The Positive Impact of Failures on Energy Efficient Virtual Machines Consolidation

Andres J. Gonzalez and Bjarne E. Helvik
Department of Telematics *
Norwegian University of Science and Technology
O.S. Bragstads plass 2E, N-7491. Trondheim, Norway
Email: {andresgm,bjarne}@item.ntnu.no
Phone: (+47) 735 93684 , 735 92667 Fax: (+47) 735 96973

***Abstract*—Failures are undesirable events that should be always prevented. This paper agrees with this concept, but it is also aware that failures are unavoidable events. The main objective of this paper is to show how the failure handling through clever fault management mechanisms produces a direct positive impact on energy efficient cloud computing deployment. We address the virtual machine consolidation problem, and its associated bin packing problem. In spite of being studied for several years, it is, to the authors knowledge, the first time that this is studied under the presence of server failures that force the migration of some virtual machines at unexpected random times. We found that the extra dynamics generated by the compulsory reallocation of virtual machines produces a reduction in the number of servers needed, allowing some servers to be switched off, and hence having a more energy efficient cloud operation. Through some case studies, we show the amount of energy saved by the effect of random servers failures. Finally, we show that under some conditions, the optimal number of active servers may be reached.**

***Index Terms*—Cloud Computing; Virtual Machines Consolidation; Energy Efficient Mechanisms; Bin Packing.**

I. INTRODUCTION

This paper is focused on Infrastructure as a Service (IaaS) cloud computing scenarios, where virtual machines with specific computational requirements are allocated in servers with finite capacity. In this scenario, one of the most popular issues is how to allocate the virtual machines in order to use as less servers as possible, and thereby reduce the energy consumption. This is known as the virtual machine (VM) consolidation problem. Solving it appropriately has become a very hot topic for the research community. The reasons behind its popularity are important, and they have been well studied for years. For instance, in [1] is estimated that cloud computing services consumed near 662 billion kWh of energy during 2007. This value and its increasing trend has been confirmed by many related works, such as the ones produced by the Environmental Protection Agency ENERGY START. In [2], this consumption is linked directly with CO_2 emissions, with alarming results that motive actions.

The simplest version of the VM consolidation problem is when all the incoming request are known in advance (offline), allowing a better VM allocation planning. Even in this case, the optimal formulation (see Section II-A) has been probed to be NP-Hard. Section II-C will present some of the heuristic procedures used to solve the offline VM consolidation. Taking advantage of the knowledge of the whole VM requests, those algorithms may provide solutions close to the optimal value.

A more complicated scenario is posed when virtual machines arrive and depart at random instants of time, and only the incoming VM request and the current servers occupation are known in order to take allocation decisions. This problem is considered fully dynamic, and it may be solved in two different ways: First, by using pure online algorithms, which do not allow virtual machine reallocation (Section II-B). Second, by using semi-online algorithms, which do allow reallocation (Section II-D).

An additional increase on the dynamics of the problem is the presence of server failures. They are unavoidable events that may happen very often [3], [4] and [5]. The effect of failures is very relevant, because they generate forced and unplanned migration of virtual machines. To the authors' knowledge, this is the first time that this extra dynamics is considered in the VM consolidation problem. Our results show that the proper fault management of failures has a positive effect on the number of active servers used in the cloud, optimizing involuntarily a problem that has been studied and addressed voluntarily for years. This result is interesting, since failures are negative events that only may have positive effects in an indirect way. However, the results of this paper show a direct positive impact. We simulate several cloud computing scenarios, in order to prove our findings. In addition, at the end of the paper, we suggest a planned mechanism to optimize the energy consumption of a cloud.

This paper is organized as follows. In Section II, we present the virtual machine consolidation problem. Section III describes the failure and energy consumption models used. In Section IV, some case studies illustrate the impact of failures in the VM consolidation problem. Finally, Section V concludes the paper.

II. VIRTUAL MACHINE CONSOLIDATION

The virtual machine consolidation problem deals with the allocation of a list of virtual machines $V = \{V_1, ..., V_v, ...V_m\}$ on a set of physical servers $S = \{S_1, ..., S_s, ...S_n\}$.

* This work was sponsored by Telenor Research (Telenor ASA) under the Robust Networks and Services in Cloud Computing project.

Each virtual machine demands a computational capacity $\{k_1, ..., k_v, ...k_m\}$, and each server has a maximum computational capacity $\{C_1, ..., C_s, ...C_n\}$.

The main objective of the virtual machine consolidation problem is to use the minimum number of servers in order to have an energy-efficient operation. It can be modeled as a bin packing problem, *"one of the oldest and most well-studied problems in computer science"* [6] and [7]. In this section, we will mention four of the different versions of the virtual machine consolidation problem.

A. Optimal Consolidation

The bin packing problem has been studied since the beginning of the 70's, and in computational complexity theory, it is recognized as a combinatorial NP-hard problem [8]. Considering the set of servers S and virtual machines V previously presented , in this section, we present an integer linear programming formulation to minimize the number of active servers, using the following binary variables:

- y_s is equal to 1 if server s allocates virtual machines (ON), and 0 otherwise (OFF).
- $x_{v,s}$ is equal to 1 if the virtual machine v is allocated on server s.

$$\min \sum_{s=1}^{n} y_s \tag{1}$$

subject to constraints (s.t.)

$$\sum_{v=1}^{m} k_v x_{v,s} \leq C_s y_s \ , \ \forall (s) \in S \tag{2}$$

$$\sum_{s=1}^{n} x_{v,s} = 1 \ , \ \forall (v) \in V \tag{3}$$

Constraint (2) is used to avoid exceeding the server capacity, and constraint (3) ensures that a virtual machine can only be assigned to one server.

Optimal solutions to very large instances of the problem can be produced with sophisticated algorithms [6], [7] and [9]. In addition, many heuristics offline algorithms (Section II-C) can provide solutions very close to the optimal. Having an optimal value as reference is a matter of prime importance for this kind of works, in order to compare the quality of a solution given by offline, online, or semi-online algorithms.

B. Online Consolidation

In online consolidation, virtual machines arrive one at a time, and the allocation should be done immediately. This is a challenging scenario due to its high dynamics, and the time constraints in order to take decisions. According to the strict online definition, once the virtual machine is allocated, it has to remain in the same server until its departure. One of the most popular online approaches is called *first-fit algorithm*. It receives the request with its respective capacity k, and it looks server by server until finding one with enough free capacity to allocate the request. The first server with enough resources will be used to allocate the virtual machine. If the allocation is not possible, a new server is switched on, and the VM is allocated there.

This algorithm is rather simple, but it does not provide an optimal allocation. A better approach is the *best-fit algorithm*. Here, given that the active servers and current allocated VMs are known, the algorithm will chose the server that fulfills the next conditions: **i.)** The free capacity of the server is bigger than the capacity request k. **ii.)** The server with the smallest remaining free capacity.

Bounding the number of servers that may allocate an incoming VM is a technique used in many online algorithms. This is done in order to improve the computational complexity, and the number of active servers. One of the most relevant works in this line is the HARMONIC algorithms [10], [11] and [12], which propose the partition of the server capacity in intervals and classify them according to this criteria.

More advanced algorithms have been proposed in the literature. However, they imply, either the knowledge of future requests, or the reallocation of VMs. The next sections will explain these two cases.

C. Offline Consolidation

The main feature of offline consolidation is the knowledge of all the incoming requests to be allocated, i.e., the number of virtual machines and its capacity demands remain unmodified during the entire problem. In this sense, the problem is static, and hence there is a better chance of planning mechanisms to obtain solutions close to the optimal value. Two of the basic and most implemented offline techniques are the extension of the online first-fit and best-fit algorithms. Using the VMs knowledge, these methods provide solutions closer to the optimal by sorting the incoming requests by their capacity, and then allocating them in decreasing order. These techniques are known as *first-fit decreasing* and *best-fit decreasing*.

Two of the most cited works in offline consolidation are: the APTAS algorithm proposed in [13], and the exact algorithm proposed in [14]. Finally, the work presented in [15] proposes a decentralized approach by using ant colony optimization.

D. Semi-Online Consolidation

Semi-online consolidation is an intermediate method between offline and online. As defined in [16], it allows the reallocation of a number of already allocated virtual machines. The work presented in [17] proposes a semi-online algorithm for the bin packing problem, whereby the packing of an item, a total disregard for already packed items of smaller size is done. This initial step may then cause some small items to be repacked. Another approach proposed in [18] allows a constant number of elements to move from one bin to another, as a consequence of a new arrival. They propose two versions of the algorithm. The first one, where no more than three elements are allowed to move, and the second one, where as many as seven items can be reallocated. In [19], the reallocation process is considered each time a new arrival occurs, giving priority to virtual machines with

smaller capacity requests, since *"the more resources a virtual machine requires, the more cost to reallocate it"*. In addition, they propose a mechanisms to control that any virtual machine will be reallocated at most once.

One of the last works focused directly on cloud computing scenarios is the ENACLOUD proposed by Li et. al. in [20]. In this work, each time a new virtual machine v arrives, the algorithm *"tries to displace the allocated virtual machines with smaller capacity than* k_v". Then, the displaced virtual machines and the new arrival v are sorted and packed again using the best-fit algorithm. Using this policy is possible to save around 10% of the energy consumption in comparison with the consumption by using only the best-fit algorithm. Finally, in semi-online algorithms, it is also common to split the server capacity in small subintervals, and label the servers according to the maximum subinterval fulfilled, when virtual machines are allocated.

III. Failure and Energy Consumption Models

A. Failure Handling in Virtualized Environments

Hardware vendors and cloud designers follow best practices and put considerable efforts in order to minimize the presence of failures. However, real clouds are not failure free. They may be generated among others, due to hardware malfunction, software problems, network disconnections, electrical problems, environmental factors, or even human mistakes. The consequences of failures may be considerable, and they may happen very often [3], [4] and [5].

Based on the dynamics posed by the iteration of failure and repair processes in a cloud computing environment, the servers may be classified in three different groups: *active* servers, *spare* servers, and *on-repair* servers. The active server group contains the servers that are currently running virtual machines, being they the main source of energy consumption in the cloud. The spare servers are functional devices waiting for virtual machines that eventually may need them. Depending on the operator policies, they may be: **i.)** On standby, which is not an energy efficient policy, as Section III-B will show, but it allows a fast response in case of failures **ii.)** Power-off state, which is energy efficient, but with a slower reaction to failures. Finally, *on-repair* servers are servers that were affected by any kind of failure, and hence they are not operational, neither able to allocate virtual machines.

Current cloud computing technologies make use of different fault tolerance techniques in order to keep high levels of availability. In this way, when a server is hit by a failure, the affected VMs can be recovered independently of the physical hardware, allowing a fast and efficient restoration.

One of the most advanced fault tolerance techniques is active replication e.g., vSphere Fault Tolerance [21]. It uses identical/parallel VM-images running. Under this technique, a VM runs on two different physical servers, and every input is duplicated. In addition, both servers are constantly synchronized in order to keep coherence between the virtual images. The idea is that in case of a server failure, the virtualization platform is able to keep the VM running by using its mirror image, without generating any downtime from the user point of view. This concept is very useful in order to perform online virtual machine migrations in clouds, and it will be used in Section IV-C.

B. Energy Consumption

One of the biggest problems for the deployment of energy efficient clouds is the fact that with the current available technologies, idle servers still use more than 60% of the total energy compared to the peak consumption, even if power management functions are enabled [20], [22] and [23]. For this reason, the most effective option to save energy is to switch off as many servers as possible.

We will use a standard way of modeling the energy consumption of a server as proposed among others in [24] and [25]. Defining the server utilization at a given time t ($u_s(t)$) as a continuous function with minimum value of 0 when the server is idle, and maximum value 1 when it is at peak load, the power consumed by a server at time t may de defined as:

$$P[u_s(t)] = P_{idle} + [u_s(t) \cdot (P_{max} - P_{idel})] \tag{4}$$

where P_{max} and P_{idle} are the power consumption when the server is fully utilized and idle respectively.

Based on experimental measurements made in [24] on servers Dell PowerEdge 1950, we will assume a value of 218 Watt for P_{max}, and 171 for P_{idle}.

Finally, given that we are addressing dynamic scenarios, the utilization of the servers is constantly changing due to the arrival and departures of virtual machines, or due to failures. The total energy consumption of a server may be obtained by making the integral of the power consumed $P[u_s(t)]$ over the evaluated time interval.

IV. Case Studies

A. Failure-free vs Failure-present scenarios

In this section, we will present a case study that compares *failure-present* (real case) with *failure-free* (not real, but commonly assumed) scenarios, by analyzing two parameters: **i.)** The number of active servers on the cloud. **ii.)** The energy consumption in kWh.

We implement a Montecarlo simulation, where we consider that VMs arrive and depart dynamically at random times. For the arrival, we assume a Poisson process, where the time between two consecutive events is negatively exponentially distributed (*n.e.d.*), with an expected value of 50 minutes. Once the VM request is received, it may be allocated according to the two policies presented in section II-B (First-Fit or Best-Fit algorithm). When the virtual machine is allocated in one server, it will stay in the cloud for a *n.e.d.* sojourn time, with a mean of 35 days.

We assume servers having a maximum capacity of 10000 MIPS, and the capacity of each individual virtual machine requests will be obtained from a sample of a uniform distribution with values 1000, 2000, 3000, 4000, 5000, 6000, 7000 and 8000 MIPS.

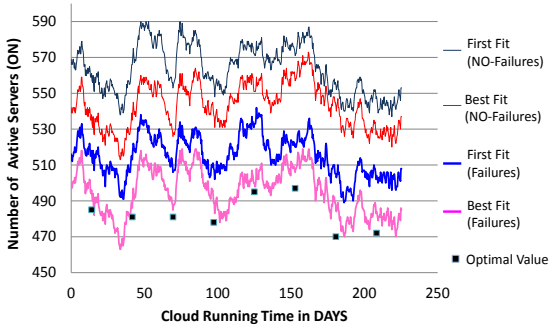

Fig. 1. Number of Active Servers (ON).

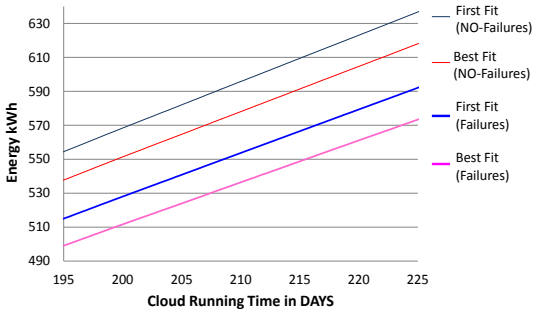

Fig. 2. Energy Consumption (kWh).

The main objective is to compare failure-free with failure-present clouds. Therefore, in the first scenario (NO-FAILURES), we assume that servers never fail, and hence the only dynamics in given by the arrival and departure of virtual machines. The second scenario (FAILURES) considers the existence of failures. We assume that servers are hit by independent failure processes that occur according to a negative exponential distribution with an expected waiting time of 30 days. Immediately after a server failure, the affected virtual machines are reallocated in the cloud in decreasing order of capacity, and according to the original allocation mechanism used on arrival, i.e., first-fit or best-fit.

Figure 1 shows the number of active servers (ON) in the cloud after simulating the previously described settings . The simulation was ran at the beginning during one year without taking any measurement, in order to start evaluating the cloud after some operational time, where its state is more stable. Therefore, in the x axis of Figure 1, the running time zero (Day 0) represents the start of the measurement of the number of active servers after one year of operation. An important information included in Figure 1 is the optimal number of active servers. This information is used for comparison purposes, and it was obtained by making a *static picture* of the state of the cloud, starting from a sample at the end of day 13th, and repeating the procedure each 28 days. The *static picture* considers only the virtual machines on the cloud and their respective capacities k at the sampling time, giving the chance to address the problem in an offline way, and hence calculating the optimal value, using the procedures shown in Section II-A.

Figure 1 confirms that Best-Fit performs much better than First-Fit, under similar conditions (both considering or ignoring failures), as it has been proved in all previous studies. In addition, the considerable reduction in the number of active servers produced by the effect of random server failures is a remarkable result. The simulation results also show that First-Fit considering failures may perform better than Best-Fit in a failure free environment. This is a very interesting result, since even trivial and well known concepts such as the better performance of best-fit over first-fit may change, depending on the consideration or not of failures.

An additional interesting information provided by Figure 1 is the close operation to the optimal value of the Best-Fit algorithm when failures are taken into account. This result is positive and at the same time unexpected, since failures are unavoidable events that happen without any intention of being beneficial. Therefore, the fact that they may help to obtain solutions close to the optimal is surprising. In Section IV-B, a more detailed analysis of how close could be the solution to the optimal will be presented. Having a number of active servers close to the optimal by using only best-fit may reformulate the advantages of most of the algorithms mentioned in Section II. The additional complexity of those mechanisms may not produce the expected reduction in the number of active servers, since this value may be already close to the minimum.

Figure 2 shows the total energy consumption in the four cases presented in Figure 1. The x axis contains the last month of the case study (day 195 to day 225). The energy consumption was calculated using the energy model presented in Section III-B, assuming a consumption of 171 Watt per Idle-active server, and 218 Watt per a fully utilized server. According to Figure 2, one can observe that the presence of failures may produce energy savings close to 10%.

B. The Effect of the Failure Intensity

Previous studies (such as those mentioned in Section II-D) are aware that the reallocation of virtual machines after their initial assignment produces a reduction in the number of active servers. In this respect, this paper presents two novel concepts: First, the reallocation studied here is not the result of a planned optimization mechanisms, but just the consequence of unavoidable and involuntary events. Second, the potential proximity to the optimal value.

Figure 3 shows the number of active servers at a fixed time t in a cloud, considering different server failures intensities. For this, we ran a Montecarlo simulation, where each server has an independent failure process with a time to failure *n.e.d.* with intensities from one failure per day, to one failure per year. In this case study, virtual machines also arrive and depart dynamically. The waiting time until the next arrival is *n.e.d.* with an expected value of 50 minutes. Once the VM is allocated using the best-fit algorithm, it stays in the cloud for a *n.e.d.* sojourn time with mean of 35 days. In order to

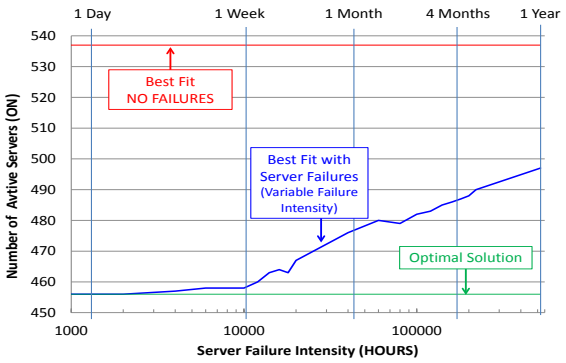

Fig. 3. Number of Active Servers (ON), under different server failure intensities (one failure expected per unit of time)

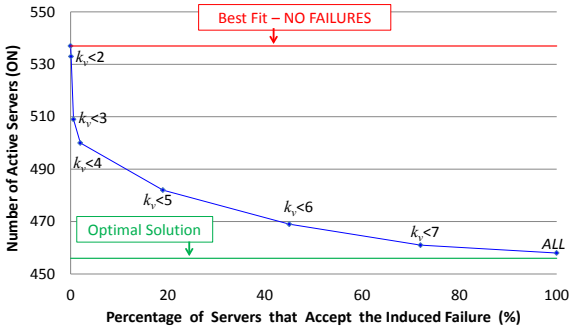

Fig. 4. Number of Active Servers (ON), when only a percentage of the servers accept the induced failure.

have a reference to compare the results obtained, Figure 3 also includes: **i.)** The optimal value which represent the minimum numbers of servers that should be activated in order to allocate the virtual machines present at time t. **ii.)** The number of servers needed to allocate the VMs at time t if best-fit had been used in a hypothetical failure-free environment.

Figure 3 shows that operating with a number of active servers close to the minimum is possible. In this case study, it happens mainly from very short failures intensities, up to values where one server failure per week is expected. In real operational clouds, providers make huge effort in order to have failures as seldom as possible. Therefore, an intensity of one failure per week is not very likely in a real cloud. For this reason, the range of Figure 3 includes intensities where in average, one failure is expected in several months, representing more realistic operational values. Despite of being not very close to the optimal point, in these cases (failure intensity in the order of months), the reduction in the number of active servers is still valuable. For instance, for an intensity of one failure per year, the number of additional servers ON is reduced in more than 50%.

C. Induced Failures

Previous sections show the effect of involuntary failure events on the number of active servers needed to allocate a set of virtual machines. Given the positive impact observed, their use as induced artificial events is an issue that deserve being considered. This idea becomes more feasible given the fact that online migration is a main feature in many commercial virtualization software. In this way, an induced failure can be planned in such a way that it does not generate any downtime (See Section III-A). In addition of having the possibility of producing planned failures with zero-downtime, induced failures may be also properly scheduled in moments where their impact could be minimum, e.g., late in the evening.

As a general policy, semi-online algorithms II-D are aware that; the more chances a cloud provider has to redistribute virtual machines, the higher the chance of operating with an optimal number of active servers. On the other hand, these reallocation processes are associated with instability; therefore, many studies try to minimize the amount of such events. We are aware that generating artificial failures may create instability in the cloud. However, we want to highlight five facts that may justify their use: **i.)** Having failure handling mechanism is compulsory anyhow. Therefore, the implementation of induced failures does not imply much additional effort for the provider. **ii.)** Real non-induced failures are events that will happen, making smaller the amount of induced failures needed. **iii.)** Current technologies allow zero downtime migrations. **iv.)** Most of the semi-online algorithms in the literature (see Section II-D) consider the use of reallocation with each VM arrival or departure. The use of induced failures may reduce that number considerably. **v.)** We found that inducing failures only in some of the servers will provide results still close to the optimal, reducing even more the amount of migration events needed. This last concept will be explained in more detail as follows.

Our assumption is that the migration of virtual machines with bigger capacity does not produce the same saving on the number of active servers than the migration of small virtual machines. To analyze this concept, we made a Montecarlo simulation similar to the one presented in the previous section, but this time, we assume that all failures are induced, and hence there is the possibility to choose if the server will accept the failure or not. We will filter the induced failures based on the capacities of the virtual machines allocated on the server. For instance, if the rejection criteria is set at 7000 MIPS, all the servers that allocate virtual machines with $k_v = 7000$, 8000, 9000, or 10000 MIPS will reject the induced failures. The simulation settings used to obtain Figure 4 are: Induced failure arrivals with *n.e.d.* time to failure with main of 1 week. *n.e.d.* virtual machine inter-arrival and sojourn time with mean of 50 minutes and 35 days respectively. Server capacity $C = 10000$ MIPS, and uniformly distributed VM capacities with a range between 1000 to 8000 MIPS.

Figure 4 shows the results obtained from the simulation. The idea is to evaluate the number of active servers needed when some of the servers reject the induced failures. For instance, the case *ALL* means that all the servers accept the induced failure, and hence the migration is made in 100% of the cases. The next case $k_v < 7$ means that the servers that allocate a

virtual machine with a capacity k equal to 8000 MIPS will reject the failures. In this case, near 30% of the failures are rejected, and the increase on the additional servers needed in reference to the minimum and the maximum value is just 3%. An interesting situation is given when $k_v < 5$. In this case, only the servers that allocate virtual machines with less than half of the capacity of the server ($k_v = [1000-4000]$) will take the induced failure, obtaining a failure rejection of more than 80%, and the number of additional servers needed in reference to the minimum and the maximum value is near 30%.

V. CONCLUDING REMARKS

The main contribution of this paper is to show the positive impact of failures on the virtual machines consolidation problem. A results that looks in principle contradictory to the conventional perception of failures. We show that the effect of failures may provide a solution very close to the optimal value under some conditions. This result is very interesting, since it is produced by random involuntary events. The virtual machine consolidation and its associated bin packing problem are very old problems. However, to the author knowledge, this problem has not been studied in the presence of random failures that force the migration of virtual machines. This paper provides a new contribution, complementing the already wide knowledge on this problem.

Many works try nowadays to develop energy efficient mechanisms through smart VM consolidation. The results in this paper are a warning for those works, since a variation on the expected reduction of active servers may be produced due to: **i.)** Failures are unavoidable events. **ii.)** They make the operational point more efficient that it was thought. The failure intensity, and the dynamics of the arrivals and departures of virtual machines are key factors for the amount of servers that can be switched off. Normal clouds operate with intensities in the order one failure per several months. Under these conditions, the optimal value can not be reached. However, the energy saving is still considerable, since even with an intensity of one failure per year, more than 50% of the additional servers needed in a failure free scenario can be switched off.

Finally, in addition to the natural improvement obtained by the presence of unavoidable failures, this paper proposes the use of artificial induced failures as an alterative to operate with values close to the optimal. We are aware of the potential instability produced by inducing failures, but we raise five points that may motivate this practice, in case of being needed. **i.)** Failure handling mechanisms are part of the cloud anyhow. **ii.)** Real failures will always happen, reducing the amount of induced failures. **iii.)** Current technologies allow zero-downtime migrations. **iv.)** Not all the servers have to accept the induced failures. **v.)** The expected number of migration events is less than the redistribution events proposed in other related works.

REFERENCES

[1] G. International, "How clean is your cloud," 2012. [Online]. Available: http://www.greenpeace.org/international/Global/international/publications/climate/2012/iCoal/HowCleanisYourCloud.pdf

[2] J. M. Kaplan, W. Forrest, and N. Kindler, "Revolutionazing data center energy efficient." *Technical Report*, 2008.

[3] D. Ford, F. Labelle, F. I. Popovici, M. Stokely, V.-A. Truong, L. Barroso, C. Grimes, and S. Quinlan, "Availability in globally distributed storage systems," in *9th USENIX conference on Operating systems design and implementation*, ser. OSDI'10, 2010, pp. 1–7.

[4] P. Gill, N. Jain, and N. Nagappan, "Understanding network failures in data centers: measurement, analysis, and implications," in *Proceedings of the ACM SIGCOMM 2011 conference*. New York, NY, USA: ACM, 2011, pp. 350–361.

[5] A. Gonzalez, B. Helvik, J. Hellan, and P. Kuusela, "Analysis of dependencies between failures in the UNINETT IP backbone network," *IEEE Pacific Rim Dependable Computing Symposium. PRDC*, Dec 2010.

[6] E. G. Coffman, Jr., M. R. Garey, and D. S. Johnson, "Approximation algorithms for np-hard problems," pp. 46–93, 1997. [Online]. Available: http://dl.acm.org/citation.cfm?id=241938.241940

[7] S. S. Seiden, "On the online bin packing problem," *J. ACM*, vol. 49, no. 5, pp. 640–671, Sep. 2002. [Online]. Available: http://doi.acm.org/10.1145/585265.585269

[8] R. Karp, *Reducibility among Combinatorial Problems*, ser. The IBM Research Symposia Series, R. Miller, J. Thatcher, and J. Bohlinger, Eds. Springer US, 1972.

[9] W. Fernandez de la Vega and G. Lueker, "Bin packing can be solved within 1 + ε in linear time," *Combinatorica*, vol. 1, no. 4, pp. 349–355, 1981.

[10] C. C. Lee and D. T. Lee, "A simple on-line bin-packing algorithm," *J. ACM*, vol. 32, no. 3, pp. 562–572, Jul. 1985.

[11] S. S. Seiden, "On the online bin packing problem," *J. ACM*, vol. 49, no. 5, pp. 640–671, Sep. 2002.

[12] J. Balogh, J. Bksi, and G. Galambos, "New lower bounds for certain classes of bin packing algorithms," vol. 6534, pp. 25–36, 2011.

[13] W. Fernandez de la Vega and G. Lueker, "Bin packing can be solved within 1 + ε in linear time," *Combinatorica*, vol. 1, no. 4, pp. 349–355, 1981. [Online]. Available: http://dx.doi.org/10.1007/BF02579456

[14] S. Martello and P. Toth, *Knapsack problems: algorithms and computer implementations*. New York, NY, USA: John Wiley & Sons, Inc., 1990.

[15] E. Feller, L. Rilling, and C. Morin, "Energy-aware ant colony based workload placement in clouds," *Proceedings of the 2011 IEEE/ACM 12th International Conference on Grid Computing. IEEE Computer Society*, 2011.

[16] B. Jnos, B. Jzsef, G. Gambor, and R. Gerhard, "On-line bin packing with restricted repacking." *Journal of Combinatorial Optimization*, pp. 1–17, 2012.

[17] Z. Ivkovic and E. Lloyd, "Fully dynamic algorithms for bin packing: Being (mostly) myopic helps," *SIAM Journal on Computing*, vol. 28, no. 2, pp. 574–611, 1998.

[18] G. Gambosi, A. Postiglione, and M. Talamo, "Algorithms for the relaxed online bin-packing model," *SIAM Journal on Computing*, vol. 30, no. 5, pp. 1532–1551, 2000.

[19] Y. Ho, P. Liu, and J.-J. Wu, "Server consolidation algorithms with bounded migration cost and performance guarantees in cloud computing," *Utility and Cloud Computing (UCC), 2011 Fourth IEEE International Conference on*, pp. 154–161, 2011.

[20] B. Li, J. Li, J. Huai, T. Wo, Q. Li, and L. Zhong, "Enacloud: An energy-saving application live placement approach for cloud computing environments," *IEEE International Conference on Cloud Computing*, pp. 17–24, 2009.

[21] VMware, "Protecting Mission-Critical Workloads with VMware Fault Tolerance," 2009, white Paper.

[22] S. Srikantaiah, A. Kansal, and F. Zhao, "Energy aware consolidation for cloud computing," *Proceedings of the 2008 conference on Power aware computing and systems. USENIX Association.*, vol. 10, 2008.

[23] A. Berl1, E. Gelenbe, M. D. Girolamo, G. Giuliani, H. D. Meer, M. Q. Dang, and K. Pentikousis, "Energy-efficient cloud computing," *The Computer Journal. OXFORD JOURNALS*, pp. 1045–1051, 2010.

[24] E. Feller, L. Rilling, and C. Morin, "Energy-aware ant colony based workload placement in clouds," in *Proceedings of the 2011 IEEE/ACM 12th International Conference on Grid Computing*, ser. GRID '11. IEEE Computer Society, 2011, pp. 26–33.

[25] A. Beloglazov, J. Abawajy, and R. Buyya, "Energy-aware resource allocation heuristics for efficient management of data centers for cloud computing," *Future Generation Computer Systems*, vol. 28, no. 5, pp. 755 – 768, 2012.