Optimal Scheduling in the Hybrid-Cloud

Mark Shifrin
Faculty of Electrical Engineering
Technion, Israel
Email: shifrin@tx.technion.ac.il

Rami Atar
Faculty of Electrical Engineering
Technion, Israel
Email: atar@ee.technion.ac.il

Israel Cidon
Faculty of Electrical Engineering
Technion, Israel
Email: cidon@ee.technion.ac.il

***Abstract*—The emerging hybrid cloud architecture allows organizations to optimize their computation needs and costs by maintaining their private computational infrastructure at high utilization and meeting peak requirements by offloading selected tasks to the public cloud. Consequently, there is a need to devise efficient systems equipped with online task *cloudbursting* algorithms that optimize the overall cost while maintaining adequate quality of service. Such algorithms must take into account the difference in communication and computational requirements associated with different tasks. For example, it is clear that when two tasks have the same local computational requirements, the one with the lower cloudbursting cost is a better candidate to be off-loaded and sent to the cloud.**

In this paper, we address the case in which arriving tasks have the same computational cost but different communication costs. We design scheduling system based on online decision algorithms driven by the user's local infrastructure constraints. We model the online scheduling problem as Markov Decision Process (MDP) problem and provide optimal policies for scheduling tasks either locally or remotely. We further explore the usage of MDP in different scenarios and prove the structural properties of the optimal policies in order to incorporate them into the decision engine. The design of the practical scheduling system is supported by the analytical results and numerical evaluations. We demonstrate the practical computational advantage of threshold type policies and provide an insight into their dependence on system parameters.

I. Introduction

The emergence of cloud computing is changing the ways in which organizations address their information technology (IT) needs. Cloud computing is a new way of increasing computational and storage capacity without investing in new infrastructure, training new personnel, or licensing new software. In theory, the cloud's agility and elasticity make it the best IT infrastructure solution. However, many practical issues, such as limited network speeds, lack of strict SLA guarantees and cloud standards, limit the full adoption of a cloud model [2]. Consequently, there is a topical trend to leverage the best of both worlds by keeping the minimal essential IT infrastructure while adopting the public cloud either when it is more cost effective or when it is dictated by performance requirements. One of the terms which are frequently used to describe this paradigm is a "hybrid cloud". This term refers to organizations that own some of the computation infrastructure, while also utilizing the services of a cloud provider to augment or supplement the private infrastructure.

Another topical term associated with a hybrid cloud is "cloudbursting". According to this concept, in case the private data center runs out of computing resources, the organization "bursts" (i.e. offloads) workload to an external cloud on an on-demand basis. The internal computing resource is the "Private Cloud", while the external cloud is typically a "Public Cloud", for which the organization gets charged on a pay-per-use basis. Effective cloudbursting offers the organization a good trade-offs between overall computation costs, availability and performance. We term the online decision process which determines whether to locally process the tasks at the private cloud or to cloudburst them to the public cloud as a *scheduling* process.

The hybrid cloud's environment poses new research challenges associated with effective task scheduling. If all the computational tasks were identical, a simple scheduling mechanism would track the backlog of tasks waiting for local execution; if the backlog crossed a certain value associated with the maximal allowed latency or maximal space limit, arriving tasks would be directed to the cloud. As the IT tasks are heterogeneous in terms of computation, communication and storage requirements, the cost-effective design of such task scheduling mechanisms becomes more challenging. Therefore, the goal of this paper is to explore the modeling of these new problems and to find the algorithmic solution to them.

To that end, we first propose a *user-level scheduling system* which comprises task scheduling algorithms. We assume that the goal of the cloud user is to minimize the cost of serving the arriving tasks by using the combination of a resource limited private cloud and cloudbursting while meeting a predefined QoS level. The model definition involves finding scheduling algorithms to handle the flow of several task types with different resource consumption requirements. Such algorithms must take into account the difference in communication and in computational requirements among different task types. The difference in requirements impose different cloudbursting costs.

Our scheduling system is equipped with optimal online decision algorithms for modeling and solving several associated MDP problems. The new system is designed to handle the task influx which consists of several task types with different cloudbursting costs. We show that the optimal scheduling policy has a *threshold structure*. That is, the optimal decision is to direct the tasks to a private cloud in case the number of already present tasks is below a certain threshold. Otherwise, the tasks must be cloudburst to the public cloud. The scheduling rule can be easily implemented due to the threshold structure of the optimal policy. Hence, the proof to structural properties of the optimal policy is provided.

To the best of our knowledge, this is the first work that proposes *optimal scheduling policies* utilizing control theory and MDP methods for the hybrid cloud environment, in which tasks can be either served in an existing private cloud or cloudburst onto the public cloud.

The optimization of a scheduling process naturally lends itself to an optimization of queuing system which can be tackled as an MDP problem. The direct techniques of finding the optimal policy for the corresponding MDP are related to the methods of value iteration, [7],[22],[31]. This process is known to become computationally consuming once the state space of a problem increases. Consequently, the mere understanding that

978-3-901882-50-0

the optimal policy has a simple structure such as a threshold type, facilitates the computational effort of the scheduling process. Our proposed system is enhanced with estimation techniques which can be applied in order to calculate the optimal policy associated with MDP problems with significant reduction in complexity. Particularly, when system parameters change, *policy adjustment* can be effectively applied.

To summarize, the contribution of this paper is two-fold:

- A user-level scheduling system provided with the MDP based techniques, which calculates the decision whether to serve heterogeneous tasks in the private cloud or to cloudburst them onto the public cloud.
- The derivation of the optimal scheduling policy for the scenario presented above. We present and prove several structural properties of the optimal policy, i.e. the threshold-type policy. The threshold policy facilitates the online policy calculation (adjustment) with reduced complexity. The corresponding algorithm is provided.

A. Related Work

We refer to two related subtopics: cloudbursting decision systems that are related to the practical problem addressed in our paper and works which address MDP threshold policies that are related to our analytical results.

Cloudbursting: Only few works have addressed cloudbursting as a decision problem. An online scheduling system in a hybrid cloud is presented in [26]. Unlike this work, where the addressed performance metric is system response time which is heuristically minimized, the objective of our system is an optimization of the combined cost. [19],[20],[25] suggest a software solution to the interaction between the users and a cloud, but do not present efficient scheduling algorithms. [23] develops a rate-limiting architecture for the cloud. [13] presents the solution to the permanent migration of tasks to the cloud, while our paper addresses dynamic cloudbursting. Some works which present task scheduling combined with resource allocation in other systems can be applied to the cloud computing, e.g. [6],[11]. The methods used in the aforementioned papers are heuristic and therefore suboptimal, while we consider the optimal scheduling in hybrid cloud by using the solution to the corresponding MDP. The scheduling of resources inside the public cloud constrained by cost-aware metrics is formulated as a linear programming (LP) problem in [27]. The system closest to ours is described in [30]. However, similarly to [27], it employs LP methods.

MDP and threshold policies: The primary tools that we utilize are Markov Decision Process and proofs to the structural properties of the underlying optimal policy. Regarding the derivation of threshold-type policy for networking systems, numerous notable works can be found, e.g. [12],[10],[33],[29],[8],[18]. Other works are related to the analytical methods used in our paper. Specifically, tools from fluid and diffusion approximations have been applied to address models that are closely related to those we introduce here. In particular, a significant amount of work has been done in recent years on models with a large number of servers (see [3],[4],[5],[14] and references therein). The approach presented in those papers extends the task influx model beyond the Poisson distribution constraint, which is the assumption throughout this paper. Arrival processes of other than Poisson distribution pose different analytical challenges and are left for the future work.

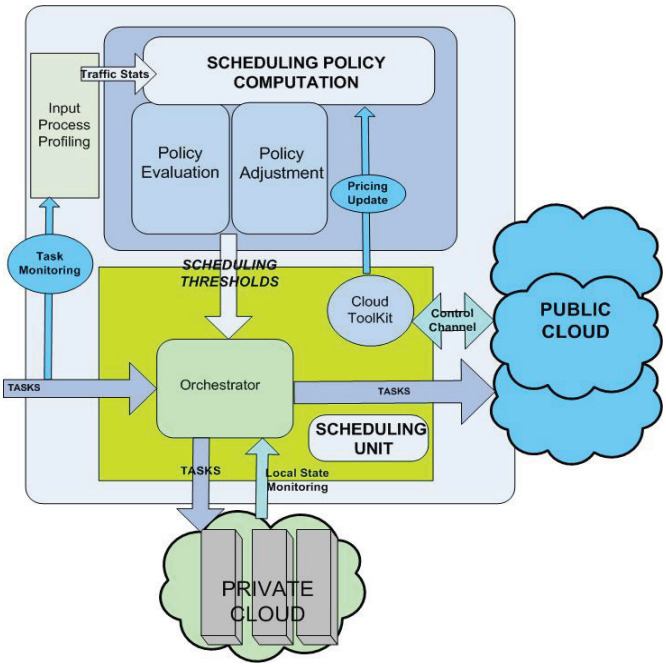

Fig. 1. Cloudbursting Scheduling System

The rest of the paper is organized as follows. In section II, we describe the user perspective hybrid cloud scheduling system. We provide a detailed description of each component of the system. In section III, we concentrate on the MDP solution and on its structural properties, where the cases of a single server for all tasks and a dedicated server for each task type are treated separately. Consequently, we show the corresponding one-dimensional or multi-dimensional thresholds. The final section is devoted to numerical results and practical models.

II. User Perspective Hybrid Cloud Scheduling System

We start with presenting a user perspective scheduling system in a hybrid cloud environment. The system controls the user environment, which contains a private cloud with limited computational resources and an access to a public cloud. Particularly, it performs a real-time scheduling of the influx of heterogeneous tasks. The block representation of the system is illustrated in Figure 1. The components of the system include the analysis of input processes, the computation of scheduling policy by solving the MDP problem, policy adjustment, the scheduling unit which implements the optimal policy, and the cloud toolkit which maintains the control channel with the public cloud. We elaborate in detail the objective of each unit and show how different units interconnect. Relevant details about the configuration of the private and the public clouds, as well as the underlying assumptions, are brought therein too.

Private cloud configuration: We assume that the computational capabilities of a private cloud are limited and fixed. Clearly, in case there is a performance independence among different task types, the problem can be effectively split into separate independent problems. However, in a practical system, there is a common resource shared by all tasks regardless of the task type. We express this limitation by using a finite shared task *buffer*, and that can be viewed either as a waiting delay constraint or as a task capacity constraint. We view two basic server settings (configurations) of a private cloud:

- Single-server model. Specifically, a private cloud has a single server resource which serves all the tasks in a FIFO order with equal service rates. In this case, tasks of different types compete with each other over the computational resources.
- Multi-server model. Here, we assume that each task type can only be processed by a dedicated server, but the tasks share a common storage infrastructure. In this case, tasks of different types compete with each other over the storage resources.

In order to simplify the mathematical treatment of the MDP, we assume no maintenance (holding) cost in a private cloud. However, it can be easily shown that convex, e.g. linear holding cost inflicts no complication regarding the optimal solution and no change in the structure of the optimal policy.

Input process profiler: The task input is fed to the scheduling unit and tracked by a special unit which performs input process analysis (profiling), denote it as (IP). The main objective of the IP is task profiling, i.e. an estimation of the arrival task rate corresponding to each task type. Throughout the paper, we assume that the arrival rate of each task type can be estimated as constant over long time periods, i.e the arrival processes are quasi-constant.

Scheduling policy computation: The scheduling policy computation is provided by a designated unit. The input to this unit is the average arrival rates of the tasks which are fed by the IP, as well as the prices corresponding to each task type. The output of this unit is the optimal policy structure which is calculated by solving the corresponding MDP. As demonstrated in the next section, the optimal policy is a threshold policy. Thus, the decision rule which is provided for the scheduling unit is rather simple to implement.

To this end, sub-units of this unit are Policy Evaluation (PE) sub-unit and Policy Adjustment (PA) sub-unit. The PE sub-unit implements the value iteration of the MDP resulting in a numerically calculated value function. Then it finds the optimal policy which corresponds to the value function. (See [7] for MDP definition and solution). The PA sub-unit comes in use once a particular change in parameters occurs, such as an update of pricing for the cloudbursting of one of the task types onto the public cloud, or a change in arrival rate. Consequently, a policy estimation technique might be applied, while an "old" policy is used in order to find the updated policy. The implementation involves the details of the value iteration procedure and is referred to in section IV. The utilization of the PA sub-unit is especially effective for the multi-dimensional case, in which the state space of the MDP is comparatively large, and the value iteration process is a resource-consuming procedure. The computational effectiveness of the policy estimation procedure is influenced by the fact that the optimal policy has a threshold type structure. Therefore, providing the proof to the threshold type structure of the optimal policy is important. For simple settings, in which the state space is small, the operation of the PA component is optional, and thus the PE component may be triggered every time a new decision rule is needed.

Scheduling Unit: The Scheduling Unit (SU) consists of two sub-units: the Orchestrator which implements the scheduling rule and the Cloud Toolkit (CT) which allocates the tasks designated for the public cloud. Note that this separation is schematic, and the unit might be implemented as a single integrated block. The input received by the Orchestrator from the MDP computation is the optimal policy, i.e the scheduling rule. The SU implements a real-time decision according to this rule, depending on the current state of the private cloud. Consequently, the Orchestrator tracks the current state of a private cloud and keeps updates of the state upon either any task being sent to the private cloud or any task accomplishment. Those tasks which are to be cloudburst are passed to the allocation in the public cloud. The mission of the CT can be fulfilled with a few known tools, such as OpenNebula [21] or Eucalyptus [20]. CT tracks available cloud resources and makes allocations of VMs on the on-demand basis according to the type of the task being cloudburst. Consequently, there is a compliance with the cloudbursting process and the VM allocation performed by the CT.

The pricing model of cloud computing is the focus of the ongoing research, e.g. [15] and [32]. Therefore, we assume no precisely defined SLA pattern associated with a specific public cloud. The allocation might be performed within or out of the existing long-term SLA with pay-as-you-go charges. Moreover, the on-demand VM allocation can be performed within more than one cloud provider. The general pricing model that we adopt is a charge for VM usage on the time basis. The examples of such pricing patterns can be found at Amazon EC2 [1] and at Fujitsu Global Cloud Platform [9]. Note that MDP formulation assumes that the cloudburst tasks are charged in a per-task manner. Therefore, a CT must be able to translate the cost of each task type into the MDP "language". Provided that the service and the communication requirements of the task types are clearly estimated, this translation can be done by an ad-hoc calculation. Any updates in task pricing are transferred by the CT to the MDP computation part block, which recalculates the policy accordingly and outputs a new decision rule to the Orchestrator.

III. Threshold structure of the optimal policy

This section addresses the threshold structure of the optimal policy. The private cloud is modeled as a server (e.g.a server cluster) that processes an inflow of tasks of several types. As mentioned before, we distinguish between two possible settings, where all the task types are served by a common FIFO server, or by FIFO servers dedicated to each task type. We prove the structural properties of the optimal policy for the both cases. The difficulty posed by both problems is the finite buffer, and is not extensively addressed in previous works. For example, the problem analyzed in [8] considers maximization objective, therefore it differs from our settings. As is elaborated below, the single server model dictates a one-dimensional state space, regardless of the number of task types. The dimension of the state space of the multi-server model is equal to the number of task types, i.e. the number of dedicated servers. We present two different proofs of existence of threshold-type optimal policies for the both settings. In the multi-dimensional case the proof, which is presented in III-B is based on theorems proved in [16], while for the one-dimensional case we present a novel proof using different techniques.

A. Threshold policy of single-server case

In the following, we present the details of the single-server model. Arriving tasks that are backlogged for local execution are queued. Such tasks cannot be cloudburst any longer. Service is FIFO and non-interruptible. The number of tasks awaiting service never exceeds a certain limit denoted as B. Therefore, a new arrival occurring while there are already B tasks in the queue is sent to the cloud. As mentioned above, the tasks influx is heterogeneous, meaning that different communicating costs are associated with each task type if handed to the cloud. Denote by k the number of task types. A

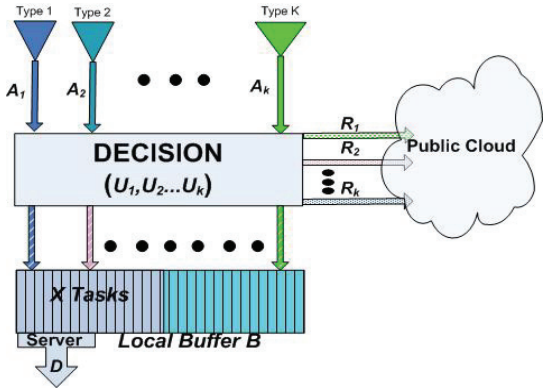

Fig. 2. MDP model chart

cost C_i (includes processing and communication) is incurred whenever a type-i task is sent to the cloud. The types are labeled in such a way that

$$C_1 \geq C_2 \geq \cdots \geq C_k > 0. \tag{1}$$

For the sake of simplicity, we assume that the task processing time distribution at the private cloud is independent of the task type. Upon each arrival, a decision is to be made (assuming the buffer is not full) whether to accept the task to the private cloud or to offload it to the public cloud and pay the corresponding cost. Denote by A_i and R_i, $i = 1, \ldots, k$, the counting processes for arrival and, respectively, remote offloading (cloudbursting), for type i. Namely,

$A_i(t)$ represents the number of tasks of type i that have arrived up to time t,

$R_i(t)$ represents the number of tasks of type i sent to the cloud up to time t.

All counting processes mentioned in this paper are assumed to have right-continuous sample paths. Further, A_i are modeled as independent Poisson processes of intensities λ_i, respectively, and the service time distribution is assumed to be exponential of rate $\mu > 0$, independent of the task type. Next, for each i, let $U_i(t)$ be a process taking values in $\{0, 1\}$, describing the control decisions determining R_i from A_i. Namely, $U_i(t) = 1$ if and only if a type-i task arriving at time t is sent to the cloud. As a result, we can write

$$R_i(t) = \int_0^t U_i(s) dA_i(s) \tag{2}$$

The total number of tasks being enqueued in the private cloud is denoted by $X(t)$. The initial condition of X is denoted by $x \in \{0, 1, ..., B\}$. The process X is given by

$$X(t) = x + \sum_{i=1}^{k} A_i(t) - \sum_{i=1}^{k} R_i(t) - D(t), \tag{3}$$

where D denotes the departure process, counting the number of completed tasks (of all types). The logical diagram which describes the MDP model is shown in Figure 2. The total discounted cost associated with a control process $U(t) = (U_1(t), \ldots, U_k(t))$ is given by

$$J(x, U) = E\Big[\int_0^\infty e^{-\gamma s} \sum_{i=1}^{k} C_i dR_i(s)\Big], \tag{4}$$

where $\gamma > 0$ is a discount factor.

A control process U is said to be *admissible* if (i) it is adapted to the filtration generated by $(\{A_i\}, X)$; and (ii) under U, the process $X(t)$ satisfies the constraint

$$X(t) \leq B \text{ for all } t. \tag{5}$$

The first condition expresses the requirement that control decisions are made based on events from the past and present, so that the decision maker has no access to information from the future. The second condition addresses the buffer limit: If for some t one has $X(t) = B$ and a task of type i arrives then $U_i(t)$ must be set to 1. The class of all admissible control processes is denoted by $\mathcal{U}$. The value function for the optimal control problem is defined as

$$V(x) = \inf_{U \in \mathcal{U}} J(x, U), \qquad x \in \{0, 1, ..., B\}. \tag{6}$$

This is a problem of continuous time Markov decision processes. For such a problem a principal tool is the characterization of the function V as the solution to a Bellman equation. Using this tool we can show that a policy of threshold type is optimal.

Remark 3.1: It is natural to work with an average cost rather than a discounted one. However, it is well known that, provided $\gamma > 0$ is sufficiently small, the optimal policies with and without discount, are the same. We later detail about what is known as Blackwell optimality.

Denoting $\delta = (\mu + \gamma + \sum_i \lambda_i)^{-1}$, the value function uniquely solves the Bellman equation (see e.g., [31] Chapter 8)

$$V(x) = \delta\mu V(x-1) + \sum_{i=1}^{k} \delta\lambda_i \min[V(x) + C_i, V(x+1)], \quad x \in \{1, 2, \ldots, B-1\}, \tag{7}$$

with boundary conditions

$$V(0) = \delta\mu V(0) + \sum_{i=1}^{k} \delta\lambda_i \min[V(0) + C_i, V(1)],$$
$$V(B) = \delta\mu V(B-1) + \sum_{i=1}^{k} \delta\lambda_i [V(B) + C_i]. \tag{8}$$

Denoting $\delta' = (\gamma + \sum_i \lambda_i)^{-1}$, the boundary condition at zero could be written in a more standard form as $V(0) = \sum_{i=1}^{k} \delta'\lambda_i \min[V(0) + C_i, V(1)]$. However, the form (8) will be useful in the analysis.

The threshold structure is provided by the following.

Theorem 3.1: There exist constants $B - 1 = b_1 \geq b_2 \geq b_3 \geq \cdots \geq b_k$ such that the following policy is optimal:

- $U_1(t) = 0$ if and only if $X(t) \leq b_1$; that is, always accept type-1 tasks unless the buffer is full;
- For $i = 2, \ldots, k$, $U_i(t) = 0$ if and only if $X(t) \leq b_i$; that is, accept type-i tasks if and only if the buffer contains b_i or fewer tasks awaiting service.

The rest of this section is devoted to the proof of this result. The first step will be to prove that V is nondecreasing and convex. To this end, consider the operator T, acting in the

space of functions from $\{0,1,\ldots,B\}$ to $\mathbb{R}$, defined as

$$TU(x) = \delta\mu U(x-1) + \sum_{i=1}^{k} \delta\lambda_i \min[U(x)+C_i, U(x+1)], \qquad x \in \{1,2,\ldots,B-1\}$$

$$TU(0) = \delta\mu U(0) + \sum_{i=1}^{k} \delta\lambda_i \min[U(0)+C_i, U(1)],$$

$$TU(B) = \delta\mu U(B-1) + \sum_{i=1}^{k} \delta\lambda_i (U(B)+C_i), \tag{9}$$

for $U : \{0,1,\ldots,B\} \to \mathbb{R}$. Then the Bellman equation reads $TV = V$. Denote

$$\|U\| = \max_x |U(x)| \tag{10}$$

and let S be the set of functions from $\{0,1,\ldots,B\}$ to $\mathbb{R}$ that are nondecreasing, convex, and having slope bounded by C_1, that is

$$U(x+1) - U(x) \le C_1, \qquad x \in \{0,1,\ldots,B-1\}.$$

The following lemma asserts that T preserves S, and moreover, acts on it as a strict contraction.

Lemma 3.1: One has $TS \subset S$. Moreover, there exists a constant $a \in (0,1)$ such that

$$\|TU - TW\| \le a\|U - W\| \text{ for every } U, W \in S.$$

Proof: To prove the first assertion, let $U \in S$ be given. Then for $2 \le x \le B-1$,

$$\begin{aligned} TU(x) - TU(x-1) &= \delta\mu(U(x-1) - U(x-2)) \\ &+ \sum_{i=1}^{k} \delta\lambda_i \{\min[U(x)+C_i, U(x+1)] \\ &\quad - \min[U(x-1)+C_i, U(x)]\}. \end{aligned} \tag{11}$$

Hence, using the nondecreasing property of U, $TU(x) - TU(x-1) \ge 0$. A similar calculation for $x = 1$ and $x = B$ gives $TU(x) - TU(x-1) \ge 0$ as well, and the nondecreasing property of TU follows.

To show that the slope of TU is bounded by C_1, we use again (11). Since U satisfies such a condition, it follows that $U(x-1) - U(x-2) \le C_1$ and that each of the expressions in curly brackets is bounded by C_1. Since $\delta\mu + \sum_i \delta\lambda_i \le 1$, it follows that $TU(x) - TU(x-1) \le C_1$. A similar calculation for $x = 1$ and $x = B$ gives an analogous result, and it follows that the slope of TU is bounded by C_1.

To prove that TU is convex, we use the fact that if W is any convex function mapping $\{0,1,\ldots,B\}$ to $\mathbb{R}$ and C a constant, then the function $Z : \{0,1,\ldots,B\} \to \mathbb{R}$ defined by

$$Z(x) = \begin{cases} \min[W(x)+C, W(x+1)] & \text{if } x \le B-1, \\ W(B)+C & \text{if } x = B, \end{cases}$$

is also convex. The elementary proof of this fact is omitted. Denote the transformation mapping W to Z by T_C. That is, $Z = T_C W$. Then TU can be written as

$$\delta\mu\tilde{U} + \sum_{i=1}^{k} \delta\lambda_i Z_i,$$

where $Z_i = T_{C_i} U$, and

$$\tilde{U}(x) = \begin{cases} U(x-1) & \text{if } x > 0, \\ U(0) & \text{if } x = 0. \end{cases}$$

Owing to the fact that U is convex and nondecreasing, $\tilde{U}$ is seen to be convex. It follows that TU is convex, as the sum of $k+1$ convex functions.

We have thus shown $TU \in S$. Since $U \in S$ is arbitrary, this proves $TS \subset S$.

To prove the second assertion, let $U, W \in S$. Consider first $x \in \{1,2,\ldots,B-1\}$. By (9), denoting $a \vee b = \max(a,b)$, $a \wedge b = \min(a,b)$ and using the inequality

$$|(a \wedge b) - (c \wedge d)| \le |a-c| \vee |c-d|,$$

we have

$$\begin{aligned} &|TU(x) - TW(x)| \\ &\le \delta\mu |U(x-1) - W(x-1)| \\ &\quad + \sum_{i=1}^{k} \delta\lambda_i [|U(x) - W(x)| \vee |U(x+1) - W(x+1)|] \\ &\le a\,\|U - W\|, \end{aligned} \tag{12}$$

where $a = \delta\mu + \sum_{i=1}^{k} \delta\lambda_i$. By the definition of δ, $a < 1$. For $x = 0$ and $x = B$, the calculation is similar, and gives the same result, namely $|TU(x) - TW(x)| \le a\|U - W\|$. We conclude that $\|TU - TW\| \le a\|U - W\|$. ∎

Proof of Theorem 3.1.

We use the contraction mapping principle (see e.g. [24, Theorem V.18]). The set S, equipped with the metric $\rho(U,W) = \|U - W\|$ is a complete metric space. The map $T : S \to S$ is a strict contraction, as shown in the above lemma. As a result, T has a unique fixed point. That is, there exists a unique $U \in S$ for which $TU = U$. Recall that V is the unique solution to the same equation in the space of *all* functions from $\{0,1,\ldots,B\}$ to $\mathbb{R}$. As a result, $V = U$. This shows $V \in S$, namely, that V is nondecreasing and convex.

In order to show the threshold property of the policy, we employ the method from [31], which builds on convexity. One can read off an optimal feedback control from the Bellman equation (16), as follows. Given $0 \le x \le B-1$, if a class-i arrival occurs when $X(t) = x$, send it to the cloud (and pay C_i) if and only if

$$V(x) + C_i < V(x+1). \tag{13}$$

Since V is convex, $V(x+1) - V(x)$ is nondecreasing in x, and so, if (13) holds for some (i,x), it also holds for (i,x') for all $x < x' \le B-1$. In other words, class-i task acceptance occurs if and only if $X(t) \le b_i$ for suitable constants b_i. The ordering of b_i, as alluded to in the statement of the theorem, is also clear by this argument. It remains to show that $b_1 = B-1$. By the above discussion, it suffices to show that $V(x) + C_1 \ge V(x+1)$ for all $0 \le x \le B-1$. This, however, follows from the fact that the slope of V is bounded by C_1, as $V \in S$. ∎

B. Threshold policy in a multi-server setting

We now turn to the setting of dedicated servers. Since the buffer is common for all k types of tasks the state space is k-dimensional. The process X is given by

$$X(t) = \sum_{i=1}^{k} X_i(t) = x_i + \sum_{i=1}^{k} A_i(t) - \sum_{i=1}^{k} R_i(t) - \sum_{i=1}^{k} D_i(t), \tag{14}$$

where D_i denotes the departure process, counting the number of completed tasks of type i. There is no change in the definitions of A_i, R_i, U_i. The control process $U = \{U_1, U_2, \cdots, U_k\}$ is said to be *admissible* if (i) it is adapted

to the filtration generated by $(\{A_i\}, X)$; and (ii) under U, the process $X(t)$ satisfies the constraint

$$X(t) = \sum_{i=1}^{k} X_i(t) \leq B \text{ for all } t. \quad (15)$$

The definitions of $J(x, U)$ and $V(x)$ undergo no change.

Denote the service rates of the dedicated servers as μ_i, $i \in \{1, .., k\}$, and $\delta = (\sum_i \mu_i + \gamma + \sum_i \lambda_i)^{-1}$. The value function uniquely solves the Bellman equation which is given as follows:

$$V(x) = \delta \sum_{i}^{k} \mu_i V(x - e_i)^+ +$$
$$\sum_{i=1}^{k} \delta\lambda_i \min[V(x + e_i) + C_i, V(x + e_i)] + (\sum_{i}^{k} x_i - B)^+ Y \quad (16)$$

where x is the initial state vector, e_i is a vector of size k with all zeroes at coordinates $j \neq i$ and 1 at coordinate i and Y is some sufficiently large constant. The last component stands for the constraint of no scheduling beyond the buffer limit B. This formulation covers all the boundary conditions and will be useful for the proof (see chpt. 10 of [16] for this method of notation). The scheduling to a private cloud is performed then $U_i = 0$, and the cloudbursting is done then $U_i = 1$. We term the property of switching of U_i from 0 to 1 in adjacent states x, x', $x < x'$, as an increasing of U_i in x.

The policy structure is provided by the following.

Theorem 3.2: For any task type i and any state $\{x_j\}, j \in \{1, \ldots, k\}$, U_i is nondecreasing in x_j.

We prove this result relying on theory presented in [16].

To this end, consider the operator T

$$TU(x) = \delta \sum_{i}^{k} \mu_i V(x - e_i)^+ +$$
$$\sum_{i=1}^{k} \delta\lambda_i \min[V(x + e_i) + C_i, V(x + e_i)] + (\sum_{i}^{k} x_i - B)^+ Y \quad (17)$$

for $U : \{0, 1, \ldots, \infty\} \to \mathbb{R}$. Then, the Bellman equation reads $TV = V$. Note that we do not restrict the state space to $\{0, 1, \ldots, B\}$, because addition of the last component in equation (18) assures that any state for with $\sum_i^k X_i > B$ is unreachable. Consider the norm defined as in (10), and let S be the set of functions from $\{0, 1, \ldots, \infty\}$ to $\mathbb{R}$ that are nondecreasing, convex and supermodular. We stick to the definition of convexity and supermodularity of Chapter 6 in [16] as follows:

- $f(X)$ is convex in x_i if
  $2f(X) \leq f(X + e_i) + f(X - e_i)$
- $f(X)$ is supermodular in x_i, x_j if
  $f(X + e_i + e_j) - f(X + e_j) \leq f(X + e_i) - f(X), \forall j \in \{1, \ldots, k\}$

The following lemma asserts that T preserves S, and moreover, acts on it as a strict contraction.

Lemma 3.2: One has $TS \subset S$. Moreover, there exists a constant $a \in (0, 1)$ such that

$$\|TU - TW\| \leq a\|U - W\| \text{ for every } U, W \in S.$$

Proof: To prove the first assertion, let $U \in S$ be given. Next we split T to the separate operators, associated with departure, controlled arrival (admission) and with the constant Y. We rewrite (18) as follows:

$$TU(x) = \delta \sum_{i}^{k} T_{D1(i)} U(x) + \delta \sum_{i=1}^{k} T_{CA(i)} U(x) + T_Y U(x) \quad (18)$$

To prove that TU is convex, nondecreasing and supermodular we use Theorem 7.2 and Theorem 7.3 of [16], which prove all these properties for the operators $T_{CA(i)}$ and $T_{D1(i)}$ correspondingly. It is easy to see that the operator T_Y preserves the properties as well.

The proof of the second assertion is similar to that of in theorem 3.1 and thus omitted. ■

Proof of Theorem 3.2. Similarly to the proof of 3.1, we use the contraction mapping principle (see e.g. [24, Theorem V.18]) to show that $V \in S$, namely, that V is nondecreasing, convex and supermodular.

In order to show the nondecreasing property of the policy, we again employ the method from [31], which builds on convexity. In a similar way, one sees that since V is convex, $V(X + e_i) - V(X)$ is nondecreasing in x_i, and since V is supermodular, $V(X + e_i) - V(X)$ is nondecreasing in x_j, $\forall j \neq i$. Consequently, class-i task acceptance is nondecreasing in number of tasks of any type present in the private cloud. ■

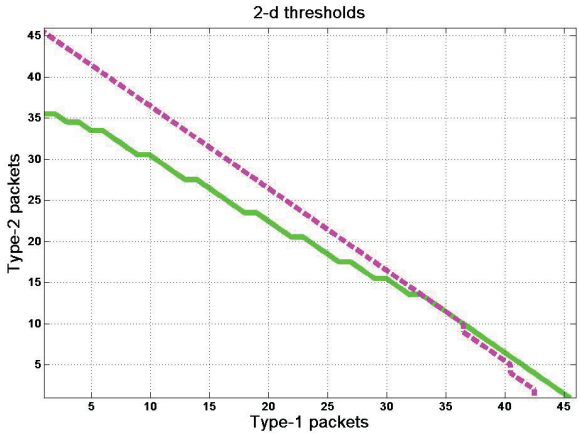

Fig. 3. Example of 2-d threshold policy

The optimal policy demonstrated in Theorem 3.2 implies geometrical properties of scheduling rules, so that there exists a continuous region where the tasks are scheduled to the private cloud. The example of the optimal policy of scheduling in a system of two task types with two dedicated servers is demonstrated in Figure 3. There are two lines which form the regions where the tasks of the corresponding type are scheduled to the private cloud. The intersection of the regionsstands for the states where both task types are accepted to the private cloud.

C. Connection to average cost objectives

We have demonstrated properties of solutions to hybrid cloud optimization problems using MDPs with *discounted* cost/reward criteria. For the long-run goals it makes sense to optimize average cost criteria. The relation between the two types of cost is well-understood (Blackwell optimality, see e.g. [7, Chapter 4.1]). In particular, when the discount factor γ is sufficiently close to zero, the optimal policy for discounted cost is also optimal for the long-run average cost

(ibid.). Consequently, our results are valid for analogous MDP formulations with long-run average cotsts/rewards.

IV. Numerical evaluation and policy adjustment

This section addresses the implementation of optimal scheduling policy computation. The optimal solution and its dependence on the system parameters are numerically evaluated. In the absence of any optimal policy, the numerical calculation process, denoted as *basic value iteration*, is performed as follows:

1) Initialize the vector of the value functions of the size corresponding to the state space.
2) Iteration step: Use the contracting property of the corresponding Bellman equation to perform iterations for the vector of the value functions.
3) Calculate the difference with the results of the previous iteration. If the difference is smaller than a predefined error level, go to the final step, otherwise go to the iteration step.
4) Calculate the optimal policy for each state by applying the part of the Bellman equation which contains the min operator. (Equation 16).

Since the final value function is unknown, the initial values at step $1)$ may be arbitrarily chosen. Note that step $4)$ does not need to be performed for the entire vector of the value functions, but until the threshold is found. The latter stems from the threshold property of the optimal policy. In this section, we present an algorithm which facilitates the implementation of the PA unit, as defined in section II. Note that we invoke the PA once there is a change in one of the parameters, e.g. the pricing of one of the task types. In order to effectively adjust the new scheduling policy which corresponds to this change, we aim to use the previous value functions and the previous policy.

The usage of an approximate value function is aimed to overcome the "curse of dimensionality", the known phenomenon which refers to the complexity of the MDP in systems with large state spaces. The approach is generally addressed in [17], where the approximate representation of the value function and the solid understanding of the system are exploited for the effective solution. In order to fulfil our objective, we study the influence of variations in system parameters on the value functions and on the optimal policy. We give selected examples of threshold policies for sample systems in order to explore the impact of system parameters. Next, we formulate a computationally-effective algorithm for the policy adjustment.

A. Numerical results for the scheduling model

We first observe the thresholds for different systems. For simplicity, we focus on single-server systems but our conclusions are valid for multi-server settings as well. We use current real-world pricing data, taken from leading cloud providers. The examples for one-dimensional models are presented in Table I. Each line refers to a different case, and the prices are taken from the Amazon EC2 pricing list, [1]. In particular, the first case reflect sper-hour prices of high-memory Reserved Instances (RI. The three prices refer to extra large, double extra large, and quadruple extra large, all lightly loaded instances (note that the prices are higher for lightly loaded resources), where each instance is selected to fit the corresponding task type. The second line represents prices for the standard extra large RI, standard large RI, high-memory extra large RI, for medium,light and heavy loads correspondingly. The first two cases in table I demonstrate the dependence of the thresholds on the price differences. As the difference between the prices

TABLE I. *User Hybrid Cloud - Threshold Examples*

private cloud Buffer size	Prices	Thresholds
35	0.288, 0.576, 1.152	6, 31, 35
35	0.185, 0.196, 0.2	14, 34, 35
18	0.049, 0.288, 1.61	7, 16, 18
18	0.025, 0.74, 1.379, 2.605	-1, 9, 17, 18

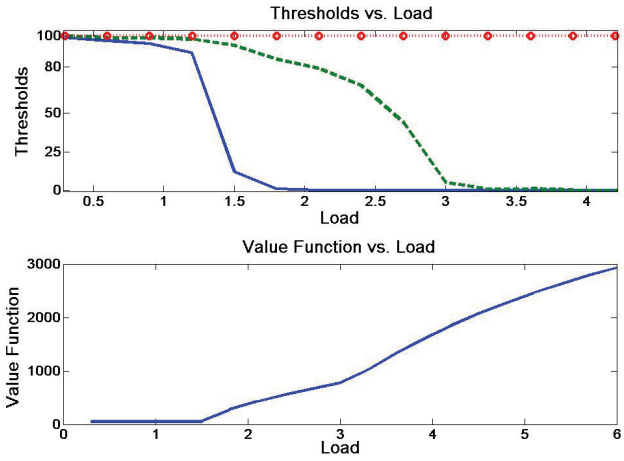

Fig. 4. Load impact simulation

decreases, the thresholds of the cheaper task types increase. The two other cases represent different task settings which include standard, high-memory, cluster GPU, high-CPU, RI under different loads. Note that a threshold value of "-1" means that the task is always cloudburst to the public cloud.

1) Impact of the load: Next, we consider the impact of the local load (i.e. at the private cloud) $\rho = \frac{\mu}{\sum_{i=1}^{K} \lambda_i}$ on the thresholds and on the value function. The setting of three task types priced as in the third case in table I, and of the private cloud buffer of size $B = 100$ is tested under a variable load starting at 0.3 and ascending to 6.0. The results are seen in Figure 4. While the value function increases with the load, the thresholds of the tasks with the lowest price decrease. The latter stems from the fact that the system "prefers" to reserve the space for the most expensive task type. Note that the ordering of the thresholds stays constant. We conclude that the optimal policy for the higher/lower load can be estimated once the optimal policy for the lower/higher load is known.

2) Variations in task prices: We evaluate the effect of the variations in task prices. The price for the certain task type can be altered due to both the change in pricing offered by the public cloud (e.g. dynamic pricing on Amazon, [1]), and to the tasks specification (e.g. tasks need to change from standard VM to advance VM, [9]). Consider the evaluation of a system with three task types. The first and third tasks are as in first case of table I. The pricing of the second task type varies, such that its initial value is equal to the pricing of the first task while the final one is higher than that of the third. The load $\rho < 1$ is fixed. The results are shown in Figure 5. The threshold of the second task type gradually increases until it surpasses the threshold of the third task type. The value function increases as well, while the threshold of the cheapest task decreases.

The "intensity" of the variations in the threshold highly depends on the load. We observe this in the same setting but with the load $\rho = 5$, where $\lambda_i > \mu$ for all i. In this case, the thresholds alternate right once the prices become equal. This is due to the fact that for the high load most of the tasks are

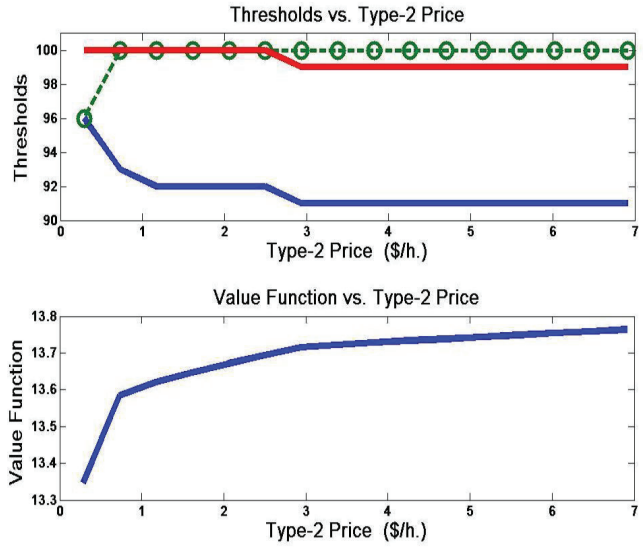

Fig. 5. Value function and the thresholds vs. type-2 price, moderate load

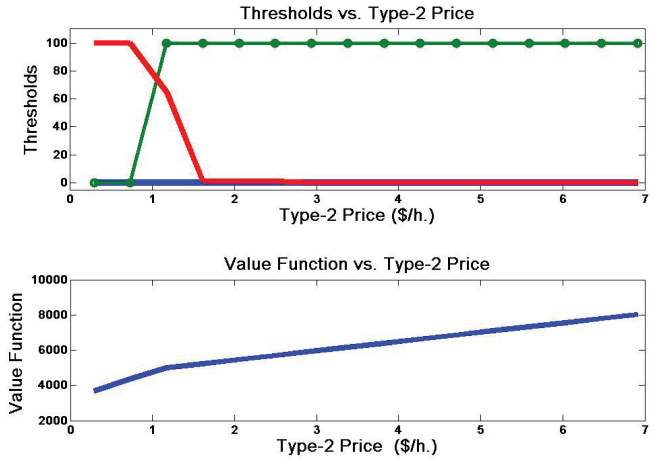

Fig. 6. Value function and the thresholds vs. type-2 price, high load

cloudburst. However, if the arrival rate of the type 2 tasks is significantly lower than μ (see Figure 6), the decrease in the threshold of the type 3 tasks is moderate. This is due to the given low arrival rate and to the observed small cloudbursting probability. The latter causes low cloudbursting intensity. For the same reason, the value function increases very slowly once $C_2 > C_3$.

We conclude that the impact of the system parameters highly depends on the setting and on the interdependence between the different parameters. Consequently, the system dynamics must be well understood, and the update in parameters must be carefully examined. See [28] for more numerical results, examples and discussion.

B. Adjustment of the threshold-type policy

Based on the observations above, we propose a computation-effective algorithm for finding the optimal policy for the system which undergoes changes in its parameters. As an example, consider a single-server system whose thresholds and the vector of the value functions are known. Observe some arbitrary task type i whose threshold is equal to $b(i) < B$. Clearly, $V(x)+C_i \geq V(x+1)$ for all $x < b(i)$. Next, suppose that the pricing of task type i increases. We estimate the new threshold as, say, $b'(i) = b(i) + 1$. Therefore, we increase the value functions by an amount proportional to the increase in the price of task type i in order to get a new approximated vector of $V'(x)$, so that $V'(x) + C_i \geq V'(x+1)$ for all $x < b'(i)$. Then, the policy iteration is continued by using the values of $V'(x)$. The algorithm for policy adjustment is summarized as follows:

1) Observe the optimal solution for the initial system and estimate the new thresholds in accordance with the parameter which has been altered.
2) Estimate the new value function based on the estimation performed in the previous stage.
3) Continue the calculation of the new policy as in *basic policy iteration* using the new estimation in the initialization step.

Although the approximation of the new value function is heuristic, the eventual convergence to the optimal solution in step 3) is guaranteed. We could observe that the speed of the convergence is several orders faster than when estimation step is not used. The actual improvement in complexity increases with the space state. One understands from [17] that a similar method can be applied when the state-space of the new system is increased, e.g. when an additional task type appears or when the private cloud buffer is augmented. In these cases, the estimation step 2) might be less straightforward, although the complexity reduction is significant. We leave the details for the future work.

V. Conclusion

In this paper, the problem of scheduling and cloudbursting in a hybrid-cloud environment is analyzed. We present the system design for the optimal scheduling in a hybrid cloud and the corresponding analytical framework based on Markov Decision Processes. We present a novel approach to optimization problems in cloud computing by means of control theory and MDP methods. The proposed system is logically divided into units which perform tasks of a optimal policy computation, policy adjustment and scheduling. Our work presents the proofs to the threshold-type structure of the optimal policies. Our numerical results are based on current pricing data from leading cloud providers. The results demonstrate the threshold structure of the optimal policy. We exploit the latter for definition of policy adjustment algorithm.

Motivated by the current work, we analyze, in a work in progress, an extended version of the model, with general class-dependent service time distributions, under heavy traffic diffusion approximations. Our results show further structural properties of the optimal control problem in this asymptotic regime. In particular, the problem is shown to undergo a dimensionality reduction, in the form of a state space collapse. This constitutes an additional powerful tool for treating scheduling problems in cloud computing.

Acknowledgments

Research supported in part by the ISF (Grant 1315/12), the US-Israel BSF (Grant 2008466), and the Technion fund for promotion of research.

REFERENCES

[1] Amazon ec2 pricing. Amazon web services. [Online]. Available: http://aws.amazon.com/ec2/pricing/

[2] M. Armbrust, A. Fox, R. Griffith, A. Joseph, R. Katz, A. Konwinski, G. Lee, D. Patterson, A. Rabkin, I. Stoica *et al.*, "Above the clouds: A view of cloud computing," Berkeley, Tech. Rep., 2010.

[3] M. Armony, "Dynamic routing in large-scale service systems with heterogeneous servers," *Queueing Systems*, vol. 51, no. 3, pp. 287–329, 2005.

[4] R. Atar, A. Mandelbaum, and M. Reiman, "Scheduling a multi class queue with many exponential servers: Asymptotic optimality in heavy traffic," *The Annals of Applied Probability*, vol. 14, no. 3, pp. 1084–1134, 2004.

[5] R. Atar, A. Mandelbaum, and G. Shaikhet, "Simplified control problems for multiclass many-server queueing systems," *Mathematics of Operations Research*, vol. 34, no. 4, pp. 795–812, 2009.

[6] M. Bennani and D. Menasce, "Resource allocation for autonomic data centers using analytic performance models," in *Autonomic Computing. Proceedings. Second International Conference on*, 2005, pp. 229–240.

[7] D. Bertsekas, *Dynamic programming and optimal control*. Athena Scientific Belmont, MA, 1995, vol. 1, no. 2.

[8] E. Çil, E. Örmeci, and F. Karaesmen, "Structural results on a batch acceptance problem for capacitated queues," *Mathematical Methods of Operations Research*, vol. 66, no. 2, pp. 263–274, 2007.

[9] Fujitsu global cloud platform. Fujitsu. [Online]. Available: http://globalcloud.us.fujitsu.com/portal/ctrl/top

[10] H. Ghoneim and S. Stidham, "Control of arrivals to two queues in series," *European Journal of Operational Research*, vol. 21, no. 3, pp. 399–409, 1985.

[11] D. Guo and L. Bhuyan, "A QoS aware multicore hash scheduler for network applications," in *INFOCOM, Proceedings IEEE*, 2011, pp. 1089–1097.

[12] B. Hajek, "Optimal control of two interacting service stations," *Automatic Control, IEEE Transactions on*, vol. 29, no. 6, pp. 491–499, 1984.

[13] M. Hajjat, X. Sun, Y. Sung, D. Maltz, S. Rao, K. Sripanidkulchai, and M. Tawarmalani, "Cloudward bound: planning for beneficial migration of enterprise applications to the cloud," *ACM SIGCOMM Computer Communication Review*, vol. 40, no. 4, pp. 243–254, 2010.

[14] J. Harrison and A. Zeevi, "Dynamic scheduling of a multiclass queue in the halfin-whitt heavy traffic regime," *Operations Research*, vol. 52, no. 2, pp. 243–257, 2004.

[15] D. Kondo, B. Javadi, P. Malecot, F. Cappello, and D. Anderson, "Cost-benefit analysis of cloud computing versus desktop grids," in *Parallel & Distributed Processing. IEEE International Symposium on*, 2009, pp. 1–12.

[16] G. Koole, *Monotonicity in Markov reward and decision chains: Theory and applications*. Now Pub, 2007, vol. 1.

[17] G. Koole and P. Nain, "On the value function of a priority queue with an application to a controlled polling model," *Queueing Systems*, vol. 34, no. 1, pp. 199–214, 2000.

[18] V. Kulkarni and T. Tedijanto, "Optimal admission control of markov-modulated batch arrivals to a finite-capacity buffer," *Stochastic Models*, vol. 14, no. 1-2, pp. 95–122, 1998.

[19] H. Liu and D. Orban, "Gridbatch: Cloud computing for large-scale data-intensive batch applications," in *Cluster Computing and the Grid. 8th IEEE International Symposium on*, 2008, pp. 295–305.

[20] D. Nurmi, R. Wolski, C. Grzegorczyk, G. Obertelli, S. Soman, L. Youseff, and D. Zagorodnov, "The eucalyptus open-source cloud-computing system," in *Cluster Computing and the Grid. 9th IEEE/ACM International Symposium on*, 2009, pp. 124–131.

[21] OpenNebula. OpenNebula.org Project. [Online]. Available: http://www.opennebula.org/

[22] M. Puterman, *Markov decision processes: Discrete stochastic dynamic programming*. John Wiley & Sons, Inc., 1994.

[23] B. Raghavan, K. Vishwanath, S. Ramabhadran, K. Yocum, and A. Snoeren, "Cloud control with distributed rate limiting," in *ACM SIGCOMM Computer Communication Review*, vol. 37, no. 4, 2007, pp. 337–348.

[24] M. Reed and B. Simon, *Methods of modern mathematical physics: Functional analysis*. Gulf Professional Publishing, 1980, vol. 1.

[25] M. Salehi and R. Buyya, "Adapting market-oriented scheduling policies for cloud computing," *Algorithms and Architectures for Parallel Processing*, pp. 351–362, 2010.

[26] Y. Seung, T. Lam, L. Li, and T. Woo, "Cloudflex: Seamless scaling of enterprise applications into the cloud," in *INFOCOM, Proceedings IEEE*, 2011, pp. 211–215.

[27] U. Sharma, P. Shenoy, S. Sahu, and A. Shaikh, "Kingfisher: Cost-aware elasticity in the cloud," in *INFOCOM, Proceedings IEEE*, 2010, pp. 206–210.

[28] M. Shifrin, R. Atar, and I. Cidon, "To cloud or not to cloud: Optimizing cloudbursting costs," Tech. Rep. [Online]. Available: http://webee.technion.ac.il/publication-link/index/id/603

[29] S. Stidham Jr, "Optimal control of admission to a queueing system," *Automatic Control, IEEE Transactions on*, vol. 30, no. 8, pp. 705–713, 1985.

[30] R. Van den Bossche, K. Vanmechelen, and J. Broeckhove, "Cost-optimal scheduling in hybrid iaas clouds for deadline constrained workloads," in *Cloud Computing (CLOUD), IEEE 3rd International Conference on*, 2010, pp. 228–235.

[31] J. Walrand, *An introduction to queueing networks*. Prentice Hall Englewood Cliffs, NJ, 1988, vol. 165.

[32] H. Wang, Q. Jing, R. Chen, B. He, Z. Qian, and L. Zhou, "Distributed systems meet economics: pricing in the cloud," in *Proceedings of the 2nd USENIX conference on Hot topics in cloud computing*. USENIX Association, 2010, pp. 6–6.

[33] R. Weber and S. Stidham Jr, "Optimal control of service rates in networks of queues," *Advances in applied probability*, pp. 202–218, 1987.