On Fault-Tolerant Bin Packing for Online Resource Allocation

Chuanyou Li, Xueyan Tang

Abstract—We study an online fault-tolerant bin packing problem that models reliable resource allocation. In this problem, each item is replicated and has $f + 1$ replicas including one primary and f standbys. The packing of items is required to tolerate up to f faulty bins, *i.e.*, to guarantee that at least one correct replica of each item is available regardless of which f bins turn to be faulty. 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. We then study three heuristic algorithms named mirroring, shifting and mixing under a particular setting where all items have the same size. The mirroring algorithm has a low utilization of the bin capacity. Compared with the mirroring algorithm, the shifting algorithm requires fewer 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. The mixing algorithm adopts a dual average strategy to 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. Finally, to pack items with different sizes, we extend the mirroring algorithm by adopting the First-Fit strategy and extend both the shifting and mixing algorithms by involving the harmonic strategy. The asymptotic competitive ratios of the three extended algorithms are analyzed respectively.

Index Terms—Fault-tolerance, Bin packing, Heuristic, Online

1 Introduction

Cloud-based applications and services are growing explosively. 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.

Resource allocation can be modeled as bin packing [1] in which the bins and items correspond to the servers and application instances respectively. In the classical bin packing problem, 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. In the online setting, each item must be placed into a bin without the knowledge of subsequent items.

Involving fault-tolerance schemes is a common approach to enhance reliability. In this paper, we consider a primary-standby replication scheme to achieve fault tolerance: each item is replicated and has $f + 1$ replicas which are composed of one primary and f standbys. For the primary-standby replication scheme, the primary is indispensable. It normally takes more responsibilities and has a higher workload than a standby replica [2]. When the primary fails, one of its available standbys will switch to be the new primary. After a standby switches to be the primary and takes over new responsibilities, its workload increases. Thus, the selection of the standby must ensure that it does not cause any overflow after switching to be the primary.

To achieve both packing efficiency and system reliability, we propose and study a new version of the bin packing problem called fault-tolerant bin packing. This problem requires the packing of items to tolerate up to f faulty bins which stop serving any items. In this new 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. The space constraint comes from the capacity planning and requires that each correct bin always has sufficient available space to serve its items. In our fault-tolerant model, a faulty primary will cause one of its available standbys to become the new primary and expand the workload. Thus, the space constraint needs to deal with various cases of failures: even when there are up to f faulty bins, it is always possible to select available standbys to become the new primaries without causing any overflow.

Given a set of items, we first derive a lower bound on the number of bins required by any feasible algorithm. The lower bound turns to be a root of an quadratic equation, which is associated with f, L and η (L is the aggregate workload of all primary replicas and η is the workload ratio between the primary and a standby). We then focus on packing algorithms in the online setting which is usually used to model resource allocation in the cloud [3]. In the basic case of uniform item sizes, we propose three heuristic algorithms named mirroring $\mathfrak{A}_m$, shifting $\mathfrak{A}_s$ and mixing $\mathfrak{A}_x$. In the mirroring algorithm $\mathfrak{A}_m$, any bin that is used to place the primary replicas has another f bins dedicated to placing the corresponding standby replicas. Our analysis shows that the utilization of the bin capacity by $\mathfrak{A}_m$ is generally inefficient.

- *Chuanyou Li is with the School of Computer Science and Engineering, MOE Key Laboratory of Computer Network and Information Integration, Southeast University, Nanjing, China.*
  E-mail: cyli@seu.edu.cn
- *Xueyan Tang is with the School of Computer Science and Engineering, Nanyang Technological University, Singapore.*
  E-mail: asxytang@ntu.edu.sg

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 reserved space should be maintained in the bin. Different from $\mathfrak{A}_m$, for primary replicas hosted in the same bin, the shifting algorithm $\mathfrak{A}_s$ disperses their standby replicas into f bin groups. The amount of reserved space is saved since in each bin at most one standby replica might switch to primary in the case of failures. Compared with $\mathfrak{A}_m$, $\mathfrak{A}_s$ reduces the number of bins used. However, in online packing, the process of opening bins by $\mathfrak{A}_s$ is not smooth. Even for packing a few items, $\mathfrak{A}_s$ needs to quickly open a large number of bins. In order to smooth the process of opening bins, we propose the mixing algorithm $\mathfrak{A}_x$. The idea is generalized from a dual average strategy by which all the primary and standby replicas are uniformly distributed among the open bins. Different from $\mathfrak{A}_m$ and $\mathfrak{A}_s$, each bin opened by $\mathfrak{A}_x$ hosts the primary and standby replicas together. The reserved space in each bin can accommodate one standby replica switching to primary for handling failures. We propose a switching strategy to meet the space constraint in the case of failures. Our analysis shows that the mixing algorithm $\mathfrak{A}_x$ achieves better performance than the mirroring algorithm $\mathfrak{A}_m$. Meanwhile, compared with the shifting algorithm $\mathfrak{A}_s$, the mixing algorithm $\mathfrak{A}_x$ opens new bins much more smoothly in the online setting.

In the general case where items can have different sizes, we involve additional strategies to extend the three heuristic algorithms. For the mirroring algorithm, we adopt the First-Fit strategy to pack primary replicas. For the shifting and mixing algorithms, we adopt the harmonic strategy to pack items in different ranges of size separately. We analyze the asymptotic competitive ratios of these extended algorithms.

For ease of reference, Table 1 summarizes the notations used in this paper. The rest of this paper is organized as follows. Section 2 sketches out the related work. Section 3 introduces the model and provides a formal definition of the fault-tolerant bin packing problem. Section 4 establishes a lower bound on the number of bins needed by any feasible algorithm. In Section 5, we propose three feasible heuristic algorithms and analyze their performance. In Section 6, we extend these algorithms to pack items of different sizes. Section 7 briefly discusses future work. Some preliminary results of our work were presented as a brief announcement [4].

TABLE 1: Summary of notations

Notation	Description
$\mathfrak{B}$	the set of bins
B_i	the i^{th} bin
$\mathfrak{R}$	the set of services to be released
R_i	the i^{th} service to be released
p_i	the primary replica of R_i
s_i^k	the k^{th} standby replica of R_i
f	the assumed upper bound on the number of failures
ℓ_i	the workload of the primary replica p_i of R_i
η	the workload ratio between the primary and a standby
L	the aggregate workload of all primary replicas of $\mathfrak{R}$
$\mathfrak{A}_m$	the mirroring algorithm
$\mathfrak{A}_s$	the shifting algorithm
$\mathfrak{A}_x$	the mixing algorithm
$\mathfrak{A}_{fm}$	the First-Fit mirroring algorithm
$\mathfrak{A}_{hs}$	the Harmonic shifting algorithm
$\mathfrak{A}_{hx}$	the Harmonic mixing algorithm

2 Related work

The classical bin packing problem has been extensively studied in both the offline and online settings [1] [5]. It is well-known that the classical bin packing problem is NP-hard [6]. So far, the best known lower bound on the competitive ratio of online bin packing is 1.54037 [7]. The best upper bound on the competitive ratio of online bin packing is 1.57828956, which is achieved by the advanced harmonic algorithm [8].

Fault tolerance is a key issue in parallel and distributed computing. Replication is a fundamental mechanism to achieve fault tolerance. Based on various replication schemes, there are a large number of fault-tolerance protocols (*e.g.*, a family of protocols based on Paxos [9]) which focus on the consistency problem among different replicas. Recently, replication is receiving increasing attention to enhance the reliability of resource allocation in the cloud. Shen *et al*. [10] proposed an Availability-on-Demand mechanism which 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 uses a First-Fit strategy and ensures the primary and the backups are located on different servers. Yanagisawa *et al*. [2] studied dependable VM allocation in a bank private cloud. Similar to our work, they also considered that the resource demands might change 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. Beaumont *et al*. [11] studied the impact of the reliability constraint on the complexity of resource allocation problems. Korupolu *et al*. [12] defined an adversarial optimization problem which aims at finding a fractional number of tasks placed in each server. They also proposed a randomized approach to reach the optimum value of the fractional solution in expectation. Both [11] and [12] target to minimize the number of faulty items in the case of failures. In contrast, we seek to ensure that each item always has at least one surviving replica as long as the number of faulty bins is no more than a given number.

Our problem is closely related to a fault-tolerant server consolidation problem studied recently [13], [14], [15]. 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 this model is multi-tenant database systems. Schaffner *et al*. [13] studied robust tenant placement for in-memory database clusters. Daudjee *et al*. [14] conducted a follow-up study and proposed a heuristic algorithm called the shifting algorithm. Compared with the mirroring algorithm proposed in [13], the shifting algorithm achieves a better performance. Mate *et al*. [15] proposed a CUBEFIT algorithm which extends the shifting algorithm to tolerate multiple server failures while ensuring no server becomes overloaded. 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 [13] [2]. The studies of [14] and [15], however, did 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

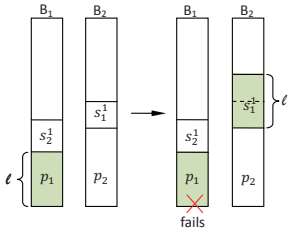


Fig. 1: Primary fails and standby expands

take over the faulty servers when failures occur. We also study both the mirroring and shifting algorithms for our problem. We further propose a new algorithm named mixing which can balance the number of bins used and the process of opening bins.

Another recent variant of the bin packing problem for modeling cloud resource allocation is called MinUsageTime dynamic bin packing [16] [17]. 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. Besides, no fault-tolerance constraint was considered in the above variant.

3 Model and Problem

3.1 Model

We consider a system which consists of homogeneous servers $\mathfrak{B} = \{B_1, B_2, B_3, ...\}$. The system is not reliable: servers may suffer from crash failures and stop executing any tasks. Failures can happen at any time on any server and are not foreknown. We refer to the applications undertaken by the system as services. We seek to achieve the following fault-tolerance guarantee: as long as the number of faulty servers is no more than a given number f, the system should keep its correctness such that no running services are blocked or discarded.

For each service, the system needs to allocate resources to run it. We consider the online setting: services are released sequentially and when allocating resources for a new service, the system has no idea about when and how many services will be released in the future. Let $\mathfrak{R} = \{R_1, R_2, R_3, ...\}$ denote the sequence of services to be released.

To achieve fault tolerance, we consider the primary-standby replication scheme. Specifically, when a service is released, it is replicated and has $f + 1$ replicas (to avoid FLP impossibility [18], we suppose the system is synchronous, such that all the $f + 1$ replicas can make agreement and always keep a consistent state with each other.). To facilitate presentation, replicas placed on faulty servers are also called faulty and replicas placed on correct servers are called correct. Among the $f+1$ replicas, one is named primary and the other f are named standbys. In general, the primary takes more responsibilities than a standby, *e.g.*, acting as a coordinator or synchronizing the states of standby replicas. Thus, a correct primary is indispensable: when the primary turns to be faulty, one of its correct standbys will switch the role and become the new primary. Based on this replication scheme, we model each service R_i as a set of $f + 1$ replicas $R_i = \{p_i, s_i^1, s_i^2, ..., s_i^f\}$, where p_i represents the primary replica and s_i^k $(1 \leq k \leq f)$ represents the k^{th} standby replica. In the following text, we also use s_i to denote one of R_i's standby replicas.

Each server has a limited amount of resources. We assume the capacity of each server is fixed and equal to 1. The amount of resources to run a service R_i is called R_i's workload. Let $\ell_i \in (0, 1)$ denote the workload of the primary replica p_i of service R_i. As discussed, the primary takes more responsibilities and has a higher workload than a standby. We assume the workload of each standby replica s_i^k is equal to ℓ_i/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 example, in Figure 1, there are two servers B_1, B_2 and two services R_1, R_2. Each service has 2 replicas. Suppose primary p_1 and standby s_2^1 are placed in B_1 and primary p_2 and standby s_1^1 are placed in B_2. If B_1 fails (so that p_1 turns to faulty), s_1^1 becomes the new primary of service R_1 and its workload increases to ℓ.

3.2 Fault-Tolerant Bin Packing

When a service is released, it is assigned to run on a set of servers. We assume that once the service (including all its $f + 1$ replicas) is assigned, it will not be moved to other servers due to reasons such as interruption to the service or high migration overheads. We model the resource allocation for services as a bin packing problem which we refer to as the fault-tolerant bin packing problem. The objective of the problem is to place the items (replicas) into a minimum number of bins (servers) while satisfying an *exclusion constraint* and a *space constraint*.

Definition 1. *A fault-tolerant bin packing algorithm satisfies the exclusion constraint if and only if it never places two (or more than two) replicas of the same service in the same bin.*

The exclusion constraint arises from the fault-tolerance requirement. To tolerate up to f faulty bins, clearly the $f+1$ replicas of the same service should be placed in $f + 1$ different bins. The space constraint arises from the capacity limitation of a bin. The space constraint in our problem is more complex than that in classical bin packing. Besides ensuring that the total workload of the replicas in a bin does not exceed the bin capacity at the time of placement, we also need to consider replica switching in the case of failures: when a faulty primary arises, a correct standby will have to switch the role and become the new primary without violating the space constraint.

Definition 2. *Given a set of faulty bins, a switching strategy (standby to primary) satisfies the following two properties:*

1) *for each faulty primary replica, one of its correct standby replicas is selected to become the new primary;*
2) *there are no overloaded bins after workload expansion.*

Definition 3. *A fault-tolerant bin packing algorithm 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 switching strategy always exists as long as there are no more than f faulty bins.*

Definition 4. *A fault-tolerant bin packing algorithm is named feasible if and only if it satisfies both the exclusion constraint and the space constraint.*

4 A Lower Bound

First, we derive a lower bound on the number of bins needed by any feasible fault-tolerant bin packing algorithm.

Theorem 1. *To place a set of services $\mathfrak{R}$ with a total workload L of all primary replicas, the number of bins required by any feasible fault-tolerant bin packing algorithm is at least:* $\frac{(\eta L+fL+\eta f)+\sqrt{(\eta L+fL+\eta f)^2-4\eta(f^2L+fL)}}{2\eta}$.

Proof. Suppose t bins are opened in total to host the set of services $\mathfrak{R}$. For each bin B_i ($1 \leq i \leq t$), we know B_i may host some primary replicas, some standby replicas and may also keep some reserved space for handling failures. Consequently, the total capacity of B_i can be divided into three disjoint parts: W_i^p, W_i^s and W_i^e. W_i^p is the part that is occupied by primary replicas, W_i^s is the part that is occupied by standby replicas and W_i^e is the remaining reserved space. Obviously, at least one of W_i^p and W_i^s is greater than 0. According to the space constraint, we have:

$$\sum_{i=1}^{t} W_i^p \binom{t-1}{f-1}(1-\frac{1}{\eta}) \leq \sum_{i=1}^{t} W_i^e \binom{t-1}{f} \tag{1}$$

Inequality (1) arises from the fact that there should always be enough reserved space to accommodate the workload expansion for handling f failures, where $\binom{t-1}{f-1}$ and $\binom{t-1}{f}$ are combinatorial numbers $t-1$ over $f-1$ and $t-1$ over f respectively. Let us consider a particular scenario that the first f bins (B_1 to B_f) are faulty. In this case, we have $\sum_{i=1}^{f} W_i^p(1-\frac{1}{\eta}) \leq \sum_{i=f+1}^{t} W_i^e$, where the left side is the total workload expansion requirement, which should be no more than the right side which is the total reserved space of the remaining $t-f$ correct bins. As any f bins can fail, we have $\binom{t}{f}$ similar inequalities. Inequality (1) is derived by adding up all these inequalities.

Moreover, the total workload of primary replicas $\sum_{i=1}^{t} W_i^p = L$ plus the total workload of standby replicas $\sum_{i=1}^{t} W_i^s = \frac{f\sum_{i=1}^{t} W_i^p}{\eta} = \frac{fL}{\eta}$ plus the total reserved space $\sum_{i=1}^{t} W_i^e$ should be equal to the total capacity of t bins, *i.e.* t. Thus, we have $L + \frac{fL}{\eta} + \sum_{i=1}^{t} W_i^e = t$. Together with the inequality (1), we can derive that $\eta t^2 - (\eta L + fL + \eta f)t + (f^2L + fL) \geq 0$. Therefore, we have $t \geq \frac{(\eta L+fL+\eta f)+\sqrt{(\eta L+fL+\eta f)^2-4\eta(f^2L+fL)}}{2\eta}$. □

Note that when considering η and f as constants and L to be a large value, we could use the asymptotic notation to rewrite the lower bound provided in Theorem 1 as $t \geq L + \frac{fL}{\eta} + o(L)$, where $L + \frac{fL}{\eta}$ is the space occupied by all primary and standby replicas and $o(L)$ is the reserved space.

5 Three Heuristic Algorithms

We start by considering a basic case in which the primary replicas of all services have the same workload $\ell \in (0,1)$. We present and study three heuristic algorithms. For ease of presentation, we shall refer to the total workload of all the replicas (primary and standby) placed in a bin as the level of the bin.

5.1 The Mirroring Algorithm

The original idea of the mirroring algorithm $\mathfrak{A}_m$ is from [13] which places primary replicas and standby replicas separately. Any bin that is used to place primary replicas has f "mirror" bins for standby replicas. Specifically, the algorithm uses Next-Fit to pack the primary replicas and creates f copies of each bin opened by Next-Fit to pack the corresponding standby replicas.

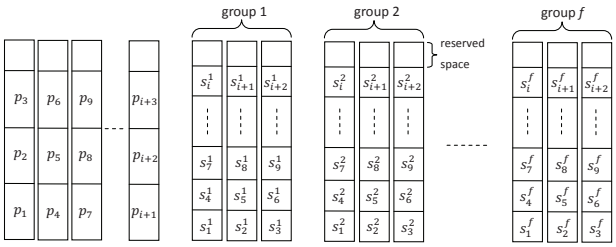


Fig. 2: Example of the algorithm $\mathfrak{A}_s$

Apparently, the algorithm $\mathfrak{A}_m$ satisfies the exclusion constraint. It also meets the space constraint obviously: even if all the standby replicas in a bin switch to primary together, the level of the bin increases to $\ell\lfloor\frac{1}{\ell}\rfloor$, which is still no more than 1.

Theorem 2. *The mirroring algorithm $\mathfrak{A}_m$ is feasible.*

Theorem 3. *To place a set of services $\mathfrak{R}$ with a total workload L of all primary replicas, the mirroring algorithm $\mathfrak{A}_m$ requires $\lceil\frac{L}{\ell\lfloor 1/\ell\rfloor}\rceil(f+1)$ bins.*

Theorem 3 is straightforward. $\mathfrak{A}_m$ needs to open $f+1$ bins to host every $\lfloor\frac{1}{\ell}\rfloor$ services. Therefore, to place all services of $\mathfrak{R}$, $\mathfrak{A}_m$ requires $\lceil\frac{L}{\ell\lfloor 1/\ell\rfloor}\rceil(f+1)$ bins. Thus, the bound provided in Theorem 3 is tight. Note that there exists at most one bin hosting primary replicas whose level is no more than $\frac{1}{2}$. Thus, a simpler but looser bound on the number of bins opened is $2(f+1)L + c$ (where c is a constant). For the mirroring algorithm, the utilization of the bin capacity is generally inefficient. Because of failures, all standby replicas located the same bin might switch to primary and expand their workloads together, which requires a large amount of reserved space.

5.2 The Shifting Algorithm

To reduce the number of bins used, we study a new algorithm named shifting which is derived from [14]. We generalize the basic idea by replication and adapt it to tolerate any f faulty bins in our model.

The shifting algorithm $\mathfrak{A}_s$ also places primary replicas and standby replicas separately. To save the amount of reserved space, the key idea of $\mathfrak{A}_s$ is to let each bin hosting standby replicas keep a reserved space which can accommodate only one standby switching to primary: the reserved space is set to $\ell(1-1/\eta)$. Suppose that q is the number of standby replicas hosted by a bin. Then, we should have $\ell(1-1/\eta) + q \cdot \ell/\eta \leq 1$. Thus, each bin can host $q = \lfloor\frac{1-\ell(1-1/\eta)}{\ell/\eta}\rfloor = \lfloor\frac{\ell+\eta-\ell\eta}{\ell}\rfloor$ standby replicas. The algorithm uses Next-Fit to pack the primary replicas and then runs the Round-Robin algorithm f times on $\lfloor 1/\ell\rfloor$ bins to pack the standby replicas. As a result, for the primary replicas hosted in the same bin, their standby replicas are shifted into f groups of bins, where each group contains $\lfloor\frac{1}{\ell}\rfloor$ bins (Figure 2 shows an example by assuming $\lfloor\frac{1}{\ell}\rfloor = 3$). $\mathfrak{A}_s$ ensures that no two standby replicas share a common bin if their primary replicas are placed in the same bin.

Theorem 4. *The shifting algorithm $\mathfrak{A}_s$ is feasible.*

Proof. The algorithm definition of $\mathfrak{A}_s$ directly ensures the exclusion constraint. For the space constraint, it is easy to show

that a switching strategy always exists if the number of failures is no more than f. Suppose x ($0 \leq x \leq f$) bins which host primary replicas turn to faulty and up to $f - x$ bins hosting standby replicas also turn to faulty. Then, at least x groups of bins for hosting standby replicas do not have any failures. Therefore, a switching strategy can be found by constructing a one-to-one mapping between the x faulty bins hosting primary replicas and the x groups of correct bins hosting standby replicas. For each primary replica in a faulty bin, its standby in the corresponding group of correct bins is switched to primary. □

Theorem 5. *To place a set of services $\mathfrak{R}$ with a total workload L of all primary replicas, the shifting algorithm $\mathfrak{A}_s$ requires at most $\lceil \frac{L}{\lfloor \frac{\ell+\eta-\ell\eta}{\ell} \rfloor \lfloor \frac{1}{\ell} \rfloor \ell} \rceil (\lfloor \frac{\ell+\eta-\ell\eta}{\ell} \rfloor + f \lfloor \frac{1}{\ell} \rfloor)$ bins.*

We know that one bin can host $\lfloor \frac{\ell+\eta-\ell\eta}{\ell} \rfloor$ standby replicas. It implies that to pack $\lfloor \frac{\ell+\eta-\ell\eta}{\ell} \rfloor \lfloor \frac{1}{\ell} \rfloor$ services, $\mathfrak{A}_s$ requires $\lfloor \frac{\ell+\eta-\ell\eta}{\ell} \rfloor + f \lfloor \frac{1}{\ell} \rfloor$ bins, where $\lfloor \frac{\ell+\eta-\ell\eta}{\ell} \rfloor$ is the number of bins hosting primary replicas and $f \lfloor \frac{1}{\ell} \rfloor$ is the number of bins hosting standby replicas. Therefore, to pack the set of services $\mathfrak{R}$, $\mathfrak{A}_s$ requires at most $\lceil \frac{L}{\lfloor \frac{\ell+\eta-\ell\eta}{\ell} \rfloor \lfloor \frac{1}{\ell} \rfloor \ell} \rceil (\lfloor \frac{\ell+\eta-\ell\eta}{\ell} \rfloor + f \lfloor \frac{1}{\ell} \rfloor)$ bins. The bound is asymptotically tight (the gap is at most a constant number of bins no more than $\lfloor \frac{\ell+\eta-\ell\eta}{\ell} \rfloor + f \lfloor \frac{1}{\ell} \rfloor - (f+1)$).

For easier comparison with the mirroring algorithm, we can derive another bound for $\mathfrak{A}_s$ in a simpler form. Except for at most one bin, the level of each bin hosting primary replicas is greater than $\frac{1}{2}$. Similarly, except for f groups of bins, the level of each bin hosting standby replicas is greater than $\frac{1}{2}(1-\ell(1-\frac{1}{\eta}))$, since otherwise there is no need to open another f groups of bins to host new standby replicas. Consequently, we have a bound $2L(1+\frac{f}{\ell+\eta-\ell\eta})+c$ (where c is a constant), where $2L$ corresponds to the number of bins hosting primary replicas and $\frac{2Lf}{\ell+\eta-\ell\eta}$ corresponds to the number of bins hosting standby replicas. Recall that the mirroring algorithm $\mathfrak{A}_m$ requires $2L(1+f)+c$ bins. Since $\eta > 1$ ensures $\ell + \eta - \ell\eta > 1$, it implies that $\mathfrak{A}_s$ achieves a better performance than $\mathfrak{A}_m$ especially when L is large.

The shifting algorithm $\mathfrak{A}_s$ decreases the amount of reserved space by guaranteeing that in each bin at most one standby might switch to primary in the case of failures. However, the process of opening bins by $\mathfrak{A}_s$ is not smooth in the online setting, because for primary replicas located in the same bin, their standby replicas span over $f \lfloor \frac{1}{\ell} \rfloor$ bins. Even for packing a few services, $\mathfrak{A}_s$ needs to quickly open a large number of bins, especially when ℓ is small. Aggregating several small services into one larger service for packing purposes could relieve the smooth problem. However, the aggregation would increase the amount of reserved space needed and decrease the packing performance. Thus, we are interested in designing a new algorithm which can balance the performance and the process of opening bins without resorting to aggregation.

5.3 The Mixing Algorithm

5.3.1 An Ideal Case

Let us first study an ideal case by adding two simplifying assumptions. First, the workload ℓ is a rather small value (*e.g.*, a value approaching 0) such that no matter how the total workload L is divided, the workload of each partition can be considered an integral multiple of ℓ. Second, $f = 1$: at most one faulty bin can arise. We propose an offline packing algorithm $\mathfrak{A}_{avg}$ that attempts to mix primary replicas and standby replicas in the same bin.

As $f = 1$, each released service has two replicas: one primary and one standby. The specification of $\mathfrak{A}_{avg}$ is based on a "dual average" process. Assume $\mathfrak{A}_{avg}$ requires t_{avg} bins to pack the set of services $\mathfrak{R}$. First, all primary replicas are uniformly distributed in t_{avg} bins, *i.e.*, in each bin B_i ($1 \leq i \leq t_{avg}$), the total workload of primary replicas is equal to $\frac{L}{t_{avg}}$. Second, for the primary replicas hosted by B_i, their standby replicas are uniformly distributed in the other $t_{avg} - 1$ bins. Thus, in each bin B_i, the aggregate workload of standby replicas is $\frac{L}{\eta t_{avg}(t_{avg}-1)}(t_{avg}-1) = \frac{L}{\eta t_{avg}}$.

Suppose that a bin B_i becomes faulty. In each correct bin B_j ($1 \leq j \leq t_{avg}$, $j \neq i$), there exist standby replicas with the aggregate workload $\frac{L}{\eta t_{avg}(t_{avg}-1)}$ whose primary replicas are hosted by B_i. Therefore, after standby switching to primary, the workload expansion in B_j will be $(1-\frac{1}{\eta})\frac{L}{t_{avg}(t_{avg}-1)}$. To avoid overloading, the reserved space in each bin should be $(1-\frac{1}{\eta})\frac{L}{t_{avg}(t_{avg}-1)}$. Accordingly, the total reserved space of t_{avg} bins should be $(1-\frac{1}{\eta})\frac{L}{t_{avg}-1}$. As t_{avg} bins are opened in total, we have: $L + \frac{L}{\eta} + (1-\frac{1}{\eta})\frac{L}{t_{avg}-1} \leq t_{avg}$. Thus, we can derive that $t_{avg} \geq \frac{(\eta L+L+\eta)+\sqrt{(\eta L+L+\eta)^2-8\eta L}}{2\eta}$. Note that this result is exactly the lower bound obtained in Theorem 1 by assuming $f = 1$. Thus, $\mathfrak{A}_{avg}$ is optimal for the ideal case and Theorem 1 is tight for this case.

Based on the observations from the above ideal case, we next introduce an algorithm which follows the idea of "dual average".

5.3.2 The Algorithm $\mathfrak{A}_x$

One possible implementation of the "dual average" idea is to place primary replicas in a group of bins in a round-robin manner and place standby replicas for the i^{th} ($i \geq 1$) primary replica in bin j in bins $j+(i-1)f+1, j+(i-1)f+2, \ldots, j+i \cdot f$ (modulo the size of the bin group). In this way, except for the first service, at most one new bin is opened for each new service released. Although this implementation can smooth the process of opening bins compared with the shifting algorithm, there is room for further improvement. In this section, we propose a novel algorithm named mixing that aims to reuse open bins for new services as much as possible before opening a new bin in the online setting.

In a nutshell, the mixing algorithm $\mathfrak{A}_x$ opens new bins gradually and organizes them into a sequence of bin groups. Each bin hosts at most α primary replicas. α is a key parameter used by $\mathfrak{A}_x$. At this stage, we only claim that α is a positive integer and bounds the number of primary replicas hosted in each bin. After we elaborate $\mathfrak{A}_x$, we shall analyze the value of α. When a new service is released, if there exists one bin in the current bin group hosting less than α primary replicas but there are not enough bins to host f more standby replicas, $\mathfrak{A}_x$ opens f new bins and adds them into the current group. If no bin in the current group can host any more primary replica, $\mathfrak{A}_x$ opens $2f$ new bins and organizes them as a new group. To simplify presentation, we use G_g to represent the g^{th} bin group created and when we mention a bin $B_i \in G_g$, the subscript i ($i \geq 1$) stands for the i^{th} opened bin of G_g. For each released service, $\mathfrak{A}_x$ always places its $f+1$ replicas in $f+1$ different bins of the same group. The notation $|G_g|$ is used to express the cardinality (the number of bins) of the group G_g. $|G_g|$ increases as more bins are added into G_g. We use ϑ to represent the final cardinality of G_g (*i.e.*, at the moment when G_{g+1} is created). The placement of $\mathfrak{A}_x$ has a simple feature: for the α primary replicas hosted in the same bin B_i, their $f\alpha$ standby replicas are dispersed in the next $f\alpha$ bins starting from $B_{(i+1) \bmod \vartheta}$ to $B_{(i+f\alpha) \bmod \vartheta}$.

```
1:  g ← 1;  % the index of the current group;
2:  G_g ← ∅;
3:  When a new service R_i is released;
4:  if G_g = ∅ then
5:      G_g ← G_g ∪ {open 2f bins};
6:      PlaceService(G_g, R_i, B_1);
7:  else
8:      Set B̃ ← TryPlacePrimary(G_g, p_i);
9:      if B̃ ≠ ∅ then
10:         for each B_j ∈ B̃ do
11:             c ← TryPlaceStandby(G_g, s_i, B_j);
12:             if c = j then
13:                 break;
14:             end if
15:         end for
16:         if c = default then
17:             G_g ← G_g ∪ {open f bins};
18:             c ← B̃.front();
19:         end if
20:         PlaceService(G_g, R_i, B_c)
21:     else
22:         g ← g + 1;
23:         G_g ← ∅;
24:         G_g ← G_g ∪ {open 2f bins};
25:         PlaceService(G_g, R_i, B_1);
26:     end if
27: end if
```

Fig. 3: The pseudo code of $\mathfrak{A}_x$

```
    Input: G_g, R_i, B_c;
1:  B_c ← p_i;
2:  B_j ← findCompatible(B_{(c+1) mod |G_g|});
3:  for each s_i^k do   % 1 ≤ k ≤ f
4:      B_{(j+k−1) mod |G_g|} ← s_i^k;
5:  end for
```

Fig. 4: The method $PlaceService$

The pseudo code of the algorithm $\mathfrak{A}_x$ is given in Figure 3. The parameter g is the index of the current bin group. Initially, g is set to 1 expressing the first group G_1 (line 1). Each group G_g is empty at the beginning (line 2). When a new service R_i is released (line 3), depending on whether G_g is empty (line 4) or not (line 7), there are two cases to consider. If G_g is empty, $\mathfrak{A}_x$ opens $2f$ new bins for G_g (line 5). The method $PlaceService(G_g, R_i, B_1)$ (line 6) targets to place the service R_i in the group G_g. It includes two sub-processes: one for placing the primary replica p_i and one for placing the f standby replicas s_i. A detailed description of $PlaceService$ is shown in Figure 4. The parameter B_c actually specifies the bin that will host p_i (line 1, Figure 4). To place the f standby replicas, we first identify a compatible bin B_j (line 2, Figure 4). The term compatible means B_j has not hosted any standby replica whose corresponding primary replica is located in B_c. The identification process starts from the bin $B_{(c+1)\ mod\ |G_g|}$ and goes clockwise ($B_{(c+1)\ mod\ |G_g|}$, $B_{(c+2)\ mod\ |G_g|}$, and so on). The first compatible bin B_j encountered is identified. Then, each standby replica s_i^k $(1 \leq k \leq f)$ is placed in the bin $B_{(j+k-1)\ mod\ |G_g|}$ (lines 3-5, Figure 4). Now let us go back to line 6, Figure 3. In this particular scenario where $2f$ new bins are just opened for an empty group G_g, the primary replica p_i is placed in B_1 and each standby replica s_i^k $(1 \leq k \leq f)$ is placed in B_{1+k}.

The operations from line 8 to line 26 handles the case that G_g is not empty. In this scenario, to place the new service R_i, the first task is to identify the bin B_c where the primary replica p_i will be placed (lines 8-19). $TryPlacePrimary(G_g, p_i)$ is a testing method (line 8). It checks whether there is a bin of the group G_g that still can accommodate one more primary replica. $TryPlacePrimary$ captures all the bins in G_g that can host p_i and put them in a set $\tilde{\mathfrak{B}}$. If each bin in G_g has already hosted α primary replicas (which implies that no bin in G_g can further host p_i), then $\tilde{\mathfrak{B}}$ is empty and the algorithm turns to open a new group G_{g+1} of bins (lines 22-25). Now consider the case that $\tilde{\mathfrak{B}}$ is not empty (lines 9-20). For each $B_j \in \tilde{\mathfrak{B}}$, a testing method $TryPlaceStandby$ examines whether there are f compatible bins to host f standby replicas if p_i is placed in B_j. Suppose B_j has hosted x $(x < \alpha)$ primary replicas before hosting p_i. The testing process proceeds as follows: if $|G_g| - (1 + fx) \geq f$, it returns the index j $(c = j)$, otherwise it returns a default value $(c = default)$. In the testing inequality $|G_g| - (1 + fx) \geq f$, $1 + fx$ represents the bin B_j and another fx bins hosting the fx standby replicas whose primary replicas are in B_j. The testing requires that except for these $1 + fx$ bins, the bin group G_g still has enough bins to pack f standby replicas in f different bins. $c = default$ (line 16) implies there are not enough compatible bins to accommodate the f standby replicas. In this scenario, $\mathfrak{A}_x$ opens another f new bins and adds them into the group G_g (line 17). Then c is redirected to the value $\tilde{\mathfrak{B}}.front()$ which returns the minimum index among all the bins in the set $\tilde{\mathfrak{B}}$ (line 18). After identifying the bin B_c to place the primary replica, $\mathfrak{A}_x$ executes the placement of service R_i: $PlaceService(G_g, R_i, B_c)$ (line 20).

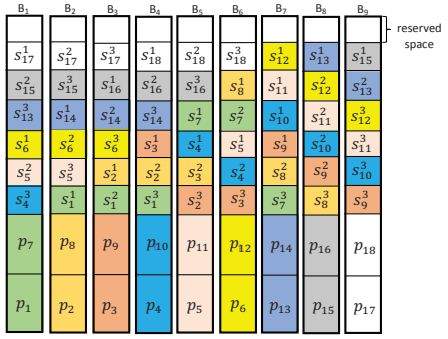


Fig. 5: Example of the algorithm $\mathfrak{A}_x$ $(f = 3)$

Figure 5 shows an example by assuming $\alpha = 2$ and $f = 3$. When the first service R_1 is released, the group is empty. $\mathfrak{A}_x$ opens 6 bins and places p_1 in B_1, s_1^1 in B_2, s_1^2 in B_3 and s_1^3 in B_4 (lines 4-6). When R_2 is released, the current group is not empty anymore. B_2 will be identified as the bin to place p_2 (line 11). Note that in the method $TryPlaceStandby$, the bin B_1 fails to pass the testing, as $|G_g| - (1 + fx) = 2 < 3$. The 3 standby replicas of R_2 will be placed in the bins B_3 to B_5. For the services R_3 to R_6, the placement is similar to R_2. When the service R_7 is released, there are not enough compatible bins to place R_7's 3 standby replicas. $\mathfrak{A}_x$ opens three new bins (B_7, B_8, B_9) and adds them into the current group. Then, the primary replicas p_7 is placed in B_1. The 3 standby replicas of R_7 is placed from B_5 to B_7. After placing the service R_{18}, the current group cannot accommodate primary replicas anymore $(\tilde{\mathfrak{B}} = \emptyset)$. $\mathfrak{A}_x$ turns to open a new group of bins (line 22).

For a given bin B_i, suppose p_j is a primary replica placed in B_i. Property 1 and Property 2 below show the locations of p_j's f standby replicas.

Property 1. *$\forall B_i \in G_g$, if p_j is the first primary replica placed in B_i, we have:*

1) *if $1 \leq i \leq f$, p_j's f standby replicas are placed in f consecutive bins from B_{i+1} to B_{i+f};*
2) *if $i > f$ and $i \bmod f = 0$, p_j's f standby replicas are placed in f consecutive bins from B_1 to B_f;*
3) *otherwise, p_j's f standby replicas are placed in the following f bins: from B_1 to $B_{i \bmod f}$ and from B_{i+1} to $B_{i+f-(i \bmod f)}$.*

Property 2. *$\forall B_i \in G_g$, if p_j is the x^{th} primary replica placed in B_i, where $1 < x \leq \alpha$, we have:*

1) *if $1 \leq i \leq f$, p_j's f standbys are placed in f consecutive bins from $B_{i+(x-1)f+1}$ to B_{i+xf};*
2) *if $f+1 \leq i \leq xf$, p_j's f standbys are placed in the f consecutive bins from B_{xf+1} to $B_{(x+1)f}$;*
3) *if $i \geq xf+1$, there are two subcases: (a) $i \bmod f = 0$, p_j's f standbys are placed in f consecutive bins from $B_{(x-1)f+1}$ to B_{xf}; (b) $i \bmod f \neq 0$, p_j's f standbys are placed in f consecutive bins from $B_{i \bmod f+(x-2)f+1}$ to $B_{i \bmod f+(x-1)f}$.*

Property 1 shows the placement of p_j's f standbys when p_j is the first primary placed in B_i. Note that the first two cases use a special term "f consecutive bins". In the following text, we use the word "consecutive" to exclude the placement of standbys turning around to B_1. More formally, during the placement of p_j's f standbys, the indices of the f successive bins need to keep increasing. When a new bin group is created, we add $2f$ bins. Clearly, if p_j is placed in one of the first f bins, its f standbys can be placed in the next f consecutive compatible bins. Another case is that p_j's f standbys are placed from B_1 to B_f. This happens only when p_j is located in a bin B_i where $i > f$ and $i \bmod f = 0$. For all the other cases, the placement of p_j's f standby replicas needs to be turned around to B_1. For example, in Figure 5, p_6's standbys are placed in three consecutive bins B_1, B_2, B_3, and p_5's standbys are placed in B_6, B_1, B_2 (note that when placing the services R_5 and R_6, the three bins B_7, B_8 and B_9 are not opened yet).

Property 2 shows the placement of p_j's f standbys when p_j is not the first primary placed in B_i. In this case, the f standbys are always placed in f consecutive bins. As shown in Figure 5, p_7's standbys are placed in three consecutive bins B_5, B_6 and B_7. Note that the method $TryPlaceStandby$ in line 11 ensures there exist enough compatible bins in G_g to host f more standbys. For example, when the service R_7 is released, $TryPlaceStandby$ returns a default value. Hence, before placing the service R_7, the bins B_7, B_8 and B_9 are opened first.

Corollary 1. *$\forall B_i \in G_g$, for any primary replica p_j located in B_i, if p_j's f standbys are located in f consecutive bins, then these f bins start from a bin whose index modulo f is equal to $(i+1) \bmod f$.*

Corollary 1 can be directly obtained from Property 1 (the first two cases) and Property 2.

Corollary 2. *$\forall B_i \in G_g$, for any primary replica p_j located in B_i and any integer $r \in [0, f-1]$, $\mathfrak{A}_x$ ensures that there exists a bin $B_y \in G_g$ which hosts one of p_j's standby replicas and satisfies $y \bmod f = r$.*

According to Property 1 and Property 2, if p_j's f standby replicas are located in f consecutive bins, the above claim is straightforward. Otherwise, p_j's f standby replicas must be placed from B_1 to $B_{i \bmod f}$ and from B_{i+1} to $B_{i+f-(i \bmod f)}$ (case 3 of Property 1). Again, $\forall r \in [0, f-1]$, the claim holds.

Corollary 3. *Every standby replica of each service is placed in a compatible bin.*

According to Property 1 and Property 2, it is easy to see that for all the α services whose primary replicas are located in B_i, none of their standbys are placed in the same bin.

Property 3. *When $\mathfrak{A}_x$ creates a new group of bins G_{g+1}, the final cardinality ϑ of G_g is equal to $f(\alpha+1)$ and $\forall B_i \in G_g$, there are α primary replicas and $f\alpha$ standby replicas placed in B_i.*

Property 3 shows the final cardinality of a bin group and the composition of each bin. In the specification of $\mathfrak{A}_x$, a new group of bins is opened only when each bin in the current bin group already hosts α primary replicas (the result of $TryPlacePrimary(G_g, p_i)$ is an empty set, line 8). Note that $\mathfrak{A}_x$ never places $f+1$ replicas of the same service across different bin groups. Furthermore, based on Corollary 3, for the α primary replicas located in the same bin, we know the corresponding $f\alpha$ standby replicas are placed in another $f\alpha$ different bins. Hence, when a new bin group is created, the current bin group includes at least $f\alpha+1$ bins. On the other hand, the cardinality of a bin group is always an integral multiple of f. We can get this conclusion through the increasing of a bin group's cardinality. $|G_g|$ increases at two places. One is opening $2f$ bins for an empty group (line 5 and line 24) and the other is adding f bins when there are not enough bins to place standby replicas (line 17). Hence, the final cardinality of a bin group should be $f(\alpha+1)$, the minimum integral multiple of f which is larger than $f\alpha+1$. From Property 1 and Property 2, we can see that for all the α primary replicas located in B_i, the $f\alpha$ standby replicas are placed in the $f\alpha$ bins next to B_i clockwise. Therefore, we can conclude that each bin accommodates $f\alpha$ standby replicas (whose primary replicas are located in $f\alpha$ different bins).

Property 4. *$\forall B_i \in G_g$, there is a set E_i of f bins not hosting any standby replica whose primary is placed in B_i. Specifically,*

1) *if $i \geq f$, $E_i = \{B_{i-f+1}, B_{i-f+2}, ..., B_i\}$.*
2) *if $1 \leq i < f$, $E_i = \{B_{\vartheta-f+i+1}, ..., B_\vartheta, B_1, ..., B_i\}$.*

We can directly get Property 4 from Property 3 and the placement of standby replicas (Property 1 and Property 2). The final cardinality of a bin group is $f(\alpha+1)$ and for α primary replicas located in the same bin, their $f\alpha$ standby replicas are placed in $f\alpha$ different bins. Clearly, there exist f bins hosting none of these standby replicas. The placement of standby replicas goes clockwise. Hence, we can identify these f bins starting from B_i by going anticlockwise.

Lemma 1. *Setting $\alpha = \lfloor \frac{\eta-\eta\ell+\ell}{\eta\ell+f\ell} \rfloor$ ensures the reserved space in each bin is greater than or equal to $\ell(1-\frac{1}{\eta})$.*

Proof. According to Property 3, there are at most α primary replicas and $f\alpha$ standby replicas located in the same bin. Let W^e be the reserved space in each bin. As the capacity of each bin is 1, we have the following inequality: $\ell\alpha + \frac{\ell}{\eta}f\alpha + W^e \leq 1$. $\ell\alpha$ stands for the space occupied by primary replicas. $\frac{\ell}{\eta}f\alpha$ is

the space occupied by standby replicas. Setting $\alpha = \lfloor \frac{\eta-\eta\ell+\ell}{\eta\ell+f\ell} \rfloor$ implies $\alpha \leq \frac{\eta-\eta\ell+\ell}{\eta\ell+f\ell}$. Thus, the reserved space W^e can be at least $1-(1+\frac{f}{\eta})\ell\frac{\eta-\eta\ell+\ell}{\eta\ell+f\ell} \geq \ell(1-\frac{1}{\eta})$. □

For each bin, a reserved space $\ell(1-\frac{1}{\eta})$ can accommodate one standby switching to primary. Therefore, Lemma 1 essentially derives the maximum value of α. Note that the value $\lfloor \frac{\eta-\eta\ell+\ell}{\eta\ell+f\ell} \rfloor$ could be 0 when $\ell > \frac{\eta}{2\eta+f-1}$. If so, we resort to the shifting algorithm to pack the services. In the following, unless stated otherwise, we focus on the scenario $\ell \leq \frac{\eta}{2\eta+f-1}$.

Corollary 4. *There is no overloaded bin before failure occurs.*

Corollary 4 directly follows from Lemma 1.

Theorem 6. *The mixing algorithm $\mathfrak{A}_x$ is feasible.*

To prove Theorem 6, we need to prove that a switching strategy exists as long as there are no more than f faulty bins. For each released service, $\mathfrak{A}_x$ places all the $f+1$ replicas into the same group of bins. Therefore, if faulty bins span over multiple groups, a switching strategy can operate independently in each group one by one. Without loss of generality, suppose all the faulty bins are in the same group G_g and let $F_g \subset G_g$ denote all the faulty bins in G_g. Recall that the bins in the group G_g are indexed from 1 to ϑ. The switching strategy we propose is based on "the bin index modulo f". Suppose $B_i \in F_g$ (the bin indexed i in group G_g) is a faulty bin. Then in general, for the faulty primary replicas in B_i, the switching strategy prefers to select the standby replicas in those correct bins whose indices modulo f are equal to $i \ mod \ f$.

We first give a few necessary definitions. The operation "modulo f" has an integer result ranging from 0 to $f-1$. For each integer $r \in [0, f-1]$, we define a set $\tilde{F}_r$ as follows: $\tilde{F}_r = \{B_i | B_i \in F_g \wedge i \ mod \ f = r\}$. In particular, if $\forall B_i \in F_g$, $i \ mod \ f \neq r$ holds, then $\tilde{F}_r = \emptyset$. Define $N_{ept} = \{r|\tilde{F}_r = \emptyset\}$, $N_{sgl} = \{r||\tilde{F}_r| = 1\}$ and $N_{mul} = \{r||\tilde{F}_r| > 1\}$. It is easy to see that $|N_{ept}| + |N_{sgl}| + |N_{mul}| = f$. Meanwhile, since the maximum number of faulty bins is up to f, we have $\sum_{r\in N_{sgl}} |\tilde{F}_r| + \sum_{r\in N_{mul}} |\tilde{F}_r| = |N_{sgl}| + \sum_{r\in N_{mul}} |\tilde{F}_r| \leq f$. Consequently, we get $|N_{ept}| \geq \sum_{r\in N_{mul}} |\tilde{F}_r| - |N_{mul}|$. Now for each nonempty set $\tilde{F}_r$, we define its correlative remainder set C_r.

Definition 5. *When $|\tilde{F}_r| = 1$, its correlative remainder set is given by $C_r = \{r\}$; When $|\tilde{F}_r| > 1$, its correlative remainder set C_r is composed of r and $|\tilde{F}_r| - 1$ elements from the set N_{ept}.*

When $|\tilde{F}_r| = 1$, $\tilde{F}_r$'s correlative remainder set is unique $C_r = \{r\}$. When $|\tilde{F}_r| > 1$, $\tilde{F}_r$ could have multiple possible correlative remainder sets as $|N_{ept}| \geq \sum_{r\in N_{mul}} |\tilde{F}_r| - |N_{mul}|$ implies $|N_{ept}| \geq |\tilde{F}_r| - 1$.

Lemma 2. *We can find a correlative remainder set C_r for each nonempty set $\tilde{F}_r$ such that for any two nonempty sets $\tilde{F}_{r_i}$ and $\tilde{F}_{r_j}$, $C_{r_i} \cap C_{r_j} = \emptyset$.*

Proof. For all the sets $\tilde{F}_r$ with cardinality greater than 1, $\sum_{r\in N_{mul}} |\tilde{F}_r| - |N_{mul}|$ gives the total number of elements needed from N_{ept}. Since $|N_{ept}| \geq \sum_{r\in N_{mul}} |\tilde{F}_r| - |N_{mul}|$, there exist non-overlapping remainder sets for these $\tilde{F}_r$'s. □

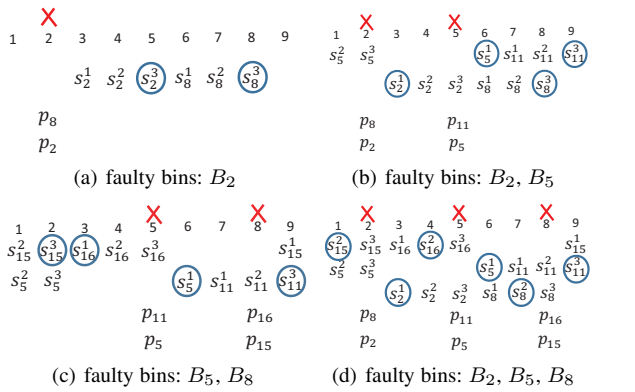


Fig. 6: Examples of $\mathcal{S}$

In what follows, we shall assume that $C_{r_i} \neq C_{r_j}$ for any $r_i \neq r_j$, $\tilde{F}_{r_i} \neq \emptyset$ and $\tilde{F}_{r_j} \neq \emptyset$. When $|\tilde{F}_r| > 1$, we use $B_{r_{max}}$ ($B_{r_{min}}$) to represent the faulty bin in $\tilde{F}_r$ that has the maximum (minimum) number. For example, suppose $f = 3$ and $\tilde{F}_1 = \{B_1, B_7, B_{10}\}$. Then $B_{1_{max}}$ stands for B_{10} and $B_{1_{min}}$ stands for B_1. Note that $\forall \tilde{F}_r \neq \emptyset$, $\tilde{F}_r$ and C_r has the same cardinality $|\tilde{F}_r| = |C_r|$. We define a one-to-one mapping as follows.

Definition 6. *ξ: $\tilde{F}_r \mapsto C_r$ is a one-to-one mapping that satisfies: if $r_{min} \leq f$, $\xi(B_{r_{min}}) = r$; otherwise, if $r_{min} > f$, $\xi(B_{r_{max}}) = r$.*

We define a strategy $\mathcal{S}$ based on the above mapping.

Definition 7. *$\forall \tilde{F}_r \neq \emptyset$, $\forall B_i \in \tilde{F}_r$, $\mathcal{S}$ returns a set of bins φ_i and $\forall B_j \in \varphi_i$, $\mathcal{S}$ switches the standby in B_j whose corresponding primary is in B_i to primary. The set φ_i is defined as follows:*

1) *$|\tilde{F}_r| = 1$. $\forall B_j \in G_g$, if $B_j \notin E_i$ and $j \ mod \ f = r$, then $B_j \in \varphi_i$;*
2) *$|\tilde{F}_r| > 1$. There are two cases:*
   a) *$\xi(B_i) \neq r$. $\forall B_j \in G_g$, if $B_j \notin E_i$ and $j \ mod \ f = \xi(B_i)$, then $B_j \in \varphi_i$;*
   b) *$\xi(B_i) = r$. $\forall B_j \in G_g$ that $B_j \notin E_i$ and $j \ mod \ f = r$: (1) if $B_j \notin \tilde{F}_r$, then $B_j \in \varphi_i$; (2) if $B_j \in \tilde{F}_r$, then $B_k \in \varphi_i$ which satisfies $B_k \in E_j$ and $k \ mod \ f = \xi(B_j)$.*

Following the placement example shown in Figure 5, we construct four examples of $\mathcal{S}$ in Figure 6. In Figure 6(a), there is one faulty bin B_2. As $2 \ mod \ 3 = 2$, the standby replicas s_2^3 and s_8^3 will switch to be the new primary replicas. In Figure 6(b), there are two faulty bins B_2 and B_5. As $2 \ mod \ 3 = 5 \ mod \ 3$, we have $|\tilde{F}_2| = 2$. Based on Definition 6, we have $\xi(B_2) = 2$. Note that $\tilde{F}_0 = \emptyset$, which implies we could define $\xi(B_5) = 0$ to get a feasible one-to-one mapping ξ. In this scenario, the standby replicas s_5^1, s_{11}^3, s_8^3 and s_2^1 will switch to become the new primary replicas. In Figure 6(c), there are two faulty bins B_5 and B_8. According to Definition 6, now $\xi(B_8) = 2$. Similarly, as $\tilde{F}_0 = \emptyset$, we get $\xi(B_5) = 0$. Therefore, s_5^1, s_{11}^3, s_{15}^3 and s_{16}^1 will become the new primary replicas. In Figure 6(d), there are three faulty bins B_2, B_5 and B_8. According to Definition 6, now $\xi(B_2) = 2$. Note that $\tilde{F}_0 = \emptyset$ and $\tilde{F}_1 = \emptyset$, we could define $\xi(B_5) = 0$ and $\xi(B_8) = 1$ to get a feasible one-to-one mapping ξ. Thus, s_2^1, s_8^2, s_5^1, s_{11}^3, s_{15}^2 and s_{16}^2 will become the new primaries.

The following Lemma 3 ensures $\mathcal{S}$ always selects correct bins.

Lemma 3. *$\forall \tilde{F}_r \neq \emptyset$, $\forall B_i \in \tilde{F}_r$, $\forall B_j \in \varphi_i$, B_j is a correct bin.*

Proof. Consider $|\tilde{F}_r| = 1$. B_i is the only one faulty bin that meets $i \ mod \ f = r$. In this case, φ_i includes B_j if $B_j \notin E_i$ and $j \ mod \ f = r$. Note that $B_i \notin \varphi_i$ as $B_i \in E_i$. Therefore, all bins in φ_i are correct bin. Consider $|\tilde{F}_r| > 1$. When $\xi(B_i) \neq r$, φ_i includes B_j if $B_j \notin E_i$ and $j \ mod \ f = \xi(B_i)$. Suppose $\xi(B_i) = r'$. According to Definition 5, we know that $r' \in N_{ept}$, *i.e.*, $\tilde{F}_{r'} = \emptyset$. Therefore, all bins in φ_i are correct. When $\xi(B_i) = r$, there are two cases. One case is $B_j \in \varphi_i$ and $j \ mod \ f = r$. In this case, the condition $B_j \notin \tilde{F}_r$ directly implies B_j is correct. The other case is $B_j \in \varphi_i$ and $j \ mod \ f \neq r$. Again, the definition of the correlative remainder set ensures B_j is correct. □

Lemma 4. *$\forall \tilde{F}_r \neq \emptyset$, for all faulty primary replicas hosted by the bins in $\tilde{F}_r$, $\mathcal{S}$ ensures that no correct bin has more than one standby replica switching to primary.*

Proof. First, consider $|\tilde{F}_r| = 1$. Suppose $B_i \in \tilde{F}_r$ is the faulty bin. If $B_j \notin E_i$ and $j \ mod \ f = r$, then $\mathcal{S}$ chooses the corresponding standby in B_j to be the new primary. Corollary 3 ensures B_j hosts at most one standby replica whose primary is in B_i, so the claim is true.

Now consider $|\tilde{F}_r| > 1$. We need to prove that $\forall B_i, B_j \in \tilde{F}_r$, $\varphi_i \cap \varphi_j = \emptyset$. There are two cases. (1) $\xi(B_i) \neq r$ and $\xi(B_j) \neq r$. As ξ is a one-to-one mapping, we have $\xi(B_i) \neq \xi(B_j)$. According to the specification of $\mathcal{S}$, we know $\forall B_k \in \varphi_i$, $k \ mod \ f = \xi(B_i)$ and $\forall B_y \in \varphi_j$, $y \ mod \ f = \xi(B_j)$. Thus, $\varphi_i \cap \varphi_j = \emptyset$. (2) Without loss of generality, suppose $\xi(B_i) = r$ and $\xi(B_j) \neq r$. $\forall B_y \in \varphi_j$, we know $y \ mod \ f = \xi(B_j)$ and $B_y \notin E_j$. $\forall B_k \in \varphi_i$, there are two scenarios: $k \ mod \ f = \xi(B_i)$ or $k \ mod \ f = \xi(B_j)$. If $k \ mod \ f = \xi(B_i)$, clearly $B_k \notin \varphi_j$. If $k \ mod \ f = \xi(B_j)$, according to the specification of $\mathcal{S}$, we have $B_k \in E_j$. Again, $B_k \notin \varphi_j$. Therefore, $\forall B_i, B_j \in \tilde{F}_r$, $\varphi_i \cap \varphi_j = \emptyset$ and the claim follows. □

Lemma 5. *$\forall \tilde{F}_r \neq \emptyset$, $\forall B_i \in \tilde{F}_r$, for any primary replica p_j hosted in B_i, $\mathcal{S}$ ensures one of p_j's correct standby replicas would switch to primary.*

Proof. Consider $|\tilde{F}_r| = 1$. Suppose $B_i \in \tilde{F}_r$ is the faulty bin. In this case, we need to prove $\forall p_j \in B_i$, there is a bin $B_y \in \varphi_i$ which hosts one of p_j's standby replicas. The algorithm $\mathfrak{A}_x$ ensures p_j has a standby replica located in a bin B_y that $y \ mod \ f = r$ (Corollary 2). Meanwhile, according to Property 4, we have $B_y \notin E_i$. On the other hand, from the specification of $\mathcal{S}$ (Definition 7), we know when $|\tilde{F}_r| = 1, \forall B_k \in G_g$ if B_k satisfies $k \ mod \ f = r$ and $B_k \notin E_i$, then $B_k \in \varphi_i$. Consequently, we get $B_y \in \varphi_i$.

Consider $|\tilde{F}_r| > 1$. We need to prove $\forall B_i \in \tilde{F}_r$ and $\forall p_j \in B_i$, there is a bin $B_y \in \varphi_i$ which hosts one of p_j's standby replicas. According to Definition 6, suppose B_{i_1} stands for a bin in $\tilde{F}_r$ that $\xi(B_{i_1}) \neq r$ and suppose B_{i_2} is the bin in $\tilde{F}_r$ that $\xi(B_{i_2}) = r$. For any primary p_{j_1} located in B_{i_1}, Corollary 2 ensures that p_{j_1} has a standby replica located in a bin B_y and $y \ mod \ f = \xi(B_{i_1})$. Meanwhile, Property 4 confirms that $B_y \notin E_{i_1}$. By the specification of $\mathcal{S}$ (Definition 7, case 2(a)), we know that $\forall B_k \in G_g$, if $k \ mod \ f = \xi(B_{i_1})$ and $k \notin E_{i_1}$, then $B_k \in \varphi_{i_1}$. Therefore, we get $B_y \in \varphi_{i_1}$.

For any primary p_{j_2} located in B_{i_2}, p_{j_2} has a standby replica located in a bin B_s and $s \ mod \ f = \xi(B_{i_2}) = r$ (Corollary 2). Meanwhile, we know $B_s \notin E_{i_2}$ (Property 4). If $B_s \notin \tilde{F}_r$, according to Definition 7 (case 2(b)), we can easily get $B_s \in \varphi_{i_2}$. Now consider $B_s \in \tilde{F}_r$. We first carry out the proof based on the hypothesis that p_{j_2}'s f standby replicas are located in f consecutive bins from B_k to B_{k+f-1} and then in the last step we prove this hypothesis is true. Among the f bins B_k to B_{k+f-1}, we need to prove that there is one bin belonging to φ_{i_2}. Corollary 1 tells us that $k \ mod \ f = (i_2 + 1) \ mod \ f$, such that $(k + f - 1) \ mod \ f = (i_2 + 1 + f - 1) \ mod \ f = i_2 \ mod \ f = r$. It implies B_{k+f-1} is exactly B_s and E_s includes the f bins from B_k to B_{k+f-1}. According to Definition 7 (case 2(b)), among the f bins from B_k to B_{k+f-1}, there exists a bin B_z belonging to φ_{i_2} and satisfying $z \ mod \ f = \xi(B_s)$.

Finally, we prove the hypothesis that p_{j_2}'s f standby replicas located in f consecutive bins is true. We prove it by contradiction: suppose p_{j_2}'s f standby replicas are not located in f consecutive bins. According to Property 1 and Property 2, we know the only case is that p_{j_2} is the first primary replica placed in B_{i_2} and $i_2 > f$ and $i_2 \ mod \ f \neq 0$. Remember that now $B_s \in \tilde{F}_r$ and B_s hosts a standby replica of p_{j_2}. According to $s \ mod \ f = i_2 \ mod \ f$ and the third case of Property 1, it can be inferred that $1 \leq s \leq f - 1$, which implies B_s must be $B_{r_{min}}$ of $\tilde{F}_r$. According to Definition 6, B_s must be the only bin in $\tilde{F}_r$ that satisfies $\xi(B_s) = r$. However, we already assume $\xi(B_{i_2}) = r$ and clearly B_s and B_{i_2} are two different bins. That leads to a contradiction. □

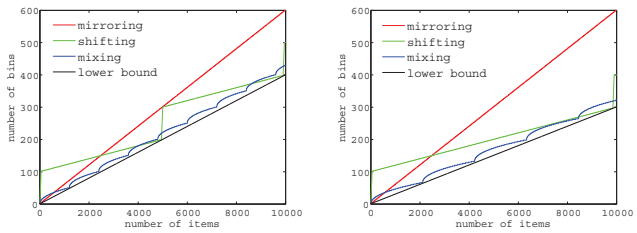
Lemma 6. *$\mathcal{S}$ is a switching strategy.*

Proof. Lemma 4 and Lemma 5 together conclude the validity of $\mathcal{S}$ for the faulty bins in each $\tilde{F}_r$. To prove $\mathcal{S}$ is a switching strategy, we still need to prove that for any two nonempty sets $\tilde{F}_{r_i}$ and $\tilde{F}_{r_j}$, no correct bin has more than one standby replica switching to primary. Lemma 2 ensures that $C_{r_i} \cap C_{r_j} = \emptyset$. According to the specification of $\mathcal{S}$, $\forall \tilde{F}_r \neq \emptyset$ and $\forall B_i \in \tilde{F}_r$, if $B_j \in \varphi_i$, then $(j \ mod \ f) \in C_r$. Therefore, $\forall B_y \in F_{r_i}$ and $\forall B_z \in F_{r_j}$, we have $\varphi_y \cap \varphi_z = \emptyset$. It implies that $\mathcal{S}$ focuses on different sets of bins for $\tilde{F}_{r_i}$ and $\tilde{F}_{r_j}$. Therefore, $\mathcal{S}$ is a switching strategy. □

Theorem 7. *To place a set of services $\mathfrak{R}$ with a total workload L of all primary replicas, the mixing algorithm $\mathfrak{A}_x$ requires at most $\frac{L}{\ell \lfloor \frac{\eta - \eta\ell + \ell}{\eta\ell + f\ell} \rfloor} + f(\lfloor \frac{\eta - \eta\ell + \ell}{\eta\ell + f\ell} \rfloor + 1)$ bins.*

According to Property 3 and Lemma 1, one bin hosts $\lfloor \frac{\eta - \eta\ell + \ell}{\eta\ell + f\ell} \rfloor$ primary replicas and one group has $f(\lfloor \frac{\eta - \eta\ell + \ell}{\eta\ell + f\ell} \rfloor + 1)$ bins. To place all services of $\mathfrak{R}$, $\mathfrak{A}_x$ requires at most $\frac{L}{\ell \lfloor \frac{\eta - \eta\ell + \ell}{\eta\ell + f\ell} \rfloor} + f(\lfloor \frac{\eta - \eta\ell + \ell}{\eta\ell + f\ell} \rfloor + 1)$ bins. The bound is asymptotically tight (the gap is at most a constant number of bins no more than $f(\lfloor \frac{\eta - \eta\ell + \ell}{\eta\ell + f\ell} \rfloor + 1)$).

For easier comparison with other algorithms, we can also have a simpler but looser upper bound. Note that by using $\mathfrak{A}_x$, each bin should keep a reserved space equal to $\ell(1 - \frac{1}{\eta})$. It implies the level of any bin is at most $1 - \ell(1 - \frac{1}{\eta})$. Consequently, for each bin (except for a constant number of bins), we know that its level is at least $\frac{1}{2}(1 - \ell(1 - \frac{1}{\eta}))$. In order to pack all the replicas, $\mathfrak{A}_x$ requires at most $\frac{fL/\eta + L}{1/2(1 - \ell(1 - 1/\eta))} + c = 2L(\frac{f}{\ell + \eta - \ell\eta} + \frac{\eta}{\ell + \eta - \ell\eta}) + c$ bins. Remember that the mirroring algorithm $\mathfrak{A}_m$ requires at most $2L(f + 1) + c$ bins and the shifting algorithm $\mathfrak{A}_s$ requires at most $2L(\frac{f}{\ell + \eta - \ell\eta} + 1) + c$ bins. When $\ell < \frac{f}{f+1}$, we have $\frac{f}{\ell + \eta - \ell\eta} + \frac{\eta}{\ell + \eta - \ell\eta} < f + 1$. Recall that $\ell \leq \frac{\eta}{2\eta + f - 1}$ is required when using $\mathfrak{A}_x$ to pack services. We have $\frac{\eta}{2\eta + f - 1} \leq \frac{f}{f+1}$ when $f \geq 1$. Therefore, when the mixing algorithm $\mathfrak{A}_x$ can be applied to pack services, it outperforms the mirroring algorithm $\mathfrak{A}_m$. The number



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

Fig. 7: Numerical examples

of bins required by the mixing algorithm may be slightly more than that by the shifting algorithm, but the process of opening bins by $\mathfrak{A}_x$ is much more smooth than that by $\mathfrak{A}_s$. Figure 7 shows two numerical examples. For both examples, we set $f = 2$ and $l = 0.02$. In Figure 7(a), η is set to 2 and in Figure 7(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 services, the mixing algorithm is still the best among the three heuristic algorithms. Moreover, when η increases, both $\mathfrak{A}_x$ and $\mathfrak{A}_s$ use fewer bins. However, $\mathfrak{A}_m$ cannot benefit from the growth of η.

6 Heterogeneous Services

We have studied the fault-tolerate bin packing problem under the assumption that all the services have the same workload. Now we extend the three proposed algorithms to pack services with different workloads. We first propose First-Fit Mirroring ($\mathfrak{A}_{fm}$) which adopts the classical packing strategy First-Fit [19] [20] to pack primary replicas. We then propose Harmonic Shifting ($\mathfrak{A}_{hs}$) and Harmonic Mixing ($\mathfrak{A}_{hx}$) algorithms. Both of them adopt the harmonic strategy which categorizes services with similar workloads into the same class. After the classification, the shifting and mixing algorithms are involved to pack services in each class separately. The two new algorithms inherit the original merits that Harmonic Shifting saves the reserved space while Harmonic Mixing balances the number of bins used and the process of opening bins in the online setting.

6.1 First-Fit Mirroring

First-Fit (FF) is a greedy packing algorithm. FF attempts to place each new item in the earliest opened bin that can accommodate the item. If no open bin can accommodate the item, FF opens a new bin for the item. By leveraging FF, we propose First-Fit Mirroring ($\mathfrak{A}_{fm}$). $\mathfrak{A}_{fm}$ uses FF to pack primary replicas. The policy to place standby replicas is the same as the original mirroring algorithm. It is clear that $\mathfrak{A}_{fm}$ is feasible.

If only one bin is used by $\mathfrak{A}_{fm}$ to place all the primary replicas, by definition, $\mathfrak{A}_{fm}$ uses a total of $f + 1$ bins. In this case, $\mathfrak{A}_{fm}$ is optimal because at least $f + 1$ bins are needed to host any service due to the exclusive constraint. If more than one bin is used by $\mathfrak{A}_{fm}$ to place the primary replicas, the average level of these bins must be greater than $\frac{1}{2}$. This is because by the definition of FF, a new bin is opened only when none of the existing bins can accommodate an item to be placed. As a result, the total level of any two bins must exceed 1 (the bin capacity) and hence, the average level of any two bins must exceed $\frac{1}{2}$. By the definition of $\mathfrak{A}_{fm}$, the average level of all the bins hosting standby replicas must be greater than $\frac{1}{2\eta}$. Since the number of bins hosting standby replicas is exactly f times that hosting primary replicas, the average level of all the bins used by $\mathfrak{A}_{fm}$ must be greater than $\frac{\frac{1}{2}+\frac{1}{2\eta}f}{1+f} = \frac{\eta+f}{2\eta(1+f)}$. From Theorem 1, we know that to pack a set of services with a total workload L of all primary replicas, it needs to open at least $L + \frac{fL}{\eta} + o(L)$ bins. Thus, the average bin level of an optimal packing algorithm is no more than $\frac{L+\frac{fL}{\eta}}{L+\frac{fL}{\eta}+o(L)}$. When L tends to be large, the asymptotic competitive ratio of $\mathfrak{A}_{fm}$ is bounded by $\frac{2\eta(1+f)}{\eta+f}$.

Theorem 8. *The asymptotic competitive ratio of First-Fit mirroring $\mathfrak{A}_{fm}$ is bounded by $\frac{2\eta(1+f)}{\eta+f}$.*

6.2 Harmonic Shifting

It is well-known that the harmonic strategy places items of similar sizes together. We now leverage this idea to adapt the shifting algorithm $\mathfrak{A}_s$ to pack services with heterogeneous workloads. We call the new algorithm Harmonic Shifting ($\mathfrak{A}_{hs}$). $\mathfrak{A}_{hs}$ divides the set of services into M (M is a positive integer) classes and places services belonging to different classes separately.

When a service R_i is released, $\mathfrak{A}_{hs}$ first decides the class that R_i belongs to. The decision is based on the workload of R_i's standby replica. If the workload of an R_i's standby replica is within the range $(\frac{1}{k+\eta}, \frac{1}{k+\eta-1}]$ $(1 \le k < M)$, then R_i belongs to the class k. If the workload is within the range $(0, \frac{1}{M+\eta-1}]$, then R_i belongs to the last class M. To pack R_i, there are two cases. If R_i belongs to a class k where $1 \le k < M$, the shifting algorithm $\mathfrak{A}_s$ is applied directly to pack R_i by treating a primary replica's workload as $\frac{\eta}{k+\eta-1}$. If R_i belongs to the last class M, R_i is also packed by $\mathfrak{A}_s$ but is considered as a part of an aggregate service. Specifically, we aggregate a set of services from class M into a larger service. We require the workload of an aggregate standby replica to be in the range $(\frac{1}{\eta+\lfloor\eta\rfloor+1}, \frac{1}{\eta+\lfloor\eta\rfloor}]$. Accordingly, each aggregate primary replica has a workload in the range $(\frac{\eta}{\eta+\lfloor\eta\rfloor+1}, \frac{\eta}{\eta+\lfloor\eta\rfloor}]$. When $M \ge 4\eta^2 + \eta + 1$, it holds that $\frac{1}{\eta+\lfloor\eta\rfloor} - \frac{1}{\eta+\lfloor\eta\rfloor+1} \ge \frac{1}{M+\eta-1}$. Consequently, we can always aggregate consecutive services in an arbitrary release sequence into an aggregate service with its standby replica's workload falling in the specified range. Thanks to the aggregation, now each aggregate replica can be treated as a replica from the class $\lfloor\eta\rfloor+1$.

For each class k where $1 \le k < M$ (the last class is treated as the class $\lfloor\eta\rfloor+1$), each bin hosts at most $\lceil\frac{k}{\eta}\rceil$ primary replicas or k standby replicas. Clearly, there are no overloaded bins for packing primary replicas. The level of a bin hosting standby replica is at most $\frac{k}{k+\eta-1}$. Thus, the available space is at least $\frac{\eta-1}{k+\eta-1}$ which is equal to the upper bound of the reserved space needed for avoiding overflow in case of failures. Together with Theorem 4 and that services from different classes are packed separately, we can conclude that $\mathfrak{A}_{hs}$ is feasible.

Now we study the packing performance of $\mathfrak{A}_{hs}$. Our analysis makes use of the method from [21]. For each class k where $1 \le k < M$, except for a constant number of bins, each bin hosts k standby replicas or $\lceil\frac{k}{\eta}\rceil$ primary replicas. Hence, we consider that each standby replica costs $\frac{1}{k}$ bin and each primary replica costs $1/\lceil\frac{k}{\eta}\rceil$ bin. For the last class M, except for a constant number of bins, the level of each bin hosting standby replicas is more than $\frac{\lfloor\eta\rfloor+1}{\eta+\lfloor\eta\rfloor+1}$ and the level of each bin hosting primary replicas is more than $\frac{\eta}{\eta+\lfloor\eta\rfloor+1}\lceil\frac{\lfloor\eta\rfloor+1}{\eta}\rceil$. Hence, a standby replica of size y costs no

more than $y \cdot \frac{\eta+\lfloor\eta\rfloor+1}{\lfloor\eta\rfloor+1}$ bin and a primary replica of size x costs no more than $x \cdot \frac{\eta+\lfloor\eta\rfloor+1}{\eta\lceil\frac{\lfloor\eta\rfloor+1}{\eta}\rceil}$ bin.

We define a weight function ω in $(0, 1]$ as follows. For a standby replica s_i with workload y_i, if s_i is from class k ($1 \leq k < M$), then $\omega(s_i) = \frac{1}{k}$; if s_i is from the last class M, then $\omega(s_i) = y \cdot \frac{\eta+\lfloor\eta\rfloor+1}{\lfloor\eta\rfloor+1}$. For a primary replica p_i with workload x_i, if p_i is from class k ($1 \leq k < M$), then $\omega(p_i) = 1/\lceil\frac{k}{\eta}\rceil$; if p_i is from the last class M, then $\omega(p_i) = x \cdot \frac{\eta+\lfloor\eta\rfloor+1}{\eta\lceil\frac{\lfloor\eta\rfloor+1}{\eta}\rceil}$. Except for a constant number of bins, the total weight of replicas hosted by each bin is at least 1. Therefore, to pack a set of services $\mathfrak{R}$, the harmonic shifting algorithm requires at most $\omega(\mathfrak{R}) + c$ bins where $\omega(\mathfrak{R})$ is the total weight of all services and c is a constant.

We now estimate an upper bound on the total weight of replicas that can be placed in one bin. Suppose B is an open bin. For ease of presentation, we use $\omega(B)$ to denote the total weight of replicas placed in B. B could host four kinds of replicas together: primary replicas from classes $1, 2, ..., M-1$, primary replicas from class M, standby replicas from classes $1, 2, ..., M-1$ and standby replicas from class M. Hence, we can write $\omega(B) = \sum_i \omega(p_i) + \sum_j \omega(p_j) + \sum_t \omega(s_t) + \sum_l \omega(s_l)$, where p_i is from a class k ($1 \leq k < M$), p_j is from class M, s_t is from a class k ($1 \leq k < M$) and s_l is from class M. Suppose p_i's workload is x_i, p_j's workload is x_j, s_t's workload is y_t and s_l's workload is y_l. We analyze them one by one. For $\omega(p_i)$, we have $\frac{\omega(p_i)}{x_i} = \frac{1}{x_i\lceil k/\eta\rceil} < \frac{k/\eta+1}{\lceil k/\eta\rceil} \leq \frac{\lceil k/\eta\rceil+1}{\lceil k/\eta\rceil} \leq 2$. The first inequality comes from the classification: if p_i is from class k, then $x_i > \frac{\eta}{k+\eta}$. Hence, we get $\omega(p_i) < 2x_i$. For $\omega(p_j)$, we have $\omega(p_j) = x_j \cdot \frac{\eta+\lfloor\eta\rfloor+1}{\eta\lceil\frac{\lfloor\eta\rfloor+1}{\eta}\rceil} \leq x_j \cdot \frac{\eta+\lfloor\eta\rfloor+1}{\lfloor\eta\rfloor+1} \leq 2x_j$. For $\omega(s_t)$, we have $\frac{\omega(s_t)}{y_t} = \frac{1}{y_t k} < \frac{k+\eta}{k}$. The inequality comes from the classification: if s_t is from class k, then $y_t > \frac{1}{k+\eta}$. Hence, we get $\omega(s_t) < y_t \cdot \frac{k+\eta}{k}$. For $\omega(s_l)$, we directly have $\omega(s_l) = y_l \cdot \frac{\eta+\lfloor\eta\rfloor+1}{\lfloor\eta\rfloor+1} \leq 2y_l$. Based on the above results, we can derive that $\omega(B) < \max(2, \frac{k^*+\eta}{k^*}) \cdot (\Sigma_i x_i + \Sigma_j x_j + \Sigma_t y_t + \Sigma_l y_l)$, where k^* is the lowest-indexed class (among $1, 2, ..., M-1$) from which B hosts a standby replica. If $\max(2, \frac{k^*+\eta}{k^*}) = 2$, clearly $\omega(B) < 2$, as the space constraint requires $\Sigma_i x_i + \Sigma_j x_j + \Sigma_t y_t + \Sigma_l y_l \leq 1$. If $\max(2, \frac{k^*+\eta}{k^*}) = \frac{k^*+\eta}{k^*}$, since B hosts standby replicas from class k^* where $1 \leq k^* < M$, the space constraint requires that the reserved space of B should be at least $\frac{\eta-1}{k^*+\eta}$. Therefore, $\Sigma_i x_i + \Sigma_j x_j + \Sigma_t y_t + \Sigma_l y_l \leq 1 - \frac{\eta-1}{k^*+\eta}$, so that $\omega(B) < \frac{k^*+\eta}{k^*}(1 - \frac{\eta-1}{k^*+\eta}) \leq 2$. Thus, we always have $\omega(B) < 2$. It follows that any bin B_{opt} opened by an optimal packing algorithm $\mathfrak{A}_{opt}$ also satisfies $\omega(B_{opt}) < 2$. Since the total weight of all services $\mathfrak{R}$ is $\omega(\mathfrak{R})$, $\mathfrak{A}_{opt}$ requires at least $\frac{\omega(\mathfrak{R})}{2}$ bins. Recall that $\mathfrak{A}_{hs}$ requires at most $\omega(\mathfrak{R}) + c$ bins. Therefore, we have the following result.

Theorem 9. *The asymptotic competitive ratio of $\mathfrak{A}_{hs}$ is bounded by* 2.

6.3 Harmonic Mixing

We also apply the idea "harmonic" to extend the mixing algorithm $\mathfrak{A}_x$ to deal with services with heterogeneous workloads. We call the new algorithm Harmonic Mixing ($\mathfrak{A}_{hx}$). $\mathfrak{A}_{hx}$ has two steps. First, it decides which class a service belongs to and second, it packs the service by the original mixing algorithm $\mathfrak{A}_x$.

The classification of services is based on the workload of the primary replica: if a service belongs to the class k ($1 \leq k < M$), its primary replica's workload is in the range $(\frac{\eta}{(k+2)\eta+(k+1)f-1}, \frac{\eta}{(k+1)\eta+kf-1}]$ and if a service belongs to the last class M, its primary replica's workload is in the range $(0, \frac{\eta}{(M+1)\eta+Mf-1}]$. This classification is derived from Property 3 in Section 5.3.2, which indicates that the mixing algorithm $\mathfrak{A}_x$ places α primary replicas and $f\alpha$ standby replicas together in one bin. By the capacity constraint, the total workload of the replicas placed in one bin plus the corresponding reserved space should be capped by the bin capacity, *i.e.*, the inequality $\ell\alpha + \frac{\ell}{\eta}f\alpha + \ell(1 - \frac{1}{\eta}) \leq 1$ holds. In this inequality, by setting a value of α, we can get a corresponding range of ℓ. For example, when $\alpha = 1$, we have $\ell \leq \frac{\eta}{2\eta+f-1}$ and when $\alpha = 2$, we have $\ell \leq \frac{\eta}{3\eta+2f-1}$. Clearly, if the workload of any primary replica is in the range $(\frac{\eta}{3\eta+2f-1}, \frac{\eta}{2\eta+f-1}]$, one bin can host 1 primary replica and f standby replicas by using $\mathfrak{A}_x$. Thus, we define $(\frac{\eta}{3\eta+2f-1}, \frac{\eta}{2\eta+f-1}]$ as the workload range for the primary replicas of the services in the class 1. Increasing α by one at a time, we can get the complete classification as given earlier.

$\mathfrak{A}_{hx}$ packs services in different classes separately. For classes 1, 2, ..., $M-1$, the original $\mathfrak{A}_x$ is directly applied by treating a primary replica's workload as $\frac{\eta}{(k+1)\eta+kf-1}$. For the class M, a set of services are aggregated together as if it is from class 1. When $M > 5$, it holds that $\frac{\eta}{2\eta+f-1} - \frac{\eta}{3\eta+2f-1} \geq \frac{\eta}{(M+1)\eta+Mf-1}$. This ensures that we can always aggregate consecutive services in an arbitrary release sequence into an aggregate service with its primary replica's workload falling in the range of class 1. $\mathfrak{A}_x$ is used to pack the aggregate items.

For each class k where $1 \leq k < M$ (the last class is treated as the class 1), the level of each bin is at most $\frac{k\eta+kf}{(k+1)\eta+kf-1}$. Thus, the available space is at least $1 - \frac{k\eta+kf}{(k+1)\eta+kf-1} = \frac{\eta-1}{(k+1)\eta+kf-1}$ which is exactly the upper bound of the reserved space needed for avoiding overflow in case of failures. Together with Theorem 6 and that services from different classes are packed separately, we can conclude that $\mathfrak{A}_{hx}$ is feasible.

We now analyze the packing performance of $\mathfrak{A}_{hx}$. For each class where $1 \leq k < M$, except for a constant number of bins (specifically, the last group of at most $f(\alpha+1)$ bins), each bin hosts k primary replicas and fk standby replicas. Thus, the bin level is at least $\frac{k(\eta+f)}{(k+2)\eta+(k+1)f-1} \geq \frac{\eta+f}{3\eta+2f-1}$. Similarly, for the class M, except for a constant number of bins, each bin hosts 1 aggregate primary replica and f aggregate standby replicas. So, the bin level is at least $\frac{\eta+f}{3\eta+2f-1}$. Therefore, overall, except for a constant number of bins, the level of each bin is at least $\frac{\eta+f}{3\eta+2f-1}$. This implies that the asymptotic competitive ratio of $\mathfrak{A}_{hx}$ is bounded by $\frac{3\eta+2f-1}{\eta+f} = 2 + \frac{\eta-1}{\eta+f}$.

Theorem 10. *The asymptotic competitive ratio of $\mathfrak{A}_{hx}$ is bounded by* $2 + \frac{\eta-1}{\eta+f}$.

The above analysis shows that $\mathfrak{A}_{hs}$ and $\mathfrak{A}_{hx}$ are better than $\mathfrak{A}_{fm}$, as both of their asymptotic competitive ratios can be bounded by a constant. Let us set $\eta = 2$, $f = 2$ and $M = 50$ to have a concrete example. Now the asymptotic competitive ratio of $\mathfrak{A}_{fm}$ is bounded by 3, the asymptotic competitive ratio of $\mathfrak{A}_{hs}$ is bounded by 2 and the asymptotic competitive ratio of $\mathfrak{A}_{hx}$ is bounded by 2.25. Figure 8 gives the numerical result in this setting, where for each service, the workload of the primary replica is randomly selected from $(0, \frac{1}{5}]$. We can see that the number of bins required by $\mathfrak{A}_{hx}$ increases smoothly. Even for packing 20000

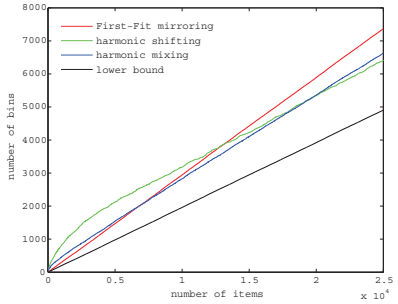


Fig. 8: Numerical example

services, $\mathfrak{A}_{hx}$ is the best among the three heuristic algorithms.

7 Future Work

In the future, we could study a recovery procedure which adds new servers/replicas into the system to maintain the fault-tolerant ability after failures occur. A straightforward recovery process is to add new servers following the configurations of the failed ones: if some primary replicas were located on the failed servers, the new joining servers will again host these primary replicas. However, this method would entail additional role switching of replicas from primary to standby on existing servers which may not be preferable due to the overheads/costs involved. A favorable recovery procedure should not involve any role switching process, *i.e.*, only adding new standby replicas into the system. Based on this idea, it will be interesting to study an extension problem which addresses self-stabilization: after carrying out a recovery procedure, the system can tolerate another up to f new failures while still meeting the exclusion and space constraints.

ACKNOWLEDGMENTS

This work is supported by Singapore MOE Academic Research Fund Tier 1 under Grants 2013-T1-002-123, 2018-T1-002-063, by the National Natural Science Foundation of China under Grant 61902063, and by the Provincial Natural Science Foundation of Jiangsu, China under Grant BK20190342.

References

[1] E. G. Coffman, J. Csirik, G. Galambos, S. Martello, and D. Vigo, "Bin packing approximation algorithms: Survey and classification," *Handbook of Combinatorial Optimization (second edition)*, pp. 455–531, 2013.
[2] H. Yanagisawa, T. Osogami, and R. Raymond, "Dependable virtual machine allocation," in *proceedings of the IEEE International Conference on Computer Communications (INFOCOM)*, 2013, pp. 629–637.
[3] W. Song, Z. Xiao, Q. Chen, and H. Luo, "Adaptive resource provisioning for the cloud using online bin packing," *IEEE Transactions on Computers*, vol. 63, no. 11, pp. 2647–2660, 2014.
[4] C. Li and X. Tang, "Brief announcement: Towards fault-tolerant bin packing for online cloud resource allocation," in *proceedings of the 29th ACM Symposium on Parallelism in Algorithms and Architectures (SPAA)*, 2017, pp. 231–233.
[5] G. Galambos and G. J. Woeginger, "Online bin packing - a restricted survey," *Mathematical Methods of Operations Research*, vol. 42, no. 1, pp. 25–45, 1995.
[6] M. R. Garey and D. S. Johnson, *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman, 1979.
[7] J. Balogh, J. Békési, and G. Galambos, "New lower bounds for certain classes of bin packing algorithms," *Theoretical Computer Science*, vol. 440-441, pp. 1–13, 2012.
[8] J. Balogh, J. Békési, G. Dósa, L. Epstein, and A. Levin, "A new and improved algorithm for online bin packing," in *proceedings of the 26th Annual European Symposium on Algorithms (ESA)*, 2018, pp. 5:1–5:14.
[9] L. Lamport, "The part-time parliament," *ACM Transactions on Computer Systems*, vol. 16, no. 2, pp. 133–169, 1998.
[10] S. Shen, A. Iosup, A. Israel, W. Cirne, D. Raz, and D. H. J. Epema, "An availability-on-demand mechanism for datacenters," in *proceedings of the 15th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (CCGrid)*, 2015, pp. 495–504.
[11] O. Beaumont, L. Eyraud-Dubois, and H. Larchevêque, "An availability-on-demand mechanism for datacenters," in *proceedings of the 27th IEEE International Symposium on Parallel and Distributed Processing (IPDPS)*, 2013, pp. 55–66.
[12] M. R. Korupolu and R. Rajaraman, "Robust and probabilistic failure-aware placement," in *proceedings of the 28th ACM Symposium on Parallelism in Algorithms and Architectures (SPAA)*, 2016, pp. 213–224.
[13] J. Schaffner, T. Januschowski, M. Kercher, T. Kraska, H. Plattner, M. J. Franklin, and D. Jacobs, "Rtp: Robust tenant placement for elastic in-momory database clusters," in *proceedings of the ACM International Conference on Management of Data (SIGMOD)*, 2013, pp. 773–784.
[14] K. Daudjee, S. Kamali, and A. Lopez-Ortiz, "On the online fault-tolerant server consolidation problem," in *proceedings of the ACM Symposium on Parallelism in Algorithms and Architectures (SPAA)*, 2014, pp. 12–21.
[15] J. Mate, K. Daudjee, and S. Kamali, "Robust multi-tenant server consolidation in the cloud for data analytics workloads," in *proceedings of 37th IEEE International Conference on Distributed Computing Systems (ICDCS)*, 2017, pp. 2111–2118.
[16] Y. Li, X. Tang, and W. Cai, "On dynamic bin packing for resource allocation in the cloud," in *proceedings of the 26th ACM Symposium on Parallelism in Algorithms and Architectures (SPAA)*, 2014, pp. 2–11.
[17] R. Ren and X. Tang, "Clairvoyant dynamic bin packing for job scheduling with minimum server usage time," in *proceedings of the 28th ACM Symposium on Parallelism in Algorithms and Architectures (SPAA)*, 2016, pp. 227–237.
[18] M. J. Fischer, N. A. Lynch, and M. Paterson, "Impossibility of distributed consensus with one faulty process," *Journal of the ACM*, vol. 32, no. 2, pp. 374–382, 1985.
[19] D. S. Johnson, "Fast algorithms for bin packing," *Journal of Computer and System Sciences*, vol. 8, no. 3, pp. 272–314, 1974.
[20] M. R. Garey, R. L. Graham, D. S. Johnson, and A. C. Yao, "Resource constrained scheduling as generalized bin packing," *Journal of Combinatorial Theory, Series A*, vol. 21, no. 3, pp. 257–298, 1976.
[21] C. C. Lee and D. T. Lee, "A simple on-line bin-packing algorithm," *Journal of the ACM*, vol. 32, no. 3, pp. 562–572, 1985.

Chuanyou Li received the Ph.D degree in computer science and engineering from Southeast University, Nanjing, China in 2014. Currently he is an assistant professor in School of Computer Science and Engineering, Southeast University and the Key Lab of Computer Network and Information, Ministry of Education. He worked as a postdoc researcher at INRIA, Rennes, France (2015-2016) and also worked as a research fellow at Nanyang Technological University, Singapore (2016-2018). His research interests includes fault tolerance, parallel and distributed computing.

Xueyan Tang received the B.Eng. degree in computer science and engineering from Shanghai Jiao Tong University in 1998, and the Ph.D. degree in computer science from the Hong Kong University of Science and Technology in 2003. He is currently an Associate Professor with the School of Computer Science and Engineering, Nanyang Technological University, Singapore. His research interests include distributed systems, cloud computing, mobile and pervasive computing. He has served as an Associate Editor of IEEE Transactions on Parallel and Distributed Systems, and a Program Co-Chair of IEEE ICPADS 2012 and CloudCom 2014.