Advanced Algorithm Design: Load Balancing

Lectured by Prof. Moses Charikar
Transcribed by Sachin Ravi*

Feb 6th, 2013

1 Load Balancing

During last lecture, we showed that using the simple randomized strategy when assigning n balls to n bins,

$$\text{max load} = O\left(\frac{\log n}{\log \log n}\right).$$

We now consider a more intelligent strategy of assigning balls to bins and show that it results in a suprising improvement over the above result.

1.1 Power of two choices

In our new strategy, we choose two bins uniformly at random and place the ball in the bin with the smaller existing load. Following this strategy results in the following bound:

Theorem 1. *In this 2-choice strategy, the max load is $O(\log \log n)$.*

We first consider a rough sketch of why this new bound makes sense and then prove it using another method which involves using properties of randomized graphs.

Suppose for a ball b in B (the set of balls),

$$height(b) = \text{load of bin in which } b \text{ is placed.}$$

Then,

$$\text{E[fraction of balls in } B \text{ w/ } height(b) \geq 2] \leq \frac{1}{2},$$

since the total number of balls in all the bins equals n.

Notice then that

$$\text{E[fraction of balls in } B \text{ w/ } height(b) \geq 3] \leq \left(\frac{1}{2}\right)^2,$$

as it is necessary that for each ball, both the bins being considered have at least two balls.

*sachinr@princeton.edu

We can repeat the same logic above and get that in general

$$\text{E[fraction of balls in } B \text{ w/ } height(b) \geq i] \leq \left(\frac{1}{2}\right)^{2^{i-2}}.$$

In order to bound the size of largest the bin, we want to find the value of i such that the probability on the right is $\frac{1}{n}$. For this we need,

$$\begin{aligned} 2^i &\sim \Omega(\log n) \\ \Rightarrow i &\sim \Omega(\log \log n). \end{aligned}$$

1.2 Randomized Graphs

We consider the above strategy with $\frac{n}{8}$ balls and n bins because it makes the analysis cleaner due to the fact that $8 > e^2$. We will model the 2-choice strategy using a graph. Specifically, consider a graph G where the vertices represent bins and edges exist between two vertices if the two vertices were being considered when trying to place a ball. Essentially, the graph starts out as a set of vertices without any edges and in each step, we uniformly at random pick two vertices and insert an edge between these vertices. All vertices start out with a count of zero and the end point with the lower count gets incremented. The maximum load in this strategy is equal to the vertex with the highest count at the end of the process.

In order to analyze the two choice strategy, we will use two specific claims (which we prove later).

Claim 1. *With probability at least* $1 - \frac{1}{poly(n)}$*, the size of the largest connected component in this graph is* $O(\log n)$*.*

Claim 2. *For* $S \subseteq V$*, the induced degree of* S*, or the number of edges with both endpoints in* S*, is at most* $8|S|$ *with probability at least* $1 - \frac{1}{poly(n)}$*,*

We use a decomposition process to interatively remove vertices from G so that the at the end of the process only the vertex with the highest count is left. We bound the size of the count of this vertex and thus the number of balls the max-sized bin, represented by this vertex, can contain. Let A_1 be the vertices in G that have degree at most 16 and in general, let A_i be the set of vertices in $G \setminus \cup_{j \in [1,i-1]} A_j$ be the set of vertices that have degree at most 16.

By Claim 2, the average degree of every induced subgraph in G is 8 and so at most half the nodes can have more than twice the average degree, meaning that at each step of the process at least half the nodes in a connected component of G are removed. By Claim 1, since the largest connected component is of size $O(\log n)$, the process should take $O(\log \log n)$ steps to terminate.

We can prove the following lemma to get the desired bound.

Lemma 1. *The maximum count possible for a vertex is at most* $16r + 1$ *if this vertex is removed from the graph in round* r*.*

Proof. We prove the lemma be induction on the number of rounds. The base case ($r = 1$) is true based on the conditions stated for A_1. For the inductive step, notice that the count for a vertex $v \in A_i$ cannot increase by more than 16 when it and its associated edges are removed from the

graph (since the degree of each such vertex is at most 16). So, if we can show that before round i starts, the count for v is at most $16(r-1)+1$, we are done.

Suppose v has count more than $16(r-1)+1$. Consider the edge (v,w) that was removed and used to increment the count of v last. The count of w must have been at least $16(r-1)+1$, otherwise the count of w would have been incremented. Then $w \in \cup_{j\ in[1,i-1]} A_j$ and has a count at least $16(r-1)+1$, which contradicts the inductive assumption. □

We now prove the two claims we used above.

Proof of Claim 1. Suppose that the size of the largest connected component S in G is at least K. In order for S to be connected, it has to have at least $k-1$ edges. Thus,

$$\begin{aligned}\Pr[|S| \geq k] &\leq \binom{n}{k}\binom{\frac{n}{8}}{k-1}\left(\frac{k}{n}\right)^{2(k-1)} \qquad (1)\\ &\leq \left(\frac{en}{k}\right)^k \left(\frac{en}{8k}\right)^k \left(\frac{k}{n}\right)^{2k} \\ &= \frac{e^{2k}}{8^k} = \left(\frac{e^2}{8}\right)^k \approx (0.93)^k,\end{aligned}$$

where in (1), the first term is the number of different subsets of size k, the second term is all the different ways to pick $k-1$ edges, and the last term is the probability that both endpoints of each of the edges lie in S. This is where we crucially need $\frac{e^2}{8} < 1$.

Letting $k = O(\log n)$, we get that $(0.93)^k = \frac{1}{poly(n)}$ and so

$$\Pr[|S| \leq O(\log n)] \geq 1 - \frac{1}{poly(n)}$$

□

Proof of Claim 2. Suppose $|S| = k$, then using the same logic as above,

$$\begin{aligned}\Pr[deg(S) \geq 8k] &\leq \binom{n}{k}\binom{\frac{n}{8}}{4k}\left(\frac{k}{n}\right)^{8k} \qquad (2)\\ &\leq \left(\frac{en}{k}\right)^k \left(\frac{en}{32k}\right)^{4k} \left(\frac{k}{n}\right)^{8k} \\ &= \left(\frac{e^{1.25}}{32}\right)^{4k} \left(\frac{k}{n}\right)^{3k} \\ &\leq \left(\frac{k}{n}\right)^{3k},\end{aligned}$$

where in (2), the second term results from the fact that to have an induced degree of at least $8k$, at least $4k$ edges are necessary.

Using the union bound, we have

$$\Pr[\exists S \subseteq V \text{ s.t. } deg(S) \geq 8k] \leq \sum_k \binom{n}{k}\left(\frac{k}{n}\right)^{3k} = \frac{1}{poly(n)}$$

Thus,

$$\Pr[\forall S \subseteq V,\, deg(S) \leq 8k] \geq 1 - \frac{1}{poly(n)}$$

□

2 Cuckoo Hashing

We now discuss a special hashing scheme called *Cuckoo Hashing*. Here, we have two independent hash functions $h_1(x)$ and $h_2(x)$. Assuming we have a table T, an item x is stored in either $T[h_1(x)]$ or $T[h_2(x)]$ and we have no linked list to store collisions. The strategy is the following: we try to place x in $T[h_1(x)]$; if this position is occupied by an element y, evict it and try to place it either in $T[h_1(y)]$ or $T[h_2(y)]$ (either $h_1(y)$ or $h_2(y)$ must differ from $h_1(x)$).

In the best case, when trying to enter x_1 we will have a chain of evictions which finally result in some result in some x_n being inserted into a spot that is not occupied (see Figure 1).

It is also possible, however, to have a cycle when following the set of evictions. In this case, we try inserting at $h_2(x)$ and a set of evictions from this spot may lead to an insertion at some empty spot (see Figure 2). How will we determine that we are indeed stuck in a cycle? We can time-out our insertion if an empty spot is not found within $O(\log n)$ steps. The truly bad case is when the set of evictions from trying to insert at both $h_1(x)$ and $h_2(x)$ cause cycles (see Figure 3). In this case, we have to re-hash all the elements in the table using a different set of hash functions.

In the next lecture we will show that the probability of re-hashing is low, specifically

$$\Pr[re - hashing] \leq O\left(\frac{1}{n^2}\right).$$

3 References

1. http://www.cs.berkeley.edu/~satishr/cs270/sp11/rough-notes/cuckoo-hash.pdf

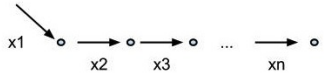

Figure 1: Case where evictions lead to insertion at empty spot.

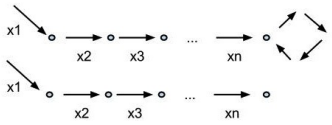

Figure 2: Case where evictions lead to one cycle and eventual insertion at empty spot.

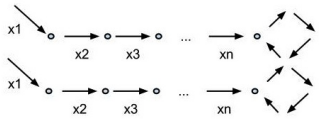

Figure 3: Case where evictions lead to two cycles, meaning re-hashing is necessary.