Admit or Reject? Preserve or Drop? Operational Dilemmas upon Server Failures on the Cloud

Nadav Lavi*
Tel-Aviv University
School of Electrical Engineering
Tel-Aviv, Israel
nadavlavi@post.tau.ac.il

Hanoch Levy
Tel-Aviv University
School of Computer-Science
Tel-Aviv, Israel
hanoch@cs.tau.ac.il

ABSTRACT

Server failures on the cloud introduce acute operational dilemmas as now the cloud management entity needs to handle existing task preservations in addition to new task admissions. These admission and preservation decisions have significant impact on the cloud performance and operational cost, as they impact future system decisions. Should a cloud manager prefer to use resources for new task admissions and increase the risk of dropping an already admitted task in the future? Or should he/she prefer to maintain resources for potential future task preservations at the expense of new task admissions? These dilemmas are even more critical in Distributed Cloud Computing (DCC) due to the small scale of the micro Cloud Computing Center (mCCC).

In this paper we will address these questions through the use of Markov Decision Process (MDP) analysis. We will show that even though the problem appears to be rather complicated (as the two decision rules are coupled), our analysis reveals that it can be significantly simplified (as one of the rules is of a trivial form). These results enables us to compose a holistic framework for cloud computing task management.

Categories and Subject Descriptors

G.3 [**Mathematics of Computing**]: Probability and Statistics—*Markov Processes*; C.4 [**Computer Systems Organization**]: Performance of Systems—*Modeling techniques*

General Terms

Performance, Theory

Keywords

Markov Decision Process, Task Management, Admission Control, Task Preservation

1. INTRODUCTION

Server failures in Cloud computing Centers (CCC) are inevitable[8, 2], due to the large number of servers and components. Such failures derive additional complexity on the CCC task management mechanism, as it is now being required to handle both admission of new tasks and preservation of existing tasks that their servers failed. In Distributed Cloud Computing (DCC), these server failures have major impact due to the small number of computing servers in micro Cloud Computing Center (mCCC) compared to traditional CCC architecture. Thus, an mCCC is required to carefully conduct task management, considering its current available resources as well as the potential future system trajectories, and collaborate with other mCCC's to overcome scenarios in which it can not admit new tasks or preserve existing tasks.

In this paper we construct a new modeling framework which provides a holistic optimization scheme for the combined problems of new task admission and existing task preservation. A policy that can be then be employed by each mCCC according to its number of servers as well as its system settings, without the need to know the status of other mCCC's. We base our model on a queuing-loss multi-server system with failures and address the optimal operation of the CCC. We formulate a reward and cost model, and evaluate the optimal policy of the combined admission/rejection and preservation/dropping decisions using a Markov Decision Process (MDP) [6]. Although the problem seems to be extremely complicated, we reveal that the optimal combined policy can be shown to be of a double switching curve form, and then can be further simplified into a set of rules in each of which one of the rules is of a trivial form. Meaning, the system is either in an always-admit or always-preserve preference. This outcome significantly simplifies the task management operation employed by mCCC. As a result, our scheme eliminates the need for complex real-time mechanisms.

2. SYSTEM MODEL

As indicated above we focus on a single mCCC and investigate the optimal admission and preservation policies. We model an mCCC as a queueing-loss system with M servers. The assumption of a fixed number of servers is reasonable as a computing facility is planned in advance for a certain capacity. The selection of a queueing-loss system is based on the fact that an arriving task will usually expect an immediate response. The fact that a task may be rejected means that it will be transferred to another mCCC. Thus our scheme provides a distributed task-management framework, enabling each mCCC to conduct its own task management using only local system information.

*Nadav Lavi is also with General-Motors Advanced Technical Center - Israel (nadav.lavi@gm.com).

DCC '2015 Portland, Oregon USA

As we investigate optimal admission and preservation policy of a single mCCC, we now model a system composed of a single computing facility. For simplicity we assume a single task type, i.e., single priority level, and all servers are identical (in terms of resources)[1]. Tasks arrive to the mCCC according to a Poisson process with rate λ. If a task is admitted then the system gains a reward $R > 0$ and incurs an operational cost of $C_a \geq 0$ due to the server allocation. If the task is rejected then the system incurs a rejection penalty of $C_p \geq 0$. We note that in our model there is no queue for un-handled tasks, therefore a rejected task can be diverted to another mCCC. Due to the expected large number of mCCC in a DCC system the Poisson characteristics of the overall arrival process into an mCCC can be assumed to hold (approximately). An admitted task service time follows exponential distribution with parameter μ.

Servers in the system are in one of three states: active (serving a task), idle (available server ready to serve a task), and failed (waiting for repair and can not be used before that). We note that both active and idle servers may fail. The duration after which an active or idle server may experience failure is exponentially distributed with parameter μ_f. If an active server fails, the task can be maintained in the system and preserved using a new server with an additional server allocation cost $C_a \geq 0$, or it can be dropped, i.e., stopped being served, with penalty of $C_d \geq 0$. We note that dropped tasks can be treated similarly to rejected tasks, as explained above. Server replacements are handled one after the other with exponentially distributed duration with parameter μ_r.

The goal of the mCCC task management, i.e., the combined admission and preservation decision rules, is to maximize the system *revenue rate*, which equals to the *reward rate* minus the operational costs (tasks' admission, rejection, preservation and drop).As the system *reward rate* is fixed and defined by λR, we can formulate an equivalent cost minimization problem and examine the expected total discounted cost over an infinite horizon. We note that in this case, the rejection cost, denoted as C_r, is consisted of the rejection penalty (C_p) plus R (returning the reward). In Section 5 we investigate the optimal policy behavior in various cost structures.

3. MDP FORMULATION

We denote the number of active, idle and failed servers as m_a, m_i, and m_f, respectively. Therefore $m_a+m_i+m_f = M$, $m_a, m_i, m_f \geq 0$. Hence, it is sufficient to use two parameters (out of the three) to describe the system state space in the MDP formulation. Although the state space formulation using (m_a, m_i) $(m_a + m_i \leq M)$, i.e., using the active and idle servers, is an intuitive approach, we choose to represent the system state space using the active and failed servers (m_a, m_f) $(m_a + m_f \leq M)$. The rational in this transformed representation is our goal to prove certain structural properties of the optimal solution. Therefore, we strive to leverage specific characteristics of the value function, namely nondecreasing, convexity and supermodularity properties, as defined below. These fundamental properties are instrumental in deriving the structure of the decision rule (Theorem 1).

[1]The proposed scheme can be easily extended to multi-task support (by a single server) through the use of task grouping. In cloud computing multi-task is supported by Virtual Machines (VMs) running on the same server.

Throughout this paper we follow the property definitions from [3]. Denoting f as a function from $\mathbb{N}_0^m$ to $\mathbb{R}$, $X \in \mathbb{N}_0^m$ $(m \in \mathbb{N})$, and e_i as a vector of same dimension as X with all zeros and a 1 at the i'th location, we now define the following three properties:

- $f(X)$ is nondecreasing in x_i if $f(X) \leq f(X + e_i)$
- $f(X)$ is convex in x_i if $2f(X+e_i) \leq f(X) + f(X + 2e_i)$
- $f(X)$ is supermodular in x_i, x_j $(1 \leq i < j \leq m)$ if $f(X + e_i) + f(X + e_j) \leq f(X + e_i + e_j) + f(X)$

We define the following operators for $f : \mathbb{N}_0^2 \to \mathbb{R}$:

$$T_{CA(1)}f(x_1, x_2) = \min\{c + f(x_1, x_2), c' + f(x_1 + 1, x_2)\}$$
$$T_{D1(2)}f(x_1, x_2) = f(x_1, (x_2 - 1)^+)$$
$$T_K f(x_1, x_2) = K(x_1 + x_2 - N)^+$$
$$T_{A(2)}f(x_1, x_2) = f(x_1, x_2 + 1)$$
$$T_{D(1,N)}f(x_1, x_2) =$$
$$\min\{x_1, N\}\mu f(x_1 - 1, x_2) + (N - x_1)^+\mu f(x_1, x_2)$$
$$T_{CP}f(x_1, x_2) = \min\{c + f(x_1, x_2 + 1), c' + f(x_1 - 1, x_2 + 1)\}$$

where, $c, c' \in \mathbb{R}$.

The operators $T_{CA(1)}, T_{D1(2)}, T_{A(2)}, T_{D(1,N)}$ model the admission-control (in our system admission of a new task), single departure from the second dimension (which in our system is the repair of a failed server), single arrival to the second dimension (a failure of an idle server in our system), and departure based on load from the first dimension (service completion of a task), respectively. The operator T_K is needed to maintain the system within the original finite state-space. $T_{CP}f(x_1, x_2)$ is a new operator, introduced by our scheme, that handles the decision upon active server failures, i.e., the preservation operator.

As indicated above, we are interested in showing certain properties of the value function. In order to prove them we will examine each of the operators above. We keep the definitions from [3] where: $T : A \to B$ means that if $f \in A$ then $Tf \in B$ $(A \subset B)$. If T maintains all properties of f we say that T preserves f.

We note that $T_{CA(1)}, T_{D1(2)}, T_{A(2)}, T_{D(1,N)}$ were previously investigated in the literature [3] (and references therein). T_K was investigated by [7]. All of the aforementioned operators preserve the nondecreasing, convexity and supermodularity properties. In this study we show that the new the new preservation operator, $T_{CP}f(x_1, x_2)$, preserves both nondecreasing and supermodularity properties, and convexity in one of the dimensions (x_1), in our scheme in the number of the active servers; see [4] for the detailed proof.

We apply the well-known *uniformization* method [3], as well as methods from [3] and [1] to enable us to formulate our state space as an infinite state space MDP. We apply a method presented in [3] and construct our value function (V) using a set of operators defined above in (1). Thus, the Bellman equation reads $TV = V$.

Where in (1), $\Lambda = \lambda + \mu_r + M\mu + 2M\mu_f$ is the overall system events rate (which is Poisson process), and δ is the discount factor $(0 < \delta \leq 1)$. Since all operators preserve the three properties of nondecreasing, supermodularity and convexity in x_1, it follows (using induction on applying the operator T) that $V(x_1, x_2)$ possesses the properties.

$$
\begin{aligned}
V(x_1,x_2) &= \delta\frac{\lambda}{\Lambda}T_{CA(1)}V(x_1,x_2)\\
&+\delta\frac{\min\{m_a,M\}\mu}{\Lambda}T_{D(1,M)}V(x_1,x_2)\\
&+\delta\frac{\mu_r}{\Lambda}T_{D1(2)}V(x_1,x_2)\\
&+\delta\frac{(M-x_1-x_2)^+\mu_f}{\Lambda}T_{A(2)}V(x_1,x_2)\\
&+\delta\frac{\min\{x_1,M\}\mu_f}{\Lambda}T_{CP}V(x_1,x_2)\\
&+\delta\frac{(M-\min\{M,x_1\}+\min\{M,x_1+x_2\})\mu_f}{\Lambda}V(x_1,x_2)\\
&+\delta\frac{(M-x_1)^+\mu}{\Lambda}V(x_1,x_2)\\
&+T_KV(x_1,x_2) \qquad (1)
\end{aligned}
$$

We note that V uniquely solves the Bellman optimality equation, and through V optimal admission and preservation policies can be derived.

4. THE OPTIMAL OPERATIONAL RULES STRUCTURE

On its face value the problem looks quite complex due to being 3 dimensional and to involve two types of operation decisions. Nonetheless - we show that the operational rule can be simplified to a double-switching curve structure. To this end we leverage (1) and use a method presented in [7], which relies on a theory presented in [3]. As a side product, many other properties result from this analysis. We briefly go through the proof structure, and note that the detailed proof appears in [4].

THEOREM 1. *For every fixed value of m_a there are two thresholds level $L_1(m_a)$ and $L_2(m_a)$, such that in state (m_a, m_f) a new task is admitted if and only if $m_f < L_1(m_a)$ and an existing task is preserved if and only if $m_f < L_2(m_a)$.*

[Double Switching Curve] The proof relies on the fact that the operators $T_{CA(1)}$, $T_{D(1,M)}$, $T_{D1(2)}$, $T_{A(2)}$ and T_K preserve the *nondecreasing*, *convexity*, and *supermodularity* properties. The preservation operator, T_{CP}, preserves the *nondecreasing*, *convexity* in one dimension (x_1), and *supermodularity* properties. Based on [3], since T is a linear combination of operators that preserve *nondecreasing*, *convexity* in x_1, and *supermodularity*, then V has the same properties. Moreover T acts as a strict contraction. As a result, both the admission and the preservation policies are of a switching-curve form. A detailed proof is provided in [4].

The outcome of Theorem 1 is that both the admission and preservation decision rules are of switching-curve form, as illustrated in Figure 1a. In the sequel we will show that this general structure can be further simplified, by showing that, depending on particular cases, one of the decision rules is of a trivial form, i.e., the system will be either in always-preserve or always-admit mode; this is illustrated in Figure 1b and Figure 1c, respectively. We will show that the selection between these two rules is based on the relation between rejection (C_r) and drop (C_d) costs. Hence, the influence of the cost structure on the double switching-curve policy is in decoupling the two switching-curves, as one of the switching-curve rules becomes a trivial rule.

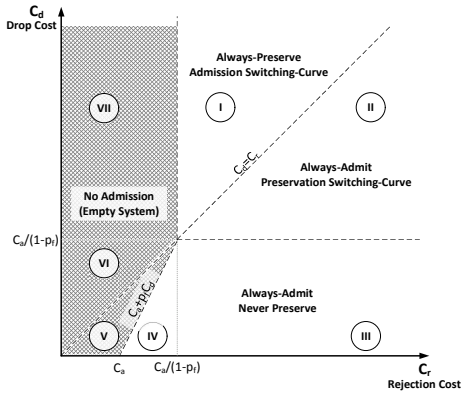


Figure 2: Policy behavior according to the cost structure.

5. DECOUPLING THE RULES

In this section we investigate the impact of the cost structure on the behavior of the admission and preservation policies and the interplay between them. Based on Section 4 the admission and preservation policies are both of a switching-curve form. We show that the general structure of dual switching curve can be further simplified, whereby one of the policies (switching curves) is of a trivial form; that is the system will be either in always-preserve or always-admit mode.

It is important to note that so far the only restriction on cost arguments was $C_a, C_r, C_d \geq 0$. We now assume $C_a < C_r$ which is a necessary condition for the system to admit new tasks, i.e., for the system to be non-empty. Under this assumption, we investigate two types of cost structures $C_d > C_r$ and $C_d < C_r$, and describe the cost regions in which the system conducts admission and/or preservations.

Figure 2 summarizes the outcome of our analysis, depicting the (simplified) operational rule to be exercised as a function of the cost parameters. In Section 5.1 we examine the cost region $C_d > C_r$ (regions I, VI, and VII), and in Section 5.2 we examine the cost region $C_d < C_r$ (regions II-V).

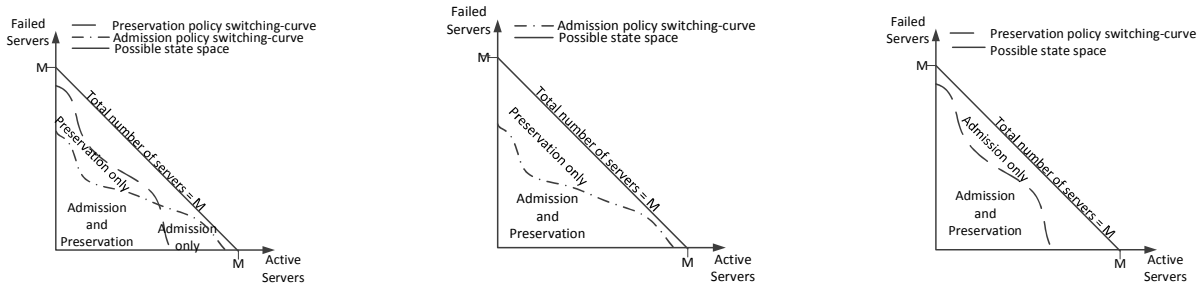
5.1 Rejection is Cheaper than Drop: Never Accept or Always Preserve Systems

In this section we assume $C_d > C_r$, which are regions I, VI, and VII in Figure 2. Under these cost settings one would expect that the system will be biased towards task preservation. We will show that in this system, the preservation policy becomes a trivial rule, that is, it is always beneficial to preserve a user (as long as there are available resources), and the preservation switching-curve coincides with $m_a + m_f = M$. We note that this cost structure relates to the common cloud use-case where preserving an already admitted task is more important than admitting a new task.

We define $p_f = \mu_f/(\mu + \mu_f)$, which is the probability for a server failure prior to task service completion.

LEMMA 1 (REGIONS VI, AND VII). *If $C_r < C_a/(1 - p_f)$ then the system will never accept a new task, i.e., the system is an empty system.*

PROOF. The proof is based on the analysis of potential events upon a task admission and comparison of the overall cost of the event sequences to the cost of immediate rejection (C_r). We will show that for any sequence of preser-



(a) Two switching curves. (b) Trivial preservation. (c) Trivial admission.

Figure 1: Switching-curves structure.

vations followed by a drop event (and incurring C_d), under the aforementioned condition the system will not accept any task.

Given that the drop of the customer occurs on the $(k+1)st$ failure the operation cost is:

$$C_a \sum_{i=0}^{k} (p_f)^i + (p_f)^{k+1} C_d, k \geq 0 \quad (2)$$

We note that $C_d > C_r$ and $C_r < C_a/(1-p_f)$ (the lemma's condition), and substitute them in (2) to yield:

$$C_a \sum_{i=0}^{k} (p_f)^i + (p_f)^{k+1} C_d > (1-p_f) C_r \sum_{i=0}^{k} (p_f)^i + (p_f)^{k+1} C_r = C_r \quad (3)$$

We emphasize that (3) holds for all $k \geq 0$. Hence, as the overall operational cost for all potential preservations and drop sequences is higher than that of rejecting the task upon arrival, the system will never accept a new task. □

We now focus on non-empty systems, that is: $C_r \geq C_a/(1-p_f)$, and investigate the preservation decision.

THEOREM 2 (REGION I). *Consider a system with finite number of servers and $C_r \geq C_a/(1-p_f)$ and $C_d > C_r$. Then for all $(x_1, x_2), x_1 \geq 1, x_2 \geq 0$ such that $x_1 + x_2 < M$ it is optimal to preserve an existing task, i.e., $C_a + V(x_1, x_2+1) \leq C_d + V(x_1 - 1, x_2 + 1)$.*

PROOF. We will prove it by a way of contradiction. For the sake of contradiction assume that the optimal policy is policy B, that in some cases does not obey the preservation rule of the theorem. Let us look at a system applying policy B (system B) and at the first such event in which system B violates the rule, that is it does not preserve a task though it has an idle server. Assume this happens at t_0.

We will construct a policy A that will perform better than policy B. System A applying policy A is identical to system B until t_0 in which it continues to preserve the task dropped by system B. We denote this task as the *extra task*, and the rest of the tasks (if exist) that are in both systems as the *mutual tasks*. Using a sample-path technique, we couple the two systems via the failure, service and arrival times, i.e., all server failures, task departures and arrivals are the same in both systems. Once system B drops the task the value function difference between the systems is $C_d - C_a$. We note that from time t_0 system A precisely imitates system B, until at some point in time, say t_1:

1. A new task arrives and system A needs to reject it (lack of idle servers), but system B can accept the task (due to the additional idle server). Thus in this case the cost difference between the systems is $C_d - C_r$.
2. An active server of one of the mutual tasks fails and system A drops the task, but system B can switch to another server (due to the extra idle server). In this case the cost difference between the two system is 0.
3. The extra task in system A is completed. In this case the cost difference between the systems remains $C_d - C_a$.
4. The server of the extra task in system A fails. System A then decides to drop the task (the delayed drop as mentioned above), which results in a penalty of C_d. In system B this server is idle and therefore system B does not incur any additional cost.

We emphasize that in all four scenarios from t_1 on system A imitates system B (both systems will behave similarly).

We note that each of the aforementioned scenarios occurs with a certain probability which we must account for. Due to the cost structure, it is obvious that $C_d - C_a > 0$ and $C_d - C_r > 0$. We focus on scenarios 3 and 4, both of the scenarios depend on a single task (the extra task) behavior. Therefore, the probabilities of scenarios 3 and 4 are $1 - p_f$ and p_f, respectively. We now evaluate the cost difference between system A and system B. It is obvious that system B overall expected cost is $C_B = C_d$. System A overall expected cost is $C_A = C_a + p_f C_d$. We compare the two costs (C_A and C_B):

$$C_A - C_B = C_a + p_f C_d - C_d = C_a - (1-p_f) C_d \quad (4)$$

Based on the conditions of the theorem, i.e., $C_r \geq C_a/(1-p_f)$, and the fact that $C_d > C_r$, it is clear from (4) that $C_A < C_B$. Thus, system A achieves an overall lower expected system cost than system B, considering all scenarios and their occurrence probabilities. We note that as our investigation is of an arbitrary task at an arbitrary point in time (and hence in system state) we can conclude that it is always beneficial to preserve an already admitted task (as long as there are available servers) when an active server fails. □

5.2 Drop is Cheaper than Rejection: Always Admit Systems

In this section we investigate regions II-V in Figure 2 in which $C_d < C_r$[2]. We first investigate the conditions for task

[2]Systems in which new users are more important than existing ones are sometimes called *overnight operation*.

preservation, and afterwards the behavior of the admission decision. We focus on Regions II-IV. The detailed proofs for all regions can be found in [4].

LEMMA 2 (REGIONS III-V). *If $C_d < C_a/(1-p_f)$ then the system will never conduct task preservations.*

PROOF. The proof relies on the fact that the expected cost of preserving a task is more costly than dropping it, and is similar to the proof of Lemma 1. □

We now investigate the admission conditions when $C_r > C_d$, namely the system is biased towards new admissions. We then prove that under these conditions the admission switching curve is of a trivial form, i.e., the system applies either an always admit or an always reject policy. For clarity we note that the analysis is based on a system with finite number of servers.

THEOREM 3 (REGION V). *If $C_d < C_a/(1-p_f)$ and $C_r < C_a + p_f C_d$ then the system will never accept new tasks and will remain empty.*

PROOF. Based on Lemma 2, when $C_d < C_a/(1-p_f)$ a task will be dropped once it's server fails. Therefore, the overall operational cost considering the potential drop is:

$$C_a + p_f C_d. \tag{5}$$

Hence, if $C_a + p_f C_d > C_r$ it is not beneficial to admit the task and it will be dropped. □

We now investigate the behavior of the admission policy according to the preservation policy. We first consider the case in which preservation is not beneficial and afterwards investigate the case in which preservation is beneficial yet rejection of a new task is more costly than dropping an existing task, i.e., system is biased towards admission.

THEOREM 4 (REGIONS III AND IV). *If $C_r > C_d$, $C_r > C_a + p_f C_d$ and $C_a/(1-p_f) > C_d$ then for all $(x_1, x_2), x_1 \geq 1, x_2 \geq 0$, such that $x_1 + x_2 < M$, it is optimal to admit a new task, i.e., $C_a + V(x_1+1, x_2) \leq C_r + V(x_1, x_2)$.*

PROOF. In this region an admitted task is dropped once its server fails (never preserve region). Nonetheless, since $C_r > C_a + p_f C_d$, even accounting for the potential drop cost the admission is beneficial. Hence, the system will always conduct admissions. □

THEOREM 5 (REGION II). *If $C_r > C_d$ and $C_d \geq C_a/(1-p_f)$ then for all $(x_1, x_2), x_1 \geq 1, x_2 \geq 0$, such that $x_1 + x_2 < M$, it is optimal to admit an existing task, i.e., $C_a + V(x_1+1, x_2) \leq C_r + V(x_1, x_2)$.*

[Always Admit] In this region we show the system is operating in an always admit regime, and preservation follows a switching-curve policy. We note that the proof somewhat resembles that of Theorem 2 and involves an analysis based on sample-path but with a different cost structure. A detailed proof is given in [4].

6. THE OPTIMAL POLICY EVALUATION

We use numerical results to compare the optimal policy with a static policy which employs an always-admit and always-preserve rules, and with a semi-static policy which admits new tasks as long as 50% of the non-active servers are idle. We use the comparison method from [5], and compare the value function at the origin ($V(0,0)$) of the systems with the static and semi-static policies to the one of the optimal switching-curve. We model a system with $M = 500$ servers and $\delta = 0.9999$ and the following set of parameters: $\lambda = 10, 25, 50, 100$, $\mu_r = 1/30, 1/60, 1/150$, $\mu = 1, 1/15, 1/30$, $\mu_f = 1/10, 1/30, 1/60$, $(C_d, C_r, C_a) = (500, 50, 1)$. Figure 3 depicts the probability density of the relative error of the static and semi-static policies for the 108 cases. It can be seen the static policy can reach high error of above 100% compared to the optimal policy. The semi-static policy behaves even worse, and reaches errors above 200%, which means 3 times higher than the optimal policy.

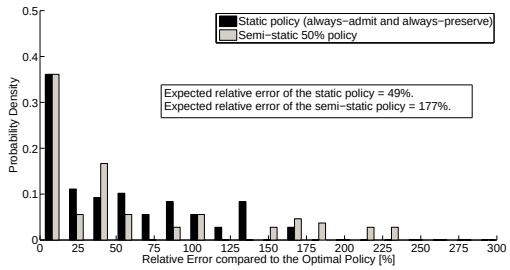


Figure 3: Probability density of the relative error.

7. REFERENCES

[1] E. Altman, T. Jimenez, and G. Koole. On optimal call admission control in resource-sharing system. *Communications, IEEE Transactions on*, 49(9):1659–1668, Sep 2001.

[2] P. Garraghan, P. Townend, and J. Xu. An empirical failure-analysis of a large-scale cloud computing environment. In *High-Assurance Systems Engineering (HASE), 2014 IEEE 15th International Symposium on*, pages 113–120, Jan 2014.

[3] G. Koole. Monotonicity in markov reward and decision chains: Theory and applications. *Foundations and Trends in Stochastic Systems*, 1(1):1–76, 2006.

[4] N. Lavi and H. Levy. Ovecoming server failures in cloud computing task management. Technical report.

[5] R. Milito and H. Levy. Modeling and dynamic scheduling of a queueing system with blocking and starvation. *Communications, IEEE Transactions on*, 37(12):1318–1329, Dec 1989.

[6] M. L. Puterman. *Markov decision processes : discrete stochastic dynamic programming*. Wiley series in probability and mathematical statistics. John Wiley & Sons, New York, 1994. A Wiley-Interscience publication.

[7] M. Shifrin, R. Atar, and I. Cidon. Optimal scheduling in the hybrid-cloud. In *Integrated Network Management (IM 2013), 2013 IFIP/IEEE International Symposium on*, pages 51–59, May 2013.

[8] K. V. Vishwanath and N. Nagappan. Characterizing cloud computing hardware reliability. In *Proceedings of the 1st ACM Symposium on Cloud Computing*, SoCC '10, pages 193–204, New York, NY, USA, 2010. ACM.