larger values of the allocation cost α (Fig. 4, (2) and (5), where $\alpha = 1.0$) the proposed algorithms lose to the benchmark ones. For NRAP, this is simply due to the fact that it allocates and deallocates resources very unevenly on every time slot, spending a lot on new allocations. For ρ-AAP, this is due to the fact that as α grows, the algorithm becomes more cautious, its allocation threshold grows, and therefore it allocates fewer cores out of the available set. We explore this effect in more detail in the next set of simulations.

In the second set of simulations (Fig. 5), we studied how the objective function changes with α: uniform values in the top row (Fig. 5, (1)-(3)) and variable values in the bottom row (Fig. 5, (4)-(6)). We see the same effect: both NRAP and ρ-AAP begin to lose to Avg and Median as α grows; for extreme values of α they even drop below the simplest benchmark algorithm, Const5.

C. Latency in the BH model

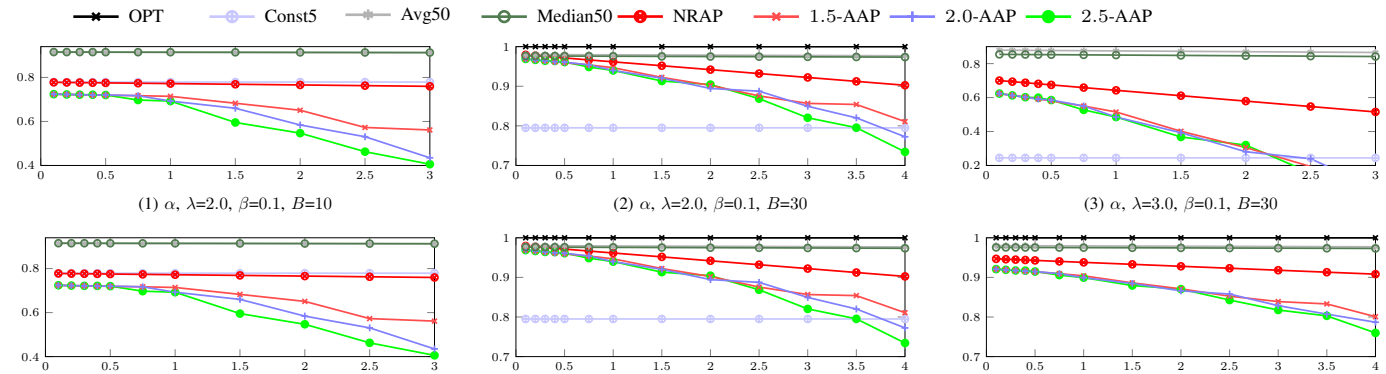
Finally, in the last set of simulations (shown on Figure 6) we study the average latency among processed service requests. There is no need to distinguish between uniform and heterogeneous values here since the value does not influence our objective function. The top row of graphs (Fig. 6, (1)-(3)) shows how latency changes as B grows. We see that the average latency of Const, Avg, and Median grows virtually linearly with B. As expected for small α, the average latency of ρ-AAP for all three considered values of ρ is virtually indistinguishable from 1, so for larger B the latency of Avg and Median grows significantly larger than that of ρ-AAP. The most interesting behaviour is exhibited by NARP: at first its latency decreases, reflecting the increasing processing power, but then it begins to increase due to the increased usage of the buffer. Still, its latency remains safely below the average latency of all other considered algorithms and approaches 1 from below as B grows. The bottom row of graphs (Fig. 5, (4)-(6)) shows the average latency as α grows. As expected, no algorithms except ρ-AAP care for α at all, and ρ-AAP's latency begins to grow for $\alpha > 1$ and further due to the effect explained above.



(1) B, λ=2.0, β=0.1, α=0.1 (2) B, λ=2.0, β=0.1, α=1.0 (3) B, λ=3.0, β=0.1, α=0.1

(4) B, λ=2.0, β=0.1, α=0.1 (5) B, λ=2.0, β=0.1, α=1.0 (6) B, λ=3.0, β=0.1, α=0.1

Fig. 4. Simulation results in the BH model with variable B. Top row: uniform values; bottom row: variable values.



(1) α, λ=2.0, β=0.1, B=10 (2) α, λ=2.0, β=0.1, B=30 (3) α, λ=3.0, β=0.1, B=30

(4) α, λ=2.0, β=0.1, B=10 (5) α, λ=2.0, β=0.1, B=30 (6) α, λ=3.0, β=0.1, B=30

Fig. 5. Simulation results in the BH model with variable α. Top row: uniform values; bottom row: variable values.

D. *The* BD *model*

Figure 7 shows how the algorithms behave in the BD model, with delays chosen according to a Zipf distribution. Note that in this model, a packet has two characteristics so the priority queue which is at the heart of every policy can come with two different priorities, sorting packets either first by slack and then by values (denoted with suffix S) or first by value and then by slack (denoted with suffix V). The results are very similar to the BH model; note, however, that algorithms that sort first by value consistently and significantly outperform their slack-preferring counterparts.

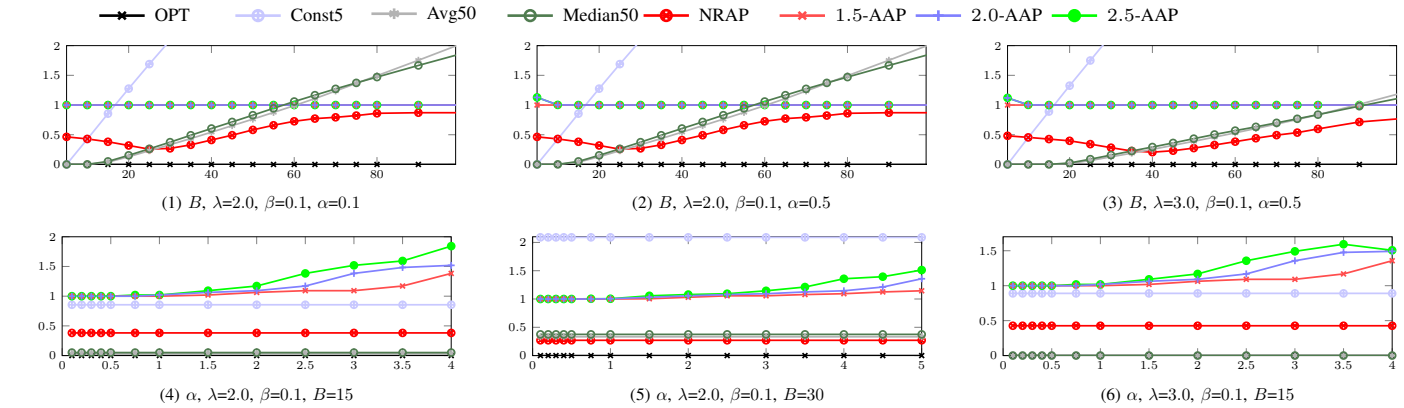
IX. Conclusion

Service providers operational environments introduce a fundamental tradeoff: between resource over-provisioning and missed service requests. In this work, we have studied the effects of delaying service requests to maximize revenues of implemented services with competitive analysis, i.e., in an adversarial setting without any assumptions on the arrival distributions. In addition, we have analyzed the incurred latency as a result of buffering service requests. We believe that this study can help to identify desired properties of resource allocation policies that optimize revenue in economic constraints.

Acknowledgements: The work of Pavel Chuprikov and Sergey Nikolenko was partially supported by the Government of the Russian Federation (grant 14.Z50.31.0030) and President grant for Young Ph.D.'s MK-7287.2016.1.

References

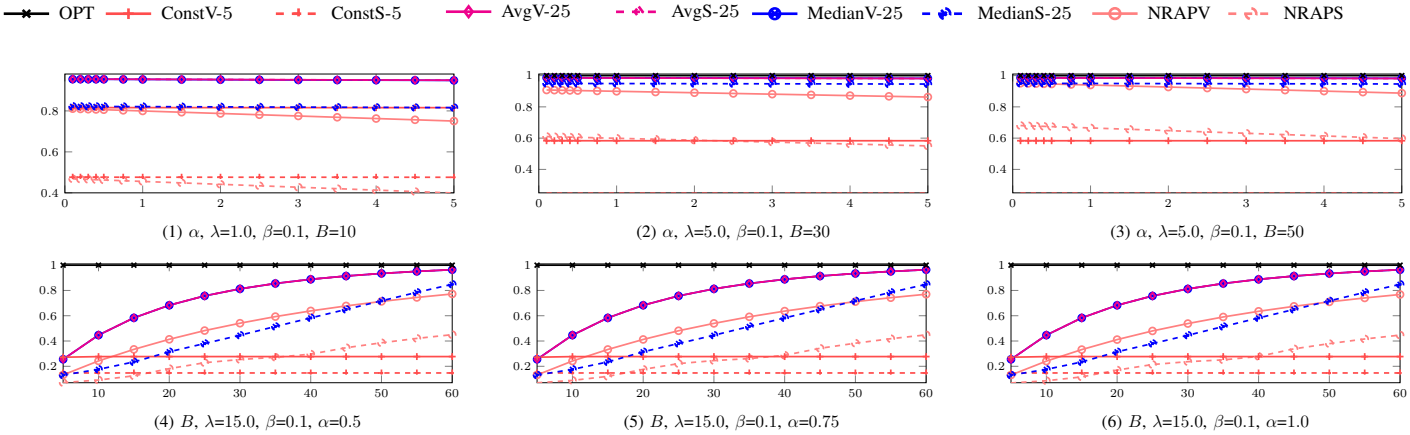
[1] A. Borodin and R. El-Yaniv, *Online computation and competitive analysis*. Cambridge University Press, 1998.
[2] M. Goldwasser, "A survey of buffer management policies for packet switches," *SIGACT News*, vol. 41, no. 1, pp. 100–128, 2010.
[3] S. Nikolenko and K. Kogan, "Single and multiple buffer processing," in *Encyclopedia of Algorithms*, 2015.
[4] P. Chuprikov, S. Nikolenko, and K. Kogan, "Priority queueing with multiple packet characteristics," in *INFOCOM*, 2015, pp. 1418–1426.
[5] I. Keslassy, K. Kogan, G. Scalosub, and M. Segal, "Providing performance guarantees in multipass network processors," *Trans. Netw.*, vol. 20, no. 6, pp. 1895–1909, 2012.
[6] K. Kogan, A. López-Ortiz, S. Nikolenko, A. Sirotkin, and D. Tugaryov, "FIFO queueing policies for packets with heterogeneous processing," in *MedAlg*, 2012, pp. 248–260.
[7] K. Kogan, A. López-Ortiz, S. Nikolenko, and A. Sirotkin, "A taxonomy of semi-fifo policies," in *IPCCC*, 2012, pp. 295–304.
[8] K. Kogan, A. López-Ortiz, S. Nikolenko, G. Scalosub, and M. Segal, "Balancing work and size with bounded buffers," in *COMSNETS*, 2014, pp. 1–8.



(1) B, λ=2.0, β=0.1, α=0.1 (2) B, λ=2.0, β=0.1, α=0.5 (3) B, λ=3.0, β=0.1, α=0.5

(4) α, λ=2.0, β=0.1, B=15 (5) α, λ=2.0, β=0.1, B=30 (6) α, λ=3.0, β=0.1, B=15

Fig. 6. Simulation results in the BH model, average latency among processed service requests.



(1) α, λ=1.0, β=0.1, B=10 (2) α, λ=5.0, β=0.1, B=30 (3) α, λ=5.0, β=0.1, B=50

(4) B, λ=15.0, β=0.1, α=0.5 (5) B, λ=15.0, β=0.1, α=0.75 (6) B, λ=15.0, β=0.1, α=1.0

Fig. 7. Simulation results in the BD model with deadlines.

[9] A. Kesselman, K. Kogan, and M. Segal, "Packet mode and qos algorithms for buffered crossbar switches with FIFO queuing," *Distributed Computing*, vol. 23, no. 3, pp. 163–175, 2010.

[10] ——, "Improved competitive performance bounds for CIOQ switches," *Algorithmica*, vol. 63, no. 1-2, pp. 411–424, 2012.

[11] ——, "Best effort and priority queuing policies for buffered crossbar switches," *Chicago J. Theor. Comput. Sci.*, vol. 2012, 2012.

[12] K. Kogan, A. López-Ortiz, S. Nikolenko, and A. Sirotkin, "Multi-queued network processors for packets with heterogeneous processing requirements," in *COMSNETS*, 2013, pp. 1–10.

[13] P. Eugster, K. Kogan, S. Nikolenko, and A. Sirotkin, "Shared memory buffer management for heterogeneous packet processing," in *ICDCS*, 2014, pp. 471–480.

[14] P. Eugster, A. Kesselman, K. Kogan, S. Nikolenko, and A. Sirotkin, "Essential traffic parameters for shared memory switch performance," in *SIROCCO*, 2015, pp. 61–75.

[15] T. Lorido-Botran, J. Miguel-Alonso, and J. A. Lozano, "A review of auto-scaling techniques for elastic applications in cloud environments," *J. Grid Comput.*, vol. 12, no. 4, pp. 559–592, 2014.

[16] M. Ferdman and et al., "Clearing the clouds: a study of emerging scale-out workloads on modern hardware," in *ASPLOS*, 2012, pp. 37–48.

[17] V. Reddi and et al., "Web search using mobile cores: quantifying and mitigating the price of efficiency," in *ISCA*, 2010, pp. 314–325.

[18] L. Tang and et al., "The impact of memory subsystem resource sharing on datacenter applications," in *ISCA*, 2011, pp. 283–294.

[19] B. Cooper and et al., "Benchmarking cloud serving systems with YCSB," in *SoCC*, 2010, pp. 143–154.

[20] C. Kozyrakis and et al., "Server engineering insights for large-scale online services," *Micro*, vol. 30, no. 4, pp. 8–19, 2010.

[21] A. Li and et al., "Cloudcmp: comparing public cloud providers," in *SIGCOMM*, 2010, pp. 1–14.

[22] A. Li, X. Yang, and M. Zhang, "Cloudcmp: Shopping for a cloud made easy," in *HotCloud*, 2010.

[23] V. Soundararajan and J. Anderson, "The impact of management operations on the virtualized datacenter," in *ISCA*, 2010, pp. 326–337.

[24] R. Han and et al., "Lightweight resource scaling for cloud applications," in *CCGrid*, 2012, pp. 644–651.

[25] H. Goudarzi, M. Ghasemazar, and M. Pedram, "Sla-based optimization of power and migration cost in cloud computing," in *CCGrid*, 2012, pp. 172–179.

[26] S. Islam and et al., "Empirical prediction models for adaptive resource provisioning in the cloud," *FGCS*, vol. 28, no. 1, pp. 155–162, 2012.

[27] Z. Gong, X. Gu, and J. Wilkes, "PRESS: predictive elastic resource scaling for cloud systems," in *CNSM*, 2010, pp. 9–16.

[28] Z. Shen and et al., "Cloudscale: elastic resource scaling for multi-tenant cloud systems," in *SOCC*, 2011, p. 5.

[29] E. Caron, F. Desprez, and A. Muresan, "Pattern matching based forecast of non-periodic repetitive behavior for cloud clients," *J. Grid Comput.*, vol. 9, no. 1, pp. 49–64, 2011.

[30] W. Iqbal, M. Dailey, D. Carrera, and P. Janecek, "Adaptive resource provisioning for read intensive multi-tier applications in the cloud," *FGCS*, vol. 27, no. 6, pp. 871–879, 2011.

[31] D. Dash, V. Kantere, and A. Ailamaki, "An economic model for self-tuned cloud caching," in *ICDE*, 2009, pp. 1687–1693.

[32] R. Pal and P. Hui, "Economic models for cloud service markets," in *ICDCN*, 2012, pp. 382–396.
[33] M. Armbrust and et al., "A view of cloud computing," *Commun. ACM*, vol. 53, no. 4, pp. 50–58, 2010.
[34] A. Meyerson, "The parking permit problem," in *FOCS*, 2005, pp. 274–284.
[35] B. Anthony and A. Gupta, "Infrastructure leasing problems," in *IPCO*, 2007, pp. 424–438.
[36] Z. Xiaoyun, Y. J. Yu, and J. Doyle, "Heavy tails, generalized coding, and optimal web layout," in *INFOCOM*, vol. 3, 2001, pp. 1617–1626 vol.3.
[37] V. Paxson and S. Floyd, "Wide area traffic: The failure of poisson modeling," *Trans. Netw.*, vol. 3, no. 3, pp. 226–244, Jun. 1995.
[38] W. Fischer and K. Meier-Hellstern, "The markov-modulated poisson process (mmpp) cookbook," *PER*, vol. 18, no. 2, pp. 149–171, 1993.
[39] H. Heffes and D. Lucantoni, "A markov modulated characterization of packetized voice and data traffic and related statistical multiplexer performance," *JSAC*, vol. 4, no. 6, pp. 856–868, Sep 1986.
[40] I. Norros, "On the use of fractional brownian motion in the theory of connectionless networks," *JSAC*, vol. 13, no. 6, pp. 953–962, Aug 1995.
[41] R. Addie, T. Neame, and M. Zukerman, "Performance evaluation of a queue fed by a poisson pareto burst process," *Computer Networks*, vol. 40, no. 3, pp. 377–397, 2002.
[42] M. Zukerman, T. Neame, and R. Addie, "Internet traffic modeling and future technology implications," in *INFOCOM*, vol. 1, March 2003, pp. 587–596 vol.1.