

# Allocating Weighted Jobs in Parallel \*

Petra Berenbrink, Friedhelm Meyer auf der Heide, Klaus Schröder

Heinz Nixdorf Institute and

Dept. of Computer Science

University Paderborn

D-33102 Paderborn, Germany

Email: {pebe, fmadh}@uni-paderborn.de, ellern@hni.uni-paderborn.de

## Abstract

It is well known that after placing  $m \geq n$  balls independently and uniformly at random (i.u.r.) into  $n$  bins, the fullest bin contains  $\Theta(\log n / \log \log n + \frac{m}{n})$  balls, with high probability. It is also known (see [Ste96]) that a maximum load of  $O(\frac{m}{n})$  can be obtained for all  $m \geq n$  if a ball is allocated in one (suitably chosen) of two (i.u.r.) bins.

Stemann ([Ste96]) shows that  $r$  communication rounds suffice to guarantee a maximum load of  $\max\{\sqrt[r]{\log n}, O(\frac{m}{n})\}$ , with high probability. Adler et al. have shown in [ACMR95] that Stemmann's protocol is optimal for constant  $r$ .

In this paper we extend the above results in two directions: We generalize the lower bound to arbitrary  $r \leq \log \log n$ . This implies that the result of Stemmann's protocol is optimal for all  $r$ . Our main result is a generalization of Stemmann's upper bound to weighted jobs: Let  $W^A$  ( $W^M$ ) denote the average (maximum) weight of the balls. Further let  $\Delta = W^A/W^M$ . Note that the maximum load is at least  $\Omega(m/n \cdot W^A + W^M)$ . We present a protocol that achieves maximum load of  $\gamma \cdot (\frac{m}{n} \cdot W^A + W^M)$  using  $O\left(\frac{\log \log n}{\log(\gamma \cdot ((m/n) \cdot \Delta + 1))}\right)$  rounds of communication. For uniform weights this matches the results of Stemmann. In particular, for  $\log \log n$  rounds we achieve optimal load of  $O(\frac{m}{n} \cdot W^A + W^M)$ . Using this lower bound it is also shown that our algorithm is optimal in the case of weighted balls for various degrees of uniformity.

\* Supported in part by DFG-Graduiertenkolleg "Parallele Rechner-netzwerke in der Produktionstechnik", ME 872/4-1, by DFG-SFB 376 "Massive Parallelität", by the SICMA Project founded by the European Community within the program on "Advanced Communication Technologies and Services", and by EU ESPRIT Long Term Research Project 20244 (ALCOM-IT)

## 1 Introduction

In the classic "balls into bins" game, we have  $m$  balls,  $n$  bins and each ball is thrown into a bin chosen independently and uniformly at random (i.u.r.). It is well known that there exists a bin receiving  $\Theta(\log n / \log \log n + \frac{m}{n})$  balls, with high probability (w.h.p.), that is, with probability  $1 - n^{-\alpha}$  for any constant  $\alpha > 0$ . The maximum number of balls found in any bin, i.e. the maximum load, can be decreased to  $\Theta(\frac{m}{n})$  by allowing a ball to select one bin among several i.u.r. chosen bins.

The problem of allocating balls to bins finds many applications. Consider, for example, the field of distributed Load Balancing. Here a client workstation issues jobs (balls) that have to be allocated to servers (bins). In this case, Load Balancing enables the effective utilization of a distributed computing system which requires efficient policies for resource allocation.

Another application is the design of Multimedia or Data Servers using disk arrays. One data unit is divided into several blocks and each block is stored on several disks chosen i.u.r. The clients' data requests represent the balls in the game. The balls "choose" bins holding a copy of the requested data. The goal is to distribute the accesses evenly among the disks.

For both applications, the balls into bins game described above only models correctly if the jobs have uniform weight (resources like time or size of requested data) which is typically not given in real applications.

Therefore, we provide an algorithm dealing with weighted balls. The algorithm allocates an unknown number of balls to  $n$  bins yielding an optimal allocation as fast as the fastest known algorithm for balls without weights. No global information on the weights of the balls is required.

Further, we generalize the lower bound of Adler et al. [ACMR95] relating the number of communication rounds to the achievable maximum load. This shows that the algorithm of Stemmann [Ste96] and our algorithm are optimal in the case of jobs without weights, and that our algorithm is also optimal in the case of weighted balls.

## 1.1 Known Results

All protocols for balanced allocations known so far assume uniform weights.

Azar et al. [ABKU94] examine a sequential protocol called *greedy process* to place  $m$  balls into  $n$  bins. For each ball they choose  $d$  bins i.u.r., and put the ball into the bin with minimum load at time of placement. They show that after placing  $n$  balls the maximum load is  $\Theta(\log \log(n)/\log(d) + 1)$ , w.h.p. Furthermore, they provide a result for an infinite version of their sequential process: in the case of a stationary distribution, the fullest bin contains less than  $\log \log(n)/\log(d) + O(1)$  balls, w.h.p., where  $n$  is both the number of balls and bins in the system. The simple sequential game has many applications and is also used as an online algorithm for competitive Load Balancing (see [ABK94], [AKP<sup>+</sup>93], and [PW93]).

Adler et al. [ACMR95] explore the problem in parallel and distributed settings for the case of placing  $n$  balls into  $n$  bins. They provide a lower bound for *non-adaptive* (possible destinations are chosen before any communication takes place) and *symmetric* algorithms (all balls and bins perform the same underlying algorithm, and all destinations are chosen i.u.r.). For any constant number  $r \in \mathbb{N}$  of communication rounds, the maximum load is shown to be at least  $\Omega\left(\sqrt[r]{\frac{\log n}{\log \log n}}\right)$  (see [Mit96b]). Additionally, they present parallelizations of the sequential strategy found in [ABKU94]. They give a two-round parallelization of the greedy process, matching the lower bound. Furthermore, they introduce a multiple-round strategy requiring  $\log \log n + O(1)$  rounds of communication and achieving maximum load  $\log \log n + O(1)$ , w.h.p. Finally, they examine a strategy using a threshold  $T$ : In each of  $r$  communication rounds each non-allocated ball tries to access two bins chosen i.u.r. and each bin accepts up to  $T$  balls during each round. They show that with  $T = \sqrt[r]{\frac{(2r+o(1)) \cdot \log n}{\log \log n}}$  this algorithm terminates after  $r$  rounds with maximum load  $r \cdot T$ , w.h.p.

Stemann [Ste96] extends the results for the case where the number of balls  $m$  is larger than the number  $n$  of bins. For  $m = n$ , he analyzes a very simple class of algorithms achieving maximum load  $O\left(\sqrt[r]{\frac{\log n}{\log \log n}}\right)$  if  $r$  rounds of communication are allowed. For constant  $r$ , this matches the lower bound presented in [ACMR95]. He generalizes the algorithm for  $m > n$  balls and achieves optimal load  $O\left(\frac{m}{n}\right)$  using  $\frac{\log \log n}{\log(m/n)}$  rounds of communication, w.h.p., or load  $\max\left\{\sqrt[r]{\log n}, O\left(\frac{m}{n}\right)\right\}$  using  $r$  rounds of communication, w.h.p.

Recently, Mitzenmacher [Mit96a] analyzed a dynamic, continuous allocation strategy: Customers (balls) arrive as a Poisson stream of rate  $\lambda n$ ,  $\lambda < 1$ , at a collection of  $n$  servers (bins). Each customer chooses  $d$  servers i.u.r. and joins the server with the fewest customers. Customers are served according to the first-come-first-serve protocol, and the service time is exponentially distributed with mean one. He calls his model the supermarket model. For a time interval of fixed and constant length  $T$ , he showed the expected waiting time to be  $O(1)$  for  $N \rightarrow \infty$ , and the maximum queue length is  $O(\log \log n + o(1))$ , w.h.p. His analysis makes use of deep

results of Kurtz ([Kur81]) on so called density dependent Markov Chains.

## 1.2 New Results

We present an algorithm allocating  $m$  weighted balls into  $n$  bins. The weights may be arbitrary positive real values. No global information on the number and the weights of the balls is given to the algorithm. The goal is to allocate the balls to the bins in such a way that the load of the bins is almost equal.

The weight  $W_i$  of a ball  $B_i$  is a positive real number. The *load* of a bin  $U_j$  is the sum of the weights  $W_i$  of all balls  $B_i$  allocated to  $U_j$ .

We define the *average weight* as  $W^A = \frac{1}{m} \sum_{i=1}^m W_i$  and the *maximum weight*  $W^M$  as the maximum of the weights, i.e.,  $W^M = \max(W_i)$ . Further let  $\Delta = \frac{W^A}{W^M}$  be the *degree of uniformity* of the weights. Note that the maximum load is at least  $\Omega\left(\frac{m}{n} \cdot W^A + W^M\right)$

**Theorem 1.1** For  $m, n \in \mathbb{N}$ ,  $m \geq n$  it holds for every  $\gamma \geq 2e^2$ : Using

$$O\left(\frac{\log \log n}{\log\left(\gamma \cdot \left(\frac{m}{n} \cdot \Delta + 1\right)\right)}\right)$$

rounds of communication, a maximum load of  $\gamma \cdot \left(\frac{m}{n} \cdot W^A + W^M\right)$  can be achieved, w.h.p. No global information on the number and the weights of the balls has to be known in advance.

If  $O\left(\frac{\log \log n}{\log\left(\frac{m}{n} \cdot \Delta + 1\right)}\right)$  communication rounds are used, the load of any bin is at most  $O\left(\frac{m}{n} \cdot W^A + W^M\right)$ , w.h.p., which is optimal. In the case of balls with unit weights, our algorithm needs at most  $O\left(\frac{\log \log n}{\log\left(\gamma \cdot \left(\frac{m}{n} + 1\right)\right)}\right)$  communication rounds, to achieve a maximum load of  $O\left(\gamma \cdot \frac{m}{n}\right)$ , w.h.p. This matches the result of Stemann [Ste96].

Furthermore we give a lower bound extending the lower bound given by Adler et al. in [ACMR95] to non-constant numbers of communication rounds and  $m \geq n$ . As in Adler et al. ([ACMR95]), our lower bound model assumes that each ball chooses two bins i.u.r. and has to be placed in one of them. Our lower bound model is described in Section 3, we call it *access graph based*.

**Theorem 1.2** For  $m, n, r \in \mathbb{N}$ ,  $r \leq \log \log n$  and  $m \geq n$  it holds with probability  $\frac{1}{4}$ :

- Any access graph based algorithm allocating  $m$  balls with unit weights into  $n$  bins requires at least  $\Omega\left(\frac{\log \log n}{\log \gamma}\right)$  rounds of communication to achieve a maximum load  $\leq \gamma'$ .
- After  $r$  rounds of communication the maximum load is at least  $\frac{1}{2} \cdot \sqrt[r]{\frac{\log n}{18 \log \log n}}$ .

This bound shows that allocating  $m$  equally weighted balls into  $n$  bins obtaining optimal load needs  $\Theta\left(\frac{\log \log n}{\log \frac{m}{n}}\right)$  rounds of communication. As a consequence of this theorem, there are no better access graph based algorithms for allocating balls with uniform weights than the one of Stemmann and ours.

Our upper bound Theorem 1.1, that approximates the optimal maximum load up to a factor  $\gamma$ , needs the same number of rounds for allocating:

- $m$  balls with unit weight
- $\Delta m$  balls with degree of uniformity  $\Delta$

This effect becomes obvious for the following set of weighted balls with uniformity  $\Delta$ :  $\Delta m$  balls get weight 1 and  $(1-\Delta)m$  balls get weight 0. Thus, under the assumptions that adding balls with weights does not change the complexity of approximating the optimal maximum load, we may conclude from our Lower Bound (Theorem 1.2): For  $m \geq n$  and  $1 \geq \Delta \geq \frac{n}{m}$  we have weighted balls with degree of uniformity  $\Delta$  in a way that  $\Omega\left(\frac{\log \log n}{\log(\gamma \frac{m}{n} \Delta)}\right)$  rounds of communication are necessary to achieve maximum load at most  $\gamma \cdot \left(\frac{m}{n} W^A + W^M\right)$ . Thus, also for weighted balls our protocol is optimal.

## 2 Adaptive Load Balancing

In order to prove Theorem 1.1 we describe an algorithm, the so called *adaptive collision protocol*. As a motivation for the adaptive collision protocol consider the *c-collision protocol* used by Stemmann in [Ste96] allocating  $m$  balls with unit weights to  $n$  bins. Let  $c = \gamma \cdot \frac{m}{n}$  for some  $\gamma > 1$ . In the *c-collision* protocol every ball chooses two bins i.u.r. In each round a ball sends a request to its chosen bins, and a bin  $U$  accepts the request of a ball if and only if  $U$  gets at most  $c$  requests. A ball which is not accepted keeps sending requests until it is accepted. Obviously,  $c$  has to be chosen properly in order to make the algorithm work, and the proper choice of  $c$  requires knowledge about the number  $m$  of the balls. To avoid this problem, assume that the algorithm is executed for all possible values of  $c$  simultaneously, i.e. in each round every ball sends a request including the values of  $c$  for which it is already accepted and each bin answers a request with the values of  $c$  for which it accepts its requests. In order to include weighted balls we compute the sum of the weights of the balls sending a request to a bin instead of just counting the number of requests.

To avoid too much communication our algorithm operates slightly different. Each ball  $B_i$  keeps the smallest value of  $c$  for which it has been accepted in the last round. We call this value  $bc_i$ . At the beginning of each round each ball  $B_i$  sends a request including  $bc_i$  and its own weight  $W_i$  to each bin it has chosen. Now every bin  $U_j$  computes the value  $uc_j$  as the minimum possible value such that every ball sending a request to  $U_j$  is accepted at  $U_j$  or in another bin. For doing this,  $U_j$  assumes that every other bin chooses the same value  $uc_j$  (while other bins may choose and assume different values). Afterwards  $U_j$  answers each request with

| $W_i$ | $bc_i$ | rank | reply    |
|-------|--------|------|----------|
| 2     | 10     | 1    | 6        |
| 2     | 9      | 2    | 6        |
| 1     | 7      | 3    | 6        |
| 2     | 6      | 4    | $\infty$ |
| 1     | 2      | 5    | $\infty$ |

Figure 1: The bin chooses  $uc_j = 6$ . The last column contains the values replied to the balls.

$uc_j$  if  $U_j$  assumes the ball to be accepted at  $U_j$ , and with  $\infty$  otherwise. Figure 1 contains an example.

The Adaptive Collision Protocol is given in Figure 2.

For every bin  $U_j$  the sum of the weights of all balls placed into  $U_j$  is called the *final load* of  $U_j$ .

To simplify our discussion we divide the weights  $W_i$  of all balls by  $W^M$ , so the maximum weight of a ball is 1 and the average weight is  $\Delta$ , the degree of uniformity.

Further we concentrate on the case where  $m \cdot \Delta = n$ , the general case may be treated analogously.

Our proof of Theorem 1.1 is done by induction: First we show some properties of the initial distribution of the requests. After that, using some knowledge on the distribution of requests in round  $t$ , we derive a similar result on the distribution in round  $t + 1$ .

Before starting with our induction, we relate the behavior of the Adaptive Load Collision Protocol to a much simpler one, thus simplifying our analysis.

The first property we state is that every ball  $B_i$  allocates itself to a bin  $U_j$  which assumes  $B_i$  to be allocated at  $U_j$ , i.e., each ball gets at least one reply  $\neq \infty$ .

**Lemma 2.1** *For each ball  $B_i$ , let  $bc_i(r)$  and  $uc_j(r)$  be the values of  $bc_i$  and  $uc_j$  in round  $r$  respectively. Then for every ball  $B_i$  it holds:  $bc_i(r) \leq bc_i(r - 1)$  and hence  $B_i$  gets at least one reply  $\neq \infty$ .*

**Proof:** The proof is done by induction on  $r$ . [ $\forall i : (bc_i(1) \geq bc_i(2))$ ]: In the step marked with (\*) all requests of each ball are answered by  $U_j$  with a value  $\neq \infty$ , so  $bc_i(1) \neq \infty$ . Let  $U_j$  be the bin returning the smallest value to  $B_i$ , (W.l.o.g we assume that  $U_j$  answers the red request of  $B_i$ ) thus  $bc_i(1)$  equals the total load of the red requests sent to  $U_j$ . Hence, while computing  $\bar{l}$  in  $U_j$  for the red requests, the sum of the  $\widehat{W}_i$  never exceeds  $bc_i(1)$ .  $B_i$  gets an reply containing  $uc_j(1)$  which equals either the sum of the weights of a part of the red requests sent to  $U_j$  or some value  $bc_{i'}(1)$  no larger than  $bc_i(1)$ . Hence  $bc_i(2) \leq bc_i(1)$ .

[ $\forall i : (bc_i(r - 1) \geq bc_i(r) \Rightarrow \forall i : bc_i(r) \geq bc_i(r + 1))$ ]: Let  $U_j$  be the bin returning the smallest value to  $B_i$  in round  $r$  (Again we assume  $U - j$  to answer the red request w.l.o.g). Further let  $\kappa(r)$  be the rank of  $B_i$ 's red request after sorting in round  $r$ . Every red request with rank larger than  $\kappa(r - 1)$  after sorting in round  $r - 1$  also has rank larger than  $\kappa(r)$  in round  $r$ . This holds, because our sorting algorithm is stable

## Adaptive Collision Protocol:

**For all balls  $B_1, \dots, B_m$  do in parallel**  
 -Choose i.u.r. three bins  $V_1, V_2$  and  $V_3$ .  
 -Send a request containing the weight  $W_i$  to each chosen bin. (The requests send to  $V_1, V_2$  and  $V_3$  are called *red, green and yellow request* respectively.)  
**For all bins  $U_1, \dots, U_n$  do in parallel**  
**For color  $\in \{ \text{red, green, yellow} \}$  do**  
 -Compute the sum  $S_j$  of the weights  $W_i$  of the *color* requests send to  $U_j$   
 -Send a reply containing  $S_j$  for each *colorequest* (\*)  
**For  $r$  rounds do**  
**For all balls  $B_1, \dots, B_m$  do in parallel**  
 -Compute  $bc_i$  as the minimum of the replied values  
 -Allocate yourself to the bin with minimum returned value (ties are broken arbitrarily).  
 -Send a request including  $bc_i$  and  $W_i$  to all chosen bins  
**For all bins  $U_1, \dots, U_n$  do in parallel**  
**For color  $\in \{ \text{red, green, yellow} \}$  do**  
 -Sort the received *color* requests according to their values  $bc_i$  in decreasing order, using a stable sorting algorithm starting with the sequence from the last round (if existing). Let  $\widehat{bc}_i$  be the value of the  $i$ -th largest value of the  $bc$ 's and let  $\widehat{W}_i$  be the corresponding weight.  

$$-\bar{l} := \max \left\{ l \mid \sum_{i=1}^l \widehat{W}_i \leq \widehat{bc}_{l+1} \right\}$$
  

$$-uc_j := \max \left\{ \sum_{i=1}^{\bar{l}} \widehat{W}_i, \widehat{bc}_{\bar{l}+1} \right\}$$
  
 -For the first  $\bar{l}$  *color* requests send a reply containing  $uc_j$ , and a reply containing  $\infty$  for the other *color* requests.  
**For all balls  $B_1, \dots, B_m$  do in parallel**  
 -If  $B_i$  allocated himself to a bin  $U_j$  place  $B_i$  into  $U_j$ .

Figure 2: The Adaptive Collision Protocol

and requests which get a reply containing  $uc_j(r-1)$  in round  $r-1$  contain a  $bc(r)$  not greater than  $bc_i(r)$  in round  $r$ . Each request answered with  $\infty$  in round  $r-1$  has a  $bc(r)$  not larger than  $bc_i(r)$ , due to the choice of  $uc_j(r-1)$ , the induction hypothesis and the stability of the sorting algorithm.

Let  $\mathcal{W}_i(r) = \sum_{\tau=1}^t \widehat{bc}_\tau(r)$ . Then we have for all  $\sigma \leq \kappa(r)$ :

$$\mathcal{W}_\sigma(r) \leq \mathcal{W}_{\kappa(r)}(r) \stackrel{(1)}{\leq} \mathcal{W}_{\kappa(r-1)}(r-1) \stackrel{(2)}{\leq} uc_j(r-1)$$

and

$$uc_j(r-1) = bc_i(r) \leq \widehat{bc}_\sigma(r)$$

This implies that  $\mathcal{W}_\sigma(r) \stackrel{(3)}{\leq} \widehat{bc}_\sigma(r)$  holds. (1) holds due to the fact that every red request with rank larger than  $\kappa(r-1)$  after sorting in round  $r-1$  has also rank larger than  $\kappa(r)$  in round  $r$ . (2) is due to the algorithms choice of  $uc_j$ .

(3) assures that  $\bar{l} \geq \kappa(r)$ . Hence, in round  $r$ ,  $B_i$  gets a reply containing a value  $\neq \infty$ . According to the choice of  $\bar{l}$ , we have  $\widehat{bc}_{\bar{l}} \geq \widehat{W}_{\bar{l}}$ . Further it holds  $bc_i(r) \geq \widehat{bc}_{\bar{l}}(r) \geq \widehat{bc}_{\bar{l}+1}(r)$ , hence  $bc_i(r) \geq \max\{\widehat{W}_{\bar{l}}, \widehat{bc}_{\bar{l}+1}\} = uc_j(r) \geq bc_i(r+1)$ . ■

Consider the *c-Load Collision Protocol* described in Figure 3. This algorithm will be used in order to analyze our Adaptive

Collision Protocol. First we show that the adaptive protocol and the *c-Load Collision Protocol* are closely related, i.e., if all balls are deleted in the latter one after  $t$  rounds, the Adaptive Collision Protocol archives a final load of at most  $3 \cdot c$ .

**Lemma 2.2** *Assume the c-Load Collision Protocol and the Adaptive Collision Protocol use the same random bins  $V_1, V_2, V_3$  for every ball. If all balls are deleted after  $r$  rounds of the Load Collision Protocol, the maximum final load of any bin in the Adaptive Collision Protocol is at most  $3 \cdot c$ .*

**Proof:** The proof is done by induction on the number of rounds.

[ $t = 1$ ]: No ball has been deleted in the *c-Load Collision Protocol*.

[ $t-1 \rightarrow t$ ]: Consider a ball  $B_i$  which is deleted before round  $t$  in the Load Collision Protocol. If the ball was deleted in round  $t' < t-1$  then  $bc_i(t') \leq c$  according to the induction hypothesis and  $bc_i(t) \leq bc_i(t') \leq c$  according to Lemma 2.1. So assume that  $B_i$  is deleted in round  $t-1$ , w.l.o.g. we assume that  $B_i$  gets an accept as reply to his red request from bin  $U_j$ . Then  $U_j$  accepted all its red requests in round  $t-1$ . Now consider the red requests send to  $U_j$  in the Adaptive Collision Protocol in round  $t-1$ . The sum of

**$c$ -Load Collision Protocol:**

**For all balls  $B_1, \dots, B_m$  do in parallel**  
 -Choose i.u.r. three bins  $V_1, V_2$  and  $V_3$ .

**For  $r$  rounds do**  
**For all balls  $B_1, \dots, B_m$  do in parallel**  
 -Send a request including  $bc_i$  and  $W_i$  to all chosen bins (The requests send to  $V_1, V_2$  and  $V_3$  are called *red, green and yellow requests* respectively)

**For all bins  $U_1, \dots, U_n$  do in parallel**  
**For color  $\in \{\text{red, green, yellow}\}$  do**  
 -Compute the sum of the weights of all *color* requests send to  $U_j$   
 -If the sum is at most  $c$ , reply the requests with *accept* (the bin is said to *accept* its requests), else reply with *non accept*

**For all balls  $B_1, \dots, B_m$  do in parallel**  
 -If  $B_i$  gets at least one reply containing *accept*,  $B_i$  is deleted (deleted balls do not participate in the protocol any more)

Figure 3: The  $c$ -Load Collision Protocol

the weights of all red requests  $R_{i'}$  with  $bc_{i'}(t-1) > c$  is at most  $c$ , so  $U_j$  answers all these requests with a value  $uc_j(t-1) \leq c$ .

If all balls are deleted at the end of the  $c$ -Load Collision Protocol,  $bc_i(r) \leq c$  holds for all balls  $B_i$  in the Adaptive Collision Protocol, so every ball  $B_i$  has allocated itself to a bin  $U_j$  sending a reply with  $uc_j \leq c$ . For each color the sum of the weights of requests getting a reply with  $uc_j \leq c$  is at most  $c$ , so for each color the sum of the weights of the balls allocating themselves to  $U_j$  is at most  $c$ . ■

For our analysis it is convenient to have the possibility of slowing down our  $c$ -Load Collision Protocol a little bit: In the first part of our proof we will consider the red and green requests only, thus we assume that no ball deletes itself due to the presence of a yellow one accepted. The yellow requests will be handled at the end of the proof.

**Lemma 2.3** *Consider a faulty version of the  $c$ -Load Collision Protocol, where some balls do not delete themselves even if they get some *accept* reply. Then it holds: At the beginning of every round  $t$  the set  $\mathcal{L}(t)$  of deleted balls in the  $c$ -Load Collision Protocol is a superset of the set  $\mathcal{F}(t)$  of balls deleted in the faulty version.*

**Proof:** The proof is done by induction on the number  $t$  of rounds.

$[t = 1]$ : At the beginning of round 1 it holds  $\mathcal{F}(1) = \emptyset \subseteq \mathcal{L}(1)$ .

$[t - 1 \rightarrow t]$ : We have  $\mathcal{F}(t-1) \subseteq \mathcal{L}(t-1)$  at the beginning of round  $t-1$ , hence every red request sent to a bin  $U_j$  in the  $c$ -Load Collision Protocol is also sent to  $U_j$  in the faulty version, so the sum of the weights of the red requests in the faulty version is not smaller than the same sum in the  $c$ -Load Collision Protocol. Hence, whenever a red request  $R$  is replied with an *accept* in the faulty version,  $R$  is accepted

in the  $c$ -Load Collision Protocol, too. The same holds for green and yellow requests. ■

Lemma 2.3 allows us to concentrate on the red and green requests first, we assume that no ball deletes himself due to an *accept* reply to a yellow request.

For the analysis we may also assume that some bins fail during the  $c$ -Load Collision Protocol, i.e. they do not accept their red or green requests even if the protocol allows them to do so (in this case the bins are not said to *accept* their requests).

Now we are ready to start our induction. During the induction, we make frequent use of the following Hoeffding Bound (see e.g. [DM90]).

**Lemma 2.4** *Let  $Y_1, \dots, Y_n$  be independent random variables with values in the interval  $[0, z]$ . Let  $\mu := E \left[ \frac{1}{n} \sum_{j=1}^n Y_j \right]$  be the mean expected value of the  $Y_j$ . Then it holds for all  $u \geq 1$*

$$\text{Prob} \left[ \sum_{j=1}^n Y_j \geq \mu \cdot nu \right] \leq \left( \frac{e^{u-1}}{u^u} \right)^{\frac{n\mu}{z}}$$

For simplicity we state our next Lemmata for red requests only, they also hold for green requests unless stated otherwise.

For  $t \in \mathbb{N}$  let  $r_j(t)$  ( $g_j(t)$ ) denote the sum of the weights  $W_i$  of all red (green) requests  $R_i$  of balls  $B_i$  send to  $U_j$  in round  $t$ . A bin which has not accepted a red (green) request in a round  $t' < t$  is said to be *red- (green-) active* in round  $t$ .

To derive a strong result on the sizes of the bins, we make use of the following function:

$$f(x) = \begin{cases} e^x & \text{if } x > \frac{c}{2} \\ 0 & \text{otherwise} \end{cases}$$

**Lemma 2.5**

$$E \left[ \sum_{j=1}^n f(r_j(1)) \right] \leq \frac{1}{2} \cdot n$$

**Proof:** Fix a bin  $U_j$ . We have

$$E \left[ \sum_{j=1}^n f(r_j(1)) \right] = n \cdot E[f(r_j(1))]$$

$$\begin{aligned} E[f(r_j(1))] &= \int_{\frac{c}{2}}^{\infty} \text{Prob}[r_j(1) = u] \cdot e^u du \\ &\leq \sum_{u=e^{\frac{c}{2}}}^{\infty} \text{Prob}[r_j(1) \in (\log(u-1), \log(u)] \cdot u \\ &\leq \underbrace{\sum_{u=e^{\frac{c}{2}}}^{\infty} \text{Prob}[r_j(1) \geq \log(u-1)]}_{(1)} \\ &\quad + \underbrace{\text{Prob}[r_j(1) \geq \log(e^{\frac{c}{2}})] \cdot e^{\frac{c}{2}}}_{(2)} \end{aligned}$$

According to the Hoeffding Bound we obtain:

$$\text{Prob}[r_j(1) \geq \log(u-1)] \leq \left( \frac{e \cdot \frac{m}{n} \cdot \Delta}{\log(u-1)} \right)^{\log(u-1)}$$

Hence

$$\begin{aligned} (1) &\leq \sum_{u=e^{\frac{c}{2}}}^{\infty} \left( \frac{e \cdot \frac{m}{n} \cdot \Delta}{\log(u-1)} \right)^{\log(u-1)} \\ &\text{for } c \geq 4 \cdot e \cdot \frac{m}{n} \cdot \Delta \\ &\leq \sum_{u=e^{c/2}}^{\infty} \left( \frac{1}{2} \right)^{\log(u-1)} \\ &\leq \frac{1}{4} \end{aligned}$$

and

$$\begin{aligned} (2) &\leq \left( \frac{e \cdot \frac{m}{n} \cdot \Delta}{c/2} \right)^{c/2} \cdot e^{c/2} \\ &\text{for } c \geq 4 \cdot e^2 \cdot \frac{m}{n} \cdot \Delta \\ &\leq \frac{1}{4} \end{aligned}$$

The next lemma shows that  $\sum_{j=1}^n f(r_j(1))$  will be at most twice the expected value, w.h.p.

**Lemma 2.6**

$$\sum_{j=1}^n f(r_j(1)) \leq n \quad \text{w.h.p.}$$

and hence, for all  $u \geq c$ , there are at most  $\frac{n}{e^u}$  bins  $U_j$  with  $r_j(1) \geq c$ , w.h.p.

**Proof:** It suffices to prove the first statement. We consider groups of balls, where each group contains balls with a total weight between  $\frac{1}{2}$  and 1, and every ball is contained in exactly one group. The number of groups is at most  $2 \cdot n$ .

Let  $Y_i$  be the random variable denoting the random choices for  $V_1$  of the balls in the  $i$ -th group (the balls in a group are *not* assumed to choose the same bin). Let

$$f'(x) = \min\{f(x), \frac{1}{8} \log n\}$$

and

$$X = \sum_{j=1}^n f'(r_j(1))$$

be a random variable — which depends on the  $Y_i$ .

Further let

$$X_i = E[X | Y_1, \dots, Y_i].$$

Then the  $X_i$  form a Martingale.

It holds

$$|X_i - X_{i+1}| \leq e^{\frac{1}{8} \log n} \leq n^{\frac{1}{8}}.$$

Azumas Inequality — as for example found in [RM95] — yields

$$\text{Prob} \left[ |X - E[X]| \geq \frac{1}{2} \cdot n \right] \leq e^{-\left( \frac{(\frac{1}{2}n)^2}{2 \cdot n \cdot n^{\frac{1}{8}}} \right)}$$

Lemma 2.5 we have  $E[X] \leq \frac{1}{2} \cdot n$ , so we are done if  $f'(r_j(1)) = f(r_j(1))$  (i.e. no bin has load higher than  $\frac{1}{8} \cdot \log n$ ) for every  $j \in \{1, \dots, n\}$ . Using the Hoeffding Bound again the latter comes up easily. ■

The next lemma shows that small (in terms of weight) subsets of balls do not form a big heap of load in any bin.

**Lemma 2.7** Consider a set  $\mathcal{R}$  of red requests  $R_1, \dots, R_k$  with total load  $\sum_{R_i \in \mathcal{R}} W_i \leq n^{\frac{3}{4}}$ . Then, for every bin  $U_j$ , the sum  $s_j$  of the weights of all requests of  $\mathcal{R}$  in  $U_j$  is at most  $\kappa \geq e$ , with probability  $1 - n^{-\frac{\kappa}{4}}$ .

**Proof:** The expected value of  $s_j$  is at most  $n^{-\frac{1}{4}}$ , so according to the Hoeffding Bound

$$\text{Prob}[s_j \geq \kappa] \leq \left( \frac{e \cdot n^{-\frac{1}{4}}}{\kappa} \right)^{\kappa}$$

■

**Lemma 2.8** Let  $B_1, \dots, B_k$  be some balls sending a red request to a fixed bin  $U_{j'}$  in round  $t$ . Let  $\mathcal{R}$  be the set of green requests send by  $B_1, \dots, B_k$ .

For  $t \leq r$  let  $U_j$  be a green-active bin in round  $t$  and  $p(t) = \text{Prob}[g_j(t-1) \geq c - \kappa]$  using  $\kappa$  from Lemma 2.7.

The probability that all green requests of  $B_1, \dots, B_k$  go to a bin  $U_{j''}$  with  $g_{j''}(t) \geq c$  is at most  $(p(t))^k$ , w.h.p.

**Proof:** Consider the set of green active bins in round  $t$ , the probability for a bin  $U_j$  to have  $g_j(t) \geq c$  under the assumption that some elements of  $\mathcal{R}$  go to  $U_j$  is at most  $p(t)$  if  $s_j$  (defined as in Lemma 2.7) is at most  $\kappa$ .

All red requests of the balls  $B_1, \dots, B_k$  go to  $U_{j'}$ , so  $\sum_{R_i \in \mathcal{R}} W_i \leq r_i(1) \leq \log n$ , w.h.p. Hence according to Lemma 2.7  $s_j \leq \kappa$ , w.h.p. ■

The next lemma shows that the sum of the random variables estimating the load in the bins decreases during each round.

**Lemma 2.9** For  $t \leq r$  assume that  $\sum_{j=1}^n f(r_j(t)) \leq q(t) \cdot n$  for some  $q(t) \leq 1$ . Then

$$E \left[ \sum_{j=1}^n f(r_j(t+1)) \right] \leq \frac{1}{2} \cdot q(t) \cdot (p(t))^{c/2} \cdot n$$

with  $p(t)$  taken from Lemma 2.8.

**Proof:** Fix a bin  $U_j$ . As in Lemma 2.5 we have

$$E \left[ \sum_{j=1}^n f(r_j(t+1)) \right] = n \cdot E[f(r_j(t+1))]$$

$$\begin{aligned} E[f(r_j(t+1))] &\leq \underbrace{\sum_{u=e^{\frac{c}{2}}}^{\infty} \text{Prob}[r_j(t+1) \geq \log(u-1)]}_{(1)} \\ &\quad + \underbrace{\text{Prob}[r_j(t+1) \geq \log(e^{\frac{c}{2}})] \cdot e^{\frac{c}{2}}}_{(2)} \end{aligned}$$

It holds

$$(1) \leq \sum_{v=e^{\frac{c}{2}}}^{\infty} \sum_{u=e^{\frac{c}{2}}}^v \text{Prob}[g_j(t) = v] \cdot \text{Prob}[g_j(t+1) \geq \log(u-1) | g_j(t) = v]$$

According to Lemma 2.8 we may use the Hoeffding Bound to determine the latter probability:

$$\begin{aligned} &\leq \sum_{v=e^{\frac{c}{2}}}^{\infty} \sum_{u=e^{\frac{c}{2}}}^v q(t) \cdot e^{-v} \\ &\quad \cdot \left( \frac{e \cdot v \cdot p(t)}{\log(u-1)} \right)^{\log(u-1)} \end{aligned}$$

$$\begin{aligned} &\leq \sum_{v=e^{\frac{c}{2}}}^{\infty} q(t) \cdot e^{-v} \cdot v^{\log v} \cdot (p(t))^{\frac{c}{2}} \\ &\leq \frac{1}{4} \cdot q(t) \cdot (p(t))^{\frac{c}{2}} \end{aligned}$$

and

$$\begin{aligned} (2) &\leq \sum_{v=e^{\frac{c}{2}}}^{\infty} \text{Prob}[g_j(t) = v] \\ &\quad \cdot \text{Prob}[g_j(t+1) \geq \log(e^{\frac{c}{2}}) | g_j(t) = v] \cdot e^{\frac{c}{2}} \\ &\leq \sum_{v=e^{\frac{c}{2}}}^{\infty} q(t) \cdot e^{-v} \\ &\quad \cdot \left( \frac{e \cdot v \cdot p(t)}{\frac{c}{2}} \right)^{\frac{c}{2}} \cdot e^{\frac{c}{2}} \\ &\leq q(t) \cdot (p(t))^{\frac{c}{2}} \cdot \sum_{v=e^c}^{\infty} e^{-v} \cdot v^{\frac{c}{2}} \\ &\leq \frac{1}{4} \cdot q(t) \cdot (p(t))^{\frac{c}{2}} \end{aligned}$$

■

**Lemma 2.10** Using the notations of Lemma 2.9 it holds:

$$\sum_{j=1}^n f(r_j(t+1)) \leq q(t) \cdot (p(t))^{c/2} \cdot n \quad \text{w.h.p.}$$

if  $q(t) \cdot (p(t))^{c/2} \cdot n \geq n^{\frac{5}{8}}$

**Proof:** As in Lemma 2.6 we consider groups of balls such that each group contains balls with a total weight between  $\frac{1}{2}$  and 1. Let  $Y_i$  be the random variable denoting whether the balls in the  $i$ -th group delete themselves in round  $t+1$  or not, depending on the load  $r_{j'}(t)$  of the bin  $U_{j'}$  where the red request of a ball goes to.

Further let

$$f^l(x) = \min\{f(x), \frac{1}{8} \log n\}$$

and let

$$X = \sum_{j=1}^n g_j(t+1)$$

be a random variable depending on the  $Y_i$  and let

$$X_i = E[X | Y_1, \dots, Y_i]$$

Then the  $X_i$  form a Martingale. According to Azuma Martingale Inequality we have

$$\begin{aligned} &\text{Prob} \left[ |X - E[X]| \geq \frac{1}{2} \cdot q(t) \cdot (p(t))^{c/2} \cdot n \right] \\ &\leq e^{-\left( \frac{(\frac{1}{2} \cdot q(t) \cdot (p(t))^{c/2} \cdot n)^2}{2 \cdot n \cdot n^{\frac{1}{8}}} \right)} \end{aligned}$$

■

Using the last lemmas we are now able to analyse the performance of the faulty version of the  $c$ -Load Collision Protocol. We assume that  $\frac{c}{2} \geq \kappa$ .

In the first round all  $n$  bins are red-active, and according to Lemma 2.6 we have  $\sum_{i=1}^n f(r_i(1)) \leq n$ . The probability  $p(1)$  of Lemma 2.8 is at most  $e^{-\frac{c}{2}} \leq e^{-\frac{c}{2}-1}$ .

In round 2 the number of red-active bins is at most  $e^{-c} \cdot n$ , according to Lemma 2.3 we defer some accepts in round 1 such that the number of active bins in round 2 is exactly  $e^{-c} \cdot n$ . We will defer some accepts in every round in order to keep our estimate for the number  $act(t)$  of active bins in round  $t$  exact all the time.

Assume that  $p(t) \leq e^{(\frac{c}{2}-1)^t}$ . According to Lemma 2.10 we have

$$\sum_{i=1}^n f(r_i(t)) \leq e^{-\left(\left(\frac{c}{2}-1\right)^{t-1}\right)}$$

Further

$$act(t) = e^{-c} \cdot \sum_{i=1}^n f(r_i(t-1))$$

Hence

$$\begin{aligned} p(t+1) &= \sum_{i=1}^n f(r_i(t)) \cdot e^{-\frac{c}{2}} \cdot act(t)^{-1} \\ &\leq e^{-\left(\left(\frac{c}{2}-1\right)^t\right) \cdot \frac{c}{2}} \cdot e^{-\frac{c}{2}} \cdot e^c \\ &\leq e^{-\left(\left(\frac{c}{2}-1\right)^{t+1}\right)} \end{aligned}$$

After

$$r = O\left(\frac{\log \log n}{\log\left(\frac{c}{2}-1\right)}\right)$$

rounds we would have

$$\sum_{i=1}^n f(r_i(r)) \leq \frac{1}{n} \cdot \sum_{i=1}^n f(r_i(r-1))$$

if Lemma 2.10 would hold for any  $q(t)$  and  $p(t)$ . So assume our analysis fails when

$$\sum_{i=1}^n f(r_i(r)) = n^{\frac{5}{8}}$$

Then according to Lemma 2.7 we can use the yellow copy to allocate the remaining balls with maximum load  $\kappa$ .

### 3 Lower Bound for Access Graph Based Protocols

In this section we prove Theorem 1.2. To do so, we define the so called *access graph*  $H = (V, E)$ . An access graph is a graph containing  $n$  nodes representing the bins. Two vertices  $u, v$  are connected by an undirected edge if there is a ball accessing the bins represented by  $u$  and  $v$ . Since multi-edges are allowed we get a random graph with  $n$  vertices and  $m$  edges.

For a fixed  $H$ , the goal of a Load Balancing algorithm is to direct the edges of  $H$  in such a way that the maximum in-degree over all vertices of the graph is minimized. Directing an edge corresponds to allocating a ball to one of its bins.

The  $r$ -neighborhood of an edge  $e$  is the subgraph of  $H$  containing the nodes and the edges within reach of the vertices of  $e$  by paths of length at most  $r-1$ . An edge  $e$  has a *symmetric*  $(r, \bar{\gamma})$ -neighborhood if both tails of the edge are roots of complete  $\bar{\gamma}$ -ary trees of depth  $r-1$ .

Consider an algorithm where two nodes  $u, v \in V$  are only allowed to communicate with each other if there is an edge  $(u, v) \in H$ . Intuitively, for each round of communication, an edge discovers a little bit more information about the graph. In  $r$  rounds the bins are able to explore their  $r$ -neighborhood. The following algorithm describes the class of access graph based algorithms fupper which our lower bound holds. The first three conditions are also required for the lower bound of [ACMR95].

**Definition 3.1** *A protocol is said to be access graph based if*

1. *Two nodes  $u, v \in V$  can only communicate with each other if there is an edge  $(u, v) \in E$ .*
2. *The decisions of the algorithm only depend on the topological structure of  $H$ .*
3. *If the neighborhood explored by an edge  $e$  is a symmetric  $(r, \bar{\gamma})$  neighborhood for some  $r, \bar{\gamma} \in \mathbb{N}$ , the edge can be assumed to be directed at random.*
4. *If an arbitrary subset of the edges of a given access graph  $H$  is deleted, the maximum load achieved does not increase.*

The 4th property which is not needed in [ACMR95] is very intuitive: removing jobs to be scheduled does not increase the load. All known algorithms respect the 4 properties, i.e. they are access based.

We make use of the following lemma:

**Lemma 3.2** *Let  $H'$  be random graph with  $n$  nodes and  $m$  edges, and let  $T$  be a fixed tree with degree  $\bar{\gamma}$  and depth  $r$ . For  $\bar{\gamma} \leq \sqrt{\frac{\log n}{18 \log \log n}}$  the probability that  $T$  is a subgraph of  $h$  is at least  $\frac{1}{2}$ .*

**Proof:** The lemma is proven in [Ste95] for the case  $m = n$ . Obviously adding more edges will only increase the probability. ■

We use this lemma to find a complete tree  $T$  of depth  $r$  in  $H$  whose inner vertices have a degree of  $\bar{\gamma} = \sqrt{\frac{\log n}{18 \log \log n}}$  as a subgraph of  $H$ . According to (4) of Definition 3.1 we can remove all edges of  $H$  not belonging to  $T$ . Now the root of  $T$  is a node of degree  $\bar{\gamma}$  whose incident edges have a symmetric  $(r, \bar{\gamma})$ -neighborhood. In  $r$  rounds the root is able to explore its  $r$ -neighborhood, which is a symmetric  $(r, \bar{\gamma})$ -neighborhood. So, according to (3) of Definition 3.1,

these edges direct themselves at random. So with probability  $\frac{1}{2}$  the root of  $T$  has in-degree  $\gamma' = \frac{\bar{\gamma}}{2}$  if  $\bar{\gamma}$  is even. Thus with probability  $\frac{1}{4}$ , there is a node with load at least  $\gamma' = \frac{1}{2} r \sqrt{\frac{\log n}{18 \log \log n}}$ . This finishes the proof of Theorem 1.2.

#### 4 Conclusions and Future Work

We have given a lower bound for allocating  $m \geq n$  balls to  $n$  bins using two i.u.r. possible destinations for each bin. The lower bound matches the best known algorithm up to a constant factor. As a main result we stated the adaptive collision protocol dealing with weighted balls and showing a tradeoff between the number of balls, the degree of uniformity of their weights, and the running time needed to achieve some given load. For balls with unit weights our protocol matches the optimal running time for that problem, further there are balls with weights such that the time our protocol requires for finding the optimal allocation is optimal.

We believe that the use of third destinations for the balls is not necessary, it is only used for our proof. Further, the lower bound should be extendable to an arbitrary constant number of destinations. We are well aware of the fact that our lower bound model is still very restrictive. It seems challenging to get such strong bound for more general models. The most important algorithmic problems concern modeling continuous allocation processes, designing and analysis them, extending Mitzenmachers ([Mit96a]) results.

#### References

- [ABK94] Yossi Azar, Andrei Z. Broder, and Anna R. Karlin. On-line load balancing. *Theoretical Computer Science*, 130:73–84, 1994. preliminary version in FOCS 92.
- [ABKU94] Y. Azar, A. Z. Broder, A. R. Karlin, and E. Ufal. Balanced allocations (extended abstract). In *Symposium on Theory of Computing*. ACM, 1994.
- [ACMR95] Micah Adler, Soumen Chakrabarti, Michael Mitzenmacher, and Lars Rasmussen. Parallel randomized load balancing (preliminary version). In *Symposium on Theory of Computing*. ACM, 1995.
- [AKP<sup>+</sup>93] Yossi Azar, Bala Kalyanasundaram, Serge Plotkin, Kirk R. Pruhs, and Orli Waarts. Online load balancing of temporary tasks. In *Proc. of 3rd Workshop on Algorithms and Data Structures*, pages 119–130, 1993.
- [DM90] Martin Dietzfelbinger and Friedhelm Meyer auf der Heide. How to distribute a dictionary in a complete network. In *Proceedings of the 22nd Symposium on Theory of Computing*, 1990.
- [Kur81] Thomas G. Kurtz. *Approximation of Population Processes*. Regional Conference Series in Applied Mathematics. CMBS-NSF, 1981.

- [Mit96a] Michael Mitzenmacher. Density dependent jump markov processes and applications to load balancing. In *FOCS*, 1996.
- [Mit96b] Michael David Mitzenmacher. *The Power of Two Random Choices in Randomized Load Balancing*. PhD thesis, Graduate Division of the University of California at Berkeley, 1996.
- [PW93] S. Phillips and J. Westbrook. Online load balancing and network flow. In *Proceedings of 25th STOC*, pages 402–411. ACM, 1993.
- [RM95] P. Raghvan R. Motwani. *Randomized Algorithms*. Cambridge University Press, 1995.
- [Ste95] Volker Stemann. *Contention Resolution in Hashing Based Shared Memory Simulations*. PhD thesis, Heinz Nixdorf Institute, University of Paderborn, 1995. appears in: HNI Verlagsschriftenreihe, Vol 3.
- [Ste96] Volker Stemann. Parallel balanced allocations (extended abstract). In *Annual ACM Symposium on Parallel Algorithms and Architectures*, 1996.