Brief Announcement: Towards Fault-Tolerant Bin Packing for Online Cloud Resource Allocation

Chuanyou Li
School of Computer Science and Engineering
Nanyang Technological University, Singapore
chuanyou.li@gmail.com

Xueyan Tang
School of Computer Science and Engineering
Nanyang Technological University, Singapore
asxytang@ntu.edu.sg

ABSTRACT

We consider an online fault-tolerant bin packing problem that models the reliable resource allocation in cloud-based systems. In this problem, any feasible packing algorithm must satisfy an exclusion constraint and a space constraint. The exclusion constraint is generalized from the fault-tolerance requirement and the space constraint comes from the capacity planning. The target of bin packing is to minimize the number of bins used. We first derive a lower bound on the number of bins needed by any feasible packing algorithm. Then we study two heuristic algorithms mirroring and shifting. The mirroring algorithm has a low utilization of the bin capacity. Compared with the mirroring algorithm, the shifting algorithm requires fewer numbers of bins. However, in online packing, the process of opening bins by the shifting algorithm is not smooth. It turns out that even for packing a few items, the shifting algorithm needs to quickly open a large number of bins. We therefore propose a new heuristic algorithm named mixing which can gradually open new bins for incoming items. We prove that the mixing algorithm is feasible and show that it balances the number of bins used and the process of opening bins.

KEYWORDS

Fault-tolerance, Bin packing, Online algorithm

1 INTRODUCTION

Cloud is a type of Internet-based computing paradigm which provides shared computing resources on demand. Cloud-based applications and services are growing explosively. Enormous application requirements promote the cloud to upgrade and equip increasing number of servers. The fast progress of cloud computing needs to face two fundamental issues. First, resource utilization, which aims at meeting the computational demands with minimum amount of cloud resources. Second, fault tolerance, whose target is to enhance the reliability, as failures are more prone to happen when the scale of the cloud grows.

Cloud resource allocation can be modeled as a bin packing problem [3]: given a set of items, the target is to pack the items into a minimum number of bins while guaranteeing that the aggregate size of the items in each bin does not exceed the bin capacity. For its online version, each item must be placed into a bin without the knowledge of subsequent items. In this paper, we consider a new version of the bin packing problem, which is called the fault-tolerant bin packing problem. This problem follows the online setting and needs to tolerate up to f faulty bins which stop serving any items.

To ensure fault tolerance, the primary-standby replication scheme is involved: each item is replicated and has $f + 1$ replicas which are composed of one primary and f standbys. For the primary-standby replication scheme, a primary replica is indispensable. The primary replica takes more responsibilities and has a higher workload than a standby replica [14]. Accordingly, for each item, a value $l \in (0, 1)$ is used to denote the workload of its primary replica and the workload of each standby replica is assumed to be l/η where $\eta > 1$ is the workload ratio between the primary and a standby. In the case of failures, when a standby replica becomes the new primary, its workload increases to l.

For the fault-tolerant bin packing problem, any feasible algorithm should satisfy an exclusion constraint and a space constraint. The exclusion constraint is generalized from the fault-tolerance requirement. For each item, its $f + 1$ replicas need to be dispersed to $f + 1$ different bins, such that no matter which f bins turn to be faulty, one correct replica is still available.

Definition 1.1. A fault-tolerant bin packing algorithm $\mathfrak{A}$ satisfies the exclusion constraint if and only if it never places two (or more than two) replicas of the same item in the same bin.

The space constraint arises from the resource limitation of a bin. We consider two scenarios. First, there is no faulty bin in the system. Second, no more than f faulty bins arise in the system. For the first case, during the placement, a fault-tolerant bin packing algorithm $\mathfrak{A}$ should ensure that the total workloads of the replicas placed in a bin does not exceed the bin capacity W. The second case is a little more complex. When a faulty primary arises, a process of switching standby to primary will be involved. Note that when $f > 1$, a service has more than 1 standby replicas. It could be the case that there exist several different switching strategies for which one standby is selected to become the new primary. Furthermore, as the switching process leads to workload expansion on the standby, it could happen that some bad switching strategies will lead to overloaded bins. Before giving a formal definition of the space constraint, we first define the concept of a valid switching strategy.

Definition 1.2. Given a set of faulty bins, a switching strategy (standby to primary) is valid if and only if it satisfies the following two properties: (1) for each faulty primary replica, one of its correct standby replica is selected to become the new primary; (2) there are no overloaded bins after workload expansion.

SPAA'17, July 24–26, 2017, Washington, DC, USA

DOI: http://dx.doi.org/10.1145/3087556.3087596

Definition 1.3. A fault-tolerant bin packing algorithm $\mathfrak{A}$ satisfies the space constraint if and only if it satisfies the following two properties: (1) there are no overloaded bins before any failure happens; (2) a valid switching strategy always exists as long as there are no more than f faulty bins.

Definition 1.4. A fault-tolerant bin packing algorithm $\mathfrak{A}$ is named feasible if and only if it satisfies both the exclusion constraint and the space constraint.

2 TECHNICAL CONTRIBUTIONS

To pack a set of items, we first derive a lower bound on the number of bins required by any feasible algorithm.

THEOREM 2.1. *To place a set of items with a total workload L of all primary replicas, any feasible fault-tolerant bin packing algorithm $\mathfrak{A}$ requires at least* $\frac{(\eta f+\eta L+fL)+\sqrt{(\eta L+fL+\eta f)^2-4\eta(f^2L+fL)}}{2\eta}$ *bins.*

We then propose three feasible heuristic algorithms named mirroring $\mathfrak{A}_{mr}$, shifting $\mathfrak{A}_{st}$ and mixing $\mathfrak{A}_{mx}$. The mirroring algorithm $\mathfrak{A}_{mr}$ places primary replicas and standby replicas separately. Any bin that is used to place primary replicas has f "mirror images", which are f bins that are used to place the corresponding standby replicas. The original idea of mirroring algorithm is from [11].

THEOREM 2.2. *The mirroring algorithm $\mathfrak{A}_{mr}$ is feasible.*

THEOREM 2.3. *To place a set of items with a total workload L of all primary replicas, the mirroring algorithm $\mathfrak{A}_{mr}$ requires at most* $(\lfloor \frac{L}{l\lfloor 1/l \rfloor} \rfloor + 1)(f+1)$ *bins.*

In the mirroring algorithm $\mathfrak{A}_{mr}$, any bin that is used to place the primary replicas has another f bins dedicated to placing the corresponding standby replicas. The utilization of the bin capacity by the mirroring algorithm is inefficient. Because of failures, all the standby replicas located in the same bin might switch to primary and expand their workloads together. To meet the space constraint, a large amount of the reserved space should be maintained in each bin hosting standby replicas. To reduce the number of bins used, we design the shifting algorithm $\mathfrak{A}_{st}$ to decrease the amount of reserved space. The idea of $\mathfrak{A}_{st}$ is derived from [4]. We generalize the basic idea by replication and make it adapt to any f faulty bins in our model. Similar to the mirroring algorithm, the shifting algorithm $\mathfrak{A}_{st}$ also places primary replicas and standby replicas separately. The difference is that for primary replicas hosted in the same bin, $\mathfrak{A}_{st}$ disperses their standby replicas into f different bins.

THEOREM 2.4. *The shifting algorithm $\mathfrak{A}_{st}$ is feasible.*

THEOREM 2.5. *To place a set of items with a total workload L of all primary replicas, the shifting algorithm $\mathfrak{A}_{st}$ requires at most* $(\lfloor \frac{L}{\lfloor \frac{l+\eta-l\eta}{l} \rfloor \lfloor \frac{1}{l} \rfloor l} \rfloor + 1)(\lfloor \frac{l+\eta-l\eta}{l} \rfloor + f\lfloor \frac{1}{l} \rfloor)$ *bins.*

The shifting algorithm $\mathfrak{A}_{st}$ saves the amount of reserved space by ensuring that at most one standby in each bin might switch to primary in the case of failures. Compared with $\mathfrak{A}_{mr}$, $\mathfrak{A}_{st}$ reduces the number of bins used. However, in online packing, the process of opening bins by $\mathfrak{A}_{st}$ is not smooth. Even for packing a few items, $\mathfrak{A}_{st}$ needs to quickly open a large number of bins. In order to smooth the process of opening bins, we propose the new mixing algorithm $\mathfrak{A}_{mx}$. $\mathfrak{A}_{mx}$ opens new bins gradually and organizes them into a sequence of bin groups. Each bin hosts at most $x = \lfloor \frac{\eta-\eta l+l}{\eta l+fl} \rfloor$ primary replicas. When a new item arrives, if there exists one bin in the current bin group hosting less than x primary replicas but not enough bins to host f more standby replicas, $\mathfrak{A}_{mx}$ opens f new bins and adds them into the current group. If no bin in the current group can host one more primary replica, $\mathfrak{A}_{mx}$ opens $2f$ new bins and organizes them as a new group. Suppose p_i represents the i^{th} arriving item's primary replica and suppose each s_i^k $(1 \le k \le f)$ represents one of the i^{th} item's standby replicas. Figure 1 shows an example of the placement by assuming $x = 2$ and $f = 3$.

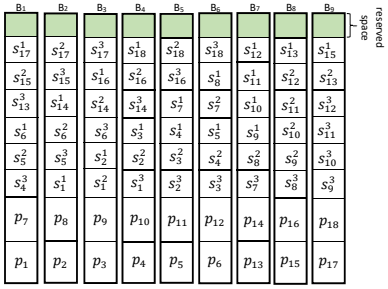
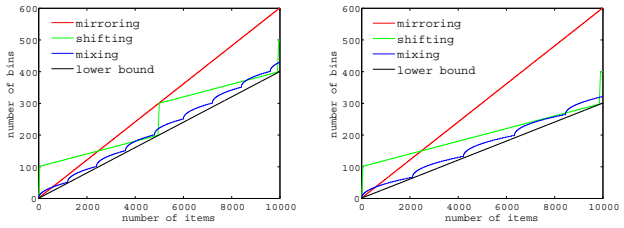


Figure 1: Example of the algorithm $\mathfrak{A}_{mx}$ ($f = 3$)



(a) $l = 0.02$, $\eta = 2$ (b) $l = 0.02$, $\eta = 4$

Figure 2: Numerical examples

THEOREM 2.6. *The mixing algorithm $\mathfrak{A}_{mx}$ is feasible.*

THEOREM 2.7. *To place a set of items with a total workload L of all primary replicas, the mixing algorithm $\mathfrak{A}_{mx}$ requires at most* $\frac{L}{l\lfloor \frac{\eta-\eta l+l}{\eta l+fl} \rfloor} + f(\lfloor \frac{\eta-\eta l+l}{\eta l+fl} \rfloor + 1)$ *bins.*

We conclude that when $l < 1/2$, the mixing algorithm $\mathfrak{A}_{mx}$ outperforms the mirroring algorithm $\mathfrak{A}_{mr}$. The number of bins required by the mixing algorithm may be slightly more than that by the shifting algorithm, but the process of opening bins for $\mathfrak{A}_{mx}$ is much more smooth than $\mathfrak{A}_{st}$. Figure 2 shows two numerical examples. For both examples, we set $f = 2$ and $l = 0.02$. In Figure 2(a), η is set to 2 and in Figure 2(b), η is set to 4. In both examples, we can see that the number of bins required by the mixing algorithm increases smoothly. Even for packing 10000 items, the mixing algorithm is still the best among the three heuristic algorithms. Besides, when η increases, both $\mathfrak{A}_{mx}$ and $\mathfrak{A}_{st}$ require fewer number of bins. However, $\mathfrak{A}_{mr}$ cannot benefit from the growth of η.

3 RELATED WORK

The classical bin packing problem has been extensively studied [3][5]. It is well known that even in the offline setting, the classical bin packing problem is NP-hard [6]. Numerous heuristic algorithms have been proposed. So far, for classical online bin packing, the best upper bound on the competitive ratio is 1.58889 which is achieved by the HARMONIC++ algorithm [12]. The best known lower bound for any online packing algorithm is 1.54037 [1].

Fault tolerance is a key issue in parallel and distributed computing and has been studied for many years. To achieve fault tolerance, replication is a fundamental mechanism. Based on various replication schemes, there are a large number of fault-tolerance protocols (*e.g.*, a family of protocols based on Paxos [8]) which focus on the consistency problem among different replicas. Recently with the rapid growth of cloud computing, replication is receiving increasing attention to enhance the reliability of resource allocation in the cloud. Shen *et al.* [13] proposed an Availability-on-Demand mechanism. The mechanism consists of a scheduler that manages computing resources. To enhance the availability, each virtual machine (VM) is replicated. When placing VMs to the physical servers, the scheduler ensures the primary and the backups are located in different servers. Yanagisawa *et al.* [14] studied dependable VM allocation in a bank private cloud. Similar to our work, they also considered the resource demands might fluctuate in the case of failures. A mixed integer programming approach was proposed to solve the resource allocation problem. The above two papers focused on empirical performance and did not provide any theoretical analysis. O. Beaumont *et al.* [2] studied the impact of the reliability constraint on the complexity of resource allocation problems. They proved several fundamental complexity results and proposed a basic randomized allocation algorithm. M. Korupolu *et al.* [7] studied failure-aware placement problems under adversarial and probabilistic failure models. Both [2] and [7] targeted to minimize the number of disrupted items in the case of failures or the probability for more than a given number of items to fail together. Different from these studies, we aim to ensure that each item always has at least one survival replica as long as the number of faulty bins are no more than f.

Our problem is closely related to a fault-tolerant server consolidation problem studied recently [11][4]. In the latter problem, to achieve fault tolerance, each task is divided into several replicas which are dispersed in different servers. Every replica inherits part of the original task workload. In the case of failures, the faulty replica's workload can be directed to other replicas hosted by correct servers. One typical application area of server consolidation is in multi-tenant database systems. Schaffner *et al.* [11] studied robust tenant placement for in-memory database clusters. Daudje *et al.* [4] conducted a follow-up study and proposed a heuristic algorithm called the shifting algorithm. Compared with the mirroring algorithm proposed in [11], the shifting algorithm achieves a better competitive ratio. A major difference of our fault-tolerant bin packing problem from the above problem is that each standby replica created has a base workload that is independent of whether the standby switches to the primary or not. This is a reasonable model for many practical scenarios. For example, in the case of database replication, read requests are normally executed by the primary only, whereas write requests must be executed by all the replicas for maintaining the consistency [11][14]. The study of [4], however, does not consider such a constraint. If standby replicas do not involve any base workload, one can in fact simply create idle servers to host standby replicas and let these servers take over the faulty servers when failures occur. In our fault-tolerant bin packing problem, both the mirroring and the shifting algorithms are also studied. The shifting algorithm has better performance. However, the process of opening bins by the shifting algorithm is not smooth in the online setting. We propose a new algorithm named mixing which can balance the number of bins used and the process of opening bins. Furthermore, our work addresses a generalized requirement to tolerate up to f failures.

Another variant of the bin packing problem for modeling cloud resource allocation has been defined and studied in recent years [9][10]. The objective of this variant is to minimize the accumulated bin usage time, which is different from our target to minimize the number of bins used in this paper. Besides, no fault-tolerance constraint was considered in the above variant.

ACKNOWLEDGMENTS

This work is supported by Singapore Ministry of Education Academic Research Fund Tier 1 under Grant 2013-T1-002-123.

REFERENCES

[1] J. Balogh, J. Békési, and G. Galambos. 2012. New lower bounds for certain classes of bin packing algorithms. *Theor. Comput. Sci.* 440-441 (2012), 1–13.
[2] O. Beaumont, L. Eyraud-Dubois, and H. Larchevêque. 2013. An Availability-on-Demand Mechanism for Datacenters. In *Proc. of the 27th IEEE International Symposium on Parallel and Distributed Processing (IPDPS)*. 55–66.
[3] E. G. Coffman, J. Csirik, G. Galambos, S. Martello, and D. Vigo. 2013. Bin packing approximation algorithms: Survey and classification. *Handbook of Combinatorial Optimization (second edition)* (2013), 455–531.
[4] K. Daudjee, S. Kamali, and A. Lopez-Ortiz. 2014. On the online fault-tolerant server consolidation problem. In *Proc. of the 26th ACM Symposium on Parallelism in Algorithms and Architectures (SPAA)*. 12–21.
[5] G. Galambos and G. J. Woeginger. 1995. On-line bin packing - A restricted survey. *Math. Meth. of OR* 42, 1 (1995), 25–45.
[6] M. R. Garey and D. S. Johnson. 1979. *Computers and Intractability: A Guide to the Theory of NP-Completeness.* W. H. Freeman.
[7] M. R. Korupolu and R. Rajaraman. 2016. Robust and Probabilistic Failure-Aware Placement. In *Proc. of the 28th ACM Symposium on Parallelism in Algorithms and Architectures (SPAA)*. 213–224.
[8] L. Lamport. 1998. The Part-Time Parliament. *ACM Trans. Comput. Syst.* 16, 2 (1998), 133–169.
[9] Y. Li, X. Tang, and W. Cai. 2014. On dynamic bin packing for resource allocation in the cloud. In *Proc. of the 26th ACM Symposium on Parallelism in Algorithms and Architectures (SPAA)*. 2–11.
[10] R. Ren and X. Tang. 2016. Clairvoyant Dynamic Bin Packing for Job Scheduling with Minimum Server Usage Time. In *Proc. of the 28th ACM Symposium on Parallelism in Algorithms and Architectures (SPAA)*. 227–237.
[11] J. Schaffner, T. Januschowski, M. Kercher, T. Kraska, H. Plattner, M. J. Franklin, and D. Jacobs. 2013. RTP: Robust tenant placement for elastic in-momory database clusters. In *Proc. of the ACM International Conference on Management of Data (SIGMOD)*. 773–784.
[12] S. S. Seiden. 2002. On the online bin packing problem. *Journal of the ACM* 49, 5 (2002), 640–671.
[13] S. Shen, A. Iosup, A. Israel, W. Cirne, D. Raz, and D. H. J. Epema. 2015. An Availability-on-Demand Mechanism for Datacenters. In *Proc. of the 15th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (CCGrid)*. 495–504.
[14] H. Yanagisawa, T. Osogami, and R. Raymond. 2013. Dependable virtual machine allocation. In *Proc. of the 32nd Annual IEEE International Conference on Computer Communications (INFOCOM)*. 629–637.