Theoretical Analysis and Tuning of Decentralized Probabilistic Auto-Scaling

Bogdan Alexandru Caprarescu, Eva Kaslik, Dana Petcu

Faculty of Mathematics and Computer Science, West University of Timișoara, Timișoara 300223, România
Research Institute e-Austria Timișoara, Timișoara 300223, România

Abstract

A major impediment towards the industrial adoption of decentralized distributed systems comes from the difficulty to theoretically prove that these systems exhibit the required behavior. In this paper, we use probability theory to analyze a decentralized auto-scaling algorithm in which each node probabilistically decides to scale in or out. We prove that, in the context of dynamic workloads, the average load of the system is maintained within a variation interval with a given probability, provided that the number of nodes and the variation interval length are higher than certain bounds. The paper also proposes numerical algorithms for approximating these minimum bounds.

Keywords: auto-scaling, decentralized computing, probability analysis, probabilistic algorithms

1. Introduction

Scalability has always been a critical non-functional requirement for parallel and distributed systems that receive a variable workload, but with the advent of cloud computing, scalability is no longer bounded by the amount of physical resources that were initially allocated to the system. The on-demand provisioning of virtualized computing nodes in the cloud allows providers of Internet services to rent only the amount of resources needed to serve the current workload and then scale in and out to cope with the request rate variations [1].

The dynamic and unpredictable nature of the workload experienced by many Web applications, which may suddenly become popular or loose most of their users in favor of other more trendy applications, requires autonomic scaling mechanisms. Therefore, the autonomic provisioning of virtualized computing nodes in the cloud allows providers of Internet services to rent only the amount of resources needed to serve the current workload and then scale in and out to cope with the request rate variations [1].

The dynamic and unpredictable nature of the workload experienced by many Web applications, which may suddenly become popular or loose most of their users in favor of other more trendy applications, requires autonomic scaling mechanisms. Therefore, the autonomic provisioning of virtualized computing nodes in the cloud allows providers of Internet services to rent only the amount of resources needed to serve the current workload and then scale in and out to cope with the request rate variations [1].

The dynamic and unpredictable nature of the workload experienced by many Web applications, which may suddenly become popular or loose most of their users in favor of other more trendy applications, requires autonomic scaling mechanisms. Therefore, the autonomic provisioning of virtualized computing nodes in the cloud allows providers of Internet services to rent only the amount of resources needed to serve the current workload and then scale in and out to cope with the request rate variations [1].

The dynamic and unpredictable nature of the workload experienced by many Web applications, which may suddenly become popular or loose most of their users in favor of other more trendy applications, requires autonomic scaling mechanisms. Therefore, the autonomic provisioning of virtualized computing nodes in the cloud allows providers of Internet services to rent only the amount of resources needed to serve the current workload and then scale in and out to cope with the request rate variations [1].

The dynamic and unpredictable nature of the workload experienced by many Web applications, which may suddenly become popular or loose most of their users in favor of other more trendy applications, requires autonomic scaling mechanisms. Therefore, the autonomic provisioning of virtualized computing nodes in the cloud allows providers of Internet services to rent only the amount of resources needed to serve the current workload and then scale in and out to cope with the request rate variations [1].

The dynamic and unpredictable nature of the workload experienced by many Web applications, which may suddenly become popular or loose most of their users in favor of other more trendy applications, requires autonomic scaling mechanisms. Therefore, the autonomic provisioning of virtualized computing nodes in the cloud allows providers of Internet services to rent only the amount of resources needed to serve the current workload and then scale in and out to cope with the request rate variations [1].

The dynamic and unpredictable nature of the workload experienced by many Web applications, which may suddenly become popular or loose most of their users in favor of other more trendy applications, requires autonomic scaling mechanisms. Therefore, the autonomic provisioning of virtualized computing nodes in the cloud allows providers of Internet services to rent only the amount of resources needed to serve the current workload and then scale in and out to cope with the request rate variations [1].

The dynamic and unpredictable nature of the workload experienced by many Web applications, which may suddenly become popular or loose most of their users in favor of other more trendy applications, requires autonomic scaling mechanisms. Therefore, the autonomic provisioning of virtualized computing nodes in the cloud allows providers of Internet services to rent only the amount of resources needed to serve the current workload and then scale in and out to cope with the request rate variations [1].

The dynamic and unpredictable nature of the workload experienced by many Web applications, which may suddenly become popular or loose most of their users in favor of other more trendy applications, requires autonomic scaling mechanisms. Therefore, the autonomic provisioning of virtualized computing nodes in the cloud allows providers of Internet services to rent only the amount of resources needed to serve the current workload and then scale in and out to cope with the request rate variations [1].

The dynamic and unpredictable nature of the workload experienced by many Web applications, which may suddenly become popular or loose most of their users in favor of other more trendy applications, requires autonomic scaling mechanisms. Therefore, the autonomic provisioning of virtualized computing nodes in the cloud allows providers of Internet services to rent only the amount of resources needed to serve the current workload and then scale in and out to cope with the request rate variations [1].
we use probability theory to analyze the behavior of the algorithm and provide the potential customer of DEPAS with methods for tuning its parameters in such a way that a certain level of trust can be guaranteed.

Actually, one of the main challenges towards the widespread use of decentralized systems in the industry comes from the difficulty to guarantee that the system will exhibit the expected behavior. Moreover, the achievement of a global property usually depends on the proper configuration of some system parameters. Finding such a proper configuration represents the tricky part of designing a decentralized system. For the DEPAS algorithm the global property to be maintained is the average load of the system and the main configuration parameter is the length of the load variation interval. By keeping the average load within the variation interval the system avoids oscillations (i.e., additions and removals mixed in a row) and the total number of nodes stays close to the optimal value.

Due to the randomized and decentralized nature of our algorithm it is not possible to guarantee that a global property is achieved in absolute terms. Therefore, we adopt a probabilistic approach and prove that the algorithm keeps the average load within the variation interval with a certain probability, called correctness probability. Intuitively, given the length of the load variation interval, the correctness probability increases with the increase in the number of nodes. Reversely, for a certain number of nodes, the correctness probability grows with the increase of the load variation interval. Thus, in both scenarios, we are interested if there are some minimum bounds so that a minimum correctness probability can be guaranteed for any configuration in which the number of nodes and the interval length are higher than the minimum bounds, respectively.

To address the above problem, in this paper, the DEPAS algorithm is modeled as a set of Bernoulli trials. In this way, using probability theory, we formally prove that: (i) given the desired load, the load variation interval, and a minimum correctness probability, there is a minimum number of nodes so that the correctness probability is higher than the given threshold for any actual load and for any number of nodes higher than the given minimum; (ii) given the desired load, the number of nodes and a minimum correctness probability, there is a minimum interval length so that the correctness probability is higher than the minimum threshold for any actual load and for any interval length higher than the minimum. Additionally, from the formal analysis of DEPAS, we derive numerical algorithms for computing upper bounds on the minimum thresholds (based on the Chernoff-Hoeffding bounds) and compare these upper bounds with an estimation of the real minimums (based on the binomial formula for Bernoulli trials). The algorithms for both scenarios can be used to tune DEPAS at design time. Moreover, the algorithm corresponding to the second scenario can be also used at runtime to dynamically adjust the variation interval length of each node according to the continuously changing size of the system.

The remaining of this paper is organized as follows. Section 2 describes the DEPAS algorithm and formulates the problem to be solved. The problem is formally analyzed and a proof is given in Section 3. The algorithms for tuning DEPAS are described in Section 4, while Section 5 discusses the tuning algorithms in the light of some experimental results. Related work is shown in Section 6. Finally, Section 7 concludes the paper.

2. Decentralized probabilistic auto-scaling (DEPAS)

The DEPAS algorithm was described in [7] as part of a middleware for deploying massively scalable services in a cloud infrastructure. In this section, the algorithm is introduced with an emphasis on the problem of finding a subset of the parameter configurations for which a minimum correctness probability can be guaranteed.

We assume a system composed of a set of computing nodes. A node can be either a physical machine or a virtual machine allocated from a cloud provider on a utility basis. The nodes are homogenous in the sense that they have the same hardware configuration and run the same piece of software comprising both the functional service and the components of our middleware: overlay management, load balancing, and auto-scaling. The overlay management algorithm organizes the nodes into an unstructured overlay network where each node has a fixed degree and a low standard deviation in-degree. This is an adapted version of the gossip-based protocol developed by Jelasity et al. [8].

For load balancing we use a combined approach: a DNS is employed to assign the address of a node to each client while an internal decentralized load balancing protocol (such as the dimension exchange protocol [9]) moves requests between neighbors in order to equalize the load across nodes.

In the following, the term node will be used to denote both the machine and the service instance running on that machine. Thus, the main parameters of a node are the capacity and the load. The capacity of a node is the maximum number of requests per second that can be processed by the service deployed on that node and is derived through
benchmarking. As the nodes are homogenous, they have the same capacity. The load of a node is computed at a given moment in time, as the percentage of the average number of requests per second that were scheduled on that node over a certain timeframe with respect to the capacity of the node. Then, in a homogenous system, the average load of the system is simply the arithmetic average of the loads of all nodes. Note that in the case when the workload received by the system overcomes its capacity, the average load is supra-unitary.

Under these considerations, the goal of the auto-scaling algorithm is to maintain the average load of the system, noted with $L$, within a given interval, $(L_0 - \delta, L_0 + \delta)$, where $L_0$ is called desired load and $\delta$ defines the allowed variation of the load (see Table 1 for a complete list of notations). To achieve this goal each node can execute two types of actions: remove itself and allocate one or more other nodes.

The DEPAS algorithm is shown in listing 1. It is periodically run by each node and begins by retrieving an estimation of the average load of the system. Note that the average load is not computed at this time, but just retrieved from the component running the average protocol. Although there are gossip-based solutions for computing the average load of the system at each node [10], in our experiments [7], for simplicity and higher scalability, we approximated the average load of the system by the average load of the node and its neighbors. If the load is less than or equal to $L_0 - \delta$, then the node computes a probability indicator using Eq. (1) and, because the indicator is sub-unitary in this case, the node uses it as the probability to remove itself. Otherwise, if the load is higher than or equal to $L_0 + \delta$, then the probability indicator is computed using the same equation. In this situation, the indicator can be supra-unitary where its integer part represents the number of nodes to be added for sure, while its fractional part is used as the probability to add another node. Note that the random() function generates a uniformly distributed random decimal number between 0 and 1.

$$pi = \frac{|L - L_0|}{L_0}$$

(1)

The desired load, $L_0$, is subject of a tradeoff. On one hand, a high value reduces the number of nodes, but, in case of a sudden workload increase, leads to a severe degradation of the system performance before the system has the chance to allocate new nodes. On the other hand, a low desired load increases the tolerance of the system to sudden traffic bursts at the cost of allocating many nodes. As the desired load has a high impact on both performance and cost, the customer of DEPAS is in the best position to set its value.

However, while the meaning and impact of the desired load is straightforward for the customer, this is not the case when it comes to the load variation threshold, $\delta$. Actually, even though the customer should not need to care about $\delta$, they are definitely interested in two aspects of the algorithm, namely correctness and accuracy, that are directly impacted by $\delta$. By correctness we understand the ability of the system to make a right provisioning decision in the first place, thus avoiding oscillations (i.e., additions and removals mixed in a row). Due to the randomized nature of DEPAS it is not possible to evaluate its correctness in absolute terms (i.e., correct or incorrect). Therefore, we introduce the notion of correctness probability and allow the customer to define the correctness of the algorithm by specifying a minimum threshold of the correctness probability (see Definition 1). Then, the accuracy of a correct algorithm is a measure of how close the allocated number of nodes is to the optimal one. Therefore, the accuracy of DEPAS is directly influenced by $\delta$: the lower $\delta$ is, the higher the accuracy is. In this paper, we provide a formal analysis of the link between correctness probability, load variation threshold, and number of nodes. Our goal is to provide the customer with an analytical method for configuring DEPAS so that it meets given correctness requirements.
Algorithm 1 DEPAS

while true do
    wait(time frame)
    $L \leftarrow \text{getAverageSystemLoad}()$
    if $L \leq L_0 - \delta$ then
        $pi \leftarrow \text{computeProbabilityIndicator}(L, L_0)$
        $p \leftarrow pi$
        if $p < \text{random}()$ then
            removeSelf()
        end if
    else
        if $L \geq L_0 + \delta$ then
            $pi \leftarrow \text{computeProbabilityIndicator}(L, L_0)$
            $m \leftarrow \lfloor pi \rfloor$
            $p \leftarrow \{pi\}$
            if $p < \text{random}()$ then
                $m \leftarrow m + 1$
            end if
            addNodes($m$)
        end if
    end if
end while

with an as high as possible accuracy.

**Definition 1.** Let $L_0$ and $\delta$ be the desired load and the load variation threshold of a DEPAS instance, respectively. Then, the correctness probability is the probability that, after DEPAS has been run by each node, the new average load is in the interval $(L_0 - \delta, L_0 + \delta)$. The correctness probability is noted with $P(L_0 - \delta < L < L_0 + \delta)$. Consequently, we say that an instance of DEPAS is correct if its correctness probability is higher than or equal to a given threshold, denoted by $P_0$.

More concretely, we consider two usage scenarios of the DEPAS algorithm. In both scenarios, the customer specifies the minimum correctness probability, $P_0$. In the first scenario – called Min $n$ – the customer is able to predict the minimum workload of the system and implicitly the minimum number of nodes and they are interested in the values of $\delta$ for which the correctness probability is higher than $P_0$. The second scenario – called Min $\delta$ – is concerned with the case when the customer has set a small value for $\delta$ in order to obtain a very accurate algorithm and wants to find out for which system sizes a correctness probability higher than $P_0$ can be guaranteed. Both scenarios are applied at design time before deploying DEPAS in the production environment. However, we can imagine the Min $\delta$ scenario being also used at runtime as a subroutine of DEPAS in order to dynamically adjust the value of $\delta$ at each node, provided that an estimation of the system size is available at each node. By applying the Min $\delta$ scenario at runtime we expect to increase the accuracy of DEPAS. However, as opposed to the design-time version, a runtime Min $\delta$ algorithm, being executed at each node, must be fast and have a low resource consumption in order not to overload the system.

In this paper, we address the above challenges for the simplified case when each node computes the same probability indicator, $pi$. This implies that each node precisely estimates the average system load. We also assume that all nodes have synchronized clocks and simultaneously run the DEPAS algorithm, although this constraint is not needed in practice. In other words, the algorithm is assumed to work in cycles where, in each cycle, each node uses the same probability to decide upon the execution of a scaling action.

Looking back at listing 1, we notice that if $pi \geq 1$, which happens when $L \geq 2L_0$, then $n[pi]$ is the number of nodes that are added for sure (where $n$ is the number of existing nodes), which represents the deterministic part of the decision. Therefore, they have no impact on the correctness probability and the case when $L \geq 2L_0$ is reducible to the case when $L \in \{L_0 + \delta, 2 \cdot L_0\}$. On the other hand, the addition and removal cases are symmetric. Consequently,
for simplicity and without loss of generality, we will consider only the addition case. Under these considerations, the formula for computing the probability of each node, \( p \), is given by Eq. (2).

\[
p = \frac{L - L_0}{L_0}, \quad L \in [L_0 + \delta, 2L_0).
\]

We want to express the correctness probability in function of the number of added nodes. For this purpose, Theorem 1 defines the notion of optimal number of nodes to be added as a rational number. In Theorem 2, the correctness probability is expressed in function of the number of added nodes and the optimal number of nodes corresponding to the bounds of the load variation interval.

**Theorem 1.** Let \( L_0 \in (0, 1) \), a system composed of \( n \) homogenous nodes, and \( L \in [L_0 + \delta, 2L_0) \) the average load of the system. Then, the optimal number of nodes to be added to the system so that the new average load will be equal to \( L_0 \) denoted by \( M(n, L, L_0) \), is given by

\[
M(n, L, L_0) = n\frac{L - L_0}{L_0}.
\]

**Proof.** Let \( m \) be the number of nodes to be added and \( C \) the common capacity of all nodes. As the system has the same workload before and after provisioning the new nodes, we have \( nLC = (n + m)L_0C \), from where it turns out that \( m = n\frac{L - L_0}{L} \).

**Theorem 2.** Let \( L_0 \in (0, 1) \), a system composed of \( n \) homogenous nodes, \( L \in [L_0 + \delta, 2L_0) \) the average load of the system, and \( L' \in (0, 2L_0) \) the average load of the system after the addition of \( m \) nodes. Then, the correctness probability is equal to the probability of \( m \in (M(n, L, L_0 + \delta), M(n, L, L_0 - \delta)) \), as expressed by the formula below:

\[
P(L_0 - \delta < L' < L_0 + \delta) = P(M(n, L, L_0 + \delta) < m < M(n, L, L_0 - \delta)).
\]

**Proof.** As the workload of the system remains unchanged, we have \( nL = (n + m)L' \), which implies that \( L' = \frac{nL}{n + m} \).

Then, \( P(L_0 - \delta < L' < L_0 + \delta) = P(L_0 - \delta < \frac{nL}{n + m} < L_0 + \delta) = P\left(\frac{1}{L_0 - \delta} > \frac{n + m}{nL} > \frac{1}{L_0 + \delta}\right) = P\left(\frac{n}{L_0 + \delta} - n < m < \frac{n}{L_0 - \delta} - n\right) = P(n\frac{L_0 - \delta}{L_0 + \delta} < m < n\frac{L_0 + \delta}{L_0 - \delta}) = P(M(n, L, L_0 + \delta) < m < M(n, L, L_0 - \delta)) \) (by applying Theorem 1).

In this section, we described the DEPAS algorithm and the problem we want to solve: finding whether and in which conditions a minimum correctness probability can be guaranteed. This section also prepared the ground for the formalization and theoretical analysis of the problem, which falls within the scope of the next section.

3. Theoretical analysis

As stated in the previous section, the DEPAS algorithm is assumed to work in cycles. In each cycle, each node uses the same probability to decide whether to add a new node or not. Consequently, a cycle of the algorithm can be modeled as a set of Bernoulli trials, \( X_1, X_2, ..., X_n \). We denote by \( S_n \) the outcome of the experiment, which is the number of nodes added in the respective cycle. Under this formalization, Theorem 3 proves that the formula for \( p \) described by Eq. (2) was correctly chosen.

**Theorem 3.** Let us consider a cycle of the DEPAS algorithm where each node adds a new node with probability \( p = (L - L_0)/L_0 \). Then, the expected number of added nodes is equal to the optimal number of nodes to be added.

**Proof.** If \( S_n \) is the number of added nodes, then taking into consideration that \( S_n \) has a binomial distribution of parameters \( n \) and \( p \) it follows that \( E(S_n) = np = M(n, L, L_0) \) (according to Theorem 1).

Moreover, the correctness probability can be computed by using the binomial probability distribution:

\[
P(M(n, L, L_0 + \delta) < m < M(n, L, L_0 - \delta)) = \sum_{i=1}^{M(n, L, L_0 - \delta)} \binom{[M(n, L, L_0 - \delta)]}{i} p^i(1 - p)^{n-i}
\]

The problem with the expression from the left-hand side of Eq. (4) is that it is discontinuous and non-monotonic with respect to \( L \) and non-monotonic with respect to \( n \) and \( \delta \), thus increasing the risk of errors when using it with
Proof. To simplify this problem, we will use the notations (re-scaling) as a set of $n$ Bernoulli trials. Then, the following two affirmations hold.

Theorem 4. Let us consider the DEPAS algorithm under the above assumptions. A cycle of the algorithm is modeled as a set of $n$ Bernoulli trials. Then, the following two affirmations hold.

1. For any $P_0 \in (0, 1)$ and $\delta = \delta_0$, there exist $n^* = n^*(P_0, \delta_0) \in \mathbb{Z}_+$ such that
   \[ P(\text{M}(n, L, L_0 + \delta_0) < S_n < \text{M}(n, L, L_0 - \delta_0)) \geq P_0, \quad \forall L \in (L_0 + \delta, 2L_0), \forall n \geq n^*. \]  \(5\)

2. For any $P_0 \in (0, 1)$ and $n = n_0 \in \mathbb{Z}_+$ satisfying
   \[ 1 - \left(\frac{1}{3}\right)^{n_0} - \left(\frac{2}{3}\right)^{n_0} > P_0, \]  \(6\)

there exists $\delta^* = \delta^*(P_0, n_0) \in (0, 1)$ such that

\[ P(\text{M}(n_0, L, L_0 + \delta) < S_{n_0} < \text{M}(n_0, L, L_0 - \delta)) \geq P_0, \quad \forall L \in (L_0 + \delta, 2L_0), \forall \delta \geq \delta^*. \]

\(7\)

Proof. To simplify this problem, we will use the notations (re-scaling)

\[ \frac{L - L_0}{L_0} = p \in (0, 1) \quad \text{and} \quad \frac{\delta}{L_0} = \varepsilon \in (0, 1). \]

Denoting

\[ \text{Prob}(p, \varepsilon, n) = P(\text{M}(n, L, L_0 + \delta) < S_n < \text{M}(n, L, L_0 - \delta)) = P\left(\frac{L - L_0 - \delta}{L_0 + \delta} < \frac{S_n}{n} < \frac{L - L_0 + \delta}{L_0 - \delta}\right) \]

\[ = P\left(\frac{p - \varepsilon}{1 + \varepsilon} < \frac{S_n}{n} < \frac{p + \varepsilon}{1 - \varepsilon}\right), \]

our problem is twofold:

a. For a fixed $\varepsilon = \varepsilon_0$, estimate the smallest value of $n \in \mathbb{Z}_+$ such that $\text{Prob}(p, \varepsilon_0, n) \geq P_0$, for any $p \geq \varepsilon_0$;

b. For a fixed $n = n_0$, estimate the smallest value of $\varepsilon \in (0, 1)$ such that $\text{Prob}(p, \varepsilon, n_0) \geq P_0$, for any $p \geq \varepsilon$.

It can be easily seen that

\[ \text{Prob}(p, \varepsilon, n) = 1 - P\left(\frac{S_n}{n} \leq \frac{p - \varepsilon}{1 + \varepsilon}\right) - P\left(\frac{S_n}{n} \geq \frac{p + \varepsilon}{1 - \varepsilon}\right). \]

First, as $0 \leq \frac{p - \varepsilon}{1 + \varepsilon} \leq p$, the theorem of Chernoff-Hoeffding bounds provides the following inequality:

\[ P\left(\frac{S_n}{n} \leq \frac{p - \varepsilon}{1 + \varepsilon}\right) \leq e^{-nD[\frac{\varepsilon}{1 + \varepsilon}, \frac{p}{1 + \varepsilon}]}, \]

where

\[ D[x, y] = x \ln \frac{x}{y} + (1 - x) \ln \frac{1 - x}{1 - y} \]
represents the Kullback-Leibler divergence between Bernoulli distributed random variables with parameters \( x \) and \( B \) respectively, \( x, y \in (0, 1) \).

On the other hand, we notice that if \( p > 1 - 2\varepsilon \), we have \( \frac{p + \varepsilon}{1 - \varepsilon} > 1 \) and hence, \( P\left( \frac{S_n}{n} \geq \frac{p + \varepsilon}{1 - \varepsilon} \right) = 0 \). However, if \( p \leq 1 - 2\varepsilon \), we have \( p \leq \frac{p + \varepsilon}{1 - \varepsilon} \leq 1 \) and the theorem of Chernoff-Hoeffding bounds provides

\[
P\left( \frac{S_n}{n} \geq \frac{p + \varepsilon}{1 - \varepsilon} \right) \leq e^{-nD}\left[ \frac{\varepsilon}{p} \right].
\]

Therefore, we obtain the following lower bounds for the probability \( \text{Prob}(p, \varepsilon, n) \):

\[
\text{Prob}(p, \varepsilon, n) \geq 1 - e^{-nD}\left[ \frac{\varepsilon}{p} \right] - e^{-nD}\left[ \frac{\varepsilon}{1 + \varepsilon} \right], \quad \forall \ p \in [\varepsilon, 1 - 2\varepsilon], \ \varepsilon \in \left(0, \frac{1}{3}\right), \ n \in \mathbb{Z}_+ \quad (8)
\]

and

\[
\text{Prob}(p, \varepsilon, n) \geq 1 - e^{-nD}\left[ \frac{\varepsilon}{p} \right], \quad \forall \ p \geq \max[\varepsilon, 1 - 2\varepsilon], \ \varepsilon \in (0, 1), \ n \in \mathbb{Z}_+. \quad (9)
\]

In the following, let \( B_1(p, \varepsilon, n) \) be the function from the right hand side of the inequality (8)

\[
B_1(p, \varepsilon, n) = 1 - e^{-nD}\left[ \frac{\varepsilon}{p} \right] - e^{-nD}\left[ \frac{\varepsilon}{1 + \varepsilon} \right], \quad \forall \ p \in [\varepsilon, 1 - 2\varepsilon], \ \varepsilon \in \left(0, \frac{1}{3}\right), \ n \in \mathbb{Z}_+
\]

and \( B_2(p, \varepsilon, n) \) the function from the right hand side of the inequality (9)

\[
B_2(p, \varepsilon, n) = 1 - e^{-nD}\left[ \frac{\varepsilon}{p} \right], \quad \forall \ p \geq \max[\varepsilon, 1 - 2\varepsilon], \ \varepsilon \in (0, 1), \ n \in \mathbb{Z}_+
\]

Both functions \( B_1 \) and \( B_2 \) are continuously differentiable on their respective domains of definition. It is easy to see that

\[
\frac{\partial B_1}{\partial \varepsilon}(p, \varepsilon, n) = D\left[ \frac{p - \varepsilon}{1 + \varepsilon} \right] e^{-nD}\left[ \frac{\varepsilon}{p} \right] + D\left[ \frac{p + \varepsilon}{1 - \varepsilon} \right] e^{-nD}\left[ \frac{\varepsilon}{1 + \varepsilon} \right] > 0
\]

because of the positiveness of the Kullback-Leibler divergence, and hence, the function \( B_1 \) is strictly increasing with respect to \( n \).

On the other hand,

\[
\frac{\partial B_1}{\partial \varepsilon}(p, \varepsilon, n) = n \frac{\partial}{\partial \varepsilon} \left( D\left[ \frac{p - \varepsilon}{1 + \varepsilon} \right] e^{-nD}\left[ \frac{\varepsilon}{p} \right] + n \frac{\partial}{\partial \varepsilon} \left( D\left[ \frac{p + \varepsilon}{1 - \varepsilon} \right] e^{-nD}\left[ \frac{\varepsilon}{1 + \varepsilon} \right] \right),
\]

where

\[
\frac{\partial}{\partial \varepsilon} \left( D\left[ \frac{p - \varepsilon}{1 + \varepsilon} \right] \right) = \frac{1 - p}{(1 + \varepsilon)^2} \left( \ln \left( 1 + \frac{2\varepsilon}{1 - p} \right) - \ln \left( 1 - \frac{\varepsilon}{p} \right) \right) > 0
\]

\[
\frac{\partial}{\partial \varepsilon} \left( D\left[ \frac{p + \varepsilon}{1 - \varepsilon} \right] \right) = \frac{1 + p}{(1 - \varepsilon)^2} \left( \ln \left( 1 + \frac{2\varepsilon}{1 - p} \right) - \ln \left( 1 - \frac{\varepsilon}{p} \right) \right) > 0
\]

for any \( \varepsilon \in \left(0, \frac{1}{3}\right) \) and \( p \in [\varepsilon, 1 - 2\varepsilon] \). Therefore, the function \( B_1 \) is strictly increasing with respect to \( \varepsilon \) as well.

In a similar way, it follows that \( B_2 \) is also strictly increasing with respect to the variables \( \varepsilon \) and \( n \).

**Case a.** When \( \varepsilon = \varepsilon_0 \) is fixed (i.e. \( \delta = \delta_0 = \varepsilon_0 L_0 \)), we have two subcases.

**a.1.** If \( \varepsilon_0 \in \left(0, \frac{1}{3}\right) \) it can be proved that for any \( p \in [\varepsilon_0, 1 - 2\varepsilon_0] \), there exists a unique \( n_1(p) \in (0, \infty) \) satisfying the equation

\[
B_1(p, \varepsilon_0, n_1(p)) = P_0.
\]

Indeed, for \( p \in [\varepsilon_0, 1 - 2\varepsilon_0] \) arbitrarily fixed, the function \( n \mapsto B_1(p, \varepsilon_0, n) \) is continuous, strictly increasing, \( B_1(p, \varepsilon_0, 0) = -1 \) and \( \lim_{n \to \infty} B_1(p, \varepsilon_0, n) = 1 \). Therefore, \( n \mapsto B_1(p, \varepsilon_0, n) \) is a bijective mapping between \((0, \infty)\) and the
interval \((-1, 1)\). As \(P_0 \in (0, 1)\), there exists a unique solution \(n_1(p) \in (0, \infty)\) of the equation \(B_1(p, \varepsilon_0, n) = P_0\). Moreover, the implicit function theorem guarantees that the function \(n_1(p)\) defined above is continuously differentiable on \([\varepsilon_0, 1 - 2\varepsilon_0]\), and hence, it is bounded. We denote

\[
n_1^* = \sup_{p \in [\varepsilon_0, 1-2\varepsilon_0]} n_1(p).
\]

On the other hand, for any \(p \in (1 - 2\varepsilon_0, 1)\), there exists a unique \(n_2(p) \in (0, \infty)\) satisfying the equation

\[
B_2(p, \varepsilon_0, n_2(p)) = P_0,
\]
given explicitly by

\[
n_2(p) = \frac{-\ln(1 - P_0)}{D \left(\frac{p - \varepsilon_0}{1 + \varepsilon_0}, p\right)}.
\]

As \(\lim_{p \to 1} D \left(\frac{p - \varepsilon_0}{1 + \varepsilon_0}, p\right) = \infty\), we obtain that \(n_2(p)\) is bounded on the interval \((1 - 2\varepsilon_0, 1)\) and let

\[
n_2^* = \sup_{p \in (1 - 2\varepsilon_0, 1)} n_2(p).
\]

Considering

\[
n^* = n^*(P_0, \varepsilon_0) = \left\{ \max\{n_1^*, n_2^*\} \right\} \in \mathbb{Z}_+
\]

from the fact that \(B_1\) is increasing with respect to \(n\), we clearly have

\[
B_1(p, \varepsilon_0, n) \geq B_1(p, \varepsilon_0, n^*) \geq B_1(p, \varepsilon_0, n_1^*) \geq B_1(p, \varepsilon_0, n_1(p)) = P_0, \quad \forall p \in [\varepsilon_0, 1 - 2\varepsilon_0], \forall n \geq n^*,
\]

and therefore,

\[
Prob(p, \varepsilon_0, n) \geq P_0, \quad \forall p \in [\varepsilon_0, 1 - 2\varepsilon_0], \forall n \geq n^*.
\]

Similarly, as \(B_2\) is increasing with respect to \(n\), we obtain

\[
B_2(p, \varepsilon_0, n) \geq B_2(p, \varepsilon_0, n^*) \geq B_2(p, \varepsilon_0, n_2^*) \geq B_2(p, \varepsilon_0, n_2(p)) = P_0, \quad \forall p \in (1 - 2\varepsilon_0, 1), \forall n \geq n^*,
\]

and therefore,

\[
Prob(p, \varepsilon_0, n) \geq P_0, \quad \forall p \in (1 - 2\varepsilon_0, 1), \forall n \geq n^*.
\]

In conclusion, we obtain:

\[
Prob(p, \varepsilon_0, n) \geq P_0, \quad \forall p \geq \varepsilon_0, \forall n \geq n^*.
\]

a.2. If \(\varepsilon_0 \geq \frac{1}{3}\), we know from inequality (9) that

\[
Prob(p, \varepsilon_0, n) \geq B_2(p, \varepsilon_0, n), \quad \forall p \geq \varepsilon_0, \ n \in \mathbb{Z}_+.
\]

For any \(p \in [\varepsilon_0, 1)\), the unique solution of the equation \(B_2(p, \varepsilon_0, n) = P_0\) is \(n_2(p)\) given by (10). Denoting

\[
n^* = n^*(P_0, \varepsilon_0) = \sup_{p \in [\varepsilon_0, 1)} n_2(p),
\]
as \(B_2\) is increasing with respect to \(n\), we obtain

\[
B_2(p, \varepsilon_0, n) \geq B_2(p, \varepsilon_0, n^*) \geq B_2(p, \varepsilon_0, n_2(p)) = P_0, \quad \forall p \in [\varepsilon_0, 1), \forall n \geq n^*,
\]

and hence

\[
Prob(p, \varepsilon_0, n) \geq P_0, \quad \forall p \geq \varepsilon_0, \forall n \geq n^*.
\]
Case b. When \( n = n_0 \in \mathbb{Z}_+ \) is fixed, we will consider two scenarios.

b.1 For \( p \leq \frac{1}{3} \), the lower bound for the probability \( \text{Prob}(p, e, n_0) \) is given by \( B_1(p, e, n_0) \) according to (8).

Let \( p \in \left( 0, \frac{1}{3} \right] \) be arbitrarily fixed. The equation

\[
B_1(p, e, n_0) = P_0
\]

has at most one solution \( e(p) \leq p \). Indeed, we have \( B_1(p, 0, n_0) = -1 \) and

\[
\lim_{e \to p} B_1(p, e, n_0) = 1 - (1 - p)^{n_0} - e^{-n_0} \left[ \frac{2}{3}, p \right]
\]

It is a simple calculus exercise to show that the function

\[
h_1(p) = 1 - (1 - p)^{n_0} - e^{-n_0} \left[ \frac{2}{3}, p \right]
\]

is strictly increasing on the interval \( \left( 0, \frac{1}{3} \right) \), \( \lim_{p \to 0} h_1(p) = -1 \) and

\[
\lim_{p \to 1} h_1(p) = 1 - \left( \frac{1}{3} \right)^{n_0} - \left( \frac{2}{3} \right)^{n_0} > P_0
\]

according to inequality (6). It follows that there exists a unique \( p^*_1 \in \left( 0, \frac{1}{3} \right] \) such that \( h_1(p^*_1) = P_0 \). Therefore, if \( p \in (0, p^*_1) \), the equation \( B_1(p, e, n_0) = P_0 \) has no solution, but if \( p \in \left( p^*_1, \frac{1}{3} \right] \), the equation \( B_1(p, e, n_0) = P_0 \) has a unique solution \( e_1 \), satisfying \( e_1(p) \leq p \). Moreover, from the implicit function theorem we obtain that \( e_1(p) \) is continuously differentiable on \( \left( p^*_1, \frac{1}{3} \right] \) and therefore, it is bounded.

Note that \( \lim_{p \to p^*_1} e_1(p) = p^*_1 \) because we have \( \lim_{p \to p^*_1} B_1(p^*_1, e, n_0) = h_1(p^*_1) = P_0 \), and hence, we can extend the function \( e_1 \) by continuity, considering \( e_1(p^*_1) = p^*_1 \).

Denoting

\[
e_1^* = \sup_{p \in \left[ p^*_1, \frac{1}{3} \right]} e_1(p),
\]

we first observe that \( e_1^* \geq e_1(p^*_1) = p^*_1 \) and therefore, using the fact that \( B_1 \) is increasing with respect to \( e \), we obtain

\[
B_1(p, e^*_1, n_0) \geq B_1(p, e_1(p), n_0) = P_0, \quad \forall p \in \left( e_1^*, \frac{1}{3} \right).
\]

(11)

b.2 For \( p > \frac{1}{3} \), the lower bound for \( \text{Prob}(p, e, n_0) \) can be expressed from (8) and (9) as

\[
B(p, e, n_0) = \begin{cases} 
B_1(p, e, n_0), & \text{if } e \in (0, \frac{1-p}{2}) \\
B_2(p, e, n_0), & \text{if } e \in \left[ \frac{1-p}{2}, p \right]
\end{cases}
\]

The function \( B \) is increasing with respect to \( e \)

\[
\lim_{e \to 0} B(p, e, n_0) = \lim_{e \to 0} B_1(p, e, n_0) = -1
\]

and

\[
\lim_{e \to p} B(p, e, n_0) = \lim_{e \to p} B_2(p, e, n_0) = 1 - (1 - p)^{n_0} > 1 - \left( \frac{2}{3} \right)^{n_0} > P_0
\]

The function \( B \) has a jump discontinuity at \( e = \frac{1-p}{2} \):
It is easy to check that the functions $h_1(p)$ and $h_3(p)$ defined above are continuous and strictly decreasing on the interval $\left(\frac{1}{3}, 1\right)$ and $h_2(p) < h_3(p)$ for any $p \in \left(\frac{1}{3}, 1\right)$. Moreover:

$$\lim_{p \to \frac{1}{3}^-} h_2(p) = 1 - \left(\frac{2}{3}\right)^{n_0} - \left(\frac{1}{3}\right)^{n_0} > P_0 \quad \text{and} \quad \lim_{p \to 1^-} h_2(p) = -1,$$

and

$$\lim_{p \to \frac{1}{3}^-} h_3(p) = 1 - \left(\frac{2}{3}\right)^{n_0} > P_0 \quad \text{and} \quad \lim_{p \to 1^-} h_3(p) = 0.$$

Therefore, there exist unique values $p_2^*, p_3^* \in \left(\frac{1}{3}, 1\right)$, such that $h_2(p_2^*) = P_0$ and $h_3(p_3^*) = P_0$.

If $p \in \left(\frac{1}{3}, p_2^*\right)$, it follows that

$$\lim_{\varepsilon \to \frac{1}{3}^+} B(p, \varepsilon, n_0) = \lim_{\varepsilon \to \frac{1}{3}^+} B_1(p, \varepsilon, n_0) = h_2(p) > h_2(p_2^*) = P_0$$

and hence, since $B_1$ is increasing with respect to $\varepsilon$, the equation $B_1(p, \varepsilon, n_0) = P_0$ has a unique solution $\varepsilon_2(p)$, satisfying $\varepsilon_2(p) \leq \frac{1-p}{p} < \frac{1}{4}$. Moreover, from the implicit function theorem we obtain that $\varepsilon_2(p)$ is continuously differentiable on $\left(\frac{1}{3}, p_2^*\right)$, and therefore, it is bounded. We can extended $\varepsilon_2$ by continuity by considering $\varepsilon_2(p_2^*) = \frac{1-p_2^*}{2}$.

Denoting

$$\varepsilon_2^* = \sup_{p \in \left(\frac{1}{3}, p_2^*\right)} \varepsilon_2(p),$$

and using the fact that $B$ is increasing with respect to $\varepsilon$, we obtain

$$B(p, \varepsilon_2^*, n_0) \geq B(p, \varepsilon_2(p), n_0) = B_1(p, \varepsilon, n_0) = P_0, \quad \forall \ p \in \left[\frac{1}{3}, p_2^*\right]. \tag{12}$$

If $p \in (p_2^*, p_3^*)$ we have $h_3(p) \geq h_3(p_3^*) = P_0$ and $\varepsilon_2^* \geq \varepsilon_2(p_2^*) = \frac{1-p_2^*}{2} > \frac{1-p}{2}$. Taking into consideration that $B_2$ is increasing with respect to $\varepsilon$, we obtain:

$$B(p, \varepsilon_2^*, n_0) = B_2(p, \varepsilon_2^*, n_0) \geq \lim_{\varepsilon \to \frac{1}{3}^+} B_2(p, \varepsilon, n_0) = h_3(p) \geq P_0, \quad \forall \ p \in (p_2^*, p_3^*]. \tag{13}$$

If $p \in (p_3^*, 1)$, since $h_3(p) < h_3(p_3^*) = P_0$, we have that

$$\lim_{\varepsilon \to \frac{1}{3}^+} B_2(p, \varepsilon, n_0) = h_3(p) < P_0 \quad \text{and} \quad \lim_{\varepsilon \to 0} B_2(p, \varepsilon, n_0) > P_0$$

and therefore, the equation $B_2(p, \varepsilon, n_0) = P_0$ has a unique solution $\varepsilon_3(p)$ such that $\frac{1-p}{2} < \varepsilon_3(p) < p$. The function $\varepsilon_3(p)$ is continuously differentiable, bounded, and it can be extended by continuity, considering $\varepsilon_3(p_3^*) = \frac{1-p_3^*}{2}$.

Denoting

$$\varepsilon_3^* = \sup_{p \in [p_3^*, 1)} \varepsilon_3(p),$$

and using the fact that $B$ is increasing with respect to $\varepsilon$, we obtain

$$B(p, \varepsilon_3^*, n_0) \geq B(p, \varepsilon_3(p), n_0) = B_2(p, \varepsilon, n_0) = P_0, \quad \forall \ p \in (p_3^*, 1). \tag{14}$$

Let

$$\varepsilon^* = \max\{\varepsilon_1^*, \varepsilon_2^*, \varepsilon_3^*\}.$$

Combining the inequalities (11), (12), (13) and (14), and taking into consideration that $B$ is increasing with respect to $\varepsilon$, we obtain:

$$B(p, \varepsilon, n_0) \geq B(p, \varepsilon^*, n_0) \geq P_0, \quad \forall \ p \geq \varepsilon^*, \ \forall \varepsilon \geq \varepsilon^*.$$
and finally:

\[ \text{Prob}(p, \varepsilon, n_0) \geq P_0, \quad \forall \ p \geq \varepsilon^*, \ \forall \varepsilon \geq \varepsilon^*. \]

The proof is now complete.

\[ \square \]

**Remark 1.** Instead of the Chernoff-Hoeffding inequalities used in the proof of the previous Theorem, it is possible to compute a lower bound of the correctness probability by means of the well known one-sided Chebyshev inequalities. For example, in a similar manner as in the previous proof, one may obtain the following formula for the estimate of \( n^* \), using Chebyshev bounds:

\[
\begin{align*}
\min n^* = \begin{cases}
\max \left\{ \frac{P_0(1 + \varepsilon_0^2) + \sqrt{(1 - \varepsilon_0^2)^2 + 4P_0^2\varepsilon_0^2}}{8\varepsilon_0^2(1 - P_0)}, \frac{P_0}{1 - P_0} \cdot \frac{(1 - 2\varepsilon_0)(1 + \varepsilon_0)^2}{2\varepsilon_0(1 - \varepsilon_0)^2} \right\}, & \text{if } \varepsilon_0 < \frac{1}{3}, \\
\frac{P_0}{1 - P_0} \cdot \frac{1 - \varepsilon_0}{\varepsilon_0}, & \text{if } \varepsilon_0 \geq \frac{1}{3}.
\end{cases}
\end{align*}
\]  

However, we note that Chernoff-Hoeffding inequalities lead to better results, since they give exponentially decreasing bounds on tail distributions, while Chebyshev inequalities yield only power-law bounds on tail decay.

### 4. Algorithms for DEPAS tuning

In this section, based on the results of the previous section, we formulate the algorithms for estimating the minimum number of nodes and the minimum load variation threshold, respectively, for which the Chernoff-Hoeffding lower bounds of the correctness probability are higher than or equal to the given \( P_0 \). In order to get an idea of how close the estimations provided by the Chernoff-Hoeffding bounds are to the real minimums, we also provide algorithms for computing \( \min n \) and \( \min \delta \) based on the binomial formula. The functions used in the algorithms together with their properties are shown in Table 2. These properties (continuity, differentiability, monotony) determine whether the functions can be used with some numerical optimization algorithms and their specific meaning was explained for each function in the proof of Theorem 4.

#### 4.1. Algorithms based on the Chernoff-Hoeffding bound

Theorem 4 used Chernoff-Hoeffding inequalities to derive a lower bound on the correctness probability. The bound is expressed by the \( B_1(p, \varepsilon, n) \) and \( B_2(p, \varepsilon, n) \) functions (see Table 2). By fixing either \( \varepsilon \) (in the Min \( n \) scenario) or \( n \) (in the Min \( \delta \) scenario), \( B_1 \) and \( B_2 \) become functions of two variables and one problem is to find the minimum value of the second variable (either \( n \) or \( \varepsilon \), respectively) so that the bound is higher than or equal to the given \( P_0 \), for any possible value of \( p \).

**Algorithm 2** \( \min y \) monotonic

**Input:** Function \( f : I_1 \times I_2 \to \mathbb{R} \) continuously differentiable on the intervals \( I_1 \) and \( I_2 \) and strictly increasing on \( I_2 \), precisions \( s_1 \) and \( s_2 \).

**Output:** Minimum \( y^* \in I_1 \) so that \( f(x, y^*) \geq 0 \forall x \in I_1 \)

Let \( l \) be an empty list

for all \( x' \in I_1 \) increasing with step \( s_1 \)

- Use a root finding method to derive the root of the equation \( f(x', y) = 0 \) with precision \( s_2 \).
- Let \( y' \) be the found root.
- Append \( y' \) to \( l \).
end for

\( y^* \leftarrow \max(l) \)

Algorithm 2 provides a solution to the above problem for a generic function of two variables \( f : I_1 \times I_2 \to \mathbb{R} \), that shares the same properties as \( B_1 \) and \( B_2 \). As in general, \( B_1 \) and \( B_2 \) are neither monotonic nor convex with respect to...
Table 2: Functions used in the tuning algorithms

| Function          | Properties                                                                 |
|-------------------|-----------------------------------------------------------------------------|
| $B_1(p, \varepsilon, n) = 1 - e^{-nD[\frac{\varepsilon}{1-\varepsilon}]} - e^{-nD[\frac{\varepsilon}{1-\varepsilon}]}$ | Continuously differentiable, strictly increasing with respect to $\varepsilon$ and $n$ when $p \in [\varepsilon, 1 - 2\varepsilon]$, $\varepsilon \in \left(0, \frac{1}{2}\right)$, $n \in \mathbb{Z}_+$ |
| $B_2(p, \varepsilon, n) = 1 - e^{-nD[\frac{\varepsilon}{1-\varepsilon}]}$ | Continuously differentiable, strictly increasing with respect to $\varepsilon$ and $n$ when $p \geq \max\{\varepsilon, 1 - 2\varepsilon\}$, $\varepsilon \in (0, 1)$, $n \in \mathbb{Z}_+$ |
| $s(p) = -\frac{\ln(1 - P_0)}{D[\frac{p - \varepsilon_0}{1 - \varepsilon_0}, p]}$ | Continuously differentiable and concave when $p \in (1 - 2\varepsilon_0, 1)$, $\varepsilon_0 \in \left(0, \frac{1}{3}\right)$ |
| $h_1(p) = 1 - (1 - p)^{\varepsilon_0} - e^{-nD[\frac{p}{1-p}]}$ | Continuously differentiable, strictly increasing on $\left(0, \frac{1}{3}\right)$ |
| $h_2(p) = 1 - e^{-nD[\frac{p}{1-p}]} - p^{\varepsilon_0}$ | Continuously differentiable, strictly decreasing on $\left(\frac{1}{3}, 1\right)$ |
| $h_3(p) = 1 - e^{-nD[\frac{p}{1-p}]}$ | Continuously differentiable, strictly decreasing on $\left(\frac{1}{3}, 1\right)$ |
| $g(n) = 1 - \left(\frac{2}{3}\right)^n - \left(\frac{1}{3}\right)^n$ | Continuously differentiable, strictly increasing on $\left(0, \infty\right)$ |
| $\text{Bin}(L, \delta, n) = \sum_{i=[M(n,L,\delta)]}^{[M(n,L,\delta)+\delta]} \binom{i}{n} p^i (1-p)^{n-i}$ | Discontinuous and non-monotonic with respect to $L$ and non-monotonic with respect to $\delta$ and $n$ |
Algorithm 3 Cernoff min n

Input: \( L_0 \in (0, 1), \delta_0 \in (0, L_0), \delta_0 \in (0, 1), \text{ precisions } s_n \text{ and } s_p \)

Output: \( n^* \in \mathbb{Z}_+ \)

Compute \( N \) with formula 15

\[
\epsilon_0 \leftarrow \frac{\epsilon}{\delta_0}
\]

if \( \epsilon_0 \in (0, \frac{1}{2}) \) then

Use algorithm 2 to find the minimum \( n_1^* \in 1..N \) for which \( f(p,n_1^*) = B_1(p,\epsilon_0,n_1^*) - \delta_0 \geq 0, \forall p \in [\epsilon_0, 1 - 2\epsilon_0] \), \( s_1 = s_p, s_2 = s_n \)

Use a concave optimization method to find \( n_2^* \leftarrow \sup_{p \in (1-2\epsilon_0, 1)} s(p) \) with precision \( s_p \)

\[
\max\{n_1^*, n_2^*\} = n^*
\]

end if

if \( \epsilon_0 \in [\frac{1}{4}, 1) \) then

Use a concave optimization method to find \( n_3^* \leftarrow \sup_{p \in \epsilon_0, 1} s(p) \) with precision \( s_p \)

\[
\max\{n_3^*\} = n^*
\]

end if

Algorithm 2 is needed by both Algorithm 3 (i.e., Cernoff min n) and Algorithm 4 (i.e., Cernoff min \( \delta \)). In fact, Algorithms 3 and 4 just translate the proof of Theorem 4 into algorithmic language. Note that Algorithm 3 uses the Chebyshev inequality (which is weaker than the Chernoff-Hoeffding inequality) to compute an upper bound of the number of nodes.

Algorithm 4 Cernoff min \( \delta \)

Input: \( L_0 \in (0, 1), n_0 \in \mathbb{Z}_+, \delta_0 \in (0, 1), \text{ precisions } s_e \text{ and } s_p \)

Output: \( \delta^* \in (0, L_0) \)

if \( 1 - \left(\frac{3}{4}\right)^{\frac{1}{3}} - \left(\frac{1}{3}\right)^{\frac{1}{3}} \leq \delta_0 \) then

exit with no solution

else

Use a root finding method to derive the unique solution \( \epsilon_1^* \in (0, \frac{1}{2}) \) of the equation \( h1(p) = \delta_0 \) with precision \( s_p \)

Use algorithm 2 to find the minimum \( \epsilon_1^* < \epsilon_1^* \) for which \( f(p,\epsilon_1^*) = B_1(p,\epsilon_1^*, n_0) - \delta_0 \geq 0, \forall p \in \left\{\epsilon_1^*, \frac{1}{2}\right\}, s_1 = s_p, s_2 = s_n \)

Use a root finding method to derive the unique solution \( \epsilon_2^* \in \left(\frac{1}{3}, 1\right) \) of the equation \( h2(p) = \delta_0 \) with precision \( s_p \)

Use algorithm 2 to find the minimum \( \epsilon_2^* < \epsilon_2^* \) for which \( f(p,\epsilon_2^*) = B_1(p,\epsilon_2^*, n_0) - \delta_0 \geq 0, \forall p \in \left(\frac{1}{3}, \epsilon_2^*\right], s_1 = s_p, s_2 = s_e \)

Use a root finding method to derive the unique solution \( \epsilon_3^* \in \left(\frac{1}{3}, 1\right) \) of the equation \( h3(p) = \delta_0 \) with precision \( s_p \)

Use algorithm 2 to find the minimum \( \epsilon_3^* < \epsilon_3^* \) for which \( f(p,\epsilon_3^*) = B_2(p,\epsilon_3^*, n_0) - \delta_0 \geq 0, \forall p \in \left[\epsilon_3^*, 1\right), s_1 = s_p, s_2 = s_e \)

\[
\epsilon^* = \max\{\epsilon_1^*, \epsilon_2^*, \epsilon_3^*\}
\]

\[
\delta^* \leftarrow \epsilon^* L_0
\]

end if

Note that, from a theoretical point of view, the accuracy of Algorithm 2 (and implicitly that of Algorithms 3 and 4) may be influenced by the precision \( s_1 \). The possible correlation between the output accuracy and the precision is checked experimentally in Section 5.
4.2. Algorithms based on the binomial formula

The binomial formula, expressed by function $Bin(L, \delta, n)$ from Table 2, allows us to compute the exact value of the correctness probability. However, it is more difficult to compute the minimum $n$ or $\delta$ based on the binomial formula than based on the Chernoff-Hoeffding bounds because the binomial function $Bin(L, \delta, n)$ is discontinuous with respect to $L$ and non-monotonic with respect to $\delta$ and $n$. A solution is given by Algorithm 5 where $x$ accounts for either $\delta$ or $n$ and $y$ represents $L$. The main idea is to check for each $y$ starting from its maximum possible value and decreasing by a given step whether the function is positive for all $x$ taken with a certain precision. The solution is given by the minimum $y$ for which the function is positive for all considered values of $x$.

Concrete algorithms for approximating the minimum $n$ (Binomial min $n$) and the minimum $\delta$ (Binomial min $\delta$) are obtained by instantiating Algorithm 5 for functions $f_1(x, y) = f_1(L, n) = Bin(L, \delta_0, n) - P_0$ and $f_2(x, y) = f_2(L, \delta) = Bin(L, \delta, n_0) - P_0$, respectively. A maximum bound for $n$ can be computed based on either Chebyshev or Chernoff-Hoeffding inequalities.

Note that, due to the discontinuity of the binomial function, the binomial algorithms are prone to numerical errors and, therefore, not recommended to be used by the customer of DEPAS. Their sole objective is to allow us to estimate the accuracy of the Chernoff algorithms as shown in the next section.

5. Experimental results

The DEPAS tuning algorithms described in the previous section were implemented in Java. We used simple root and supremum finding algorithms based on the bisection method. The DEPAS Tuning Tool, which provides both graphical and command line facilities for running the tuning algorithms in a large spectrum of scenarios, is available for download [13].

The DEPAS Tuning Tool was used to obtain the experimental results presented in this section. There are three objectives of the experiments: (i) to estimate the accuracy of Chernoff algorithms by comparing their results with the ones of the binomial algorithms, (ii) to make an idea about the execution time of the tuning algorithms, and (iii) to see how the accuracy of Chernoff algorithms is affected by the precision of the load. Two experiments (one with Min $n$ algorithms and another one with Min $\delta$ algorithms) were performed for each objective, thus resulting in a total of six experiments.

All experiments were performed on Amazon EC2 Large Instances (7.5 GB memory, 4 EC2 Compute Units, 64-bit platform) running Amazon Linux. In all experiments we set $L_0 = 0.8$, and $P_0 = 0.99$. The default precisions were $s_0 = 10^{-1}$, $s_\epsilon = 10^{-3}$, and $s_p = 10^{-3}$.

In the first experiment, the number of nodes was computed with both Chernoff and binomial algorithms for several values of $\delta$ taken from the interval $[0.05, 0.15]$ with step 0.005. From Figure 2 we can see that Chernoff min $n$ and Binomial min $n$ algorithms give relatively close results and that the results become closer with the increase of $\delta$. For example, for $\delta = 0.05$ Chernoff gives $n = 342$ and binomial gives $n = 224$, while for $\delta = 0.15$ the result of Chernoff is 41 compared to binomial’s 30.
In the second experiment the values of $\delta$ were computed with both Chernoff and binomial algorithms for several values of $n$ taken from the interval $[25,1000]$ with step 5. As shown in Figure 2, the results are close and become closer with higher values of $n$. To give just two examples, for $n=100$, Chernoff gives $\delta = 0.094$ and binomial gives $\delta = 0.075$, while for $n=1000$ Chernoff’s $\delta$ is 0.03 and binomial’s is 0.023.

The third experiment was actually a series of 32 identical, but independently performed experiments that ran the Chernoff min $n$ and Binomial min $n$ algorithms for several values of $\delta$. The average execution time of both algorithms is shown in Table 3. Note that the result of the Chernoff min $n$ algorithm was used as an upper bound in the Binomial min $n$ algorithm and, therefore, the execution time of the latter includes the execution time of the former. We can notice that Chernoff min $n$ is very fast no matter what the value of $\delta$ is, while Binomial min $n$ is slow for small values of $\delta$, but its execution time decreases with the increase of $\delta$.

The fourth experiment derived the execution time of the Chernoff min $\delta$ and Binomial min $\delta$ algorithms as an average of the execution times obtained from a set of 32 identical experiments. Several values of $n$ were considered and the results are shown in Table 4. We can see that Chernoff performs extremely fast again, while Binomial is slow for high values of $n$, improving when $n$ decreases.

The last two experiments checked whether the results of Chernoff min $n$ and Chernoff min $\delta$ algorithms are influenced by the precision of the load, $s_p$. Therefore, experiments 5 and 6 executed the Chernoff algorithms in the same
Table 3: Average execution time (in seconds) for Min n and Min \( \delta \): Chernoff vs. Binomial

| \( n \)  | Chernoff | Binomial | \( \delta \)  | Chernoff | Binomial |
|---------|----------|----------|---------------|----------|----------|
| 0.05    | 0.013    | 280.804  | 0.005         | 0.016    | 310.619  |
| 0.075   | 0.010    | 29.461   | 0.005         | 0.014    | 178.924  |
| 0.1     | 0.008    | 6.496    | 0.005         | 0.016    | 10.004   |
| 0.125   | 0.006    | 2.081    | 0.005         | 0.014    | 178.924  |
| 0.15    | 0.005    | 0.803    | 0.005         | 0.014    | 178.924  |

conditions as experiments 1 and 2, respectively, but for three different load precisions: \( 10^{-3}, 10^{-4}, 10^{-5} \). We found that the three-decimal results computed for the above load precisions were identical in all the considered cases, which means that it is worthless to use lower than \( 10^{-3} \) load precisions.

In conclusion, the Chernoff algorithms produce results that are close to the real minimums, are very fast, and their results accuracy does not improve with low load precisions. Therefore, they can be used in both Min n and Min \( \delta \) scenarios at both design time and runtime.

6. Related work

Randomized algorithms have found widespread applicability due to their simplicity and speed [14]. A brief survey on randomized algorithms [14] as well as a more recent and detailed one [15] were written by Motwani and Raghavan. The probabilistic analysis of randomized algorithms aims to provide probabilistic guarantees with respect to the likelihood of these algorithms to perform correctly or efficiently. For example, in the framework of resource allocation and admission control in transactional systems, Almeida et al. express the quality of service as the probability of executing every job within a maximum timeframe, and use the Markov and Chebyshev inequalities to compute an upper bound of this probability [2].

Particularly, randomization proved to be a powerful tool for building decentralized algorithms that run in parallel and work with local information. Thus, probabilistic decentralized algorithms were proposed for solving the load balancing problem in large networks [16, 17]. Theoretical analysis proved that, by using these algorithms, the expected or the maximum response time is less than a certain threshold with high probability. As for the Chernoff bound, it was applied in the probabilistic analysis of randomized packet routing algorithms for sparse networks and helped proving that certain algorithms are able to route all packets in a maximum number of steps with high probability [12, 18].

A recent application of the Chernoff bound can be found in the decentralized and probabilistic solution to document clustering that was proposed by Papapetrou et al. [19]. In their approach, the correctness probability is the probability of assigning each document to the right cluster and – similarly to our case – is set by the user and used to tune the parameters of the algorithm.

7. Conclusions

The decentralized probabilistic auto-scaling (DEPAS) algorithm can be used to deploy elastic service systems that can quickly grow from tens to tens of thousands of computing nodes. However, the potential customer of DEPAS may not be convinced only by the experimental results that are, after all, dependent on the specific experimental scenarios being considered.

Therefore, in this paper, we defined the correctness of DEPAS in a probabilistic manner and modeled DEPAS as a set of Bernoulli trials. Then, the Chernoff-Hoeffding bounds were used to theoretically prove that there is a subset of configurations for which a minimum correctness probability can be guaranteed. Moreover, based on the theoretical results, we designed numerical algorithms for automatically tuning DEPAS so that it can be both correct and accurate. Through a set of experiments we showed that the results given by the Chernoff-based tuning algorithms are close to the real minimums, which were estimated based on the binomial formula.

In a future paper, the Chernoff min \( \delta \) algorithm will be used at runtime to dynamically adapt DEPAS, and the expected gain in accuracy will be experimentally checked for a set of realistic workload traces.
8. Acknowledgements

This research has been partially funded by the Romanian National Authority for Scientific Research, CNCS UEFISCDI, under project PN-II-ID-PCE-2011-3-0260 (AMICAS) and by the European Commission, under project FP7-ICT-2009-5-256910 (mOSAIC). Bogdan Caprarescu is partially supported by IBM through a PhD Fellowship Award.

References

[1] M. Armbrust, A. Fox, R. Griffith, A. D. Joseph, R. Katz, A. Konwinski, G. Lee, D. Patterson, A. Rabkin, I. Stoica, M. Zaharia, A view of cloud computing, Commun. ACM 53 (2010) 50–58.
[2] J. Almeida, V. Almeida, D. Ardagna, I. Cunha, C. Francalanci, M. Trubian, Joint admission control and resource allocation in virtualized servers, J. Parallel Distrib. Comput. 70 (2010) 344–362.
[3] H. Ghanbari, B. Simmons, M. Litou, G. Iszlai, Feedback-based optimization of a private cloud, Future Generation Comp. Syst. 28 (2012) 104–111.
[4] U. Sharma, P. Shenoy, S. Sahu, A. Shaikh, A cost-aware elasticity provisioning system for the cloud, in: Proceedings of the 2011 31st International Conference on Distributed Computing Systems, ICDCS ’11, IEEE Computer Society, Washington, DC, USA, 2011, pp. 559–570.
[5] S. Meng, L. Liu, V. Soundararajan, Tide: achieving self-scaling in virtualized datacenter management middleware, in: Proceedings of the 11th International Middleware Conference Industrial track, Middleware Industrial Track ’10, ACM, New York, NY, USA, 2010, pp. 17–22.
[6] E. K. Lua, J. Crowcroft, M. Pias, R. Sharma, S. Lim, A survey and comparison of peer-to-peer overlay network schemes, IEEE Communications Surveys and Tutorials 7 (2005) 72–93.
[7] N. M. Calcavecchia, B. A. Caprarescu, E. Di Nitto, D. J. Dubois, D. Petcu, Depas: A decentralized probabilistic algorithm for auto-scaling, CoRR arXiv:1202.2509v1 (2012).
[8] M. Jelasity, S. Voulgaris, R. Guerraoui, A.-M. Kermarrec, M. van Steen, Gossip-based peer sampling, ACM Trans. Comput. Syst. 25 (2007).
[9] E. Di Nitto, D. J. Dubois, R. Mirandola, F. Saffre, R. Tateson, Applying self-aggregation to load balancing: experimental results, in: Proceedings of the 3rd International Conference on Bio-Inspired Models of Network, Information and Computing Systems, BIONETICS ’08, ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering), 2008, pp. 14:1–14:8.
[10] M. Jelasity, A. Montresor, O. Babaoglu, Gossip-based aggregation in large dynamic networks, ACM Trans. Comput. Syst. 23 (2005) 219–252.
[11] A. Montresor, A. Ghodsi, Towards robust peer counting, in: in Proc. of the 9th International Conference on Peer-to-Peer (P2P09, IEEE Computer Society, Washington, DC, USA, 2009, pp. 143–146.
[12] M. Mitzenmacher, E. Upfal, Probability and Computing: Randomized Algorithms and Probabilistic Analysis, Cambridge University Press, New York, NY, USA, 2005.
[13] Depas tuning tool, 2012. Http://bogdan.softinvent.org/research/depas/ (accessed Jan 15 2012).
[14] R. Motwani, P. Raghavan, Randomized algorithms, ACM Comput. Surv. 28 (1996) 33–37.
[15] R. Motwani, P. Raghavan, Algorithms and theory of computation handbook, Chapman & Hall/CRC, 2010, pp. 12–12.
[16] M. Mitzenmacher, The power of two choices in randomized load balancing, IEEE Trans. Parallel Distrib. Syst. 12 (2001) 1094–1104.
[17] S. Fu, C.-Z. Xu, H. Shen, Randomized load balancing strategies with churn resilience in peer-to-peer networks, J. Netw. Comput. Appl. 34 (2011) 252–261.
[18] E. Upfal, Efficient schemes for parallel communication, J. ACM 31 (1984) 507–517.
[19] O. Papapetrou, W. Siberski, N. Fuhr, Decentralized probabilistic text clustering, IEEE Transactions on Knowledge and Data Engineering 23 (2011) 339–342.