Threshold Load Balancing With Weighted Tasks

Petra Berenbrink*, Tom Friedetzky†, Frederik Mallmann-Trenn*, Sepehr Meshkinfamfard†, and Chris Wastell†

*School of Computing Science, Simon Fraser University, Burnaby, Canada
Email: petra@sfu.ca, fmallman@sfu.ca

†School of Engineering and Computing Sciences, Durham University, Durham, UK
Email: tom.friedetzky@dur.ac.uk, sepehr.meshkinfamfard@dur.ac.uk, christopher.wastell@dur.ac.uk

***Abstract*—We study threshold-based load balancing protocols for weighted tasks. We are given an arbitrary graph G with n nodes (resources, bins) and $m > n$ tasks (balls). Initially the tasks are distributed arbitrarily over the n nodes. The resources have a threshold and we are interested in the balancing time, i.e., the time it takes until the load of all resources is below the threshold.**

We distinguish between resource-based and user based protocols. In the case of resource-based protocols resources with a load larger than the threshold are allowed to send tasks to neighbouring resources. In the case of user-based protocols tasks allocated to resources with a load above the threshold decide on their own whether to migrate to a neighboring resource or not.

For resource-controlled protocols we present results for arbritary graphs. Our bounds are in terms of the mixing time (for above-average thresholds) and the hitting time (for tight thresholds) of the graph.

We relate the balancing time of resource-controlled protocols for above-average thresholds in arbitrary graphs to the mixing time of the graph and to the hitting time for tight thresholds.

Our bounds are tight and, surprisingly, they are independent of the weights of the tasks. For the user-controlled migration we consider complete graphs and derive bounds for both above-average and tight thresholds.

***Keywords*-Load balancing; Random walks; Threshold; Mixing time of random walks; Weighted Tasks.**

I. Introduction

We study threshold-based balls-into-bins schemes that can be used to balance load in distributed systems. The balls usually model tasks or data and the bins model the resources used to process the tasks or to store the data. The performance of a distributed system often depends on the maximum load of any of the machines. The higher the maximum load, the longer the execution time of the entire system. Hence, good load balancing schemes are crucial for efficient computations on distributed systems.

Most balls-into-bins games studied theoretically so far assume that the balls are of equal size. The size of a ball usually measures the computation time of the task it models, or the size of the data. However, this assumption is unrealistic in many cases. In this paper we study load balancing schemes with weighted balls which are, for example, able to model task with different runtimes.

Here we are given an arbitrary graph G with n nodes representing the resources (machines) and $m > n$ weighted tasks (balls) with a total weight of W. Initially the tasks are distributed arbitrarily over the n nodes. Similar to [1], [2], we assume every resource has a threshold which is the maximum load the resource can accept. The threshold is the same for all resources. We distinguish between a tight threshold of $W/n + w_{max}$, and above-average thresholds of $(1 + \varepsilon) \cdot W/n + w_{max}$, where w_{max} is the maximum weight of any task and ε is an arbitrary positive constant. Note that the thresholds must be chosen appropriately. i.e. they must be at least the average load. We assume that the average load can either be derived in quickly by, for example, using a diffusion load balancing process, or the threshold are given by some external restriction. In this paper we are interested in the balancing time, i.e., the time it takes until the load of all resources is below the threshold.

We distinguish between resource-based protocols where the resources with a load above the threshold (called overloaded resources) are allowed to send tasks to a neighboring resources, and user-based protocols where each task on an overloaded resource decides autonomously whether or not to migrate to a neighboring resource.

For resource-based protocols we consider arbitrary graphs. Our results are expressed in terms of the mixing time (for above-average thresholds) and hitting time (for tight thresholds) of random walks on the underlying graph. Subsequently we will show that these bounds are tight. For user-based protocols we only consider complete graphs. In all cases our bounds (for the weighted case) match the bounds of [1], [2]. All protocols we consider are decentralized and do not require a global view of the system.

A. Contributions

We follow the work of [1], [2] and study threshold-based balls-into-bins protocols, but in contrast to these

papers we assume that the balls (tasks) have arbitrary weights.

Resource-Controlled Protocols: For arbitrary graphs G and above-average thresholds we show (Theorem 2) that the balancing time is $O(\tau(G) \cdot \log m)$ w.h.p., where $\tau(G)$ is the mixing time of a random walk on G. Note that this bound does not depend on the weights of the tasks. In [2] the authors show a bound of $O(\mathcal{H}(G) \cdot \log n + \tau(G) \cdot \log m)$ on the expected balancing time, where $\mathcal{H}(G)$ is the hitting time of a random walk on G (See Section I-C). Note that our bounds for weighted tasks match their bound for uniform tasks and are even stronger in the sense that we do not have the hitting time in our bound. The reader may refer to Table I-A to find a comparison of hitting times and mixing times of several common graphs. From Theorem 3.7 of [2] it follows that our bound is tight.

Graph	Mixing Time	Hitting Time
Complete Graph	$O(1)$	$O(n)$
Reg. Expander	$O(\log n)$	$O(n)$
E.R. Random Graph	$O(\log n)$	$O(n)$
Hypercube	$O(\log n \log\log n)$	$O(n)$
Grid	$O(n)$	$O(n \log n)$

Table I
SUMMARY OF MIXING AND HITTING TIMES FOR COMMON GRAPHS

For tight thresholds we show (Theorem 5) a bound of $O(\mathcal{H}(G) \cdot \log m)$ on the expected balancing time. This bound matches the bound presented in [2]. Again, the bound is independent of the weight of the tasks. In Observation 6 we show that this bound is also tight.

User-Controlled Protocols: We consider complete graphs for above-average and tight thresholds. For above-average thresholds we derive a bound of $O(w_{max}/w_{min} \cdot \log m)$ on the expected balancing time (Theorem 9). w_{max} and w_{min} denote the maximum and minimum weight of any task respecitively. Moreover, for tight thresholds we derive (Lemma 10) a bound of $O\left(w_{max}/w_{min} \frac{\log m}{n^2}\right)$.

Both bounds match the bounds of [1] for uniform weights. However for weighted balls our bounds include the additional factor of w_{max}/w_{min}.

In Section IV, we briefly discuss experimental results highlighting cases where the theoretically derived bounds for user-controlled migration are tight and where not.

B. Related Work

There are many results about balls-into-bins games for uniform balls. Here we concentrate on protocols for weighted balls and on local protocols that use a threshold to allocate the balls.

1) Thresholds: In [3] the authors consider parallel threshold protocols, they investigate the trade-off between the number of rounds of communication between the resources and the final load. Specifically, for given number of rounds of communication r, they prove a *lower* bound on the maximum load of $\Omega(\sqrt[r]{\log n \log\log n})$ for n unit-sized balls and n resources.

The papers [1], [2] are the most closely related to our work. However both publications only consider uniform balls. In [1] the authors show results for threshold-based balancing protocols for user-controlled migration on complete graphs. For above-average thresholds they show a bound on the balancing time of $O(\log m)$ and $O(n^2 \cdot \log m)$ for tight thresholds. The results of [1] were generalized to arbitrary graphs in [2]. For the resource-controlled protocols they provide a bound of $O(\mathcal{H}(G) \log m)$ on the balancing time, where $\mathcal{H}(G)$ is the hitting time of a random walk on G. They also show an improved bound of $O(\mathcal{H}(G) \cdot \log n + \tau(G) \cdot \log m)$ for above-average thresholds, where $\tau(G)$ is the mixing time of a random walk on G. For the user-controlled protocol they provide a bound of $O\left(n^5 \cdot \mathcal{H}(G) \cdot \log m\right)$ to reach a balanced state.

In [4] the authors consider a sequential balls-into-bins processes that randomly allocates m uniform balls into n bins using thresholds. They analyze two allocation schemes that achieve a close to optimal maximum load of $\lceil m/n \rceil + 1$ and require only $\mathcal{O}(m)$ random choices.

2) Weighted balls: The authors of [5] were among the first to consider the problem of allocating weighted balls in parallel. Their key result is a generalization of the upper bound presented in [6] to weighted balls : Let w_{avg} (w_{max}) denote the average (maximum) weight. They present a protocol that achieves a maximum load of $\gamma \cdot (m/n \cdot w_{avg} + w_{max})$ using $O(\log\log n/(\log \gamma \cdot ((m/n) \cdot \Delta + 1))$ communication rounds, where $\Delta = w_{avg}/w_{max}$.

In [7] the authors investigate generic multiple-choice balls-into-bins protocols with weighted balls, showing some surprisingly counter-intuitive properties. In [8] the authors investigate a sequential balls-into-bins process where m weighted balls are allocated to n The process allocates every ball into the least loaded of two randomly chosen bins. The case where each of the m balls has unit weight had been studied in [9]. The authors show that the difference between the maximum load and the average load does not increase with m. In [8] the authors show that as long as the weight distribution has finite second moment and satisfies a mild technical condition, the gap between the load of the heaviest bin and the load of the average bin is independent of the number balls thrown. In [10] the authors consider the so-called $(1+\beta)$-process where each ball goes to a random bin with probability β and to the least-loaded of two randomly

chosen bins with a probability of $(1-\beta)$. The authors show that for the $(1+\beta)$-process the gap between the minimum and average load is at most $\Theta(\log n/\beta)$, independent of m. They also show that the gap remains $\Theta(\log n/\beta)$ in the weighted case for a large class of weight distributions.

In [11] the authors consider balls-into-bins games in a user-controlled setting. In the beginning the balls are arbitrarily distributed over the bins. The protocol works in parallel rounds. In every round every user (ball) is allowed to randomly select another bin and to move to that bin if the load is smaller than the load of its current bin. In particular, they provide an upper bound on the balancing time of $\log\log m + \text{poly}(n)$. In [12] these results are generalized to weighted balls. They shows that their protocol yields an expected balancing time of $O(nm\Delta^3\epsilon^{-2})$. Finally in [13] the latter result is generalized to weighted balls and resources with "speeds".

C. Model

Let $[m] = \{1, 2, \ldots, m\}$ be the set of tasks and let $[n] = \{1, 2, \ldots, n\}$ be the set of resources. The resources are connected by an arbitrary graph $G = (V, E)$. Let d_i be the degree of node i and d the maximum degree. Tasks on a resource r can move to a neighboring resource r' if $(r, r') \in E$.

Each task $i \in [m]$ has an associated weight $w_i \in \mathbb{N}$. Let w_{max} be the maximum weight and let $W = \sum_{i=1}^{m} w_i$ be the total weight of all tasks. We assume that $w_{min} \geq 1$. If this is not the case, then one can easily scale all parameters, such that $w_{min} = 1$.

Let

$$x(t) = (x_1(t), x_2(t), \ldots, x_n(t))$$

denote the load vector at the beginning of step t (before the task removal) where $x_r(t)$ is the load of resource r.

$$X(t) = (X_1(t), X_2(t), \ldots, X_n(t))$$

denotes the state of the system at the beginning of step t, where $X_r(t)$ is the random variable that denotes the load of resource r. We use $b_r(t)$ to denote the number of balls on machine r at time t. Our protocols will use a threshold

$$\text{Tr} = (1+\varepsilon) \cdot W/n + w_{max}$$

with $\varepsilon \geq 0$.

D. Random walks

For an undirected, connected graph G let $P_{i,j}$ be the probability that the random walk moves from node i to node j. We consider standard random walks for non-regular graphs with transition matrix P, where $P_{i,j} = 1/d$ for $i \neq j$ and $(i,j) \in E$ and $P_{i,i} = (d - d_i)/d$. Let P^t be the t-th power of P. Then $P^t_{i,j}$ is the probability that a random walk starting from node i is located at node j after exactly t steps. The stationary distribution of the random walk on G is called $\pi(G)$ and it is the uniform distribution for this random walk. Note that, in general, the results in this paper hold for all random walks where the stationary distribution equals the uniform distribution.

In [14] the authors give a bound on the *mixing time* $\tau(G)$ of the random walk, which is defined as the expected time it takes for the random walk (defined above) on G to approach its the stationary distribution. A statement of the result can be found in Lemma 12 the appendix. From Lemma 12 it follows that we can assume that $\tau(G) = 4\log n/\mu$, where μ is the spectral gap of P. With $\lambda_1 \geq \lambda_2 \geq \ldots \geq \lambda_n$ the n Eigenvalues of P we have

$$\mu := 1 - \max_{2 \leq i \leq n} \{|\lambda_i|\}.$$

The hitting time $\mathcal{H}_{u,v}(G)$ of a random walk is defined as the time for a random walk to reach node $v \in V$ when starting from node $u \in V$. We define the maximum hitting time as

$$\mathcal{H}(G) := \max_{u,v \in V} \mathcal{H}_{u,v}(G).$$

II. Resource-Controlled Migration

In this section, we consider a protocol in which each overloaded resource determines whether or not tasks should migrate (Algorithm II.1). Tasks which are currently assigned to resource r can only move to neighboring resources of r (see the algorithm below). The protocol is distributed and every node requires only knowledge of its own load and the global threshold. The algorithm in Figure II.1 shows one step of our protocol.

We assume that every resource stores all its tasks in a stack data structure. If several balls arrive at the same resource in one time step the balls are stored in an arbitrary order. The *height* $h^i_r(t)$ of task i on resource r at time t is the sum of the weights of all tasks in the data structure that are positioned before i. We say task i is *cutting the threshold* Tr if $h^i_r(t) < \text{Tr}$ and $h^i_r(t) + w_i > \text{Tr}$.

Each task on resource $r \in [n]$ can either be completely *above*, *completely below*, or *cutting* the threshold. Let $I^a_r(t)$ ($I^b_r(t)$) be the sets of tasks on resource r that are completely above (completely below) the threshold at the beginning of step t. $I^c_r(t)$ denotes the tasks on resource r which partially above threshold at the beginning of step t.

We say that a task is *accepted* by a resource if the height of the task plus its weight is less than or equal to the threshold. Note that once a task is accepted by a resource, it will never leave that resource again. We call these tasks *inactive*, and the tasks that are not accepted by a resource are called *active*.

Algorithm II.1 Resource Controlled

for all All resources r in parallel **do**
  if $x_r(t) >$ Tr **then**
    Remove any task $i \in I_r^a(t) \cup I_r^c(t)$ and reallocate the task to a neighboring resource that is chosen according to transition matrix P.
  Assign new heights to all migrated balls

A. Above-Average Thresholds

In this section, we assume thresholds are larger by some constant factor than the average load W/n. The next lemma will be used in the proof of this sections main result. It estimates the probability that a randomly chosen resource has a load below or equal to the threshold. The result is quite easy to see and it allows us in Section III to simplify the potential analysis of [1].

Lemma 1. *Assume that r is a resource chosen uniformly at random in step t. Then*

$$P[X(t+1) \leq Tr] \geq \varepsilon/(1+\varepsilon).$$

Proof: From a simple pigeonhole argument it follows that at any point in time (meaning also at the end of a step) there is a fraction of $\varepsilon/(1+\varepsilon)$ resources that can accept an additional task of any weight not larger than w_{max}. Assume this is not the case. Then there are at least

$$(1-\varepsilon/(1+\varepsilon)) \cdot n + 1$$

many resources with a weight of at least $(1+\varepsilon) \cdot W/n$. Then

$$((1-\varepsilon/(1+\varepsilon)) \cdot n + 1)(1+\varepsilon) \cdot W/n > W,$$

which is a contradiction. ∎

The following result holds for arbitrary graphs. It is stated in terms of the performance a random walk on G. Let $\tau(G)$ denote the mixing time of the underlying Markov chain. For complete graphs the result shows a balancing time of $O(\log m)$.

Theorem 2. *Assume $Tr = (1+\varepsilon) \cdot W/n + w_{max}$. Assume that G is an arbitrary graph with mixing time $\tau(G)$. Let c be an arbitrary constant. Then, with a probability of $1 - n^{-c}$ all tasks are allocated after*

$$4(c+1) \cdot \tau(G) \cdot \frac{\log m}{\log(1+\varepsilon)}$$

time steps.

Proof: Following our algorithm we can assume that every active task performs a random walk with transition matrix P, until it reaches a resource that has a load that is small enough to accept the task. If a task is accepted by a resource, the task becomes *inactive*. All other tasks are *active*. We now divide the time steps into *phases* of length $2\tau(G)$ each, where τ is the mixing time of the random walk on G.

We fix a phase j and an active task i and assume that task i is still active in the last step of phase j. Let $r_{i,j}$ be the resource that it visits in the last step of phase j. From Lemma 12 it follows for every $1 \leq k \leq n$ that

$$P[r_{i,j} = k] = n^{-1} \pm n^{-3}.$$

From Lemma 1 it follows that the probability that task i is successful in the last step in phase j is at least

$$\frac{\varepsilon n}{1+\varepsilon} \cdot \left(\frac{1}{n} - \frac{1}{n^3}\right) \geq \frac{\varepsilon}{2(1+\varepsilon)}.$$

Now assume that after $\ell \log m$ many rounds, with $\ell = 2(c+1) \cdot \log m / \log(1+\varepsilon)$, there exists a task i that is still active. A necessary condition for the task to still be active is that it was not accepted by any resource in any of the last steps of the ℓ phases. The probability that the task was not accepted by an resource in any of the last steps of each of the ℓ rounds is at most

$$\left(1 - \frac{\varepsilon}{2(1+\varepsilon)}\right)^{\ell} \leq \left(\frac{1}{m}\right)^{c+1}.$$

The result follows from the union bound. ∎

B. Tight Threshold

The next theorem shows results for a tighter threshold

$$\text{Tr} = W/n + 2w_{max}.$$

Our result bounds the balancing time in terms of the *hitting time* $\mathcal{H}(G)$.

We call an assignment of the weighted tasks to the resources *proper* if no resource has a weight of more that $W/n + w_{max}$. Note that it is trivial to calculate a proper assignment. The simple *first fit* rule will work.

To analyze the protocol we will use a potential function that counts the weights of the tasks that are partially or completely above the threshold. At any $t \geq 0$ the potential Φ is defined as

$$\Phi(X(t)) = \sum_{i \in I^a(t) \cup I^c(t)} w_i. \tag{1}$$

For any $t > 0$ the potential change between subsequent states $X(t)$ and $X(t+1)$ is defined as

$$\Delta\Phi(t+1) = \Phi(X(t)) - \Phi(X(t+1)). \tag{2}$$

The next observation shows that the potential function is non-increasing.

Observation 3. *For any $t > 0$ we have*

$$\Delta\Phi(t+1) \geq 0.$$

Proof: Based on the definition, at any time $t > 0$,

any task i can either be in $I^c(t) \cup I^a(t)$ or $I^b(t)$. For the ease of presentation we assume that the protocol considers the tasks sequentially in an arbitrary order. In the following, we call a step where one of the tasks is considered a *sub-step*. Assume that task i is one of the tasks that is moved to a neighboring resource in sub-step t'. Note that task i can be moved to another resource if and only if $i \in I^c(t) \cup I^a(t)$ at the beginning of step t. At the beginning of step $t+1$, either $i \in I^c(t+1) \cup I^a(t+1)$ or $i \in I^b(t+1)$. In the first case the potential remains unchanged, in the latter case $\Delta\Phi(t+1) = w_i$. Also, note that due to our stack ordering the potential does not change due to a task that did not move to another resource in sub-step t'. ∎

The next lemma estimates the one-step potential decrease.

Lemma 4. *Assume* $Tr = W/n + 2w_{max}$.

$$\mathbb{E}\left[\Delta\Phi(t+2\mathcal{H}(G)) \mid X(t) = x(t)\right] \geq \frac{\Phi(X(t))}{4}.$$

Proof: Here we consider a *phase* of length $2\mathcal{H}(G)$, where $\mathcal{H}$ is the hitting time. At the beginning of a phase we assign each active task to one of the resources such that the maximum load of any resource is at most $W/n + w_{max}$. This bin is called the *target* bin of the task. Then for every active task we place a pebble on its resource. The pebbles perform a random walk of length $\mathcal{H}(G)$. If a pebble hits the target bin of the corresponding task then the task is coloured *blue*. Otherwise the task is coloured *red*.

Let B (R) be the set of blue (red) tasks and $W(B)$ ($W(R)$) be the total weight of the blue (red) tasks. Recall that $\Phi(X(t))$ is the weight of the active tasks at the beginning step t and

$$\Phi(x(t)) = W(B) + W(R).$$

Let X' be the state in which all tasks are assigned to the same resource as in $X(t)$. In addition to these tokens, we assign all the blue tasks to their targets. Let $\Delta\Phi'(X(t))$ be the potential drop which happens due to the assignment of the blue tasks. Using Markov's bound it follows that a random walk hits its target with a probability of 1/2. Hence

$$\Delta\Phi'(X(t)) \geq \Phi(X(t))/2.$$

Now we consider the original process where every task performs a random walk until it reaches a resource that has a load that is small enough to accept the task. Our goal is to show

$$\Phi(X(t)) - \Phi(X(t+2\mathcal{H}(G))) \geq \frac{\Delta\Phi'(X(t))}{2} \geq \frac{\Phi(X(t))}{4}.$$

Without loss of generality, we assume that the tasks follow their pebble, as long as they were not accepted by one of the resource (in which case they are accepted by the resource and become inactive). We now split the resources into two sets. Resources that were not able to accept all incoming tasks during the phase are called *full*. These resources have a load larger than $W/n+w_{max}$. The other resources are called *good*. Note that a resource is either full or good.

Additionally, we partition the blue tasks and the red tasks into three sets. The tasks of B_g are blue tasks that are accepted by a good resource, and the tasks of R_g are red tasks that are accepted by a good resource. Note that this means that they did not steal the position of another task in its target bin. The tasks B_f (R_f) are blue (red) tasks accepted by a full resource. Finally, the tasks that are still active at the end of Phase i are in the sets B_n and R_n. Note that

$$\Delta\Phi'(X(t)) = W(B) = W(B_g) + W(B_f) + W(B_n)$$

and

$$\begin{aligned}&\Phi(X(t+2H))\\ &\quad = \Phi(X(t)) - (W(B_g) + W(B_f) + W(R_g) + W(B_f)).\end{aligned}$$

In the following, we show that $W(B_f) + W(R_f) \geq W(B_n)$. Then it follows from Observation 13 in the appendix that

$$\Phi(X(t)) \geq \frac{\Delta\Phi'(X(t))}{2}.$$

To show that $W(B_f) + W(R_f) \geq W(B_n)$ we assume that ℓ_b is the total weight of the tasks that are accepted by resource b at the beginning of the phase. Then we get that

$$W(B_n) \leq \sum_{b \in \textsc{Full}} \frac{W}{n} + w_{max} - \ell_b.$$

This holds since $\sum_{b\in\textsc{Full}} W/n + w_{max} - \ell_b$ is an upper bound on the total number of blue tasks that can assigned to a full resource during the phase (recall, A' uses the threshold $T' = W/n + w_{max}$). Additionally,

$$W(R_f) + W(B_f) \geq \sum_{b \in \textsc{Full}} \frac{W}{n} + w_{max} - \ell_b.$$

This holds due to the definition of a full resource which is a resource that has a load larger than $W/n + w_{max}$ (recall that A uses the threshold $T = W/n + 2w_{max}$). This concludes the proof of this lemma. ∎

Now we use the above lemma to show the following result.

Theorem 5. *Assume* $Tr = W/n + 2w_{max}$. *Let* $\mathcal{H}(G)$ *be the hitting time of the random walk on* G *with uniform stationary distribution. Let* T *be the time it takes until*

all balls are allocated. Then

$$\mathbb{E}[t] = O(\mathcal{H}(G) \cdot \ln(W)).$$

Proof: Since the maximum potential is bounded by W this follows from the Drift Theorem in Lemma 11 in the Appendix. ∎

The next observation shows that the bound of Theorem 5 is tight for uniform tasks. For weighted tasks the bound is not tight because the total weight W is super-polynomial in m.

Observation 6. *There is a class of graphs such that the resource based protocol converges to a balanced state in an expected number of steps of $\Omega(\mathcal{H}(G) \cdot \log m)$ for tight thresholds.*

Proof: The proof follows the same line of argument as the proof of Theorem 3.7 in [2]. Instead of two cliques glued together with k edges we use the following graph G. Let G consists of a clique K of $n-1$ nodes and one single node u. This single node is connected to exactly k nodes of the clique for some arbitrary $k < n$. The hitting time of this graph is $\Theta(n^2/k)$. Initially, we distribute the tasks on nodes of K in such a way that all nodes in K have a load of W/n the remaining balls are distributed on an arbitrary node of K.. . By using the same arguments as in [2] the required time becomes

$$\Omega(\mathcal{H} \log(m/n)) = \Omega(\mathcal{H} \log(m))$$

for $m \gg n$. ∎

III. User-Controlled Migration

In this section, we consider user-controlled protocols for complete graphs. Before defining the protocol, we first present some necessary definitions.

We assume that the resources store the tasks in a stack data structure. Recall, the *height* $h_r^i(t)$ of task i in resource r is the sum of the weights of all task that are positioned before i. We say task i is *cutting the threshold* if $h_i^r(t) < \mathrm{Tr}$ and $h_r^i(t) + w_i > \mathrm{Tr}$. Then the potential of an overloaded resource r is called $\Phi_r(t)$ and counts the weight of the task which is cutting the threshold (if there is any) plus the weights of the tasks which are above the threshold. The potential of a non-overloaded resource is zero. The potential $\Phi(t)$ at step t is defined as

$$\Phi(t) = \sum_{i=1}^{n} \Phi_i(t).$$

The user-controlled protocol works as follows. Tasks leave an overloaded resource with a probability of $\alpha \cdot \left\lceil \frac{\phi_r}{w_{max}} \right\rceil \cdot \frac{1}{b_r}$ and move to a randomly chosen resource.

Algorithm III.1 User-Controlled

for all All users do in parallel **do**
  Let $r(i)$ be the resource allocated by user i.
  if $x_{r(i)} > \mathrm{Tr}$ **then**
    With probability $\alpha \cdot \left\lceil \frac{\phi_r}{w_{max}} \right\rceil \cdot \frac{1}{b_r}$ migrate to resource choosen uniformly at random.

A. Above-Average Thresholds

To analyze the potential change we assume that the resources store the tasks in a stack data structure. We consider tasks leaving one after the other starting with all tasks leaving Resource 1, then Resource 2, and so on. For every resource we consider the tasks in the order of increasing heights. If it is clear from the context and the time step t is fixed, we will just write Φ_r, b_r, and x_r.

The potential $\Phi(t+1))$

- decreases by w_i for every task i which was above the threshold ($h_r(t) + w_i > \mathrm{Tr}$) and migrates to a resource r' such that $h_{r'}(t+1) + w_i \leq \mathrm{Tr}$
- does not change for every task i which was above the threshold ($h_r(t) + w_i > \mathrm{Tr}$) and migrates to a resource r' such that $h_{r'}(t+1) + w_i > \mathrm{Tr}$
- increases by w_i for every task i which was below the threshold ($h_r(t) + w_i \leq \mathrm{Tr}$) and migrates to a resource r' such that $h_{r'}(t+1) + w_i > \mathrm{Tr}$.

For the sake of presentation, the potential change caused by task i leaving r is accounted for at resource r as opposed to at the resource r'.

First observe the following.

Observation 7. *Let t be an arbitrary time step and $\Phi_r(t) > 0$. Then the number of tasks required to leave r such that $x(t+1) < Tr$ is at least*

$$\phi_r' = \lceil \Phi_r / w_{max} \rceil.$$

We now calculate the one-step potential change. First we assume $w_{min} = w_{max} = 1$, then the potential change can be estimated in the following way.

$$\mathbb{E}[\Delta\Phi_r | i \text{ balls leave}] \geq \begin{cases} \frac{\varepsilon}{1+\varepsilon} \cdot i & \text{if } i \leq \phi_r' \text{ balls leave} \\ \phi_r' - i & \text{if } i > \phi_r' \text{ balls leave.} \end{cases}$$

In the first case fewer than ϕ' tasks leave, meaning that the potential decreases. Note that it does not matter if tasks above the threshold or below the threshold leave since $w_{max} = w_{min}$, but also since $i \leq \phi'$ which is a key insight for the case where $w_{max} \neq w_{min}$. The second case is pessimistic and assumes 1) that the ϕ_r' tasks leaving from above the threshold move to an overloaded resource and 2) that an additional $(i - \phi_r')$ tasks leave from below the threshold and move to a resource with a load $\geq \mathrm{Tr}$ such that they are above the threshold.

From Lemma 1 it follows that there is a fraction of $\varepsilon/(1+\varepsilon)\cdot n$ resources which can accept additional tasks and which are still not overloaded at the end of a round. Hence, the probability that a leaving task decreases the potential Φ is at least $\varepsilon/(1+\varepsilon)$.

In the weighted setting, the potential decreases if $i \le \phi'_r$ tasks leave and land on an underloaded resource. Since every task leaves with the same probability we can assume that the expected size of a leaving task is x_r/b_r. If $i > \phi'_r$ tasks leave resource r, then we forget about the tasks that move from above the threshold again and assume a potential increase by at most $(i-\phi)\cdot x_r/b_r$. Hence, in the weighted setting we have the following bounds on the potential change.

$$\mathbb{E}\left[\Delta\Phi_r | i \text{ balls leave}\right] \ge \begin{cases} \frac{\varepsilon}{1+\varepsilon}\cdot i\cdot\frac{x_r}{b_r} & \text{if } i \le \phi'_r \text{ balls leave} \\ (\phi'_r - i)\frac{x_r}{b_r} & \text{if } i > \phi'_r \text{ balls leave.} \end{cases}$$

We note that for the first case, due to the definition of ϕ'_r, the potential decreases in expectation by $\frac{x_r}{b_r}$ for every leaving task even if the height of this task is below the threshold .

In the next lemma we use that observation to calculate the one-step potential drop.

Lemma 8. *Let $\alpha = \frac{\varepsilon}{120(1+\varepsilon)}$. Assume $Tr = (1+\varepsilon)\cdot W/n + w_{max}$ and $\Phi(X(t)) > 0$. We have*

$$\mathbb{E}\left[\Delta\Phi(t+1) \mid X(t)=x(t)\right] \ge \frac{1}{2}\cdot\frac{\varepsilon}{1+\varepsilon}\cdot\Phi(X(t)).$$

Proof: Let $p_r(i)$ denote the probability that *exactly* i tasks leave resource r. We emphasize that all tasks on a given resource have the same probability to leave. For $1 \le i \le b_r$ we define a random variable $Y_r(i)$ which is one if task i on resource r leaves and zero otherwise.

$$\begin{aligned}
&\mathbb{E}\left[\Delta\Phi_r(t+1) \mid X(t)=x(t)\right] \\
&= \sum_{i=0}^{b_r} \mathbb{E}\left[\Delta\Phi_r | i \text{ balls leave}\right]\cdot p_r(i) \\
&\ge \sum_{i=1}^{\phi'_r} \frac{\varepsilon}{1+\varepsilon}\cdot i\cdot\frac{x_r}{b_r}\cdot p_r(i) - \sum_{i=\phi'_r+1}^{b_r} i\cdot\frac{x_r}{b_r}\cdot p_r(i) \\
&\ge \frac{x_r}{b_r}\sum_{i=1}^{b_r}\frac{\varepsilon}{1+\varepsilon}\cdot i\cdot p_r(i) - 2\frac{x_r}{b_r}\sum_{i=\phi'_r+1}^{b_r} i\cdot p_r(i) \\
&= \frac{x_r}{b_r}\frac{\varepsilon}{1+\varepsilon}\cdot\mathbb{E}\left[\sum_{i=1}^{b_r} Y_r(i)\right] - 2\cdot\frac{x_r}{b_r}\sum_{i=\phi'_r+1}^{b_r} i\cdot p_r(i),
\end{aligned}$$

where for the last step we used the fact that all tasks move with the same probability. Since

$$E[Y_r(i)] = \alpha\cdot\left\lceil\frac{\Phi_r}{w_{max}}\right\rceil\cdot\frac{1}{b_r},$$

we get $E\left[\sum_{i=1}^{b_r} Y_r(i)\right] = \alpha\cdot\left\lceil\frac{\Phi_r}{w_{max}}\right\rceil$. We now bound $\sum_{i=\phi'_r+1}^{b_r} i\cdot p_r(i)$. We first observe that the term is maximized for small values of ϕ_r. Since $\phi'_r \ge 1$ we derive

$$\begin{aligned}
\sum_{i=\phi'_r+1}^{b_r} i\cdot p_r(i) &= \sum_{i=\phi'_r+1}^{b_r} i\binom{b_r}{i}\left(\alpha\left\lceil\frac{\Phi_r}{w_{max}}\right\rceil\cdot\frac{1}{b_r}\right)^i \\
&\quad\cdot\left(1-\alpha\left\lceil\frac{\Phi_r}{w_{max}}\right\rceil\cdot\frac{1}{b_r}\right)^{b_r-i} \\
&\le \sum_{i=\phi'_r+1}^{b_r} i\binom{b_r}{i}\left(\alpha\left\lceil\frac{\Phi_r}{w_{max}}\right\rceil\cdot\frac{1}{b_r}\right)^i \\
&\le \sum_{i=\phi'_r+1}^{b_r} i\left(\alpha\frac{e\cdot b_r}{i}\cdot\left\lceil\frac{\Phi_r}{w_{max}}\right\rceil\cdot\frac{1}{b_r}\right)^i \\
&\le \sum_{i=\phi'_r+1}^{b_r} i\left(\alpha\frac{e\cdot b_r}{\left\lceil\frac{\Phi_r}{w_{max}}\right\rceil}\left\lceil\frac{\Phi_r}{w_{max}}\right\rceil\cdot\frac{1}{b_r}\right)^i \\
&\le \sum_{i=2}^{\infty} i(e\alpha)^i \le 30\alpha^2.
\end{aligned}$$

For $\alpha = \frac{1}{120(1+\varepsilon)}$ we have

$$\begin{aligned}
&\mathbb{E}\left[\Delta\Phi_r(t+1) \mid X(t)=x(t)\right] \\
&\ge \frac{1}{(1+\varepsilon)}\left\lceil\frac{\Phi_r}{w_{max}}\right\rceil\cdot\alpha\cdot\frac{x_r}{b_r} - 60\alpha^2\cdot\frac{x_r}{b_r} \\
&\ge \alpha\cdot\frac{1}{2(1+\varepsilon)}\cdot\frac{x_r}{b_r}\cdot\frac{(\Phi_r)}{w_{max}} \\
&\ge \alpha\cdot\frac{1}{2(1+\varepsilon)}\cdot\frac{w_{min}}{w_{max}}\cdot\Phi_r
\end{aligned}$$

Summing over all resources r with $\Phi_r > 0$ yields

$$\begin{aligned}
&\mathbb{E}\left[\Delta\Phi(t+1) \mid X(t)=x(t)\right] \\
&\ge \alpha\cdot\frac{\varepsilon}{2(1+\varepsilon)}\cdot\frac{w_{min}}{w_{max}}\cdot\Phi.
\end{aligned}$$

■

We can now use the above lemma to show the following result for complete graphs.

Theorem 9. *Assume $Tr = (1+\varepsilon)\cdot W/n + w_{max}$ and $\alpha = \frac{\varepsilon}{120\cdot(1+\varepsilon)}$. Let T be the time it takes until all tasks are allocated. Then*

$$\mathbb{E}\left[T\right] = 2\cdot\frac{1+\varepsilon}{\alpha\cdot\varepsilon}\cdot\frac{w_{max}}{w_{min}}\cdot\log m.$$

Proof: The theorem follows from Lemma 8 together with the Drift Theorem (see Lemma 11). ■

B. Tight Threshold

In this section we show results for tight thresholds on complete graphs.

Theorem 10. *Assume* $Tr = W/n + w_{max}$ *and* $\alpha \leq \frac{1}{120n}$. *Let* T *be the time it takes until all tasks are allocated. Then*

$$\mathbb{E}[T] = \frac{2 \cdot n}{\alpha} \cdot \frac{w_{max}}{w_{min}} \cdot \log m.$$

Proof: It is easy to see that at any point in time there is at least one resource which can accept an additional task of any weight at most w_{max}. Therefore, by replacing $\varepsilon/(1+\varepsilon)$ with $1/n$ and setting $\alpha \ll \frac{1}{n}$ in the proof of Theorem 9 the result follows. ■

IV. Simulations

In this section, we show some simulation results for the user-controlled protocol and complete graphs. Note that our bounds for the resource-controlled protocols are tight. In our simulations we assume $w_{min} = 1$, $\varepsilon = 0.2$, and $\alpha = 1$. We also assume that all tasks are initially situated on the same resource. Each data point is obtained by averaging over 1000 trials.

In Figure 1 we have two different task sizes. All tasks have with $w_{min} = 1$ or $w_{max} = 50$. The x-axis shows the total task weight W. k denotes the number of tasks with weight w_{max} and $m(W,k) = W - k \cdot w_{\max}$ is the number of tasks with weight $w_{min} = 1$.

The simulation shows that the balancing time is proportional to the logarithm of $m(W,k) + k$. Hence, the results seems to be more or less independent of the number of big balls.

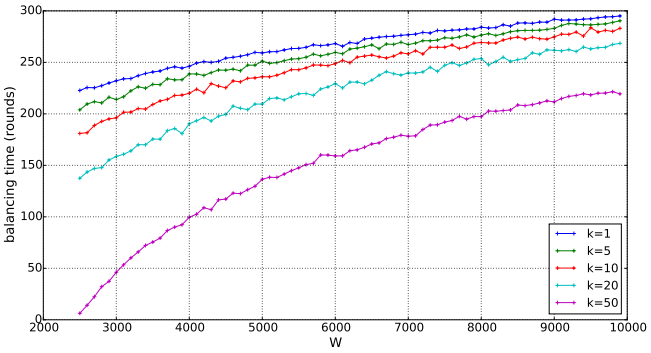

Figure 1. Balancing time in terms of k, where k denotes the number of tasks with weight $w_{max} = 50$ and $n = 1000$.

Given the outcome of the first simulation, we now consider in Figure 2 the case where there is only one task with weight w_{max}. On the x-axis we have the number of tasks and on the y-axis the balancing time, normalized by $\log m$. We show results for different sizes of maximum ball weights. This simulation suggests that the upper bound of Theorem 9 is tight up to a constant factor; the balancing time of the simulation is logarithmic in m and almost linear in w_{max}/w_{min}.

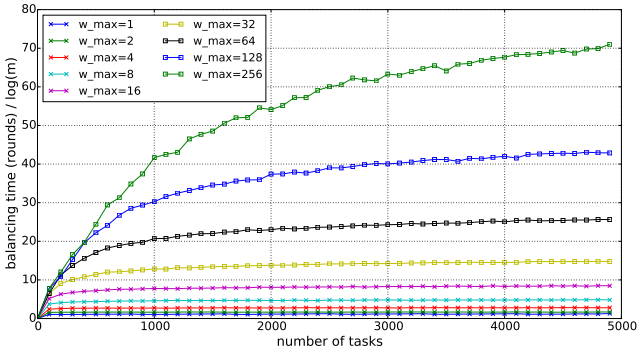

Figure 2. Balancing time in terms of w_{max} with $n = 1000$.

Our simulations show that a small value of α is not necessary. We are leaving it as an open question whether the theoretical bound can also be shown for $\alpha = 1$.

References

[1] H. Ackermann, S. Fischer, M. Hoefer, and M. Schöngens, "Distributed algorithms for qos load balancing," *Distributed Computing*, vol. 23, no. 5-6, pp. 321–330, 2011. [Online]. Available: http://dx.doi.org/10.1007/s00446-010-0125-1

[2] S. Hoefer and T. Sauerwald, "Threshold load balancing in networks," *CoRR*, vol. abs/1306.1402, 2013. [Online]. Available: http://arxiv.org/abs/1306.1402

[3] M. Adler, S. Chakrabarti, M. Mitzenmacher, and L. Rasmussen, "Parallel randomized load balancing," *Random Struct. Algorithms*, vol. 13, no. 2, pp. 159–188, 1998. [Online]. Available: http://dx.doi.org/10.1002/(SICI)1098-2418(199809)13:2<159::AID-RSA3>3.0.CO;2-Q

[4] P. Berenbrink, K. Khodamoradi, T. Sauerwald, and A. Stauffer, "Balls-into-bins with nearly optimal load distribution," in *25th ACM Symposium on Parallelism in Algorithms and Architectures, SPAA '13, Montreal, QC, Canada - July 23 - 25, 2013*, 2013, pp. 326–335. [Online]. Available: http://doi.acm.org/10.1145/2486159.2486191

[5] P. Berenbrink, F. Meyer auf der Heide, and K. Schröder, "Allocating weighted jobs in parallel," in *Proceedings of the Ninth Annual ACM Symposium on Parallel Algorithms and Architectures*, ser. SPAA '97. New York, NY, USA: ACM, 1997, pp. 302–310. [Online]. Available: http://doi.acm.org/10.1145/258492.258522

[6] V. Stemann, "Parallel balanced allocations," in *Proceedings of the Eighth Annual ACM Symposium on Parallel Algorithms and Architectures*, ser. SPAA '96. New York, NY, USA: ACM, 1996, pp. 261–269. [Online]. Available: http://doi.acm.org.proxy.lib.sfu.ca/10.1145/237502.237565

[7] P. Berenbrink, T. Friedetzky, Z. Hu, and R. Martin, "On weighted balls-into-bins games," *Theor. Comput. Sci.*, vol. 409, no. 3, pp. 511–520, 2008. [Online]. Available: http://dx.doi.org/10.1016/j.tcs.2008.09.023

[8] K. Talwar and U. Wieder, "Balanced allocations: the weighted case," in *Proceedings of the 39th Annual ACM Symposium on Theory of Computing, San Diego, California, USA, June 11-13, 2007*, 2007, pp. 256–265. [Online]. Available: http://doi.acm.org/10.1145/1250790.1250829

[9] P. Berenbrink, A. Czumaj, A. Steger, and B. Vöcking, "Balanced allocations: The heavily loaded case," *SIAM J. Comput.*, vol. 35, no. 6, pp. 1350–1385, 2006. [Online]. Available: http://dx.doi.org/10.1137/S009753970444435X

[10] Y. Peres, K. Talwar, and U. Wieder, "The (1 + beta)-choice process and weighted balls-into-bins," in *Proceedings of the Twenty-First Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2010, Austin, Texas, USA, January 17-19, 2010*, 2010, pp. 1613–1619. [Online]. Available: http://dx.doi.org/10.1137/1.9781611973075.131

[11] P. Berenbrink, T. Friedetzky, L. Goldberg, P. Goldberg, Z. Hu, and R. Martin, "Distributed selfish load balancing," *SIAM J. Comput.*, vol. 37, no. 4, pp. 1163–1181, 2007. [Online]. Available: http://dx.doi.org/10.1137/060660345

[12] P. Berenbrink, T. Friedetzky, I. Hajirasouliha, and Z. Hu, "Convergence to equilibria in distributed, selfish reallocation processes with weighted tasks," *Algorithmica*, vol. 62, no. 3-4, pp. 767–786, 2012. [Online]. Available: http://dx.doi.org/10.1007/s00453-010-9482-1

[13] C. Adolphs and P. Berenbrink, "Distributed selfish load balancing with weights and speeds," in *ACM Symposium on Principles of Distributed Computing, PODC '12, Funchal, Madeira, Portugal, July 16-18, 2012*, 2012, pp. 135–144. [Online]. Available: http://doi.acm.org/10.1145/2332432.2332460

[14] D. Levin, Y. Peres, and E. Wilmer, *Markov chains and mixing times*. American Mathematical Society, 2006.

[15] B. Doerr and S. Pohl, "Run-time analysis of the (1+1) evolutionary algorithm optimizing linear functions over a finite alphabet," in *Genetic and Evolutionary Computation Conference, GECCO '12, Philadelphia, PA, USA, July 7-11, 2012*, 2012, pp. 1317–1324. [Online]. Available: http://doi.acm.org/10.1145/2330163.2330346

V. Appendix

A. Auxiliary Results

In this section we present some results that we used in our proofs. The first result is a well-known drift theorem. This version is from [15], similar results can be found in many publications.

Lemma 11. *Let $S \subseteq \mathbb{R}$ be a finite set of positive numbers with minimum s_{min}. Let $\{V(t)\}_{t\in\mathbb{N}}$ be a sequence of random variables over $S \cup \{0\}$. Let T be the random variable that denotes the first point in time $t \in \mathbb{N}$ for which $V(t) = 0$. Suppose that there exists a constant $\delta > 0$ such that*

$$\mathbb{E}\left[V(t) - V(t+1) \mid V(t) = s\right] \geq \delta s \tag{3}$$

holds for all $s \in S$ with $\mathbb{P}\left[V(t) = s\right] > 0$. Then for all $s_0 \in S$ with $\mathbb{P}\left[V(0) = s_0\right] > 0$,

$$\mathbb{E}\left[T \mid V(0) = s_0\right] \leq \frac{1 + \ln(s_0/s_{min})}{\delta}. \tag{4}$$

The following Lemma is shown in [14].

Lemma 12. *Let G be an arbitrary. Let $\mathbf{P}$ be the transition matrix of a random walk on G with . Let $t \geq 4\log n/\mu$, where μ is the spectral gap of $\mathbf{P}$. Then $\mathbf{P^t_{i,j}} = \pi_i \pm n^{-3}$.*

B. Missing Proofs

The following observation is used in the proof of Lemma 4

Observation 13. *Using the definitions from Lemma 4 we get*

$$\Delta\Phi(t + \tau(G)) \geq \frac{1}{2}\Delta\Phi'(t + \tau(G)).$$

Proof: Clearly $W(B_g) \geq 0$ and $W(R_f) \geq 0$, and therefore $W(B_g) + W(R_f) \geq 0$. Adding $W(R_f)$ on both sides gives

$$W(B_g) + 2W(R_f) \geq W(R_f).$$

Adding a $W(B_f)$ on both sides gives

$$W(B_g) + 2W(R_f) + W(B_f) \geq W(R_f) + W(B_f)$$

Since $W(B_f) + W(R_f) \geq W(B_n)$ we have

$$W(B_g) + 2W(R_f) + W(B_f) \geq W(B_n).$$

Dividing both sides by 2 gives

$$W(B_g)/2 + W(R_f) + W(B_f)/2 \geq W(B_n)/2.$$

Adding $W(B_g)/2 + W(B_f)/2$ on both sides gives

$$\begin{aligned} W(B_g) + W(R_f) + W(B_f) &\geq W(B_g)/2 + W(B_f)/2 + W(B_n)/2 \\ &= \frac{1}{2}(W(B_g) + W(B_f) + W(B_n)). \end{aligned}$$

The LHS is just $\Delta\Phi(t+\tau(G)) - W(R_g)$, and the RHS is $\frac{1}{2}\Delta\Phi'(t + \tau(G))$, so we obtain

$$\Delta\Phi(t + \tau(G)) - W(R_g) \geq \frac{1}{2}\Delta\Phi'(t + \tau(G)),$$

and as $W(R_g) \geq 0$ the claim of $\Delta\Phi(t+\tau(G)) \geq \frac{1}{2}\Delta\Phi'(t+\tau(G))$ follows. ∎